Gradient Vector Griding: An Approach to Shape-Based Object Detection in RoboCup Scenarios

Hamid Moballegh, Naja von Schmude, and Raúl Rojas

Institut für Informatik, Arbeitsgruppe Künstliche Intelligenz,
Freie Universität Berlin, Arnimallee 7, 14195 Berlin, Germany
moballegh@gmail.com, {schmude,rojas}@inf.fu-berlin.de
http://www.fumanoids.de

Abstract. This paper describes a new method of extraction and clustering of edges in images. The proposed method results a graph of detected edges instead of a binary mask of the edge pixels. The developed algorithm contains a sequential pixel-level scan, and a much smaller second and third pass on the results to determine the connectivities. It is therefore significantly faster than Canny edge detector, performing both edge detection and grouping tasks. The method is developed for a RoboCup scenario, however it can also be applied to any other image as long as the prerequisites are met. The paper explains the idea, discusses the prerequisites and finally presents the implementation results and issues.

1 Introduction

The vision system is the most important module of a robot to perceive its environment in RoboCup competitions. The task of the vision system is to analyze an incoming image for detecting important structures or areas (features) and combine these features to meaningful objects according to the needs of the other modules of the control software. As vision-based measurements are often used in closed loop control of the robots, a realtime performance is required. Due to the fact that on-board CPU power is strongly limited, high-performance and very specialized algorithms are needed.

Common approaches use image segmentation by color to deal with the object extraction[11,2]. The reason is that the relevant objects like goals, robots and the ball can be distinguished by their color. Acceptable detection rates can however only be achieved in presence of a constant and well conditioned lighting. Moreover, almost all color-based object recognition methods require a manual or automatic calibration, which makes them hard to use[5,6,7].

Shape-based approaches are rare but are getting more popular in RoboCup because the RoboCup rules change more and more from the highly specialized scenario to a more general one[8,9]. Therefore object recognition can't rely only on the color of the objects in the future, but needs to take the shape of the

T. Röfer et al. (Eds.): RoboCup 2011, LNCS 7416, pp. 162–173, 2012.

objects into account. Many methods have been introduced to detect RoboCup field objects based on their shapes. A majority of these methods focus on the field lines since the uncertainties of other landmarks are prohibitively large[9]. Different techniques have been proposed to extract the field lines. Some examples are the use of kernels[10], Hough transformation[1,10] and edge detection techniques[8].

In this paper we introduce a shape-based object detection scheme, which results a directed graph of detected edges. The graph isn't build on pixel basis but on cell basis; that means the image is overlaid with a grid formed of equal sized cells. A number of graph nodes is calculated for each cell. The nodes are then connected corresponding to the connectivity in the 8-neighborhood of each cell. This is similar to the technique suggested by A. Farag and E. Delp[4]. The algorithm allows adjusting the maximum curvature of the result paths. The idea of griding is inspired by the Histogram of Oriented Gradients (*HOG*) introduced by Dalal and Triggs[3]. However the final implementation differ greatly from the original work.

The paper is structured as follows: At first the four steps of the method are described. Several techniques are explained to increase the performance of the method as well as to improve the quality of the results. Finally achieved results from the algorithm are presented and the implementation issues are discussed.

2 Object Detection Using Gradient Vector Griding

The main idea of the Gradient Vector Griding (*GVG*) method is to reduce computation cost by reducing the image dimension to a much smaller grid. An important prerequisite for the method is that the information loss according to this stage remains small. This is provided in RoboCup case. For each cell, one or more features are calculated from the original image, where each feature represents an edge passage through the cell. These features are the so-called *edge representers* (*ER*). This is unlike *HOG* where a histogram of orientations is extracted for each cell. The next stage finds connected edges passing through neighboring cells; the connectivity is calculated. In the last step, complete edge traces are extracted out of the cell connectivity graph.

Figure 1 shows the basic implementation of the *GVG* algorithm. All stages are explained in detail in the next subsections.

2.1 Gradient Vector Calculation

Like every shape based object recognition method, the procedure begins with the calculation of the gradient vector for each pixel in the image. Sobel operators can be used to compute the components of the gradient vector in each pixel. The better alternative is Robert's cross operator described in equations 1 to 4. Note the gradient vector provided by this method is 45° rotated.

