ROBOCUP2004 SYMPOSIUM,
Instituto Superior Técnico,
Lisboa, Portugal,
July 4-5, 2004.

Getting the most from your color camera in a color-coded world

Erio Grillo[1], Matteo Matteucci[2], and Domenico G. Sorrenti[3]

[1] undergrad. stud. (Informatica) at Universitá di Milano - Bicocca
[2] Politecnico di Milano, Dept. Elettronica e Informazione,
matteucci@elet.polimi.it
[3] Universitá di Milano - Bicocca, Dept. Informatica, Sist. e Comun.,
sorrenti@disco.unimib.it

Abstract. In this paper we present a proposal for setting camera parameters which we claim to give results better matched to applications in color-coded environments then the camera internal algorithms. Moreover it does not require online human intervention, i.e. is automated, and is faster than a human operator. This work applies to situations where the camera is used to extract information from a color-coded world. The experimental activity presented has been performed in the framework of Robocup mid-size rules, with the hypothesis of temporal constancy of light conditions; this work is the necessary first step toward dealing with slow changes, in the time domain, of light conditions.

1 Introduction

Color cameras are used in many application domains, in robotics especially, where they represent a relatively cheap, but powerful sensor. Whenever the operating environment of the camera, i.e. the working environment of the embodied agent, is such that the color of the objects carries the object semantic, then we say that the agent is immersed in a *color-coded world*. This work focuses on such situations, where the agent processing can color-discriminate the objects in the scene; the camera acquisition parameters should be matched to the color codes, to facilitate such discrimination.

Unfortunately, dealing with a color-coded world does not simplify things as much as one can expect; in fact, many aspects are not fixed and contribute to the difficulty of color-discriminating the world:

1. the actual values of the pixels of the objects can often be really different, even though the colors are appear as expected; in practice what is granted is just the human-usable string which represents the name of the color. A Robocup example: according to the rules *the playground has to be green*; this just means that we are happy when the color attribute of the playground is called *green* by most humans. On the other hand its color can range from the emerald-like playground we had at the European championship in Amsterdam 2000, to the pea green we had at the worlds in Padova 2004.

2. the actual light on the playground can be different in one or more aspects (e.g., intensity, color temperature) from site to site; of course this impacts on the apparent color of the objects
3. the spectral sensitivity curve (quantum efficiency diagram) of the camera changes with the device in use; of course this impacts on the apparent color of the objects, as seen by the color classification algorithms
4. the values of the parameters which affect the functioning of the many circuits inside the camera have also a large impact on the apparent color of the objects
5. this point applies only if the algorithms for the automatic setting of some parameters are active; the actual algorithm changes with the camera; we would not be happy to discover that our processing does not provide a reasonable output any more, if we change the camera model and/or brand

Cameras functioning usually depend on some parameters, which are in charge of controlling the image acquisition process. Examples of such parameters are: the gain of the amplifier which takes in input the output of the light collecting device and gives out the usable output of the camera, the color balancing "knobs", which allow to change the relative weight of the color channels in order to deal with different color temperatures of the light, etc. Even though the values of such parameters are not usually given the appropriate relevance, i.e. many users just leave them to their default, their role is really relevant in the image grabbing process. Slight changes of some parameters turns into very different images. As their values affect the best results that can be attained by the processing which is applied to the grabbed image, we would like image grabbing parameters set so to grab the near-best images, in terms of the noise acting on the color classification. This should be compared to current praxis, consisting in taking the default values and then working hard to discriminate the colors. Notice that the colors have largely been mixed-up during image grabbing.

It is important to recall that some parameters can be automatically set by the internal algorithms of the camera, while the others are usually left to the manual intervention (if any) of the user. The camera internal algorithms, however, cannot match the requisites of a specific application, mainly because of their generality with respect to the observed scene. As an example, a mid-size Robocup robot could require to discriminate between blue and cyan, but this capability is not part of the functionalities of a normal camera. Moreover, the internal algorithms do not perform well when the camera not in very good light conditions, e.g. when the camera has in view both bright and dark areas. To perform well here means to reach a quality of the image which a human operator could reach, by acting on the camera parameters. A last consideration concerns the internal algorithms, which are usually unknown; they are regarded as part of the source of revenues by the camera builder companies: we tried to gain knowledge of their functioning both from the literature and asking the manufacturer, without usable results. Therefore such algorithms are often turned off, at least to allow the user to understand what's actually going on in the camera, even though the user knows that doing so implies working far away from optimality.

