Using Genetic Algorithms for Real-Time Object Detection

J. Martínez-Gómez[1], J.A. Gámez[1], I. García-Varea[1], and V. Matellán[2]

[1] Computing Systems Department, University of Castilla-La Mancha, Spain
[2] Dept. of Mechanical and Computer Engineering, University of León, Spain
{jesus_martinez,jgamez,ivarea}@dsi.uclm.es, vicente.matellan@unileon.es

Abstract. This article presents a new approach to mobile robot vision based on genetic algorithms. The major contribution of this proposal is the real-time adaptation of genetic algorithms, which are generally used offline. In order to achieve this goal, the execution time must be as short as possible. The scope of this system is the Standard Platform category of the RoboCup[1] soccer competition. The system developed detects and estimates distance and orientation to key elements on a football field, such as the ball and goals. Different experiments have been carried out within an official RoboCup environment.

1 Introduction

For mobile robotics, image processing has become one of the most important elements. Intelligent robots need to retrieve information from the environment in order to interact with it. Vision cameras are one of a robot's key devices. The images taken by the robot's camera need to be processed in real time with limited processing resources. The systems developed need to cope with noisy and low quality images, and in order to process the maximum number of images by second, the algorithms must be as efficient as possible.

In the RoboCup[1] environment different solutions have been proposed over the last years. These proposals use the information obtained with colour filtering processes[2]. Scan-lines[3] and edge-based[4] solutions have been one of the most widely-used for the RoboCup competition.

The approach presented here carries out object recognition by using real-time genetic algorithms[5](GAs). The number of iterations and individuals for the GA must be reduced as much as possible in order to improve efficiency (some authors propose the use of cellular GAs[6] instead of reducing the number of individuals and iterations). This is necessary because the system has to be applied in real time. In order to prevent system performance from being affected by this reduction, the individuals will be initialized using all the available information. This initial information can be obtained from previous populations and from the colour filtering process applied to the last image taken by the robot's camera. After an image showing an object o, the next image has a high probability of

[1] http://www.robocup.org/

J. Baltes et al. (Eds.): RoboCup 2009, LNAI 5949, pp. 215–227, 2010.

showing the same object. The information obtained from previous populations allows us to take advantage of the high similarity between consecutives images taken by the camera.

Our hypothesis is that the similarity between captured images, and the information obtained with the filtering process, can be used to develop a real-time vision system based on genetic algorithms. Different tests in real scenarios using the biped robot Nao have been carried out to evaluate our proposal. These tests show the object (ball and goals) recognition process on the official RoboCup football field.

The article is organized as follows: problem restrictions are outlined in Section 2. We describe the full vision system in Section 3, and in Section 4 we explain the experiments performed and the results obtained. Finally, the conclusions and areas for future work are given in Section 5.

2 Problem Restrictions

The vision system has to be valid for use in the Standard Platform category. Robot Nao[2] is the official platform for this category, and its camera takes 30 (320 x 240 pixels) frames per second. The camera's native colour space is YUV[7].

In order to reduce the amount of information to work with, the captured images are filtered. This processing removes the pixels that do not pass a colour filter. The key colours in the RoboCup environment are yellow and blue for the goals, green for the carpet, orange for the ball and white for the field lines. Football player equipment is red and dark blue. The filtering is carried out by defining a top and bottom limit for the Y, U and V colour components. A pixel will successfully pass a filter only if all its components are between these limits. Fig. 1 shows a filtering example for blue.

Fig. 1. Colour filtering for the blue goal

Object recognition has to be carried out during a football match. The environment includes objects that are partially hidden behind others, so the frames taken in a football match will not always show the complete object we want to recognize. Scan-line-based methods present a lot of problems in these situations, whereas our system works properly, as will be shown in the results section.

[2] http://www.aldebaran-robotics.com/eng/Nao.php

3 Vision System

Genetic algorithms use individuals that represent potential solutions to problems. For our vision system, individuals have to represent the detection of the object o placed at distance d with the orientation or. This information (object o at distance d with orientation or) is contrasted with the one extracted from the last frame captured by the robot's camera. The fitness will be high for individuals with information that is plausible with respect to the last image. On the other hand, the fitness will be low if the object o does not appear in the image.

