REAL-TIME INTER- AND INTRA- CAMERA COLOR MODELING AND CALIBRATION FOR RESOURCE CONSTRAINED ROBOTIC PLATFORMS

Walter Nisticò and Matthias Hebbel
Robotics Research Institute (IRF), Universität Dortmund, Otto-Hahn-Str. 8, Dortmund, Germany

Keywords: Camera modeling, color correction, evolutionary algorithms, real time.

Abstract: This paper presents an approach to correct chromatic distortion within an image (vignetting) and to compensate for color response differences among similar cameras which equip a team of robots, based on Evolutionary Algorithms. Our black-box approach does not make assumptions concerning the physical/geometrical roots of the distortion, and the efficient implementation is suitable for real time applications on resource constrained platforms.

1 INTRODUCTION

Robots which base their perception exclusively on vision, without the help of active sensors such as laser scanners or sonars, have to deal with severe additional constraints compared to generic computer vision applications. The robots have to interact with a dynamic (and sometimes even competitive or hostile) environment and must be able to take decisions and react to unforeseen situations in a fraction of a second, thus the perception process has to run in real-time with precise time boundaries. In particular, autonomous robots are typically constrained even in terms of computational resources, due to limitations in power supply, size, and cost. A very popular approach especially for indoor environments is color-based image segmentation, and even though dynamic approaches have been demonstrated (for example, (Iocchi, 2007) (Schulz and Fox, 2004)), static color classification (Bruce et al., 2000) is still the most widely used solution, due to its efficiency and simplicity. In a group of homogeneous robots, which make use of color information in their vision system, it is important that all cameras produce similar images when observing the same scene, to avoid to have to individually calibrate the vision system of each robot, a procedure which is both time consuming and error prone. Unfortunately, even when the cameras are from the same model and produced from the same manufacturer, such assumption does not always hold (Röfer, 2004).

1.1 The Platform

This work has been developed on the popular Sony Aibo ERS-7 robot (Sony Corporation, 2004), which is the only complete standard platform widely adopted for robotic applications to date. The robot is equipped with a 576MHz 64bit RISC CPU, 64MB of main memory, and a low-power CMOS camera sensor with a maximum resolution of 416×320 pixel. Images are affected by a ring-shaped dark blue cast on the corners, and different specimen of the same model tend to produce slightly different color responses to the same objects and scene. Our experiments have been conducted using the YUV color space which is natively provided by most common cameras, but the approach can be applied unaltered to the RGB color space. The Aibo production has been recently discontinued by Sony, but several new commercially available robotic kits are being introduced on the market, with similar characteristics in terms of size and power, often equipped with embedded RISC CPUs or PDAs and low quality compact flash cameras.

1.2 Related Work

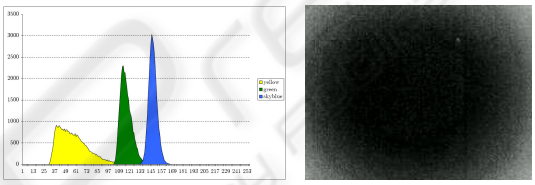
Whenever object recognition is mostly based on color classification, the dark / colored cast on the corners of

Nisticò W. and Hebbel M. (2007).
REAL-TIME INTER- AND INTRA- CAMERA COLOR MODELING AND CALIBRATION FOR RESOURCE CONSTRAINED ROBOTIC PLATFORMS.
In *Proceedings of the Fourth International Conference on Informatics in Control, Automation and Robotics*, pages 378-383
DOI: 10.5220/0001631403780383
Copyright © SciTePress