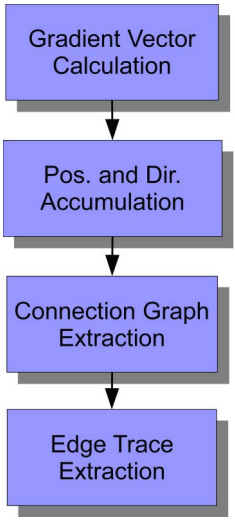


Fig. 1. Stages of Gradient Vector Griding

$$G_x : \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{1}$$

$$G_y : \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \tag{2}$$

$$|G| = \sqrt{G_x^2 + G_y^2} \text{ or } |G| = |G_x| + |G_y| \tag{3}$$

$$\theta = \arctan(G_x/G_y) + \frac{\pi}{4} \tag{4}$$

Sobel operators are less sensitive to noise, however the accumulation in the next stage of the algorithm makes the Robert's cross version pretty noise tolerant.

There are two important points to consider when calculating the gradient vector for *GVG*. First, it is necessary to calculate the direction in a -180° to 180° range, i.e. using the function $\arctan 2(y, x)$ instead of $\arctan(y/x)$. This prerequisite is discussed in detail in the next subsection. The second point is that the method requires the edge direction which is perpendicular to the gradient direction. The following convention is used in the algorithm to convert gradient vector $\mathbf{V}$ to edge vector $\mathbf{E}$, which is a simple 90° rotation.

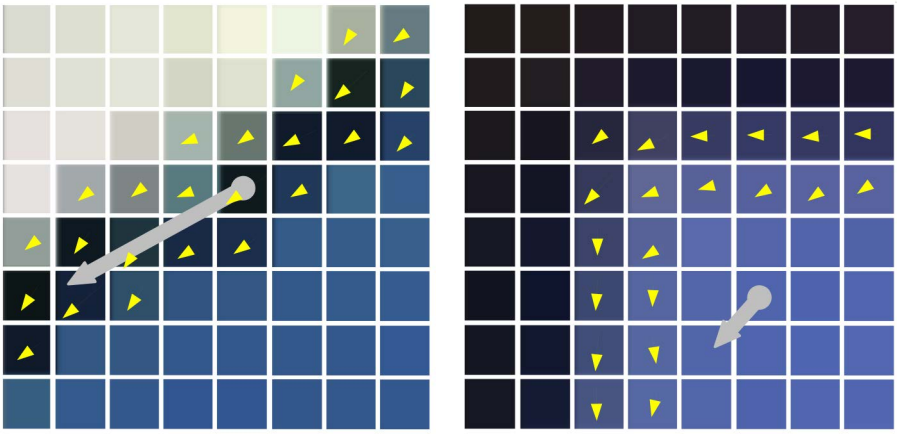
$$\mathbf{E} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \mathbf{V} \tag{5}$$

If Robert's cross operator is used for differentiating, edge directions should be calculated as follows, which is a rotation of 135°.

$$\mathbf{E} = \begin{bmatrix} -\frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \end{bmatrix} \mathbf{V} \tag{6}$$

2.2 Position and Direction Accumulation

A selective accumulation method is used to calculate a set of *ER*s for each cell. As mentioned above, an *ER* should refer to an edge passing through the cell. In the case that the cell contains a single linear edge (shown in figure 2(a)), the *ER* is simply extracted by averaging the position and edge vectors of the pixels inside the cell belonging to the edge. As the complexity of the cell contents increases, simple averaging fails as shown in figure 2(b). It is then required to distinguish between multiple edges using a more general clustering algorithm.



(a) Single Edge Case (b) Multiple Edge Case

Fig. 2. Average gradient direction and position in a cell. Averaging fails when multiple edges appear inside a cell (b) but works in the single edge case (a).

The problem cannot be solved in general form without a noticeable increase in the amount of calculations. We noticed that a partial solution should be enough for RoboCup use. Figure 3 presents some examples commonly observed in RoboCup scenario. Samples are overlaid with edge direction vectors. An often observed case is a field line or a side pole included with both side edges inside a single cell. The other but less frequent observation is a more or less 90° corner as a result of either an intersection between two lines or a part of a rectangular object. Thanks to 360° representation of the gradient direction, it is possible to separate the edges in a majority of the cases based only on the direction information. Note that the edges of an object such as a field line form two complement directions although they are parallel in the image.