3.1 General Processing Scheme

The processing starts with the arrival of new images at the robot's camera. A new image will evolve a new population for each object to be recognized. In this work, three distinct objects are considered, the blue goal, the yellow goal and the orange ball, so three different populations will be kept. After taking a new image, the colour filtering allows the robot to know the objects likely to appear in the image. The populations of the non-plausible objects will not be evolved. Fig.2 shows the general processing scheme.

```
Capture a new image and filter it with colour filters
for each object to recognize
    if we have obtained enough pixels
        Evolve a new population
        Apply local search over the best individual
        Return the estimated distance to the object
    end if
end for
```

Fig. 2. General system processing scheme

In order to avoid local optimums, the population will be restarted after a given number of iterations failing to improve the best individual. An iteration will increase the value of a counter if the best fitness of the iteration is not greater than the best global fitness. The counter value will be set to zero if the iteration obtains the best fitness. The population will be restarted if the counter reaches a limit value.

3.2 Genetic Representation

In addition to the distance between the camera and the object to be recognized, we also need to estimate the orientation between both elements. This information is not only needed for self-location tasks[8], but also for the application of the fitness function. The shape of an object in an image will depend on the distance and the orientation between object and camera.

Fig.3 presents graphically the three parameters to be estimated: d is the distance between camera and object, α is the difference of orientation in the

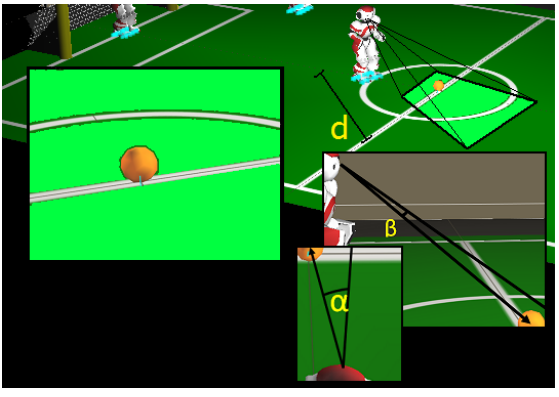

Fig. 3. Image taken with specific distance and orientation between object and camera

x-axis and β in the y-axis. With the same distance d and different α or β values, captured images will show the same ball but located at a different position within the image. The image will not show the ball with big α or β variations. A third component for the orientation difference in the z-axis is not needed, because using horizon detection techniques[9], the image can be processed to show all the objects parallel to the floor.

Each individual stores the following information (genes):

- Distance to the robot: d
- Orientation difference in the x-axis: α
- Orientation difference in the y-axis: β

All the genes are represented by a numerical value, limited by the maximum distance detection for d, and by the field of view for α and β. An additional gene is needed to perform goal detection. This gene (θ) represents the goal orientation when the frame is taken. Two frames taken with the same $< d, \alpha, \beta >$ parameters will be different if the goal orientation varies, as can be observed in Fig.4.

3.3 Obtaining the β Parameter

We can avoid modelling β if we know the angle between the camera and the floor in the y-axis, γ. Thus β can be calculated using γ, the distance d, and the

Fig. 4. Images taken varying the θ parameter

orientation in the x-axis α. With this approach, the areas of the search space that represent unreal solutions will not be explored. Using γ and the camera's field of view, we can obtain the minimum and maximum distances at which we can detect elements. For instance, if γ is close to 90 degrees, the robot will be able to recognize distant objects, but not a nearby ball.

The main problem of calculating β instead of modelling it is that our algorithm will heavily depend on γ estimation and its performance will decrease if γ is not correctly estimated. For legged robots, the movement of the robot causes an enormous variation in the camera angle, which makes it difficult to obtain a precise value for γ. For wheeled robots, the movement will not affect the camera angle as much as for legged ones and γ can be accurately calculated.

