ISSN 0005–1144

Gregor Klančar, Mišel Brezak, Drago Matko, Ivan Petrović

Mobile Robots Tracking Using Computer Vision

UDK 004.896:004.93
IFAC 4.6.2; 2.8.3

Original scientific paper

In this paper a global vision scheme applied to a fast dynamic game – robot soccer is presented. The process of robots positions and orientations estimation is divided into two steps. In the first step, the Bayer format image is acquired from camera, then the RGB image is interpolated and pixels are classified into a finite number of classes. At the same time, a segmentation algorithm is used to find corresponding regions belonging to one of the classes. In the second step, all the regions are examined. Selection of the ones that are parts of the observed object is made by means of simple logic procedures. A data filtering is used to improve identified noisy data. The novelty is focused on the optimization of the image acquisition algorithm as well as the processing time needed to finish the estimation of possible object positions.

Key words: Bayer CFA, computer vision, image segmentation, mobile robots, pixel classification

1 INTRODUCTION

This paper presents a design of a global vision system for estimating current object positions and orientations on the playground. The MiroSot category soccer robots we have are without on-board sensors. Thus a precise and fast global vision has to be designed for robots control and navigation in an unknown, dynamically changing area. When designing the vision system, the following requirements have to be accomplished:

- computational efficiency,
- high reliability,
- good precision, and
- robustness to noise, changing of lightening and different color schemes.

The last characteristic is essential for the system to function well when using it under different conditions present at competitions [8].

With color cameras, there are many possible ways to carry out the detection of robots wearing color dresses. All of them try to classify pixels of an image into one of a predefined number of classes. The most common approaches are: linear color thresholding, K-nearest neighbour classification, neural net-based classifiers, classification trees and probabilistic methods [7, 3, 6].

To obtain image of the observed scene a high speed IEEE-1394 digital color single CCD camera is used where image information on CCD sensor is given in so called Bayer format. This image is then reproduced to a full RGB image using bilinear interpolation algorithm. Further on a fast approach with constant thresholding and back-stepping algorithm is presented where a special effort is put into the efficiency aspect. The thresholds can be presented as boxes in 3-dimensional color spaces obtained by means of off-line learning. First an incoming pixel is classified into one of the predefined boxes, then the pixels belonging to one class (a connected region) are univocally labeled. With the main purpose of obtaining all fully connected regions, a back-stepping algorithm is applied. Both steps are done with just one scan of the image. Then the logic part and a simple optimization method are employed to select the proper regions from the previously generated ones. After this logic, the positions and orientations of the objects on the playground are estimated and extended Kalman filter is used to suppress noise of the estimated data.

The paper is organized as follows. In section 2 a brief overview of the system is given. Image acquisition system and algorithm for image interpolation are presented in section 3. Section 4 and 5 focuses on the algorithms for pixel classification and image segmentation. The algorithm for object estimation and data filtering is illustrated in the section 6. The paper ends with conclusions and some ideas for future work.

2 SYSTEM OVERVIEW

The soccer robot set-up, Figure 1, consists of ten MiroSot category robots (generating two teams) of size 7.5 cm cubed, rectangular playground of size 2.2×1.8 m, IEEE-1394 digital color camera and personal computer Pentium IV. The vision part of the

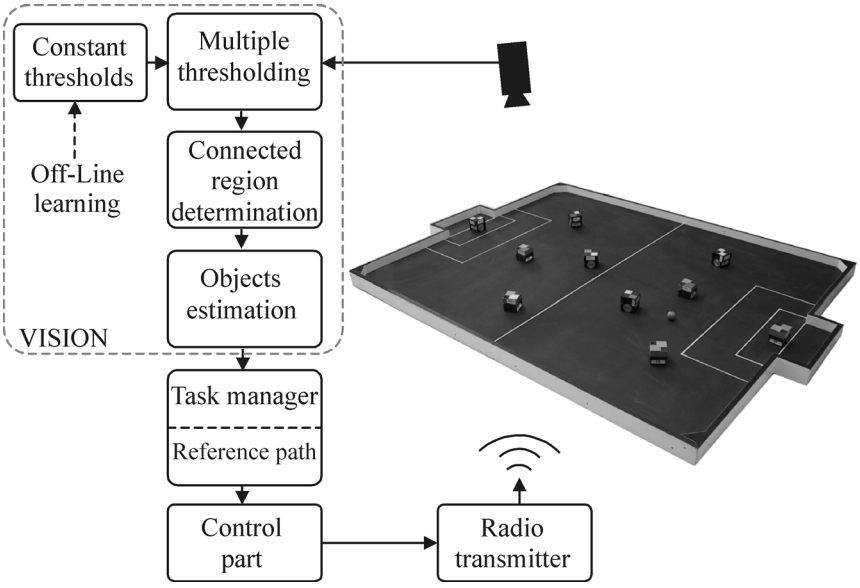

Fig. 1 System overview

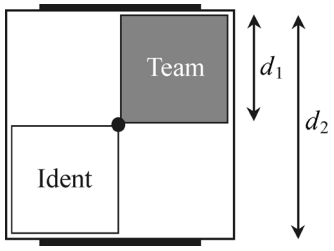

Fig. 2 Color patches on the robot

program processes the incoming images, of a resolution of 640×480 pixels, to identify the positions and orientations of the robots and the position of the ball. Finally, the control part of the program calculates the linear and angular speeds, v and ω, that the robots should have in the next sample time according to current situation on the playground. These reference speeds are sent to the robots by a radio connection.