The algorithm described in figure 5(a) is developed based on this idea. As a matter of optimization it is preferred to have a one-pass algorithm so that it can also be implemented without an entire image buffering. The algorithm functions as follows: The image is scanned pixel by pixel. Two *ER*s are used in

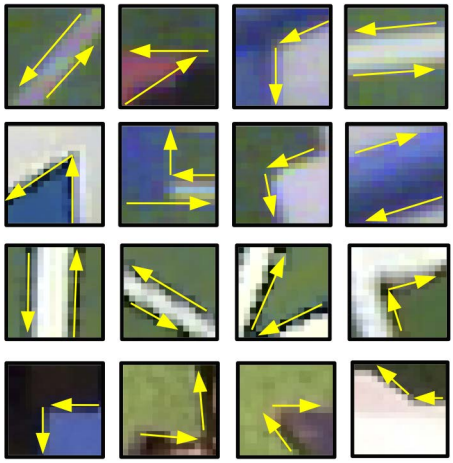


Fig. 3. Common examples of multiple edge distribution in RoboCup scenario

this implementation. Each *ER* contains a position, an orientation accumulator and a pixel counter. An edge pixel is joined to the first *ER* when its gradient orientation is closer than a certain distance to the average orientation of the *ER*. Otherwise it is compared to the second *ER*, and in the case of no match it is ignored. An empty *ER* will obviously be filled with the first edge pixel met. The angle distance is defined in equation 7.

$$|\theta_1 - \theta_2| = \begin{cases} |\theta_1 - \theta_2 + 360| & \theta_1 - \theta_2 < -180 \\ |\theta_1 - \theta_2 - 360| & \theta_1 - \theta_2 > 180 \\ |\theta_1 - \theta_2| & otherwise \end{cases} . \tag{7}$$

As a further optimization to the algorithm it is possible to replace the polar representation of the gradient vector with a cartesian one, i.e. using directly the edge vector (e_x, e_y) instead of its orientation θ.

The implementation of angle distance thresholding will then be replaced with thresholding the inner product $\mathbf{V}_1 \cdot \mathbf{V}_1 = |\mathbf{V}_1||\mathbf{V}_2| \cos\theta = e_{x_1} e_{x_2} + e_{y_1} e_{y_2}$ of the vectors $\mathbf{V}_1$ and $\mathbf{V}_2$ as shown in equation 8.

$$\mathbf{V}_1 \cdot \mathbf{V}_2 > |\mathbf{V}_1|\,|\mathbf{V}_2| \cos(thr) \tag{8}$$

$$\Leftrightarrow \quad e_{x_1} e_{x_2} + e_{y_1} e_{y_2} > \sqrt{e_{x_1}^2 + e_{y_1}^2}\sqrt{e_{x_2}^2 + e_{y_2}^2} \cos(thr) \tag{9}$$

This saves the dynamic calculation of the arctan 2 function needed in the polar presentation. According to the CPU documentation, 32 bit multiplication can be performed in one cycle. However the normalization of the vectors $|\mathbf{V}_1|$ and $|\mathbf{V}_2|$ could cause performance problems, because the square root has to be calculated.

This can also be optimized away using the z component of the cross product $\mathbf{V}_1 \times \mathbf{V}_2 = |\mathbf{V}_1||\mathbf{V}_2| \sin\theta \boldsymbol{n}$, where $\boldsymbol{n}$ represents the normal vector to $\mathbf{V}_1$ and

$\mathbf{V}_2$. As the vectors are defined in $\mathbb{R}^2$, we use zero as z component of $\mathbf{V}_i$, so the calculation of the cross product results in

$$\mathbf{V}_1 \times \mathbf{V}_2 = \begin{pmatrix} e_{x_1} \\ e_{y_1} \\ 0 \end{pmatrix} \times \begin{pmatrix} e_{x_2} \\ e_{y_2} \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ e_{x_1} e_{y_2} - e_{y_1} e_{x_2} \end{pmatrix}. \tag{10}$$

We can now write $|(\mathbf{V}_1 \times \mathbf{V}_2)_z| = |\mathbf{V}_1||\mathbf{V}_2| \sin\theta = |e_{x_1} e_{y_2} - e_{y_1} e_{x_2}|$. As $\tan\theta = sin\theta / \cos\theta$, the sinus and cosine function can be replaced with the cross and inner product described above.