3.4 Fitness Function

The fitness function returns numeric values, according to the goodness of the projection obtained with the parameters $< d, \alpha, \beta >$ of each individual. To evaluate an individual, its genes are translated into a projection of the object that the individual represents. The projection needs a start position $< x, y >$, obtained from α and β. The size of the object depends on d.

An object projection is evaluated by comparing between it with the information obtained from the filtering process. A pixel $< x, y >$ of the projection will be valid only if the pixel $< x, y >$ of the image captured by the robot's camera successfully passes the colour filter. This evaluation is illustrated in Fig. 5, where the left image shows the original image after an orange filter. The right one shows the result of evaluating 12 different individuals, where red pixels are invalid (they have not passed the colour filter) and green pixels are valid.

After this processing we obtain the number of valid and invalid pixels for each individual. Using the percentage of pixels that pass the filter as a fitness function has a serious drawback: individuals representing distant objects obtain better fitness values. Those individuals correspond to smaller projections resulting in a higher probability of having a bigger percentage of valid pixels (few right pixels mean high percentage).

Due to this problem, and using the information obtained with the filtering, we define the fitness function as the minimum value of:

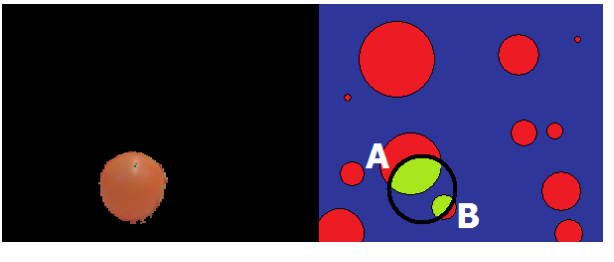

Fig. 5. Filtered image (left) and evaluation of 12 individuals to detect the ball (right)

– % of pixels of the projection that have passed the colour filter.
– % of pixels that passed the colour filter and belong to the valid projection pixels.

In order to illustrate the behaviour of the function, let's study individuals A and B in Fig. 5. Individual B has a higher percentage of pixels that passed the filter (70 versus 45). On the other hand, only 5% of the pixels that passed the orange filter belong to individual B. For A, this percentage rises to 35%. The fitness value will be 0.35 for individual A and 0.05 for B.

3.5 Population Initialization

The population is usually randomly initialized for GAs. Our approach uses additional information to initialize the first individuals. Firstly, we could use individuals from populations of previous captures. In addition to this, the information extracted from the filtering process could also be used. Such information is the number of pixels of each colour, and the x and y component of the centroid of the distribution of pixels obtained with the colour filter. According to this information, an individual can be initialized in 3 different ways:

– Randomly.
– Using the information from the filtering process.
– Cloning an individual from a previous population.

The first two ways of generating a new individual can always be used. The third one can only be used when a valid population is available. Such population must have been evolved to recognize the same object o that we want to recognize. The number of frames between the current one and the last one that evolved a population to recognize o has to be small. If the frame number difference is large, the individuals of the population will not be valid for cloning, because these individuals were evolved to solve a situation different from the current one. A draw is carried out to select the way in which an individual is initialized. All the ways have a probability that depends on the number of frames from the last frame that recognized the object we are studying. We need two parameters to obtain these probabilities:

– MW: Max probability of cloning an individual from a previous population.
– MNF: Max number of frames possible between the present frame and the last one that recognized the object we are studying.

The sum of the three parameters is normalized to be 1.0. The probability of initializing individuals by cloning them from other populations ($ClonеProb$) will decrease if the number of frames without updating the population ($NFWU$) increases. The other two probabilities are calculated using $ClonеProb$.

$$ClonеProb : MW - MW * (NFWU/MNF)$$
$$InitialInfoProb : (1 - (ClonеProb)) * 0.66$$
$$RandomlyProb : (1 - (ClonеProb)) * 0.34$$

If we increase the number of individuals that are randomly initialized, the variety of the initial population will be greater. Using the initial information,

Fig. 6. Object partially captured

the algorithm's elitism will increase (with the problem of local optimums). With individuals cloned from other populations, the algorithm will converge faster with small variations between frames. The balance between elitism and generality can be obtained through a correct combination of these three ways. We selected 0.66 and 0.34 as values to obtain a heterogeneous initial population, based on preliminary empirical tests.