It should be noted that the manual determination of near-optimal values is a time-consuming task and has to be performed by a human operator. This is especially true when working in not optimal light conditions, which is unfortunately when internal algorithms use to fail most. Moreover, the intervention of a human operator in setting these parameters introduces another problem, due to his/her inherently subjective perception of colors, which could make subsequent processing to fail. In order to leave out his/her subjectivity the operator should work basing on some quantitative index, therefore paving the way to an automatic use of the same index.

In this paper we propose to formalize the problem of selecting the acquisition parameters as an optimization task (see Section 2) and we solve such an optimization problem using the genetic meta-heuristic (see Section 3). The experimental activity, presented in Section 4, has been performed in the framework of Robocup mid-size rules, with time-constant light conditions. We consider this work as the necessary first step toward adaptation to slow time-changes of the light conditions. The overall aim of this work is to make the setup time short while minimizing the human intervention, in agreement with the short-setup issue in Robocup mid-size.

2 Setting up the problem

The Robocup mid-size league working environment is a so-called color-coded world: it can be described as follows, as ruled by current (2003) rules: the robots move on a flat and rectangular playground built with a green carpet; the two teams of four robots, like in real soccer, defend one goal and attack the other goal. Each of the two goals (one yellow, the other blue) lay on one of the two short sides of the playground. The poles of the goals are white, as the playground lines (side and touch lines, goal area lines, etc). The four corners feature a quarter circle arc white line in the playground and a pole with 3 bands of colors: the corner poles on the blue goal side have colored bands: yellow, blue, and yellow again, from the playground upward. The corners on the other side present the bands, but with the yellow and blue bands exchanged. More details, for the interested reader, can be found in the rule document [1].

The light on the playground is adjusted between 300lux and 900lux. Very slight spatial differences in the light intensity are allowed, e.g. shadows due to the goals, robots, etc. Starting from 2004 slight time changes of the light are allowed as it will be implied also by an imperfect (and cheaper) lightening system, complemented by natural light from outdoor. With such a wide range in light intensity typical issues regard highlights and shadows. With a strong light the red of the ball tends to get yellow in the un-avoidable highlights on the ball, the playground also gets filled with highlights where the green is so diluted to be easily perceived as cyan or even white. With too poor a light the blue gets undistinguishable from black and cyan, etc.

In order to describe the parameters involved in the image acquisition process, we considered those included in the IEEE1394 IIDC (DICAM, Digital Cam-

era) version 1.30 standard [2]. IIDC is an open commercial standard on top of IEEE1394, the well-known Firewire serial bus, an affordable interface for high-speed devices, like cameras. IIDC is a standard for the so-called digital-video, i.e. uncompressed video on Firewire, and, because of the processing time constraints of applications, IIDC is the usual approach in Robotics since no un-compression is required. Many IIDC cameras are available on the market, from webcam-level to professional ones, CCD or CMOS based. When claimed to be compliant with IIDC, such cameras should export settings which are a subset of the settings defined in IIDC specifications. In the view of general usability in Robocup competitions, we carried out the work with low-cost webcam-level Firewire IIDC cameras; notice that this choice just applies to the experiments, i.e. the specific set of parameters on which to conduct the optimization, and does not affect the proposal per sé.

In the following we report a list of parameters in the IIDC standard (IIDC 1394-based Digital Camera Specification v. 1.30), with a very short explanation; some of them can be set to *automatically determined by the camera internal algorithms*. Of course the experimental work has been carried out just on the ones implemented in the actual camera we used, as described in Section 4.

1. EXPOSURE: the combination of the sensitivity of the sensor, the lens aperture and the shutter speed; most webcam-level cameras can actually change just the exposure time
2. IRIS: the lens aperture control; most webcam-level cameras do not have a mechanical iris; instead they use to act on the integration time of the incoming light, i.e. the exposure time
3. GAIN: the electronic amplification of the camera circuit gain control
4. SATURATION: the saturation of the COLOR, i.e. the degree to which a color is undiluted by white light; if a color is 100 percent saturated, it contains no white light; if a color has no saturation, it is a shade of gray
5. WHITE BALANCE - RED channel: the camera can automatically adjust the brightness of the red, green and blue components so that the brightest object in the image appears white, this is done by controlling the relative intensity of the R and B values (in RGB)
6. WHITE BALANCE - BLUE channel: see the point above
7. BRIGHTNESS: the black level of the picture, a pure offset of the intensity level of each pixel coming from A/D converter
8. GAMMA: this parameter defines the function between incoming light level and output picture level, see [3] for details
9. SHARPNESS: the camera can enhance, in a digital sense, the details of edges; this is done by applying a mathematical formula across the image; this is not of interest in the contest of this work
10. HUE: the phase of the color
11. SHUTTER: the opening time of the lens aperture; most webcam-level cameras do not have a mechanical shutter; instead they use to act on the integration time of the incoming light, i.e. the exposure time
12. FOCUS: the lens focus control, most webcam-level cameras just have a mechanical lens focus handle, which is not under software control

