

Architecture of an Object-based Tracking System Using Colour Segmentation

Rafael García-Campos, Joan Batlle, and Rainer Bischoff *

Computer Vision and Robotics Group
Dep. of Electronics, Computer Science and Automation
University of Girona, Av. Lluís Santaló, s/n, 17003 Girona (Spain)
Tel. +34.72.418400 Fax +34.72.418399
email: {rafa, jbatlle}@ei.udg.es

*Institute of Measurement Science
Federal Armed Forces University Munich (Germany)
email: Rainer.Bischoff@UniBw-Muenchen.de

Abstract

This paper presents a robust specialised architecture for real time tracking, using colour segmentation. Most of the existing object-tracking algorithms extract the features which constitute the object (usually called tokens), and track them from one frame to another. We propose a new architecture to solve the tracking problem at object level. The first step is the extraction of the object from the scene. The proposed architecture uses the discriminatory properties of three colour attributes: Hue, Saturation and Intensity, in order to segment the object. The second step consists of the computation of the centroid of the object by using the enhanced image from the segmentation module. As a last step, tracking of this centroid is achieved in a sequence of images. The processor presented here can provide the sequence of centroids at video rate.

Introduction

Tracking moving objects over time is a complex problem in computer vision and has been an important research subject over the last few years [1], [2], [3]. Impressive tracking systems have been developed for some specific applications [4], [5]. We assume, initially, that we just want to track one object in the scene, but there may be other moving objects present in the scene. Some of the methods presented in the literature have a serious matching problem in this case, and often mistake one object for another. In such a situation, when the moving objects have different colour properties from the object to be tracked, our system does not suffer from the influence of other objects. Kinematics-based algorithms may not work properly when the objects abruptly change their motion from one frame to the next [6], [7]. Such situations can be due to collisions or sharp turns. On the other hand, these algorithms detect motion in a sequence of frames, by extracting *tokens* such as edges, corners, interest points, etc. [8], [9]. Methods extracting two dimensional models mistake objects when there is a change in their 2-dimensional shape [10], [11]. This paper presents a new system which can cope with the problems explained above for some special applications, because we are not tracking tokens, but the object (or, more accurately, the centroid of the object). The basic condition imposed by our method is to maintain the colour features of the object in a sequence of images.

In the first part of the paper, we describe the general architecture of our tracking processor. Different modules of the system and their interconnections are shown. Then, a detailed description of each module is provided. Finally, some results and experiments are presented.

General Overview

The simple architecture proposed takes advantage of the robustness of combining Saturation and Hue properties of the object. Furthermore, the dynamic range of the camera is increased by performing the real time re-scaling for the RGB channels as shown in [12].

First, a colour camera takes an image, and its sampled RGB signal feeds the *Colour Conversion* module. RGB is a widely known representation of colour which is not suitable for colour vision applications. Our system transforms the RGB signal into a known model: the Hue, Saturation and Intensity (HSI) model as shown in [13]. The *Colour Conversion* module performs an HSI real time conversion as stated in equations (1), (2) and (3) by means of three Look Up Tables (LUTs): one for Hue, another one for Saturation, and the last one for Intensity.

$$H = \cos^{-1} \left(\frac{\frac{1}{2}((R-G) + (R-B))}{\sqrt{(R-G)^2 + (R-B)(G-B)}} \right) \quad (1)$$

$$S = 1 - \frac{\min(R, G, B)}{\frac{R+G+B}{3}} \quad (2)$$

$$I = \frac{R+G+B}{3} \quad (3)$$

Every component of RGB is sampled into 8 bits. The 5 most significant bits are taken from each component, generating a 15-bit bus. This bus is addressing simultaneously the three 32Kb LUTs, which are programmed through the system bus using (1), (2) and (3) by means of a specific software. As the LUTs are programmed dynamically, it is possible to reconfigure them during execution, loading, for example, a different colour model as shown in [14].

The block diagram of the system is shown in Figure 1.