$$\tan\theta = \frac{\frac{|(\mathbf{V}_1 \times \mathbf{V}_2)_z|}{|\mathbf{V}_1||\mathbf{V}_2|}}{\frac{\mathbf{V}_1 \cdot \mathbf{V}_2}{|\mathbf{V}_1||\mathbf{V}_2|}} = \frac{|(\mathbf{V}_1 \times \mathbf{V}_2)_z|}{\mathbf{V}_1 \cdot \mathbf{V}_2} = \frac{|e_{x_1} e_{y_2} - e_{y_1} e_{x_2}|}{e_{x_1} e_{x_2} + e_{y_1} e_{y_2}} \tag{11}$$

To avoid the division, we multiply the tangents with the cosine part and the final angle thresholding of the vectors is then given in equation 12.

$$|(\mathbf{V}_1 \times \mathbf{V}_2)_z| < \mathbf{V}_1 \cdot \mathbf{V}_2 \tan(thr) \tag{12}$$

$$\Leftrightarrow \quad |e_{x_1} e_{y_2} - e_{y_1} e_{x_2}| < e_{x_1} e_{x_2} + e_{y_1} e_{y_2} \tan(thr) \tag{13}$$

Using this technique, vector angle thresholding is done with only four integer multiplications having a great impact on the performance of the algorithm.

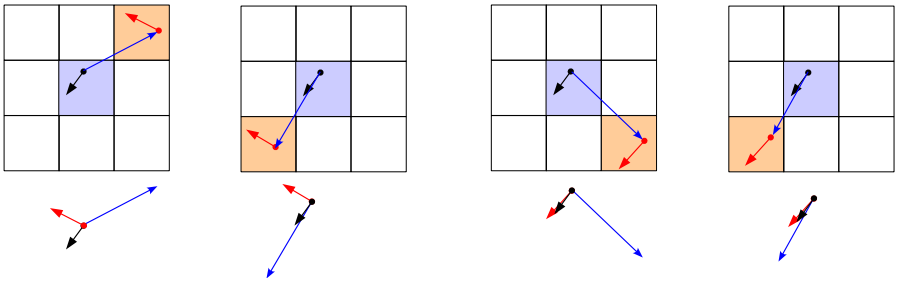
The grid structure can be chosen overlapped, i.e. each cell also covers half of every neighboring cell. This increases the smoothness of the results but is computationally more expensive.

2.3 Connection Graph Extraction

A list of *ER*s is produced through a single image scan. A connection graph should be calculated by scanning the grid and comparing *ER*s of neighboring cells. The graph is implemented using an extra field in the cell structure pointing to the following neighbor *ER* called *"outbound"* and a boolean field indicating that the *ER* is added to the trace called *"inbound"*.

The algorithm is shown in figure 5(b). Upon two conditions two neighbor *ER*s are marked as connected. The first condition verifies that both *ER*s are in the same direction. This condition is however not enough as it also holds true for separate parallel edges. Therefore the second condition verifies that the vector connecting the *ER*s is also in the same direction as both ERs. By adjusting the thresholds the maximum accepted curvature of the edge trace can be determined. Both conditions can be optimized using the technique described in the last section. Note that for the second condition the edge direction should be used. Figure 4 demonstrates different examples of neighboring *ER*s. A connection is only accepted in example 4(d).

The internal loop of the algorithm breaks as soon as a connection is found. This guarantees that every node in the connection graph has an out degree of at most one. It is encouraged also to reduce the in degree of the nodes to at most



(a) Cond1 does not hold. Cond2 does not hold.

(b) Cond1 does not hold. Cond2 holds.

(c) Cond1 holds. Cond2 does not hold.

(d) Cond1 holds. Cond2 holds.

Fig. 4. Examples of connected and unconnected neighbors

one. This is implemented by refusing a connection if the destination node has already been connected by checking its *inbound* property.

It is theoretically not guaranteed that the connection graph becomes free from loops, however if a loop exists it should be the result of an uninterrupted semi-circular edge in the image which does not usually appear in RoboCup images.