Fig. 1. The RGB cube and the 8 color codes relevant for the Robocup application

13. TEMPERATURE: the temperature inside the camera
14. TRIGGER: this is used to control the frame grabbing in applications which require accurate synchronization, e.g. with stroboscopic light
15. ZOOM the lens zoom control, not available on most webcam-level cameras
16. PAN movement: the motion around a vertical axis, not available on most webcam-level cameras
17. TILT movement: same as above, but referred to an horizontal axis, not available on most webcam-level cameras
18. OPTICAL FILTER CONTROL: changes the optical filter of a camera, not available on most webcam-level cameras

We propose a formalization of the task as an optimization problem, where the independent variables are the camera parameters; the dependent variable has to reflect the quality of the grabbed image, and, most important, the relationship between the independent and the dependent variables is not known. Under a pure theoretical point of view a complete model of image formation could be developed. Apart the complexity of such a model, the intrinsic limit of an approach which would try to create a complete model of the image formation is the difficulty and/or the un-accuracies in the determination of the values of the model parameters. As an example consider the many different physical incarnation of the playground, of the robot body, of the goal color, lights, etc. Therefore we consider not applicable a classical operating research approach, where the relationship $dependent = f(independent)$ has to be known.

Our formalization of the problem starts with the observation that the color-codes in Robocup are the vertexes of the so-called RGB cube, see Figure 1; this includes the extremal conditions of minimum light (black) and maximum light (white). Hence the task can be expressed as *how to adjust the camera parameters so that each image area of one color gets RGB values which are very near to the corresponding vertex of the RGB cube.* Here the vertexes are the human-defined color, which, in the following, we will call *color prototypes* for short.

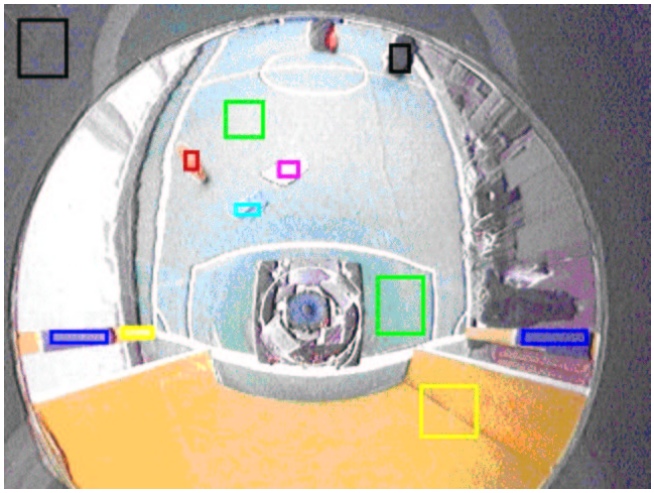


Fig. 2. An example of ground truth for color prototypes selected by the operator.

Suppose we have an image where some areas have been collected and classified by a human operator and is taken as ground-truth, see Figure 2. We would like to push/stretch each color toward its corresponding *prototype* by acting on the acquisition parameters, ending up with pixel values (at least for the selected areas) which are easy-to-discriminate into the set of color prototypes. To perform this color space deformation in an automatic way, we first define an evaluation function, aiming at capturing the distance of the cloud of points of each color from its prototype and, after that, we seek the camera setting which minimize such a function. Of course the camera will not be moved during the whole optimization process.