To identify the orientations, each robot has to have two color patches. One is the team color and the other is the identification color patch. According to FIRA (Federation of International Robot-soccer Association) rules, the team color must be blue or yellow, the ball must be orange and identification colors can be any color except the team and ball color. The patch positions and shapes can be chosen freely. In our case, square patches were used. They were placed diagonally, with the team color being closer to the front part of the robot, Figure 2.

3 IMAGE ACQUISITION AND INTERPOLATION

Image is acquired using industrial digital color camera. It uses IEEE-1394 (Firewire) bus, which offers several advantages, e.g. relatively high bandwidth (400 Mbits/s today, several Gbits/s in the future), guaranteed bandwidth (allows uninterrupted live video to be transmitted without interruptions over the otherwise shared network), standardized specification for the cameras (D-Cam specification), flexibility and lower price of the entire vision system.

In computer vision intended for robot soccer, it is very important that camera has frame rate as high as possible, because control of the robots that move at very high speeds (up to 2.4 m/s) is required, and there is no other control feedback than the image acquired with the camera. Having that in mind, an IEEE-1394 digital camera is used that can operate at 80 fps. Because of high frame rate and limited bus bandwidth, camera cannot transmit full RGB image, but transmits image in so called Bayer format.

This means that, in order to record color images with a single sensor, the sensor is covered with color filter so that different pixels receive different colored illumination. This type of filter is known as a color filter array (CFA). The different color filters are arranged in well-known patterns across the sensor, though the most common arrangement is the Bayer CFA, which consists of filters with bandpasses in three primary colors, red, green and blue (Figure 3). Since each pixel registers only the illumination of the covering's filter color, only one color per pixel is transferred via bus to the host computer, and the remaining color information is not detected by that pixel and must be estimated in host computer. Hence, in order to construct a full color image, interpolation algorithm has to be used [1].

Color interpolation algorithms can be classified into two groups, namely non-adaptive algorithms and adaptive algorithms. Non-adaptive algorithms denote those algorithms that perform interpolation in a fixed pattern for every pixel within a group. While adaptive algorithms imply that those algo-

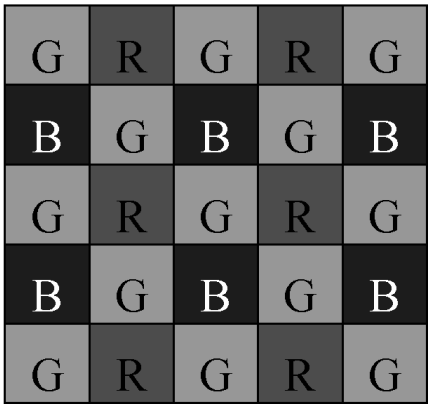

Fig. 3 Bayer CFA pattern

rithms can detect local spatial features present in the pixel neighborhood, and then make effective choices as which predictor should be used for that neighborhood. For the purpose of robot tracking, the non-adaptive algorithm called bilinear interpolation is chosen, because of its low computational time, possibility for further optimization using MMX instructions of the processor and acceptable quality of the resulting image.

Bilinear interpolation explores four points neighboring the point, and assumes that the brightness function is bilinear in this neighborhood. Thus, a weighted value of the surrounding four pixels is calculated. Unfortunately, these weighted values produce a smoothening effect on the output image, which results with zipper effect on edges in color images (edges look like a zipper). Let $R_{i,j}$, $G_{i,j}$ and $B_{i,j}$ be the brightness of the red, green and blue pixel at position (i, j), respectively. If the camera has Bayer pattern like shown in Figure 1 then we interpolate missing color information as follows.