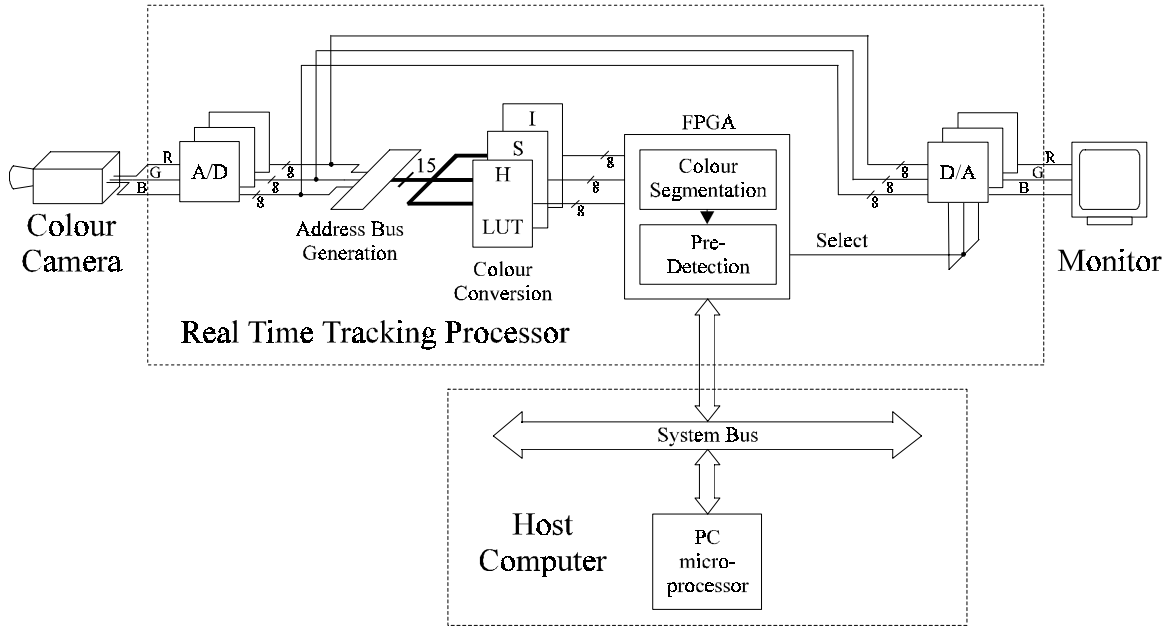


Figure 1: Block Diagram of the System

Once the HSI conversion has been performed, the image can be thresholded by choosing the preferred minimum and maximum values for Hue, Saturation and Intensity, depending on the colour characteristics of the object and the illumination conditions of the scene. For the same comparator (see Figure 2) we can define multiple intervals, allowing to represent an object by a certain number of colour attributes. This operation will be performed in the *Colour Segmentation* module, which is inside the FPGA. The upper and lower limits of Hue, Saturation and Intensity are set up by means of a specific software running on the PC, and these values are multiplexed in the FPGA in order to save I/O pins. For every pixel sampled in the A/D converter, we obtain its Hue component. The time for this computation is equal to the access time to the Hue LUT (approx. 50 ns). This value is passed to the FPGA where it will be compared to the minimum and maximum levels allowed for the Hue, and the comparator will generate a signal indicating whether or not this value is in-between the limits. The same idea applies for Saturation and Intensity.

All the circuitry in the FPGA has been developed using parameterised VHDL descriptions. In this way, future improvements of the system can be easily performed; such as increasing the number of objects to track.

We will define the region R as the set of pixels which are part of the object in the segmented image. This image is used to compute the centroid of the object to be tracked. The computation is performed in two independent phases, which can be pipelined: a first stage where the data from the thresholded image is acquired and accumulated (*Pre-detection* module), and a second one which actually computes the centroid of the object.