the images captured by the on board camera is a serious hindrance. Vignetting is a radial drop of image brightness caused by partial obstruction of light from the object space to image space, and is usually dependent on the lens aperture size (Nanda and Cutler, 2001) (Kang and Weiss, 2000). These approaches, as well as (Manders et al., 2004), treat the problem in terms of its physical origins due to geometrical defects in the optics, and are mostly focused on radiometric calibration, i.e. ensuring that the camera response to illumination after the calibration conforms to the principles of homogeneity and superposition. However, none of the proposed methods deals with chromatic distortion, as is the case of our reference platform, and other inexpensive low power CMOS sensors. Recently, a few papers have attempted to tackle such problem. These solutions share a similar approach to minimize the computational costs by using lookup tables to perform the correction in real time, while the expensive calibration of the correction tables is performed off-line. In (Xu, 2004) the author uses a model based on a parabolic lens geometry, solved through the use of an electric field approach. No quantitative analysis of the results is provided, but this technique has been successfully used in practice by one team of autonomous robots in the RoboCup Four-Legged League.[1] Another successful technique used in RoboCup has been presented in (Nisticò and Röfer, 2006), based on a purely black-box approach where a polynomial correction function is estimated from sample images using least square optimization techniques. Since this approach does not rely on assumptions concerning the physics of the optical system, we feel that it can be more effective in dealing with digital distortions such as saturation effects. Again no quantitative analysis has been presented, and both papers do not address the problem of inter-robot camera calibration, which has been treated in (Lam, 2004) with a simple linear transformation of the color components considered independently.

2 COLOR MODEL

The first step to understand the characteristics of this chromatic distortion, was to capture images of special cards that we printed with uniform colors, illuminating them with a light as uniform as possible, trying to avoid shadows and highlights.[2] Then we calculated the histograms of the three image spectra, with a number of bins equal to the number of possible values that each spectrum can assume, i.e. 256. Under these conditions, the histograms of such uniform images should be uni-modal and exhibit a very narrow distribution around the mode (in the ideal case, such distribution should have zero variance, i.e. all the pixels have exactly the same color) due only to random noise. Instead, it could be observed that the variance of the distribution is a function of the color itself; in case of the U channel, it appears very narrow for cold / bluish color cards, and very wide for warm / yellowish cards (Figure 1(a)). Consequently, we model the chromatic distortion d_i for a given spectrum i of a given color I as a function of I_i itself, which here we will call *brightness component* $\lambda_i(I_i)$.



(a) (b)

Figure 1: a) Histograms of the U color band for uniformly colored images: yellow, green and skyblue. Notice how the dispersion (due to the vignetting) increases inverse proportionally to the position of the mode. b) Brightness distribution of the U color band for a uniformly yellow colored image.

The distribution itself is not centered around the mode, but tends to concentrate mostly on one side of it. The reason for this becomes apparent by observing the spatial distribution of the error (cf. Figure 1(b)); the phenomenon itself is nothing but a ring shaped blue/dark cast, whose intensity increases proportionally to the distance from the center of the distortion (u_d, v_d), which lies approximately around the optical center of the image, the principal point. So, let $r = \sqrt{(x-u_d)^2 + (y-v_d)^2}$, then we define the *radial component* as $\rho_i(r(x,y))$. Putting together brightness and radial components, we obtain our distortion model:

$$d_i(I(x,y)) \propto \rho_i\left(r(x,y)\right) \cdot \lambda_i\left(I_i(x,y)\right) \quad (1)$$

Now, due to the difficulty to analytically derive $\rho_i, \lambda_i, \forall i \in \{Y,U,V\}$ about which little is known, we decided to use a black-box optimization approach. Both sets of functions are non-linear, and we chose to approximate them with polynomial functions

[1] RoboCup is an international joint project to promote AI, robotics, and related fields. *http://www.robocup.org/*

[2] However, this is not so critical, and the use of a professional diffuse illuminator is not necessary, as our approach can deal well with noise and disturbances (see Section 2.1).

(McLaurin series expansion).

$$\begin{aligned}\rho_i(r) &= \sum_{j=0}^{n} \varrho_{i,j} \cdot r^j \\ \lambda_i(I_i) &= \sum_{j=0}^{m} l_{i,j} \cdot I_i^j\end{aligned} \tag{2}$$

The unknown polynomial coefficients $\varrho_{i,j}$ and $l_{i,j}$ can be estimated, from a set of samples, using naturally inspired machine learning techniques. So, the final vignetting correction function is as follows:

$$I'_i(x,y) = I_i(x,y) - \rho_i\left(r(x,y)\right) \cdot \lambda_i\left(I_i(x,y)\right) \tag{3}$$

where $I'_i(x,y)$ is the corrected value of the spectrum i of the given pixel.