Interpolation of green component for pixels where we know the value of red or blue component is done by averaging the upper, lower, left and right green pixel values:

$$G_{i,j} = \frac{1}{4}(G_{i-1,j} + G_{i,j-1} + G_{i,j+1} + G_{i+1,j}). \tag{1}$$

Interpolation of red or blue component for pixels where green component is known can be computed as the average of two adjacent pixel values in matching color:

$$\begin{aligned} R_{i,j} &= \frac{1}{2}(R_{i,j-1} + R_{i,j+1}) \\ B_{i,j} &= \frac{1}{2}(B_{i,j-1} + B_{i,j+1}). \end{aligned} \tag{2}$$

Interpolation of red or blue component for pixels where blue or red component is known is computed as the average of four neighboring diagonal pixel values:

$$\begin{aligned} R_{i,j} &= \frac{1}{4}(R_{i-1,j-1} + R_{i-1,j+1} + R_{i+1,j-1} + R_{i+1,j+1}) \\ B_{i,j} &= \frac{1}{4}(B_{i-1,j-1} + B_{i-1,j+1} + B_{i+1,j-1} + B_{i+1,j+1}). \end{aligned} \tag{3}$$

The example of interpolated image is given in Figure 2. It can be seen that intensive zipper effect is present on edges, but quality of the image is still acceptable. The described bilinear interpolation algorithm is implemented using MMX assembly instructions of the processor, and this way interpolation time of 2.5 ms on Pentium IV 3 GHz processor is achieved, which is a very good result.

Fig. 4 Interpolated image (original and interpolated image)

4 PIXEL CLASSIFICATION

To enable detection of different color patches, each pixel has to be classified into one of the predefined colors.

4.1 Color Space Transformations

The color image can be presented by the use of different color space notations such as *RGB*, *HSI*, *YUV* and others. When using simple thresholds for pixel classification, *HSI* and *YUV* color spaces are most appropriate [4]. They code the information about chrominance in two dimensions (*H* and *S* or *U* and *V*) and only one dimension includes the information about intensity (*I* or *Y*). A particular color on the playground can be then described with wide areas between thresholds in the intensity dimension, while the threshold areas for other chrominance dimensions are narrow. However, these color spaces are more robust to different lightening conditions.

Both *RGB* and *YUV* spaces were used in our experiments with constant thresholds. The second color space gave slightly better classification results. However, to obtain the *YUV* color representation, a time consuming transformation from the original *RGB* space had to be done. Although this transfor-

mation was optimized by using lookup tables it still requires more than 20 ms, while the rest of the program takes only 10 ms to identify objects from the image. Therefore, *YUV* or any other color space should be used only when it can be directly obtained from the frame grabber.

4.2 Thresholding

The basic idea is to classify each pixel according to the remembered color thresholds of each object. Initially, this part was done with the following code:

```
for i=1 to number of colors on playground
 if (R >= R_lower_boud AND R <=R_upper_bound AND
   G >= G_lower_boud AND G <=G_upper_bound AND
   B >= B_lower_boud AND B <=B_upper_bound)
Pixel_color =i-th color;
 else Pixel_color=background;
end
```

This simple part of the code requires 6 relational and 5 AND operations for each pixel classification. This code is repeated for each color that we want to classify.

To improve this operation, i.e. to check all colors at the same time, an idea of parallelism was considered. Three N×32-bit integer arrays were allocated. Where N corresponds in size to the number of color levels (usually N=256) and the maximum number of colors to be classified is 32 respectively (Figure 5). Each bit in a 32-bit memory location is associated with one color. Although the algorithm is able to classify 32 colors, only 13 different colors are enough for the purpose of the robot soccer game. Because the computer microprocessor has 32-bit arithmetics, the computational burden is the same as for only the 16-bit memory location.

Let us suppose that we want to classify a yellow patch with the color values in the following range:

```
200 <= R <=220
230 <= G <=250
10  <= B <= 30
```

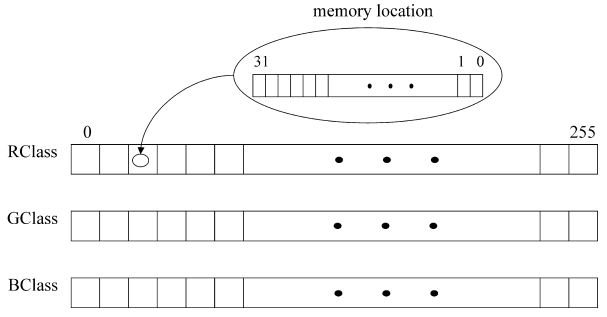

Fig. 5 Look-up tables for pixel classification

Suppose this color is associated with the 31st bit of each memory location. So in this case the highest bits in the memory locations between 200 and 220 in *RClass*, 230 and 250 in *GClass* and 10 and 30 in *BClass* are set to 1, respectively. The same procedure is performed also for bits 0–30 for other color patches.

At the run time, memory locations with the index corresponding to current pixel *R*, *G*, *B* values are taken. The bitwise AND operation between the chosen memory locations gives the information about the classification of pixels. If the result has the 31st bit set to 1, then the pixel is recognized as yellow.

With this methodology, a multiple thresholding (the thresholds for all color patches checked at the time) is made in only one scan of the image. As the multiple threshold operation takes just two AND operations, it significantly reduces computational burden.