The pre-detection consists of accumulating the x-coordinate for all the pixels in the image which are part of the region R . The same idea is applied to the y-coordinate, as well as counting the number of pixels in this region. So $x_c(R)$ is given by (4) and (5), that is

$$x_c(R) = \frac{\sum_{(x,y) \in R} x}{A(R)} \quad (4)$$

$$y_c(R) = \frac{\sum_{(x,y) \in R} y}{A(R)} \quad (5)$$

where $A(R)$ is the area for the region R (the number of pixels in this region). The robustness in the computation of $x_c(R)$ and $y_c(R)$ is achieved because (4) and (5) are actually working as noise filters. This computation is carried out by the PC microprocessor because the hardware implementation of a divisor requires too much space in the FPGA.

Once the coordinates for the centroid of region R have been found, this centroid can be expressed as (6).

$$c(R) = (x_c(R), y_c(R)) \quad (6)$$

Figure 2 shows the block diagram of the internal structure of the FPGA.

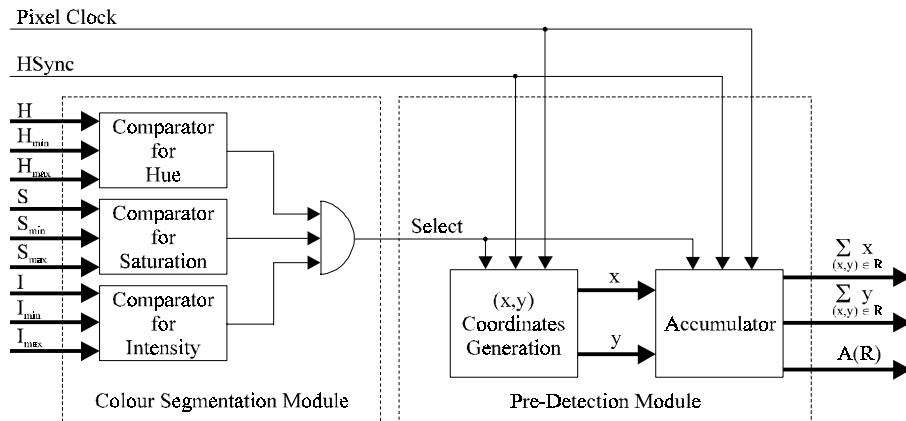


Figure 2: *Internal Structure of the FPGA*

So, the processor is organised in a two stage pipeline. The first stage is dedicated to the acquisition of data from the scene. While this data is being acquired and accumulated for frame k ($\sum_{(x,y) \in R} x, \sum_{(x,y) \in R} y, A(R)$), the second stage is

processing previous data from frame $k-1$. In this way, new data may be acquired while the previous data is being processed. This allows the processor to compute $c(R)$ at video rate.

The time available to compute the centroid of the object in one scene is a whole frame period (20 ms in the CCIR standard) because of the pipeline. It means that the centroid of the frame k will be obtained 20 ms later. So, the tracking is performed with a delay of one frame. This compares to systems using a frame-grabber, which normally have to wait for the acquisition of the frame into memory before starting to process it. This means that our processor has already pre-processed the image while normal frame-grabbers are still storing.

Eventually, a sequence of centroids is provided by the microprocessor in real time in order to track the object in a 2-dimensional approach, and the marked object is displayed on a monitor.

Experiments and Results

Two experiments have been carried out. In the first test we used a video sequence of motorbikes in a circuit where different characteristics of visually similar colours were present. We could keep track of them although the perspective and form of bikes and pilots changed. In the second experiment we tracked a painted mobile robot. This mobile robot was painted in two colours: one for the front part and another for the back. The centroid of each colour was computed, obtaining the direction vector of the vehicle.

The only condition imposed in our experiments is the difference in the colour properties between the object to be tracked and the other objects and the background, respectively. Although this condition may look too restrictive, experience has shown that very few objects have the same *Hue* and the same *Saturation* in the tested images. Tests with the *Intensity* property showed that it is too susceptible to lighting conditions, and only in a very few cases it can help in the segmentation process.