Let GT^c be the set of ground-truth pixels of color c, i.e. $GT^c = \{p \mid \text{declared by the user to be of color } c\}$. The prototype of color c is denoted by $prot^c$. The evaluation function f^c, i.e. computed with respect just to color c, is:

$$f^c = \frac{1}{|GT^c|} \sum_{\forall p \in GT^c} \|p(r,g,b) - prot^c(r,g,b)\| \tag{1}$$

where by $\|A - B\|$ we denoted a distance between two points A and B, in the RGB color space, and by $|Set|$ the cardinality of Set. Let $p(x)$, $x = r, g, b$ be the r or b or g coordinate of a pixel, then Equation 1, if we choose the classic Euclidean norm, translates into:

$$f^c = \sqrt{\sum_{x=r,g,b} [\overline{p}^c(x) - prot^c(x)]^2} \tag{2}$$

where:

$$\overline{p}^c(x) = \left(\frac{1}{|GT^c|} \cdot \sum_{\forall p \in GT^c} p(x) \right) \quad x = r, g, b. \tag{3}$$

If we consider $\overline{p}^c(x)$, $x = r, g, b$ as the r, g, b coordinates of the barycenter $\overline{P}^c$ of GT^c, Equation 2 can be written shortly as:

$$f^c = \|\overline{P}^c - prot^c\| \tag{4}$$

Put as above, the problem is a minimization problem, i.e. the smaller the evaluation function the better the solution. A relevant point is that this minimization has to be carried out for all the relevant colors at the same time. Fulfilling the constraint for just one color is an easy, but not useful, task.

In the case of mid-size Robocup league we have the 8 distinct color prototypes mentioned before; however the generality of the proposal is in its independency on the number of color-codes, on their specific location in the color space, and, to some extent, on the particular color space; this turns into applicability in other domains.

To account for the multi-objective aim of the work, which is to minimize the evaluation function for all the colors at the same time, we considered as the evaluation function average the average of the evaluation function of the single colors. So, let CC be the set of color codes, which in our example application are:

$$CC = \{white, black, yellow, blue, red, green, cyan, magenta\}$$

the overall evaluation function for a given set of parameters could be:

$$f^{CC}_{ave} = \frac{1}{|CC|} \sum_{\forall c \in CC} f^c, \tag{5}$$

however, this simple function turns out not to be capable to consistently reach good results because its formulation inherently allows a compensation of very good results for some colors with very bad ones for the others. Therefore, we propose to use a different function, a sum of quadratic terms, which aims at weighting more large errors, so to drive the optimization toward a more homogeneous treatment of all the colors. Homogeneous here means forcing at the same time each color cloud towards each prototype.

$$f^{CC}_{SSD} = \sum_{\forall c \in CC} (f^c)^2. \tag{6}$$

3 How to solve the (optimization) problem

The search space of the optimization problem introduced in the previous section is the space of the camera settings; the goal is to minimize an evaluation function

computed on few color characteristics of the image obtained from the camera. However, it is not possible to face this optimization task with classical analytical or numerical optimization techniques due to several reasons:

- the absence of an analytical form for the relationship between the independent variables and the measured values,
- the evaluation function is noisy,
- the evaluation function is not linear nor continuous

Due to the previous reasons and given that we have no clue about the convexity of the problem, we propose to use a randomized algorithm to perform the optimization, in particular we suggest to use a genetic meta-heuristic.

Genetic algorithms [4] have proved to be a powerful search tool when the search space is large and multi-modal, and when it is not possible to write an analytical form for the error function in such a space. In these applications, genetic algorithms excel because they can simultaneously and thoroughly explore

Algorithm 1 Sketch of Goldberg's Simple Genetic Algorithm

Begin Simple Genetic Algorithm
Create a Population P with N Random Individuals
for all $i \in P$ **do**
  Evaluate(i)
end for
repeat
  repeat
    Select i_1 and i_2 according to their Fitness
    with probability p_{cross}
      $i'_1, i'_2 \leftarrow$ **Cross**(i_1, i_2)
    otherwise {with probability $1 - p_{cross}$}
      $i'_1 \leftarrow i_1$ and $i'_2 \leftarrow i_2$
    end with probability
    with probability p_{mut}
      $i''_1 \leftarrow$ **Mut**(i'_1)
    otherwise {with probability $1 - p_{mut}$}
      $i''_1 \leftarrow i'_1$
    end with probability
    with probability p_{mut}
      $i''_2 \leftarrow$ **Mut**(i'_2)
    otherwise {with probability $1 - p_{mut}$}
      $i''_2 \leftarrow i'_2$
    end with probability
    Add Individuals i''_1 and i''_2 to New Population
  until (Created a new Population P')
  for all $i \in P'$ **do**
    Evaluate(i)
  end for
until (Stopping criterium met)
End Simple Genetic Algorithm

many different parts of a large space seeking a suitable solution. At first, completely random solutions are tried and evaluated according to a fitness function, and then the best ones are combined using specific operators. This gives the ability to adequately explore possible solutions while, at the same time, preserving from each solution the parts which work properly.