To increase robustness of the presented algorithm to poor lightening conditions one must increase color thresholds for each object. However this increases number of classified pixels, because not only pixels belonging to the searched objects appear but also the ones belonging to camera noise and similar color patches. This can be circumvented if all classified pixels are transformed to some nonlinear color space such as HSI (Hue, Saturation, Intensity) and then again thresholded, but only with the first two dimensions (Hue and Saturation). This step filters out most of the noisy pixels. The important advantage of this approach is also its computational efficiency because only classified pixels (using RGB thresholds) are transformed to HSI color representation and not the whole image.

5 IMAGE SEGMENTATION AND COMPONENT LABELLING

To estimate patch positions, first the regions belonging to the ball, team and opponent team patches and all identification patches have to be located. The number of those regions on the playground (K) can be higher than the number of all patches due to noise. Image segmentation in K regions and labeling is done fulfilling the following five conditions [7]:

i. $\bigcup_{i=1}^{K} R_i = R,$

ii. $R_i \bigcap R_j = \varnothing, \quad i, j = 1, 2, \ldots, K, \text{ and } i \neq j,$

iii. R_i is a connected region of pixels,

iv. $P(R_i) = 1, \forall i,$

v. $P(R_i \bigcup R_j) = 0, i \neq j, \text{ and } R_i, R_j \text{ are neighbors},$

where $P(x)$ is a logical predicate, which takes the value one if all the pixels of the region accomplish a criterion of homogeneity. In our case, the homogeneity criterion is the equality in color.

According to the first and second conditions, the regions R_i together must occupy the entire image R and the regions must not have common pixels. Due to the third condition, there must be at least one path of pixels of the same color connecting any two pixels in the region. Moreover, the regions must be homogeneous with respect to the color and the neighbour regions must not have the same color, as stated in conditions four and five.

5.1 Image Segmentation and Labelling Algorithm

Pixel classification and image segmentation are merged by the aid of a corresponding algorithm. Its main property is that the image classification and segmentation is done with only one pass through the image, what considerably contributes to time efficiency. The results of the mentioned algorithm are labelled regions with the following information:

- region number,
- color,
- number of pixels belonging to this region,
- pointers to each pixel belonging to this region,
- coordinates of the centre of the region, x_{avg} and y_{avg}.

Before the algorithm starts running, one color space is selected for each component to be identified: ball, robot1, …, robot5, team, opponent robots and opponent team.

