Aleksander NAWRAT
Silesian University of Technology at Gliwice
Karol JĘDRASIAK
Polish-Japanese Institute of Information Technology at Bytom

FAST COLOUR RECOGNITION ALGORITHM FOR ROBOTICS

Key words

Image processing, target recognition, human-machine interface.

Summary

Developing fast and accurate 2D image processing algorithms is an important task for the practical use in mobile robotics. This paper presents an algorithm for fast and accurate blob detection and extraction based on the usage of two parameters ξ and χ. The algorithm is aimed to work in the colour domain to prevent any loss of information but can also be implemented on grey-scale images. Achieved regions of interest can be further processed to achieve high level description. The algorithm is implemented in Java environment in order to adduce results on different video devices and system platforms.

Introduction

The most important part of computer vision is the detection of the required features for further processing. No exact definition was developed for a feature; therefore, it is important to specify whether edges, corners, or regions are required. Numerous approaches of feature detection were developed, such as edge and corner detectors or blob detectors. In cybernetics, it is important to detect specific objects by their colour or shape, or both, in order to maximise the chance for further successful recognition. Detection is a low-level image processing operation that often appears computationally expensive. Each of the detectors varies in terms of speed and accuracy. Classic blob detection aims at detecting regions in the image that are described by the intensity value different from that of the surroundings. The most popular approaches are the Laplacian of Gaussian (LoG) and the difference of Gaussian (DoG). It is required to convolve the image by a Gaussian kernel and compute the LoG or DoG operator. It is an efficient technique, but the convolving phase of the process with kernels 5 by 5 or bigger is computationally expensive.

For many years, the research in the feature detection field concentrated on grey-scale image processing. In colour images, the human vision model can be used to improve the accuracy of the algorithm. The colour detection problem is often solved using a colour comparator. Thanks to its simplicity and efficiency, the Euclidean Distance Vector is often applied to solve the colour detection problem. The result depends on the definition of distance and the colour model. Different results will be achieved using Minkowski distance or Chebyshev distance. The colour model used in the most of the image acquisition devices and displays is the RGB colour model; therefore, it is the standard model for colour domain images. This model stores individual values for red, green, and blue. It uses additive colour mixing to achieve complex colours. The colours of a real-life object depend on illumination. It is easier to detect intensity changes in HSV (Hue, Saturation, Value) colour model, but additional conversion from RGB colour space to HSV colour space is required. In HSV colour model colours are separated by 120° angles and have a different saturation level and intensity value. Instead of using the HSV model, a Vector Angle metric was introduced. However both solutions, or even their joint usage, have certain limitations. The techniques mentioned above are used for edge detection, but can also be used for seeking for regions of interest. Though fast and accurate blob detection algorithms, which could be used in cybernetics, are still yet to find.

1. Algorithm overview

The algorithm aims at fast processing and accuracy of detecting specific colour and extracting blobs for further analysis, usually edge detection. The speed requirement is fulfilled by iterating over the image only twice and using as simple methods as possible. Accuracy is controlled by two parameters: ξ and χ.

At the outset, the algorithm iterates over pixels vertically. For each pixel luminous intensity is calculated using the following formula:

$$I_v = \frac{299r_p + 587g_p + 114b_p}{1000} \tag{1}$$

where: r_p – red colour intensity,
g_p – green colour intensity,
b_p – blue colour intensity,
I_v – pixel luminous intensity.

At the same step, the distance vector between current pixel and reference colour is counted. To perform this task, the previously extracted colour intensities are used in the formula below:

$$\delta = \sqrt{(r_p - r_r)^2 + (g_p - g_r)^2 + (b_p - b_r)^2} \tag{2}$$

in this particular algorithm the following faster estimation is used:

$$\delta = |r_p - r_r| + |g_p - g_r| + |b_p - b_r| \tag{3}$$

where: r_r – reference pixel red colour intensity,
g_r – reference pixel green colour intensity,
b_r – reference pixel blue colour intensity,
δ – length of distance vector between current pixel and reference pixel.