2.4 Edge Trace Extraction

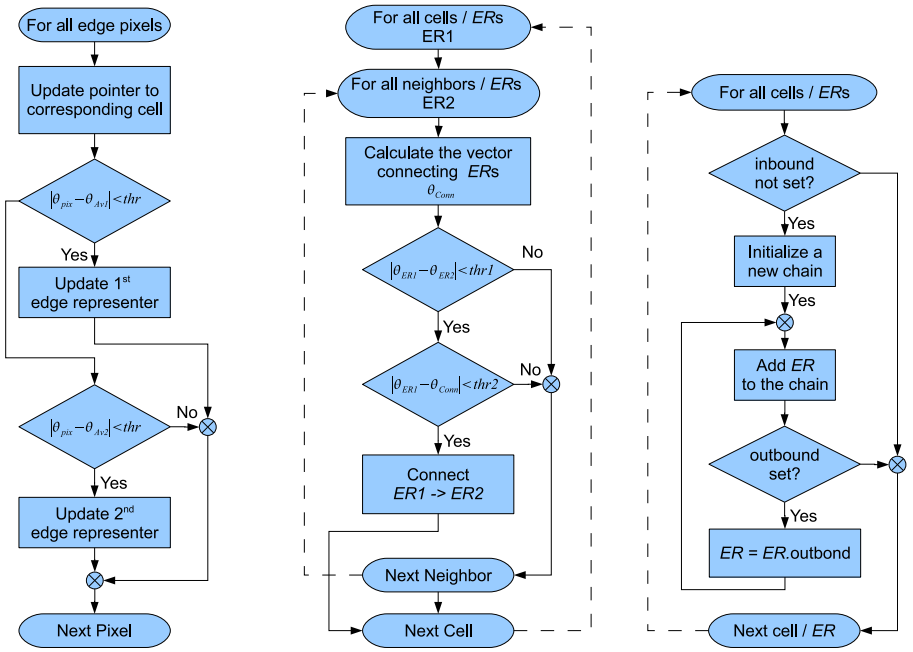
The final stage of the algorithm produces an array of edge traces. Each component of the array is a connected *ER* chain. The algorithm is demonstrated in figure 5(c). It searches the graph for source nodes, which are simply *ER*s with the *inbound* property not set. Upon a match, the trace is followed using the *outbound* pointers and the matching *ER*s are pushed in the trace.

3 Implementation and Experimental Results

The algorithm was implemented as the low level part of the vision software for participation in RoboCup 2010. Two *ER*s are calculated per cell. The grid contains 40×30 non-overlapping cells, each covering 256 pixels of the captured image.

To achieve the required frame rate and still have enough CPU power free for other processes running, the following optimizations are applied to the algorithm.

Quarter Resolution Scan. Color digital cameras usually provide images with a so called *"Bayer pattern"*. A VGA image contains therefore 640×480 single channel values. This is 1/3 of the information recorded in common RGB pattern. The remaining values are interpolated in such representations. Hence it is possible to skip every other pixel and every other image line without a distinct loss of information.