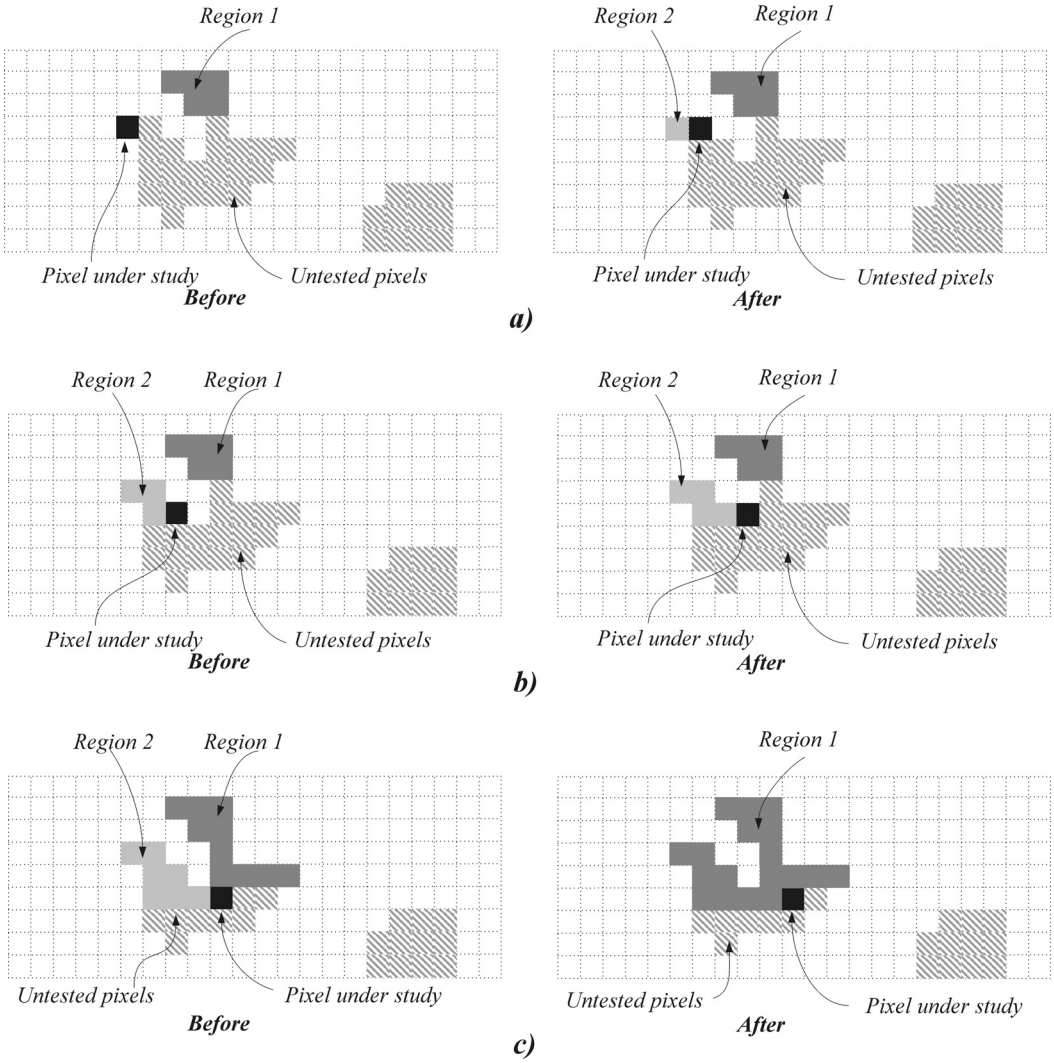

Fig. 6 Image segmentation and labeling

5.2 Algorithm

The algorithm for image segmentation and labeling can be summarized as follows:

- The algorithm starts analysing the 1st pixel of a given region of interest (in our case the whole image).
- If the color of the pixel under study is a valid color and is different from the color of the upper and left neighbor pixels, a new region is created (Figure 6a).
- If the color of the pixel under study is a valid color (it belongs to one of the predefined color labels) and is equal to the color of the upper or left pixel, then the pixel under study is added to the region of the upper or left pixel (Figure 6b).
- If the color of the pixel under study is a valid color and is equal to the upper and left pixel colors, then (Figure 6c):
  - if the upper and left pixels belong to the same region, the pixel under study is added to this region,
  - otherwise, the pixel under study is added to the region with a bigger number of pixels, the pixels of the region with lower quantity of elements are copied to the bigger region, and then the region is deleted.

6 POSITION ESTIMATION

Objects are marked with color patches as shown in Figure 3. To estimate objects position first correct regions which belong to the object has to be selected. Objects positions are then calculated from these regions. To suppress noise of the calculated position and orientation data extended Kalman filter is implemented.

6.1 Object Estimation

From all the possible valid regions identified as described in the previous section, a proper number of regions with the biggest area are selected. First, the team and opponent team regions are investigated. The region is a probable team patch if it is classified as team color and if the positions of other team patches satisfy the condition:

$$dist(region, team_i) > \frac{3}{4} d_1 \tag{4}$$

where *dist* is Euclidean distance, *region* is the current testing region, $team_i$ are already chosen team regions and d_1 is the size of the color patch (Figure 2). To find the right identification region among all which are classified as a particular identification color, the following condition must be fulfilled:

$$\frac{3}{4} d_1 < dist(region, ident_i) < d_2 \tag{5}$$

where *region* is the current testing region, $ident_i$ are already chosen other identification regions (other robots) and d_2 is the robot size (Figure 2).

Considering condition (1) and (2) all possible pairs (team and identification patch) which could represent mobile robots should be found. However situations where wrong pairs could be estimated still exist and are shown in Figure 7. This could happen if more identification patches are associated with one team patch.

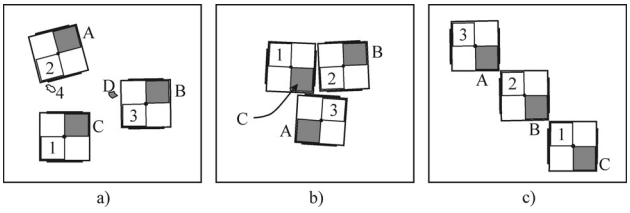

Fig. 7 Different robots placement

In Figure 7, team patches are shaded and marked with different letters, while the identification patches are marked with numbers. Situation in Figure 5a is not problematic because robots are far apart (correct pairs are A2, C1 and B3). The regions 4 and D are not considered because their area is small, and are probably due to the noise. But in Figure 5b wrong pairs could be estimated as C3 or C2. Similarly in Figure 5c wrong estimates could be A2 and B1. The selection of correct pairs (representing the robots) should therefore globally satisfy (for all estimated robots) the condition that each team patch has its own identification patch.

When the right regions representing color patches are found, the final estimated patch position can be improved by taking the weighted average of all region positions with the same classified color and less than distance d_1 away.

From known positions of the regions belonging to the objects, the object positions and orientations are calculated. The position of the ball is equal to its region position, while the *i*-th robot data (position x_i, y_i and orientation φ_i) are calculated as follows:

$$\begin{bmatrix} x_i \\ y_i \\ \varphi_i \end{bmatrix} = \begin{bmatrix} \frac{x_{T_i} + x_{I_i}}{2} & \frac{y_{T_i} + y_{I_i}}{2} & \tan^{-1}\left(\frac{y_{T_i} - y_{I_i}}{x_{T_i} - x_{I_i}}\right) - \frac{\pi}{4} \end{bmatrix}^T \tag{6}$$

with x_{Ti}, y_{Ti} denoting *i*-th position of the team patch and x_{Ii}, y_{Ii} denoting *i*-th position of the identification patch.