3.6 Partial Object Occlusion

Vision systems must cope with hard environments. For instance, the objects to recognize can be partially hidden behind other robots, or the images captured by the robot's camera may show only parts of the desired object, due to the camera's orientation. Our proposal performs the individual's evaluation using the entire object's projection and not partial characteristics. This is the reason that our system works properly with occlusions.

4 Experiments and Results

The experiments were carried out on a RoboCup Standard Platform football field, with the official goals, a 6 x 4 metre carpet and a ball. We used a Nao robot, taking 2 images per second. The format of the images is YUV and their size is 320 x 240 pixels. While the experiments were being carried out, the absolute difference between the real and estimated distance to the object we wanted to detect was stored per frame. The estimated distance was the value of the d gene of the individual with the best fitness. Lighting conditions were stable throughout the experiments, and the colour filters were optimal.

The execution time for each image was variable. We decided to use two frames per second because the maximum execution time was never greater than 450 milliseconds.

After the filtering process ($\approx$ 80 msec), the execution time was never greater than 370 milliseconds (183 for the goal and 187 for the ball).

4.1 Genetic Algorithm Parameters

The experiments were carried out with the following parameters:

– Individual number: 12 and Iteration number: 24

- Mutation probability: 5% and Crossover type: point
- Replacement: generational algorithm
- Restart after 25% iterations without improving the global optimum
- MW: 0.5 and MNF: 10

The algorithm uses a limited number of individuals and iterations. The mutation probability and the crossover type are standard, and the entire population is replaced with the offspring at the end of the iteration. Due to this, the quality of the population can decrease while the search progresses. Evolution is performed without taking into account robot's odometry.

After evolving the population, a simple local search process (Hill Climbing) is applied to the best individual. This processing will allow us to improve the best fitness. The local search is applied by evaluating positive and negative variations for the genes of the individual. The algorithms that combine concepts and strategies from different metaheuristics are called memetic algorithms [10].

4.2 Experiment 1 - Hypothesis Validation

The objective of the first experiment was to prove that the system is able to work in the given time-frame, recognizing the environment elements and estimating the distance to them. We used the standard parameters described above and we executed the same tour over the football field 6 times. 30 frames were taken per tour (15 seconds). The frames captured the yellow goal placed between 360 and 300 cm, and the orange ball placed at distances between 260 and 200 cm.

The experiment consisted of 180 different frames (6 x 30). We stored the absolute difference between real and estimated distance (denoted $DBRED$) and the fitness of the best individual of the population by frame. These fitness values were used to generate different data sets. Each one of these data sets had only the detections carried out with individuals whose fitness values were greater than certain thresholds. Table 1 shows, taking the ball and yellow goal separately, and with four different threshold values (0, 0.25, 0.5 and 0.75), the average of the $DBRED$. It also gives the percentage of frames that obtained an individual with a fitness value greater than the threshold.

Table 1. Average $DBRED$ and % of frames with a fitness value over certain thresholds

Fitness		> 0.0	> 0.25	> 0.5	> 0.75		> 0.0	> 0.25	> 0.5	> 0.75
Average (cm)	Ball	42.62	40.57	31.77	22.75	Yellow	40.03	37.88	33.1	32.69
Frames (%)		68.89	68.33	57.78	8.89	Goal	99.44	93.33	44.44	8.89

It can be seen that the fitness function properly represents the goodness of the individuals. This is because using individuals with higher fitness values reduced the average of the differences between real and estimated distances. Table 2 shows the percentage of frames that obtained a difference between estimated and real distance lower than certain thresholds.