2.1 Reference Color Estimation

To be able to model our error functions, we must first estimate how the pictures should look like, if they were not affected by any chromatic distortion. Since most of the image area exhibits little to no distortion, we define as *reference color* of a single image the most frequent color appearing in its spectra. So we calculate the histograms of the 3 color spectra (number of bins = number of color levels = 256) and we find the modes $\bar{r}_i^Y, \bar{r}_i^U, \bar{r}_i^V$, which represent the reference color for image i. However, a single image can be affected by other sources of noise, such as a temporary change of light intensity, strobing (aliasing between the camera shutter speed and the light source power frequency, which affects fluorescent lights) or shadows: consequently, the histograms' modes might have temporary fluctuations, or exhibit multiple nearby modes. To make our system more robust toward this kind of noise, we collect multiple pictures of each colored cards in a log file, and we partition the images therein contained into image classes[3], where a class represents a certain color card (e.g. yellow, orange, green, blue) at a certain light intensity and camera settings. For each image class j we want to have a single reference color $\bar{R}_j^Y, \bar{R}_j^U, \bar{R}_j^V$: of course this could be obtained by averaging the references from all the images which belong to the class, but this would still be affected by outliers. Instead, we track the current reference of a given class using a simple first order linear Kalman filter (Welch and Bishop, 2003):

- For each image i in the log, a reference value is estimated for the 3 spectra $\bar{r}_i^Y, \bar{r}_i^U, \bar{r}_i^V$, as the modal value of the corresponding histogram
- Constant value process model: the predicted reference color $\hat{R}_j^Y, \hat{R}_j^U, \hat{R}_j^V$ for class j at the following step (image i) remains the same as the one after the measurement update of image $i-1$
- We use the output of the Kalman filter to perform the partitioning into color classes on the fly; a new class is generated when:

$$\exists c \in \{Y,U,V\} : \left|\hat{R}_j^c - \bar{r}_i^c\right| > \vartheta \tag{4}$$

  where ϑ is a confidence threshold; so if at least one of the spectra in the reference of the current image differs too much from the expected reference for the current class, then we need to create a new color class. A new class reference is initialized to the value extracted from the current image.
- Measurement model: we update the running class reference $\hat{R}_j^Y, \hat{R}_j^U, \hat{R}_j^V$ using the reference extracted from the current image

2.2 Inter-camera Model

In the general case, it is possible that to completely correct the difference in color response between two cameras it is necessary to rotate the color space of one camera to align with the other. This however, is not feasible for a real time implementation for robotic applications. To capture the dependencies between the 3 color components and the distance from the image center, we would need a $256^3 \cdot \rho_{max}$ look-up table, which would have a size of over 2GB even for our low resolution camera. Otherwise, we could perform the rotation with the multiplication of a 3×3 matrix by our color vector; since such costly operation would have to be performed for every pixel, it would slow down the image processing too much.

In (Lam, 2004) the author suggests a simple linear transformation of the 3 color components of the camera to be calibrated, treating them independently from each other; such an approach is used in image processing programs to perform adjustments in the color temperature of a picture. Since we are going to use an evolutionary approach to the optimization process, we have decided to give more freedom to the inter-camera transformation, by using higher order polynomials instead:

$$I''_i(x,y) = A_i\left(I'_i(x,y)\right) = \sum_{j=0}^{4} a_{i,j} \cdot \left(I'_i(x,y)\right)^j \tag{5}$$

$I'_i(x,y)$ is the i-th color component of pixel x,y of the camera that we want to calibrate, $a_{i,j}$ are the transformation coefficients which have to be learned, and $I''_i(x,y)$ is the resulting value, for a certain color

[3] With *image class* or *color class* here we will refer to a color card captured under certain lighting settings, i.e. the same color card can be represented by different color classes, like blue-dark and blue-light.

spectrum $i \in \{Y, U, V\}$, which should match the color as it would appear if taken by the camera of the reference ("alpha") robot. Further, we must obtain I'_i from I_i by applying the vignetting correction as described. All robots in the team have to be calibrated to match the color representation of the "alpha" robot, hence each robot will have its own set of coefficients $a_{i,j}$.