6.2 Data Filtering

Calculated robot positions and orientations are corrupted with (camera and frame-grabber) noise and with other disturbances that appear during robot control, such as: wireless communication, wheel sliding and others. Noise level of the estimated data mostly depends on vision system initialization (determination of proper thresholds for certain patches). At proper process initialization the estimated data do not differentiate from the real data more than ±1 mm for positions in ±1° for orientations. However, noise level increases at worse conditions. Furthermore, the estimated data contains approximately one sample delayed information because of image processing. All these problems can be efficiently solved by the use of the corresponding filter. To improve performance of motion controller, extended Kalman filter (EKF) is used [9].

Kinematics of a mobile robot with differential drive is defined by equation:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} 0.5\cos\theta & 0.5\cos\theta \\ 0.5\sin\theta & 0.5\sin\theta \\ 1/b & -1/b \end{bmatrix} \begin{bmatrix} v_R \\ v_L \end{bmatrix}, \tag{7}$$

where b is the distance along the axle between the centers of the drive wheels, and v_L and v_R are circumferential speeds of the left wheel and right wheel, respectively (Figure 8).

Measurement vector of EKF is:

$$\boldsymbol{z} = \begin{bmatrix} x_c & y_c & \theta_c \end{bmatrix}. \tag{8}$$

State variables that are estimated by EKF are just filtered measurement variables, giving measurement model needed by EKF:

$$z(k) = \boldsymbol{x}_k(k) + \boldsymbol{n}(k) \tag{9}$$

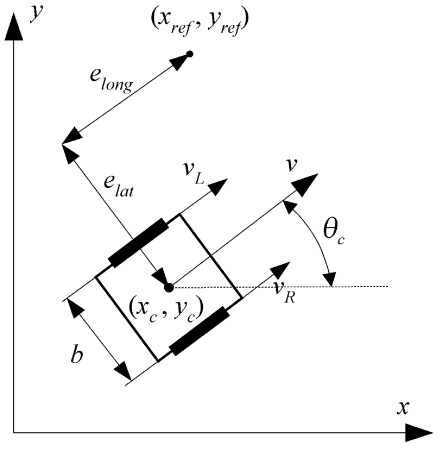

Fig. 8 Scheme of the robot

where $\boldsymbol{n}(k)$ is measurement noise with covariance matrix $\boldsymbol{R}$. EKF is used to estimate expected values of these measurement variables:

$$\hat{\boldsymbol{x}} = \begin{bmatrix} \hat{x} & \hat{y} & \hat{\theta} \end{bmatrix}. \tag{10}$$

State model used for EKF is obtained by discretizing equation (7) with discretization period T. Thus, time update equations of EKF are:

$$\hat{\boldsymbol{x}}^-(k) = f(\hat{\boldsymbol{x}}(k-1), \boldsymbol{u}(k-1)) = \begin{bmatrix} \hat{x}_{k-1} + \frac{T}{2}\left(v_{L,ref}(k-1) + v_{R,ref}(k-1)\right)\cos\hat{\theta}_{k-1} \\ \hat{y}_{k-1} + \frac{T}{2}\left(v_{L,ref}(k-1) + v_{R,ref}(k-1)\right)\sin\hat{\theta}_{k-1} \\ \hat{\theta}_{k-1} + \frac{T}{b}\left(v_{R,ref}(k-1) - v_{L,ref}(k-1)\right) \end{bmatrix} \tag{11}$$

$$\boldsymbol{P}^-(k) = \boldsymbol{A}(k-1)\,\boldsymbol{P}(k-1)\,\boldsymbol{A}^T(k-1) + \boldsymbol{Q} \tag{12}$$

where $\boldsymbol{P}^-$ is *a priori* estimate error covariance matrix, $\boldsymbol{P}$ is a posteriori estimate error covariance matrix, $\boldsymbol{Q}$ is process noise covariance matrix, and $\boldsymbol{A}$ is Jacobian matrix of partial derivatives given by:

$$A_{[i,j]} = \frac{\partial f_{[i]}}{\partial x_{[j]}}(\hat{\underline{\boldsymbol{x}}}, \underline{\boldsymbol{u}}). \tag{13}$$

Measurement update equations of EKF are:

$$\begin{aligned} \boldsymbol{K}(k) &= \boldsymbol{P}^-(k)\left[\boldsymbol{P}^-(k) + \boldsymbol{R}\right]^{-1} \\ \hat{\boldsymbol{x}}(k) &= \hat{\boldsymbol{x}}^-(k) + \boldsymbol{K}(k)\left[\, z(k) - \hat{\boldsymbol{x}}^-(k) \,\right] \\ \boldsymbol{P}(k) &= \left[\boldsymbol{I} - \boldsymbol{K}(k)\right]\boldsymbol{P}^-(k) \end{aligned} \tag{14}$$

where $\boldsymbol{K}$ is the Kalman gain matrix. Designer has to set the values for covariance matrixes $\boldsymbol{R}$, $\boldsymbol{Q}$ and $\boldsymbol{P}(0)$, and for initial values of state estimation $\hat{\boldsymbol{x}}(0)$.