Table 2. Percentage of frames that obtained a $DBRED$ lower than certain thresholds

	Percentage of frames under					Percentage of frames under			
	100 cm	75 cm	50 cm	30 cm		100 cm	75 cm	50 cm	30 cm
Ball	63.63	56.11	44.44	35.55	Yellow Goal	92.77	87.78	72.22	51.67

The results obtained show a high degree of robustness, especially for the yellow goal. In an environment with a maximum distance of 721 cm, a high (37.37% and 51.67%) percentage of frames obtained differences for the distance estimation under 30 centimetres.

Ball recognition (with our genetic algorithm) was more complicated than goal recognition, because only individuals which are very close to the solution (perfect detection) obtain fitness values different from zero. Due to the small size of the ball in the frames captured, only the projections of individuals close to the solution have pixels in common with the image obtained after the colour filtering process. The convergence of a GA with this kind of individuals will not be constant. 83.83% of correct ball recognitions (fitness$>$ 0) were carried out with fitness values greater than 0.5. For the goal, this percentage descends to 44.69%.

4.3 Experiment 2 - β Study

The main objective of the second experiment was to test whether the β parameter can be calculated using the other parameters. The performance of the algorithm obtaining β instead of modelling it was studied. The robot made the same tour as in experiment 1.

For this experiment, the individuals did not use the β gene, but the parameter is needed for the fitness function and has to be calculated. This was done using the parameters d (distance to the object), α (orientation difference in the x-axis) and γ (orientation difference between the robot's camera and the floor in the y-axis). γ is obtained using the robot's sensors. The experiment consisted of 180 frames again and a summary of the results obtained is shown in table 3.

Table 3. Average $DBRED$ and % of frames with a fitness value over certain thresholds

Fitness		> 0.0	> 0.25	> 0.5	> 0.75		> 0.0	> 0.25	> 0.5	> 0.75
Average (cm)	Ball	18.70	18.05	16.89	27.7	Yellow	33.66	32.81	34.31	27.5
Frames (%)		69.44	68.33	57.78	5.55	Goal	100.0	95.00	40.56	1.11

The first conclusion drawn from the results is that the number of correct detections (frames that obtained fitness values greater than 0) has increased. However, the percentage of frames with a fitness value greater than 0.5 and 0.75 decreased. This is because modelling β instead of obtaining it from the other parameters lets the algorithm to reach situations that are not right according to

the model, but which are valid due to noise or the difference between the real and estimated γ value.

The average difference between the real and estimated distance($DBRED$) decreased considerably. With lower gene numbers and the same iterations, GAs converge faster to better solutions. In order to establish a complete comparison between modelling β and calculating it with other parameters, table 4 provides the percentage of frames that obtained a $DBRED$ lower than certain thresholds.

Table 4. Percentage of frames that obtained a $DBRED$ lower than certain thresholds

	Percentage of frames under					Percentage of frames under			
	100cm	75cm	50cm	30cm		100cm	75cm	50cm	30cm
Ball	68.89	68.89	65.56	54.44	Yellow Goal	96.67	93.33	76.67	48.89

If we compare table 4 and 2, we can see that the robustness of the algorithm has improved. The faster convergence of the algorithm with fewer genes makes it possible to obtain a higher percentage of frames with a small $DBRED$ to the object.

The main conclusion drawn from the data is that the number of genes should always be as small as possible. If one of the parameters that are modelled can be obtained from other parameters, this parameter should be removed. In order to use fewer genes, we have to use all the possible information retrieved from the environment, the platform and the elements to recognize. This information allows us to include our knowledge about the problem in the algorithm, and with such information the algorithm will only reach individuals representing real situations (according to the robot and the environment).

4.4 Experiment 3 - MW Study

The third experiment shows how MW affects the vision system. This parameter defines the maximun probability of cloning an individual for initialization from previous populations. MW defines the weight of previous frames for the process. If the value of this parameter increases a higher number of individuals from the initial population will represent solutions reached for previous frames.