In Algorithm 1 we show the general scheme for the genetic algorithm we used. The initial population is randomly initialized choosing, for each individual, a random vector of camera parameters; each parameter range is sampled extracting a value out of a uniform distribution. After the definition of the first population of random solutions, each individual is evaluated by computing its fitness and ranked according to it. We evolve the population by selecting the individuals according to their fitness and stochastically applying to them the genetic operators *crossover* and *mutation*. Once a new offspring has been generated, the fitness of the new individuals is evaluated. This process continues until a stopping criterium is met, e.g. a maximum number of generations.

The basic genetic algorithm used in our implementation is the Simple Genetic Algorithm Goldberg describes in his book [4]. At each generation it creates an entirely new population of individuals by selecting from the previous population, and then mating them to produce the offspring for the new population. In all our experiments we use *elitism*, meaning that the best individual from each generation is carried over to the next generation. By using elitism we ensure the algorithm to be a monotonic any-time algorithm, meaning that the optimization process can be stopped at any point while getting always a reasonably good solution.

Solution are coded in genotypes by means of integer valued strings; genetic optimization is known to loose efficiency when individuals are coded by too long genotypes, hence we did not use binary coding. Each position, i.e. allele, represents a camera parameter. The alleles assume values in a limited range according to the camera specifications. We use uniform random initialization and classical uniform crossover; for the mutation we use a Gaussian probability mutation to select values in the neighbors of the actual value instead of a completely random

Algorithm 2 Evaluation of the fitness of an individual

Begin Evaluate (*Individual i)*
Set the camera parameters according to the genotype
Grab a couple of images to make the new parameter set effective
$fitness = 0$
for all $c \in CC$ **do**
  Grab an image
  Compute $\overline{P}^c$, the barycenter in the color space of pixels of color c
  Compute the barycenter distance $f^c = \|\overline{P}^c - prot^c\|$ from the c color prototype
  Set $fitness = fitness + (f^c)^2$
end for
return $fitness$
End Evaluate

ones. The fitness function described in Algorithm 2 is used to evaluate each individual according to the problem definition of Section 2. Experimental practice evidenced that changing the parameters takes effect after some time, so a couple of fake images are grabbed before each individual evaluation.

4 Experimental activity

We performed the experiments with one of the cameras in use on our robots; it is a quite widespread camera in the Robocup community. It is, as mentioned before, a webcam-level camera, with a single color CCD from Sony, progressive scan, capable to provide, via the Firewire interface, 640x480 pixel, 24bpp at 15 frames/s. that this is an upperbound, 30 frames/s are not possible at the maximum resolution). The builder is OrangeMicro and the model IBot. After some tweaking we now have the camera reliably running onboard our goalkeeper.

The camera exports a limited subset of the 18 parameters in the IIDC specification, the others being not software implemented or being related to physical feature which are not present on the camera, e.g. the mechanical iris. The set of parameters for our camera hence reduces to 8:

1. EXPOSURE, nominal range [0, 498]
2. IRIS, nominal range [0, ..., 4]
3. GAIN, nominal range [0, ..., 255]
4. SATURATION, nominal range [0, ..., 255]
5. WHITE BALANCE - RED channel, nominal range [0, ..., 255]
6. WHITE BALANCE - BLUE channel, nominal range [0, ..., 255]
7. BRIGHTNESS, nominal range is [0, ..., 511]
8. GAMMA, nominal range [0, 1]

It is possible to set EXPOSURE, WHITE BALANCE (both RED and BLUE), and BRIGHTNESS to take a value decided by the camera internal algorithm. Considering the nominal range of discrete values, the search space has a finite, but quite large number of points. The time required by the fitness evaluation is bounded by the image grabbing time: the camera gives out 15 frames/s, which means a period of about 67ms. Considering that more than one image transfer has to be awaited, in order to have the new settings operating, then the time required by a brute-force approach would be about:

$$\begin{aligned} BruteForceTime &= 499 \cdot 5 \cdot 256 \cdot 256 \cdot 256 \cdot 256 \cdot 512 \cdot 2 \cdot 180\ ms \\ &= 65313835\ years. \end{aligned}$$

During the experimental activity we switched on all the lights, and obtained what we call an *intermediate* level of light intensity, i.e. in the range [260, 370] lux, which is quite homogeneous, in our opinion, although the light intensity is not as high as it should be in Robocup mid-size. It is worthwhile to notice that the conditions of this experiment are those where the *current praxis* is more likely to give good results, because of the homogeneity of the light intensity.