(a) Position and direction accumulation algorithm

(b) Connection graph extraction algorithm

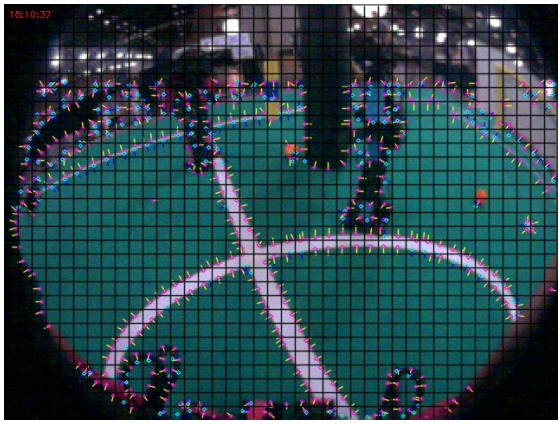
(c) Edge extraction algorithm

Fig. 5. Algorithms

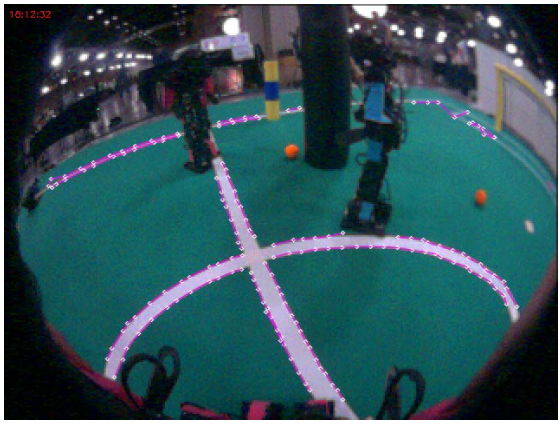
Over Horizon Skip. The only objects which partly appear over the horizon are the landmarks and other robots. To detect and calculate the position of these objects it is always enough to find the lowest point belonging to them. Therefore there is no need to detect objects over the horizon. As the camera is equipped with a wide angle lens, the horizon is projected as a curve in the captured image. This can however be estimated with a horizontal line touching the actual projection in the top-most point. There are different methods to calculate the horizon in the image. Two solutions currently used in FUmanoids are IMU sensors and field range detection. Upon detection of the horizon, scanning the image can vertically begin from this line.

Out of Circle Skip. Fish eye optics used in FUmanoid robots project the image inside a circle. The rest of the imaging surface is covered with black pixels. This effect can be observed in figure 6. It is possible to skip these pixels by calculating the horizontal extents for each image line.

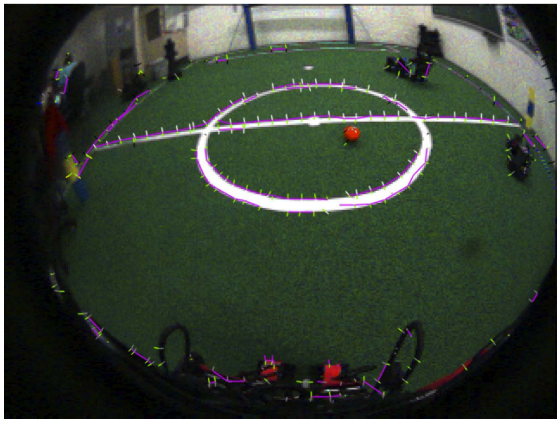
Random Line Skip. Apart from the fish-eye optics deformation, the camera provides a perspective view of the field. Therefore much more information is available from near objects, observed in the lower area of the image, than from far ones appearing in the upper area. Due to this fact it is possible to ignore more and more lines as the scan gets closer to the bottom of the image. This is done by comparing a uniformly distributed random number



(a) Visualization of the *ER*s.



(b) Extracted edges overlaid on the test image.



(c) Typical results of the test case.

Fig. 6. Implementation results of GVG

with a dynamic threshold calculated form the scan lines Y coordinate. The following formula is used to skip a line:

$$r < Y/640 \tag{14}$$

Here r is a random real number between 0 and 1 and Y is the vertical component of the scan line coordinate.

Implementation results are summarized in table 1 on two available processor platforms. The first platform is Gumstix Verdex Pro which is equipped with a 600 MHz PXA270 and the second platform is a Gumstix Overo with a 720 MHz OMAP3 3530 processor. Each row of the table shows the optimization added to the last state. The last row shows the highest optimization level achieved by applying all described techniques.

A standard test procedure is used for this performance measurement. The robot is placed upright on one of the penalty points directed to the goal placed on the other side of the field as shown in figure 6(c). Two parameters are measured, which together show the performance of the algorithm. These are processed frame rate and CPU usage. Since the camera delivers a limited number of frames per second and the CPU is also capable of performing a limited amount of processing, there are two possible scenarios.

1. A frame can be entirely processed before the next frame is available. In this case the processor goes idle and the CPU shows less than 100% of usage. Frame rate remains constant in this mode and CPU usage is used as the indicator of the performance.
2. A new frame gets available while the last frame is still being processed. This leads to a frame buffer overflow, which is in turn handled with dropping frames. CPU is never idle in this case and the usage indicator shows always 100%. Frame rate is then used as the performance indicator.