Kalman filter includes the system motion model in its filtering algorithm which enables efficient filtering. However when having a larger disagreement between the used motion model and the reality, the filtered states differ from the real system ones. In the case of the robot soccer set-up this could occasionally happen during objects collisions between robots or the ball. When such cases occur, the best solution is to initialize the filter with the current state values which enables a faster recovery of the filter. Unfortunately, there are also some phenomena which cannot be reliably or even approximately predicted, such as: wheel sliding on different surfaces (clean, unclean, rough, smooth), battery condition, and the like.

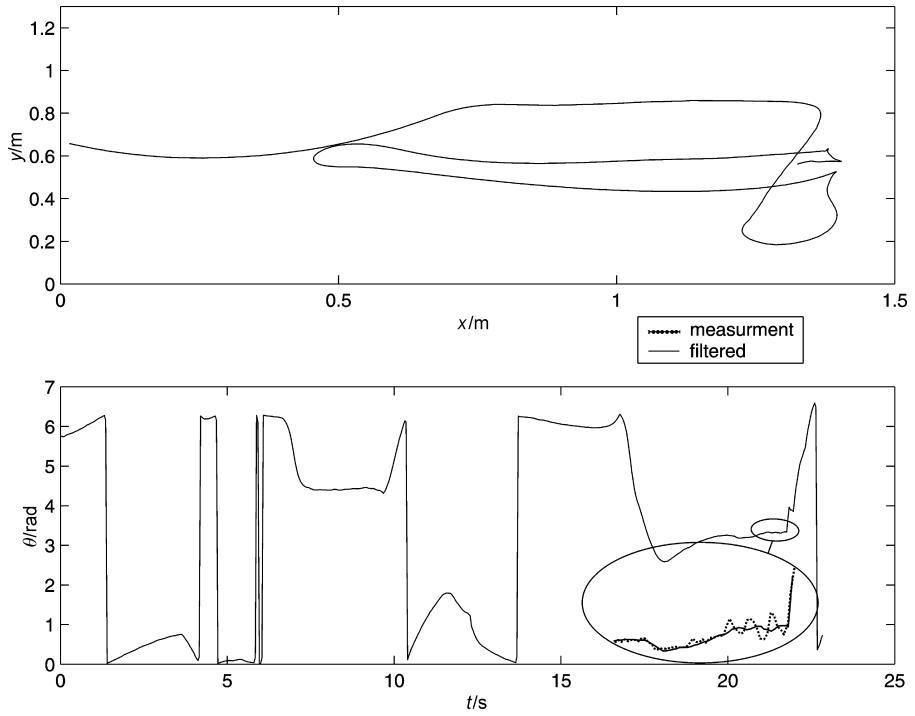

Fig. 9 Robot trajectory and orientation; real and filtered data

The implementation of Kalman filter is validated on a real robot. First, noise variances were determined from an observing standing robot. The results from the above-presented filter for the robot can be seen in Figure 9 where camera estimated data and filtered data are shown.

From Figure 9 it can be concluded that orientation data are more subjected to noise than position data. Therefore Kalman filter proved useful especially at orientation estimation. An important advantage is also prediction of the robot states in future or at least for the delay time resulting from image processing. The used model also does not include collisions among robots or between robots and boundary. Therefore in such cases the estimated data values could vary from real data. The best and simplest strategy and solution to collision situations is to initialize a filter with the current robot state values enabling a faster recovery of the filter. Another problem can appear when the robot data are wrongly estimated due to pure illumination conditions or bad vision system initialization. These outliers are partly filtered out because such behavior is unlikely according to robot model. The solution to these situations is to somehow detect the outlier measurements and simply ignore their values in equation (14).

As already mentioned, at corresponding vision system initializations and good lightening conditions the noise level does not contaminate estimated data more than ±1 mm for positions in ±1° for orientations (evaluations taken when robots stand still). However, noise level increases at worse conditions. The data filtering presented is useful because it suppresses noise and other disturbances and eliminates delays resulting from image processing by predicting future robot states. The obtained filtered data therefore enables a better (more accurate and faster) robot control.