When different cameras exhibit similar vignetting characteristics, the vignetting correction polynomials can be calculated only once for all cameras, then the inter-camera calibration can be performed independently and using much less sample images. In fact, in our experiments we have seen that learning both polynomial sets at the same time can easily lead to over-fitting problems, with the inter-camera calibration polynomials which also end up contributing to compensate the vignetting distortion (for example by clipping the high and low components of the color spectra), but this results in a poor generalization to the colors which are out of the training set.

2.3 Realtime Execution

After optimizing off-line the parameters $\varrho_{i,j}, l_{i,j}, a_{i,j}$ with the techniques presented in Section 3, we are able to calculate the corrected pixel color values (Y'', U'', V''), given the uncorrected ones (Y, U, V) and the pixels' position in the image (x, y). Performing the polynomial calculations for all the pixels in an image is an extremely time-consuming operation, but it can be efficiently implemented with Look-Up Tables:

- $radialLUT[x, y] = \sqrt{(x - u_d)^2 + (y - v_d)^2}$ stores the pre-computed values of the distance of all the pixels in an image from the center of distortion (u_d, v_d);
- $colorLUT[r, l, i] = A_i\left(l - \rho_i\left(r\right) \cdot \lambda_i\left(l\right)\right)$ where $r = radialLUT[x, y]$, $l = I_i(x, y) \in [0 \dots 255]$ and $i \in \{Y, U, V\}$ is the color spectrum. We fill the table for all the possible values of r, l, c, the size of the table is $256 \cdot r_{max} \cdot 3$ elements, so it occupies only $\approx$ 200KBytes in our case

The look-up tables are filled when the robot is booted; afterward it is possible to correct the color of a pixel in real-time by performing just 2 look-up operations.

3 PARAMETER OPTIMIZATION

The goal is to find an "optimal" parameter set for the described color model given a set of calibration images as previously described. Thus, we consider as optimal the parameter set which minimizes the "function value" (from now on referred to as *fitness*), defined as the sum of squared differences of the pixels in an image from the reference value calculated for the color class in which the image belongs. To calculate the fitness of a certain parameter set, given a log file of images of colored cards, we proceed as follows:

- For each image $I_{i,k}$ in the log file (i is the color band, k the frame number), given its reference value previously estimated $R_{i,k}$, the current fitness F_i^k is calculated as:

$$F_i^k = \sum_{(x,y)} \left(I''_{i,k}(x, y) - R_{i,k}\right)^2 \qquad (6)$$

- The total fitness F_i is calculated as the sum of the F_i^k where each k is an image which belong to a different color class; this to ensure that the final parameters will perform well across a wide spectrum of colors and lighting situations, which otherwise might only fit a very specific situation.

The optimization process is performed independently for each color band $i \in \{Y, U, V\}$. To optimize the vignetting correction, we use for $A()$ (from Equation 5) the identity function, and the references $R_{i,k}$ which are extracted from the same log (and same robot) that we are using for the optimization process. In case of inter-robot calibration instead, only $A_i()$ will be optimized, $\rho_i(), \lambda_i()$ will be fixed to the best functions found to correct the vignetting effect, and the references $R_{i,k}$ are extracted from a log file generated from *another* ("alpha") robot, which is used as reference for the calibration. If we want to optimize the center of distortion (u_d, v_d), we need a set of $\rho_i(), \lambda_i()$ calculated in a previous evolution run[4]: as fitness for this process we use the sum of the fitnesses of all 3 color channels, under the assumption that there is only one center of distortion for all color bands.

3.1 Simulated Annealing (SA)

In Simulated Annealing (Kirkpatrick et al., 1983) in each step the current solution is replaced by a random "nearby" solution, chosen with a probability that depends on the corresponding function value ("Energy") and on the annealing temperature T. The current solution can easily move "uphill" when T is high (thus jumping out of *local minima*), but goes almost exclusively "downhill" as T approaches zero. In our work we have implemented SA as follows (Nisticò and Röfer, 2006).

[4]Otherwise, changing u_d, v_d would have no effect on the fitness

In each step, the coefficients $\varrho_{i,j}, l_{i,j}$ (vignetting reduction) or $a_{i,j}$ (inter-robot calibration) or (u_d, v_d) (center of distortion) are "mutated" by the addition of zero mean gaussian noise, the variance of which is dependent on the order of the coefficients, such that high order coefficients have increasingly smaller variances (decreasing order of magnitude) than low order ones, following the idea that small changes in the high order coefficients produce big changes in the overall function. The mutated coefficients are used to correct the image, as in Equation 5.