Using the Canny algorithm from the OpenCV implementation as edge detection, the described test procedure shows the results listed in table 2. Compared to the *GVG*, the Canny edge extraction is slower by a factor of 10. The Canny algorithm also only results a bitmap of the edges and doesn't perform any edge

Table 1. GVG Implementation Results

	Gumstix Overo		Gumstix Verdex	
	Frame rate (FPS)	CPU usage (%)	Frame rate (FPS)	CPU usage (%)
No optimization	12	100	10	100
Quarter res. scan	18	100	14	100
+Over horizon skip	20	100	17	80
+Out of circle skip	26	80	17	75
+Random line skip	26	65	17	60

Table 2. Canny Implementation Results

	Gumstix Overo		Gumstix Verdex	
	Frame rate (FPS)	CPU usage (%)	Frame rate (FPS)	CPU usage (%)
Canny	2.7	100	1.5	100

grouping, so this has to be done additionally. Therefore we can conclude that the *GVG* algorithm significantly outperforms the standard Canny procedure in the RoboCup scenario. Figure 6(a) shows the visualization of the *ER*s for a typical image captured by the robot. Despite the rough grid a relatively high level of detail is detected. In figure 6(b) the result of the edge trace extraction is demonstrated for field lines.

4 Conclusion

In this work we introduced an edge detection and grouping method based on gradient vector directions. The method was described and some key ideas to increase the performance of the method in each stage were presented. Implementation results and issues were given and discussed in the paper. It was shown as well that the new algorithm outperforms the Canny edge detector in the RoboCup scenario.

The developed method was successfully used as the pixel-level processor in the vision system of the RoboCup team FUmanoids in 2010. The results provided by this method facilitated robust object recognition as well as rapid localization of the robot. The team won the second place of the Humanoid Kid-Size competition.

References

1. Bais, A., Sablatnig, R., Novak, G.: Line-based landmark recognition for self-localization of soccer robots. In: Proceedings of the IEEE Symposium on Emerging Technologies, pp. 132–137 (2005)
2. Bandlow, T., Klupsch, M., Hanek, R., Schmitt, T.: Fast Image Segmentation, Object Recognition, and Localization in a RoboCup Scenario. In: Veloso, M.M., Pagello, E., Kitano, H. (eds.) RoboCup 1999. LNCS (LNAI), vol. 1856, pp. 174–185. Springer, Heidelberg (2000)
3. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 886–893 (2005)
4. Farag, A.A., Delp, E.J.: Edge linking by sequential search. Pattern Recognition 28(5), 611–633 (1995)
5. Gunnarsson, K., Wiesel, F., Rojas, R.: The Color and the Shape: Automatic On-Line Color Calibration for Autonomous Robots. In: Bredenfeld, A., Jacoff, A., Noda, I., Takahashi, Y. (eds.) RoboCup 2005. LNCS (LNAI), vol. 4020, pp. 347–358. Springer, Heidelberg (2006)

6. Heinemann, P., Sehnke, F., Streichert, F., Zell, A.: Towards a Calibration-Free Robot: The ACT Algorithm for Automatic Online Color Training. In: Lakemeyer, G., Sklar, E., Sorrenti, D.G., Takahashi, T. (eds.) RoboCup 2006. LNCS (LNAI), vol. 4434, pp. 363–370. Springer, Heidelberg (2007)
7. Mayer, G., Utz, H., Kraetzschmar, G.: Towards autonomous vision self-calibration for soccer robots. In: Proceedings of the 2002 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems (2002)
8. Röfer, T., Jüngel, M.: Fast and Robust Edge-Based Localization in the Sony Four-Legged Robot League. In: Polani, D., Browning, B., Bonarini, A., Yoshida, K. (eds.) RoboCup 2003. LNCS (LNAI), vol. 3020, pp. 262–273. Springer, Heidelberg (2004)
9. Schulz, H., Liu, W., Stückler, J., Behnke, S.: Utilizing the Structure of Field Lines for Efficient Soccer Robot Localization. In: Ruiz-del Solar, J., Chown, E., Plöger, P. (eds.) RoboCup 2010. LNCS (LNAI), vol. 6556, pp. 397–408. Springer, Heidelberg (2010)
10. Strasdat, H., Bennewitz, M., Behnke, S.: Multi-cue Localization for Soccer Playing Humanoid Robots. In: Lakemeyer, G., Sklar, E., Sorrenti, D.G., Takahashi, T. (eds.) RoboCup 2006. LNCS (LNAI), vol. 4434, pp. 245–257. Springer, Heidelberg (2007)
11. Wasik, Z., Saffiotti, R.: Robust color segmentation for the robocup domain. In: Proc. of the Int. Conf. on Pattern Recognition (ICPR), pp. 651–654 (2002)