Conclusions

We have briefly presented here a simple but robust architecture, which allows the 2-dimensional tracking of an object even when there are other moving objects in the same scene. The system is robust enough to keep track of both rigid and non-rigid objects even if there are changes in their perspective. As far as we are not using any heuristics we are not affected by brusque changes in the motion and/or occlusions as in kinematics-based systems. Our approach just computes the centroid of the object, while other approaches like in the token-based tracking, every token (feature) is characterised by its position and orientation [15]. The orientation will not be provided by our system, but, on the other hand, we need less computation to determine the position. Moreover, tracking is achieved without the need of a memory for storing and scanning the image, saving time and costs.

Further Work

This system can easily be extended due to its scalar architecture. It allows to track more than one moving object in the scene by adding more comparators and accumulators to the FPGA. The two-stage pipeline leaves enough time for the computation of multiple centroids. Accuracy of the colour segmentation module will be improved by increasing the number of bits for the HSI conversion from 5 to 7/8 per channel. A second prototype is being developed where the divisions from equations (4) and (5) are performed in the FPGA, instead of passing the values to be divided over the system bus to the CPU. The use of parameterised VHDL descriptions helps to rapid testing and prototyping.

Future research work will extend to multiple camera configurations in order to extract depth information. Our real time tracking processor could then be used in a wide range of applications, such as object grasping by assembly robots, visual feedback for automated navigation, etc.

References

- [1] Lowe, D.G., "Robust model-based motion tracking through the integration of search and estimation," *International Journal of Computer Vision*, vol.8:2, pp. 113-122, 1992.
- [2] Coombs, D., and Brown, C., "Real-time smooth pursuit tracking for a moving binocular robot," *Proc. IEEE*, pp. 23-28, 1992.
- [3] Huttenlocher, D.P., Noh, J.J., and Rucklidge, W.J., "Tracking non-rigid objects in complex scenes," *Proc. IEEE*, pp. 93-101, 1993.
- [4] Dickmanns, E.D., Graefe, V., "Applications of dynamic monocular machine vision," *Machine Vision and Applications*, pp. 241-261, vol. 1, 1988.
- [5] Frau, J., Casas, S., Balcells, Ll., "A dedicated pipeline processor for target tracking applications," *Proc. IEEE International Conference on Robotics and Automation*, pp. 599-604, 1992.
- [6] Shariat, H., Price, K. E., "Motion Estimation with more than Two Frames," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 417-432, vol. 12:5, 1990.
- [7] Horn, B. K. P., Schunck, B., "Determining Optical Flow," *Artificial Intelligence*, pp. 185-203, vol. 17, 1981.
- [8] Zhuang, X., Huang, T.S., Ahuja, N., Haralick, R.M. () "A simplified linear optic flow-motion algorithm," *Computer Vision, Graphics and Image Processing*, pp. 334-344, vol. 42, 1988.
- [9] Davis, L.S., Wu, Z., and Sun, H., "Contour-based motion estimation," *Computer Vision, Graphics and Image Processing*, pp. 313-326, 1982.
- [10] Bergevin, R., and Levine, M.D., "Extraction of line drawing features for object recognition," *Pattern Recognition*. Vol.25: 3, pp. 319-334, 1992.
- [11] Cédras, C., and Shah, M. "Motion-based recognition: a survey," *Image and Vision Computing*. Vol.13:2, pp. 129-155, 1995.
- [12] Regincós, J., and Batlle, J., "A system to reduce the effect of CCDs saturation," *Proc. IEEE International Conference on Image Processing*, 1996 (*To appear*).
- [13] Gonzalez, R.C., and Woods, R.E., "Digital Image Processing," *Addison-Wesley Publishing Company*. 1992.
- [14] Pujas, Ph., and Aldon, M.J., "Robust colour image segmentation," *Proc. International Conference on Advanced Robotics*, pp. 145-155, 1995.
- [15] Deriche, R., and Faugeras, O., "Tracking line segments," *Proc. European Conference on Computer Vision*, pp. 259-268, 1990.