For each image $I_{i,k}$ in the log file (i is the color spectrum, k the frame number), given its reference value previously estimated $R_{i,k}$, the current "energy" E for the annealing process is calculated as in Equation 6. The "temperature" T of the annealing is lowered using a linear law, in a number of steps which is given as a parameter to the algorithm to control the amount of time spent in the optimization process. The starting temperature is normalized relative to the initial energy, so that repeating the annealing process on already optimized parameters has still the possibility to perform "uphill moves" and find other optimum regions. The process ends when the temperature reaches zero; the best parameters found (lowest energy) are retained as result of the optimization process.

This approach has proved to work well with our application, however it has 2 shortcomings:

- The search space is very irregular and has multiple local and even *global* optima. One reason for the latter is that the function to be optimized is the product of different terms ($\rho_i() \cdot \lambda_i()$), so that exactly the same result can be achieved by multiplying one term by a factor and the other by its reciprocal, or inverting the sign for both terms, etc.
- The variances used to mutate the coefficients give a strong bias to the final results, as it is not possible to fully explore such a wide search space in a reasonable time; the depicted approach lacks the ability to find "good" mutation parameters, apart from the simple heuristic of decreasing the variance order at the increase of the coefficient order

Both issues can be dealt with efficiently by Evolution Strategies.

3.2 Evolution Strategies (ES)

Evolution Strategies (Schwefel, 1995) use a parent population of $\mu \geq 1$ individuals which generate $\lambda \geq 1$ offsprings by *recombination* and *mutation*. ES with *self-adaption* additionally improve the control of the mutation strength: each parameter which has to be optimized (object parameter) is associated with its own mutation strength (strategy parameter). These strategy parameters are also included in the encoding of each individual and are selected and inherited together with the individual's assignments for the object parameters.

An offspring is created by recombination of the parents. We use two parents to generate an offspring, then such offsprings are subject to mutation.

The selection operator selects the parents for the next generation based on their fitness. In case of the (μ, λ)-strategy only the best μ individuals out of the offsprings are chosen to be the parents of the next generation. Our implementation works similarly as the annealing process, with the following exceptions:

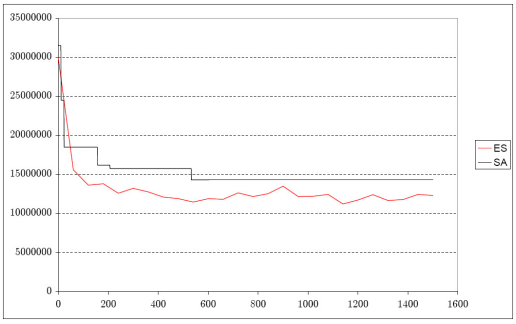
- (μ, λ) strategy with self-adaption, the strategy parameters are initialized with the same sigmas used in the annealing process;
- We terminate the evolution process when n generations have passed without any fitness improvement

Having $\mu > 1$ means that several different "optimal" areas of the search space can be searched in parallel, while the self-adaptation should make the result less dependent from the initial mutation variances.

4 EXPERIMENTS AND RESULTS

At first we compared the two optimization techniques to see what is most suitable for our application. We ran the optimization on a log file containing 9 image classes, the evolution strategy used 8 parents and 60 offsprings, stopping the evolution after 10 generations without improvements. For simulated annealing, we set the number of steps to match the number of fitness evaluations after which ES aborted, ≈ 5000; the total optimization time, for both algorithms, is around 5 minutes, on a 1.7GHz Pentium-M CPU. As it can be seen in Figure 2, both algorithms significantly reduce the image error, but ES found better optima than SA; the results of these techniques are affected by random factors, however except for extremely low evolution times ($< 1min$), ES outperforms SA consistently.

To test the performance of our approach for inter-robot calibration, we created 2 log files containing images representing 6 different color cards taken from 2 robots (one of which we use as reference and call "alpha robot") which show a very different color response, especially for the yellow and pink cards. Table 1 shows that the optimization succeeded in reducing the error standard deviation often by a factor of 3 or more, as well as shifting the modes of the im-



(a) The fitness curves for the correction of the U-channel.