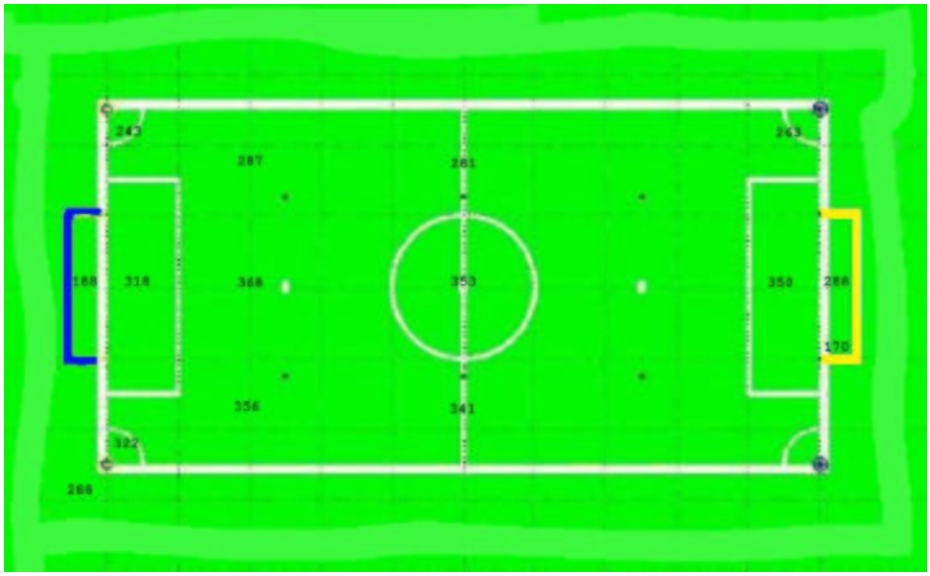


Fig. 3. Distribution of light in intermediate light conditions

We call *current praxis* the situation where the *manual-only* parameters have been set to their default values and the others have been set so that the camera decides on them. Figure 4 reports the evolution of the fitness for the best individual in the population compared to the fitness value of the image acquired with the *current praxis* parameters. As it can be seen in the plot, the steady-state is reached much before of the end of the process; the fitness value is lower than the one computed on an image grabbed with the *current praxis*, just after a very few generations. This experiment uses $p_{cross} = 0.7$, $p_{mut} = 0.2$, and a population of 20 individuals. These values have not been optimized and we feel that better results could be obtained by a more accurate selection of them. In our preliminary experiments, aiming at demonstrating the convergence of the proposal, the evolutionary process is terminated after a large number of generations; in the move to on-line applications we will define different stopping criteria such as maximum execution time, or permanence in a steady-state for a given number of generations.

Table 1 compares the *current praxis* situation and the best solution found by our approach. The values in the parentheses are the values given out by the camera during *auto* functioning. We observed that such values, which coincide with the default values of each parameter, do not change with the light conditions, whilst they should. Therefore we consider that the onboard software replies with just the default values; this supports our claim about the difficulties to discover the internal functioning of the camera. Just to give an idea of the quality of the