The robot captured 20 different images from a static position. While the frames were being captured, the robot's camera orientation was quickly varied. All the frames show the blue goal placed at 250 cm and the orange ball situated at 150 cm. Most of the frames only partially show these elements due to the camera movements (only the orientation changed). We used the standard parameters for the genetic algorithm, and β was modelled as a gene. The variations in MW defined the different configurations. The experiment was repeated 9 times with each different configuration to obtain a final set of 180 frames (20 * 9). 4 different configurations were tested, with MW values of 0, 25, 50 and 75%. Table 5 shows the results obtained for the experiment.

Table 5. Average $DBRED$ and % of frames with a fitness value over certain thresholds

	MW	Fit>0	Fit>0.25	Fit>0.5	Fit>0.75	Fit>0	Fit>0.25	Fit>0.5	Fit>0.75
Ball	0.00	47.37	47.37	36.93	31.75	93.59	93.59	78.84	35.26
	0.25	43.10	41.43	34.26	34.27	91.66	91.02	80.12	44.87
	0.50	41.37	41.26	33.63	33.67	89.74	89.10	75.64	29.49
	0.75	43.48	42.08	32.72	33.49	89.10	87.18	75.00	32.69
Blue Goal	0.00	58.02	49.48	27.15	12.78	100.0	82.68	47.49	12.85
	0.25	53.64	42.63	26.71	19.72	98.32	83.80	56.42	13.97
	0.50	51.22	43.54	21.76	14.16	98.88	87.71	55.87	13.97
	0.75	44.16	37.60	24.45	15.39	98.88	89.94	64.25	12.85
		Average $DBRED$				Percentage of frames			

We can observe how the changes applied to MW do not produce big variations in the difference between the real and estimated distance. Table 5 shows how the percentage of frames that obtained better fitness values increases with greater MW values. For the blue goal, this happens for all the MW values. For the ball, the optimum point for the MW value is 0.25. The performance of the algorithm gets worse if MW is greater than 0.25.

Table 6. Percentage of frames that obtained a $DBRED$ below certain thresholds

		Percentage of frames under						Percentage of frames under			
	MW	100cm	75cm	50cm	30cm		MW	100cm	75cm	50cm	30cm
Ball	0.00	82.69	72.44	62.18	33.33	Blue Goal	0.00	77.09	68.71	55.31	32.96
	0.25	85.90	79.49	71.79	31.41		0.25	81.00	73.74	62.57	34.08
	0.50	85.90	75.00	67.31	35.30		0.50	81.56	75.41	61.45	43.01
	0.75	80.77	73.72	64.10	32.69		0.75	87.15	78.77	72.07	48.60

Finally, table 6 presents the percentage of frames that obtained differences between the real and estimated distance below certain thresholds.

The robustness of the algorithm noticeably improved when the value of MW increased. For the ball, the best results were obtained again for a MW value of 0.25. The behaviour of the algorithm varies for the different objects to be detected when MW increases.

The ball is always captured as a small round orange object and very few frames show the ball partially hidden behind other objects. Because of this, the filtering process gives us useful information for the initialization of the new individuals. The $< x, y >$ position of the ball inside a frame will be close to the centroid $< x, y >$ obtained for the orange pixels after the filtering process. If we excessively increase the number of individuals cloned from previous iterations, the number of individuals initialized with the filtering information will be lower than the number needed for optimal convergence.

In spite of these drawbacks, a small percentage of individuals from previous iterations improves the system's convergence, because the algorithm will have a more diverse initial population. The offspring obtained by crossing individuals

initialized in different ways will be able to obtain better fitness values. The individuals from previous iterations will be very useful if the initial information (obtained via the filtering process) was noisy.

The situation is completely different for goal detection. The shape of the goals in the frame depends on the position and orientation between camera and goal. The size of a goal's projection is bigger than that obtained for the ball, as can be observed in Fig.6. Individuals that are far from the solution can obtain fitness values greater than zero, due to the useful information stored in their genes. The risk of falling into local optimums is much greater for goal detection and the filtering information is less useful. Initializing individuals in different ways will help the algorithm to escape from local optimums. The solution represented by individuals from previous iterations will usually be closer to the global optimum than the one represented by the individuals initialized with the filtering information, especially for minor changes between frames.