7 CONCLUSIONS

An approach towards establishing a fast and robust vision system for the purpose of robot soccer game is presented. Special consideration is given to optimization of computational work and robustness issues. Robustness is achieved by time-efficient algorithms which enable global image processing. Contrary, some vision systems used by other robot soccer teams employ local image processing to obtain the desired frame rate of vision system. The major disadvantage of these latter algorithms is loss of one or more objects (robots or ball) because of some unpredicted reasons (lightening conditions, collisions, bugs). The local search areas have to be increased until objects are found, which results in larger and irregular sample time. This could not happen with global image processing. However, disadvantage of the presented approach can appear if

large number (more than 15) of different color patches have to be followed. Some of them could then become quite similar on camera image which could result in wrong objects estimation. The problem will be dealt with in future work by inclusion of object tracking algorithms.

From the vision part perspective, our intention was to efficiently merge the following algorithms: image acquisition, classification, connected region determination and labeling. The applied approach is confirmed by a short time (10 ms on 2 GHz Pentium IV computer) needed for an incoming image processing and all objects estimations.

REFERENCES

[1] T. Sakamoto, C. Nakanishi, T. Hase, **Software Pixel Interpolation for Digital Still Cameras Suitable for a 32-bit MCU.** IEEE Transactions on Consumer Electronics, vol. 44, no. 4, pp. 1342–1352, Nov. 1998.

[2] A. M. I. Nilsson, P. U. W. Nordblom, **Interpolation in Color Filter Arrays, Master Thesis.** Malmö University, School of Technology and Society, Sweden, 2002 (http://www.ts.mah.se /utbild/tdtby/exjobb-klara.htm; accessed 25 Nov. 2005).

[3] C. M. Bishop, **Neural Networks for Pattern Recognition.** Oxford University Press, 1995.

[4] J. Bruce, T. Balch, M. Veloso, **Fast and Cheap Color Image Segmentation for Interactive Robots.** Proceedings of IROS--2000, pp. 2061–2066, Takamtsu, Japan, Oct. 2000.

[5] K. Han, M. Veloso, **Reactive Visual Control of Multiple Nonholonomic Robotic Agents.** Proceedings of the International Conference on Robotics and Automation, pp. 3510–3515, Leuven, Belgium, 1998.

[6] C. G. Looney, **Pattern Recognition using Neural Networks. Theory and Algorithms for Engineers and Scientists.** Oxford University Press, 1997.

[7] N. Pavešić, **Raspoznavanje vzorcev: Uvod v analizo in razumevanje vidnih in slušnih signalov** (Pattern Recognition; An Introduction into the Analysis and Understending of Visual and Audio Signals). Faculty of Electrical Engineering, Ljubljana, Slovenia, 2000.

[8] G. Wyeth, B. Brown, **Robust Adaptive Vision for Robot Soccer, Mechatronics and Machine Vision in Practice.** J. Billingsley, Ed., Research Studies Press, pp. 41–48, 2000.

[9] G. Welch, G. Bishop, **An Introduction to the Kalman Filter, Technical Report.** Deptartment of Computer Science, University of North Carolina at Chapel Hill, 2002 (http://www.informatik.uni-bonn.de/III/lehre/seminare/Mustererkennung/WS99/kalman.pdf, accessed 25 Nov. 2005).

Praćenje mobilnih robota korištenjem računalnog vida. Dan je opis algoritma globalnog računalnog vida primijenjenog na brzu dinamičku igru – robotski nogomet. Proces određivanja pozicija i orijentacija robota sastoji se od dva koraka. U prvom koraku, iz kamere prenosi se slika u Bayer formatu, iz koje se potom interpolira RGB slika, a pikseli se klasificiraju u konačan broj klasa. Istovremeno, primjenjuje se algoritam segmentacije kako bi se izdvojilo odgovarajuće regije slike koje odgovaraju jednoj od klasi boja. U drugom koraku ispituju se sve pronađene regije, a odabir onih koje odgovaraju traženim objektima provodi se jednostavnom logičkom procedurom. Koristi se filtriranje kako bi se umanjio šum u izmjerenim vrijednostima. Doprinos ovog rada sastoji se u optimizaciji algoritma interpolacije slike i algoritma obrade slike za mjerenje pozicija i orijentacija objekata.

Ključne riječi: Bayer CFA, računalni vid, segmentacija slike, mobilni roboti, klasifikacija piksela, praćenje robota

AUTHORS' ADDRESSES

Dr. Gregor Klančar
University of Ljubljana, Faculty of Electrical Engineering
Tržaška 25, SI-1000 Ljubljana, Slovenia

Mišel Brezak B.Sc
University of Zagreb, Faculty of Electrical Engineering and Computing, Unska 3, Zagreb, Croatia

Prof. dr. Drago Matko
University of Ljubljana, Faculty of Electrical Engineering,
Tržaška 25, SI-1000 Ljubljana, Slovenia

Prof. dr. Ivan Petrović
University of Zagreb, Faculty of Electrical Engineering and Computing, Unska 3, Zagreb, Croatia

E-mail: gregor.klancar@fe.uni-lj.si
misel.brezak@fer.hr
ivan.petrovic@fer.hr

Received: 2005-12-19