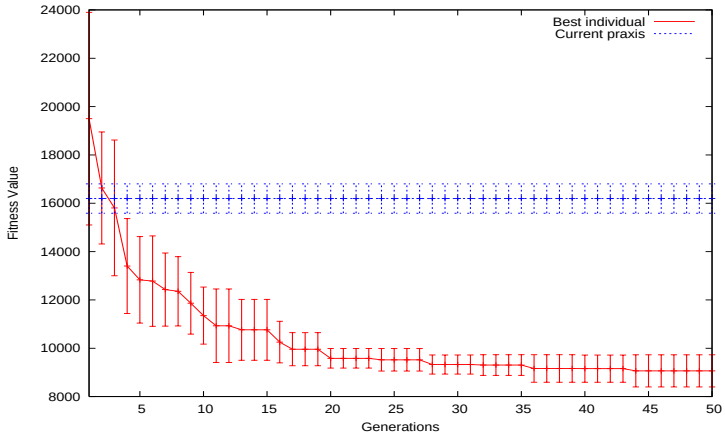


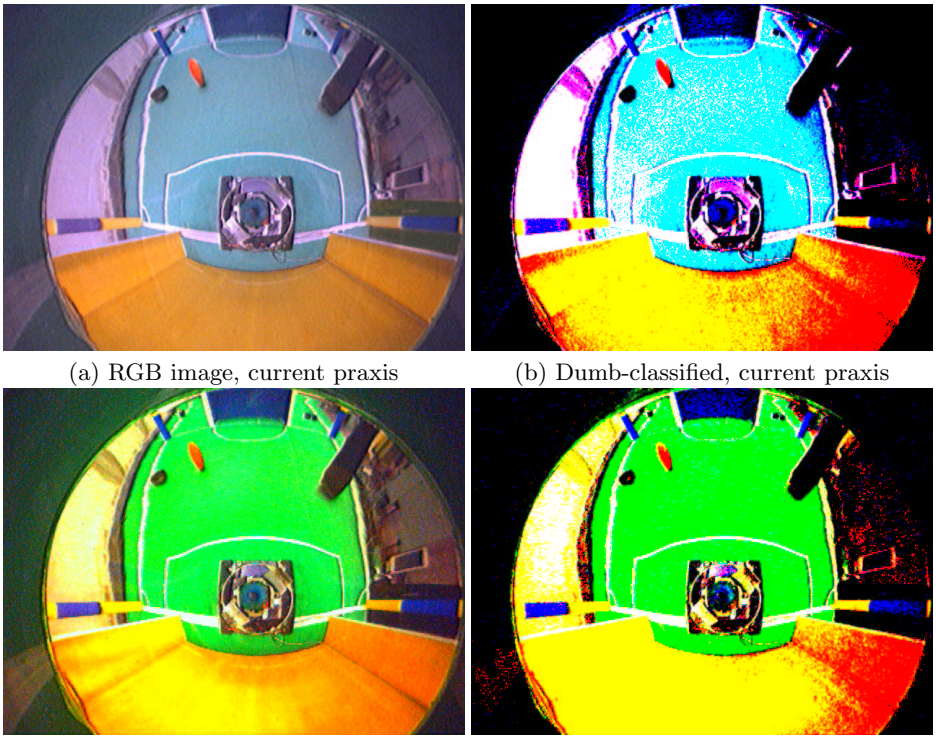
Fig. 4. An example of evolution of the fitness functions during the optimization

results, notice the last row, where the fitness has been computed for the *current praxis* case too.

In Figure 5 we present the image grabbed with the *current praxis* and with our solution. Since the final aim of the work is to ease the color-classification processing, it is worthwhile to evaluate how much our work eases this processing. We therefore introduced a very simple color-classification scheme, which we then applied to both solutions. This scheme classifies each pixel as the color of the closest prototype. As it can be easily seen the performance of our approach outperforms the current praxis: e.g. our approach turns the playground from mostly cyan to green, removing the large amount of white points; also the yellow goal is less reddish with our approach. This effects could more easily be observed in the rightmost column with the classified images.

	current praxis	Evolved Parameters
EXPOSURE	Auto (498)	485
IRIS	4	1
GAIN	87	175
SATURATION	105	227
WHITE BALANCE [RED]	Auto (84)	128
WHITE BALANCE [BLUE]	Auto (91)	102
BRIGHTNESS	Auto (314)	201
GAMMA	1	0
Overall Fitness	**16193**	**4636**

Table 1. Parameters for the *current praxis* and our best solution, with fitness



(a) RGB image, current praxis (b) Dumb-classified, current praxis

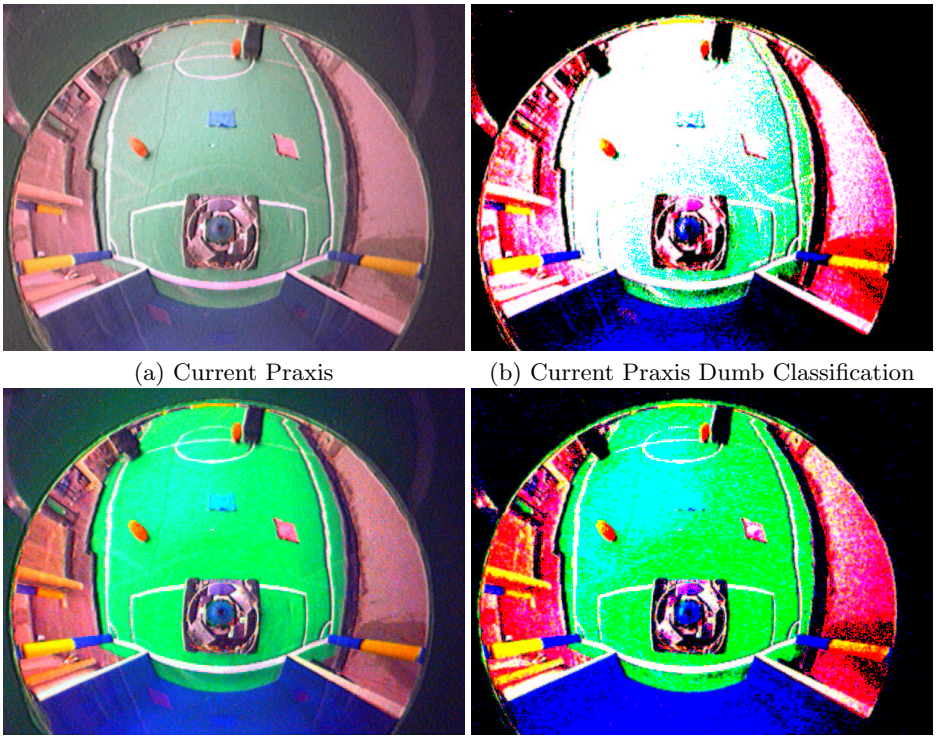
(c) RGB image, our approach (d) Dumb-classified, our approach