5 Conclusions and Future Work

According to the results obtained from the first experiment, our system is a robust alternative to traditional systems for object recognition. It uses the principles of genetic algorithms with a short execution time, which allows the system to be used in the RoboCup environment. The system works properly in the presence of occlusions, without the necessity of a case-based approach.

The β parameter should always be obtained from the other parameters. This parameter can be correctly obtained if the robot's angles are measured without error. The number of genes for the individuals should be as small as possible.

Based on the results obtained in the third experiment, the similarity between consecutive frames can be used to improve the performance of our system.

The system was originally developed for goals and ball recognition, but in view of the results obtained and the available alternatives, the main application for the system should be that of goal detection. This is because goal recognition is much more difficult than ball detection, which can be done by using other techniques.

For future work, we aim to integrate the system developed with a localization method, such as Montecarlo[11] or Kalman Filters[12]. The selected localization method should use the estimated distances and orientations to the goals and the fitness of the best individual, and in order to integrate the visual and the odometry information in an optimal way[13], the fitness of the best individual could be used to represent the goodness of the visual information.

Adding some restrictions to the initialization of the new individuals by taking into account the robot's estimated pose could also be considered.

Acknowledgements

The authors acknowledge the financial support provided by the Spanish "Junta de Comunidades de Castilla-La Mancha (Consejería de Educación y Ciencia)" under PCI08-0048-8577 and PBI-0210-7127 Projects and FEDER funds.

References

1. Rofer, T., Brunn, R., Dahm, I., Hebbel, M., Hoffmann, J., Jungel, M., Laue, T., Lotzsch, M., Nistico, W., Spranger, M.: GermanTeam 2004. Team Report RoboCup (2004)
2. Wasik, Z., Saffiotti, A.: Robust color segmentation for the robocup domain. In: Pattern Recognition, Proc. of the Int. Conf. on Pattern Recognition (ICPR), vol. 2, pp. 651–654 (2002)
3. Jüngel, M., Hoffmann, J., Lötzsch, M.: A real-time auto-adjusting vision system for robotic soccer. In: Polani, D., Browning, B., Bonarini, A., Yoshida, K. (eds.) RoboCup 2003. LNCS (LNAI), vol. 3020, pp. 214–225. Springer, Heidelberg (2004)
4. Coath, G., Musumeci, P.: Adaptive arc fitting for ball detection in robocup. In: APRS Workshop on Digital Image Analysing, pp. 63–68 (2003)
5. Mitchell, M.: An Introduction to Genetic Algorithms (1996)
6. Whitley, L.: Cellular Genetic Algorithms. In: Proceedings of the 5th International Conference on Genetic Algorithms table of contents. Morgan Kaufmann Publishers Inc., San Francisco (1993)
7. Foley, J.D., van Dam, A., Feiner, S.K., Hughes, J.F.: Computer graphics: principles and practice. Addison-Wesley Longman Publishing Co., Inc., Amsterdam (1990)
8. Borenstein, J., Everestt, H., Feng, L.: Where am I? Sensors and Methods for Mobile Robot Positioning (1996)
9. Bach, J., Jungel, M.: Using pattern matching on a flexible, horizon-aligned grid for robotic vision. Concurrency, Specification and Programming-CSP 1(2002), 11–19 (2002)
10. Moscato, P.: Memetic algorithms: a short introduction. Mcgraw-Hill'S Advanced Topics In Computer Science Series, pp. 219–234 (1999)
11. Fox, D., Burgard, W., Thrun, S.: Active markov localization for mobile robots (1998)
12. Negenborn, R.: Robot localization and kalman filters (2003)
13. Martínez-Gómez, J., José, A., Gámez, I.G.V.: An improved markov-based localization approach by using image quality evaluation. In: Proceedings of the 10th International Conference on Control, Automation, Robotics and Vision (ICARCV), pp. 1236–1241 (2008)