	Y	U	V
init	49677830	31517024	53538715
SA	23608403	14298870	16127666
ES	20240155	10819027	15581825

(b) The initial and achieved best fitnesses.

Figure 2: Results of the vignetting reduction.

Table 1: Inter-robot calibration. $\Delta\mu_{start}, \Delta\mu_{end}$ represent the difference of the mode of the robot from the reference ("alpha") before and after the calibration. $\sigma_{start}, \sigma_{end}$ are the standard deviation from the reference mode.

Color Card	$\Delta\mu_{start}$ Y/U/V	$\Delta\mu_{end}$ Y/U/V	σ_{start} Y/U/V	σ_{end} Y/U/V
green	1/0/3	1/0/4	6.2/3.8/7.4	3.2/3.0/3.9
orange	18/12/5	7/2/3	13.3/17.8/9.3	4.6/4.8/4.2
cyan	13/2/8	5/2/4	11.3/3.9/4.8	4.4/3.8/4.0
red	4/9/5	2/1/1	7.6/15.2/7.5	3.1/5.2/3.6
yellow	6/7/30	5/4/5	21.5/8.3/24.8	6.0/3.9/5.5
pink	17/6/5	3/1/2	24.6/12.6/8.9	5.3/4.3/5.6

ages much closer to the reference values provided by the alpha robot. The total optimization time, split in 2 runs (one for the vignetting, the other for the inter-robot calibration) was approximately 6 minutes.

5 CONCLUSIONS

We have presented a technique to correct chromatic distortion within an image and to compensate for color response differences among similar cameras which equip a team of robots. Since our black-box approach does not use assumptions concerning the physical/geometrical roots of the distortion, this approach can be easily applied to a wide range of camera sensors and can partially deal with some digital sources of distortion such as clipping / saturation. Its efficient implementation has a negligible run-time cost, requiring just two look-up operations per pixel, so it is suitable for real time applications on resource constrained platforms.

REFERENCES

Bruce, J., Balch, T., and Veloso, M. (2000). Fast and inexpensive color image segmentation for interactive robots. In *Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '00)*, volume 3, pages 2061–2066.

Iocchi, L. (2007). Robust color segmentation through adaptive color distribution transformation. In *RoboCup 2006: Robot Soccer World Cup X*, Lecture Notes in Artificial Intelligence. Springer.

Kang, S. B. and Weiss, R. S. (2000). Can we calibrate a camera using an image of a flat, textureless lambertian surface? In *ECCV '00: Proceedings of the 6th European Conference on Computer Vision-Part II*, pages 640–653. Springer-Verlag.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science, Number 4598, 13 May 1983*, 220, 4598:671–680.

Lam, D. C. K. (2004). Visual object recognition using sony four-legged robots. Master's thesis, The University of New South Wales, School of Computer Science & Engineering.

Manders, C., Aimone, C., and Mann, S. (2004). Camera response function recovery from different illuminations of identical subject matter. In *IEEE International Conference on Image Processing 2004*.

Nanda, H. and Cutler, R. (2001). Practical calibrations for a real-time digital omnidirectional camera. Technical report, CVPR 2001 Technical Sketch.

Nisticò, W. and Röfer, T. (2006). Improving percept reliability in the sony four-legged league. In *RoboCup 2005: Robot Soccer World Cup IX*, Lecture Notes in Artificial Intelligence, pages 545–552. Springer.

Röfer, T. (2004). Quality of ers-7 camera images. http://www.informatik.uni-bremen.de/˜roefer/ers7/.

Schulz, D. and Fox, D. (2004). Bayesian color estimation for adaptive vision-based robot localization. In *Proceedings of IROS*.

Schwefel, H.-P. (1995). *Evolution and Optimum Seeking*. Sixth-Generation Computer Technology. Wiley Interscience, New York.

Sony Corporation (2004). *OPEN-R SDK Model Information for ERS-7*. Sony Corporation.

Welch, G. and Bishop, G. (2003). An introduction to the kalman filter.

Xu, J. (2004). Low level vision. Master's thesis, The University of New South Wales, School of Computer Science & Engineering.