Fig. 5. Images for *current praxis* (a, b), and our solution (c, d)

In order to support the effectiveness of the approach we present hereafter a second experiment. In this experiment we reduced also the search space for some parameters, according to subjective considerations about the effect on the images

	current praxis	Evolved Parameters
EXPOSURE	Auto (498)	490
IRIS	4	4
GAIN	87	217
SATURATION	105	242
WHITE BALANCE [RED]	Auto (84)	187
WHITE BALANCE [BLUE]	Auto (91)	126
BRIGHTNESS	Auto (314)	115
GAMMA	1	1
Overall Fitness	**21655**	**8143**

Table 2. Parameters for the *current praxis* and our solution, with fitness.



(a) Current Praxis (b) Current Praxis Dumb Classification

(c) Optimized Image (d) Optimized Image Dumb Classification

Fig. 6. Image grabbed with *current praxis* (a)(b), and our solution (c)(d).

obtained by changing (independently) each parameter, e.g., to avoid removing colors form the image. In particular we reduced the range to $[200, 498]$ for EXPOSURE, to $[80, 255]$ for GAIN, to $[100, 255]$ for SATURATION, to $[80, 255]$ for WHITE BALANCE (both RED and BLUE), to $[100, 511]$ for BRIGHTNESS, and fixed GAMMA to 1 and IRIS to 4.

Table 2 compares the *current praxis* situation and the best solution found by our approach. Again our approach gets a lower fitness with respect to current praxis. However, the fitness is twice the fitness of the previous experiment (light conditions were very similar, and the observer is the blue, darker, goal). According to us this is due to having reduced improperly the search space. This again supports the approach of blind optimization we took, avoiding any subjective interpretation of the parameters.

In Figure 6 we present the image grabbed with the *current praxis* and with our solution for this second experiment. As it can be easily seen, even with a sub-optimal solution, the performance of our approach is better than using the current praxis.

5 Conclusions

In this work we propose an approach to set the camera parameters affecting the image grabbing process, so to ease at the maximum extent the color classification task. The problem is cast to a minimization problem which we propose to optimize with a genetic meta-heuristic. An experimental activity is presented which validates the proposal.

This work is just a first step toward an automatic tool for tracking light changes. At the present stage it heavily depends on the training set to include patches from different light conditions, in order to be able to find a globally good parameter setting. We are now working on automatic acquisition of the ground-truth, which will allow to apply the approach to sequences of images, taken in different parts of the field.

6 Acknowledgements

This work has been partially supported by the project P.R.I.N. 2003 "Tecnologie software e modelli di conoscenza per progetto, sviluppo e sperimentazione di sistemi robotici multi-agente in ambienti reali", funded by M.I.U.R.
We thank the referees for their useful observations, which allowed us to improve the quality of the paper.

References

1. Technical Committee of the Midsize League for 2003: Rules for Robocup Midsize Competitions, year 2003. available at http://wwwradig.in.tum.de/MSL-2003/ (2003)
2. 1394 Trade Association: IIDC 1394-based Digital Camera Specification Version 1.30. 1394 Trade Association, Regency Plaza Suite 350, 2350 Mission College Blvd., Santa Clara, California 95054, USA (2000) technical report id.: TA Document 1999023.
3. Poynton, C.A.: A technical introduction to digital video. Wiley (1996)
4. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley Publishing Company, Inc., Reading, MA (1989)