The parameter ξ is used to control the tolerance of detected pixels. If $\delta<\xi$, then current pixel is marked with a special colour for the next stage of process. Real-life images consist of objects that reflect light and, in natural way, produce different level shades and scales. To detect such smooth changes in pixel intensity, a method similar to adaptive threshold is used. The previous intensity value is stored as ψ variable and the intensity of the following pixel is stored as κ variable. Therefore, the estimation of standard deviation is as follow:

$$\sim \sigma = \sqrt{(I_v - \psi)^2 + (I_v - \kappa)^2} \tag{4}$$

The usage of χ parameter enhances sensitivity of the algorithm to processing shades.

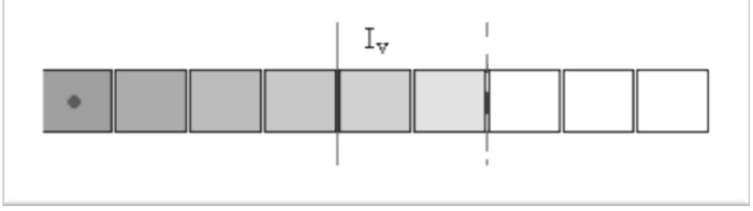


Fig. 1. A result of colour detection with usage of vector distance (line) and implemented improvement (dashed line). Red dot marks reference pixel

If $\sim\sigma<\chi$ then current pixel is also marked for next stage. To avoid detecting every pixel with slight change in intensity a formula is calculated only for pixels in close proximity of the pixel marked previously as correct.

The next stage of algorithm performs a similar approach but horizontally. It is also checked if the pixel is within the tolerance range of parameters ξ and χ. The proximity is calculated as is the previous and the following pixel in the same row. The blob extraction is also performed at this stage. Pixels marked during the prior horizontal phase and vertical phase are the subject of extracting. Storing the whole region of interest in the created blob object is inefficient, therefore only four peripheral pixels are stored. To avoid computable expensive creating and iterating over all blob objects, it is important to check whether currently examining the proximity of the pixel contains previously marked pixels. The presented approach performs a preselect stage to reduce unnecessary iterations over a possibly high number of detected blob objects.

The final stage of the algorithm finds and destroys blobs that do not match requirements such as minimum width or area. Before presenting extracted data for each blob, the centre of the mass is calculated using the simplest formula and previously found four peripheral pixels.

$$R=\left(R_x=L_x+\frac{(R_x-L_x)}{2},R_y=T_y+\frac{(B_y-T_y)}{2}\right) \tag{5}$$

where: R_x – x coordinate of the centre of the mass,
L_x – x coordinate of the left peripheral pixel,
R_x – x coordinate of the right peripheral pixel,
R_y – y coordinate of the centre of mass,
T_y – y coordinate of the top peripheral pixel,
B_y – y coordinate of the bottom peripheral pixel.

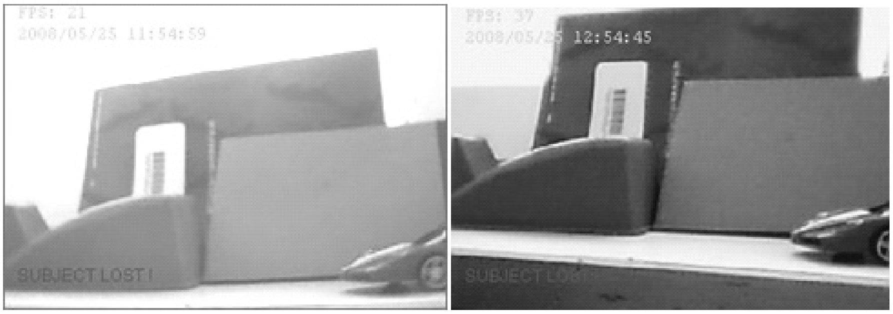
2. Results

The outcome images with marked detected blobs are presented for two different webcams. The first is Logitech Quickcam Express and the second is Trust 150 SpaceC@m Portable. Each camera will be tested as follows:

a) In two resolutions: 320x240 and 640x480;
b) Without detection, with distance vector detection, with implemented changes; and,
c) Speed comparison.

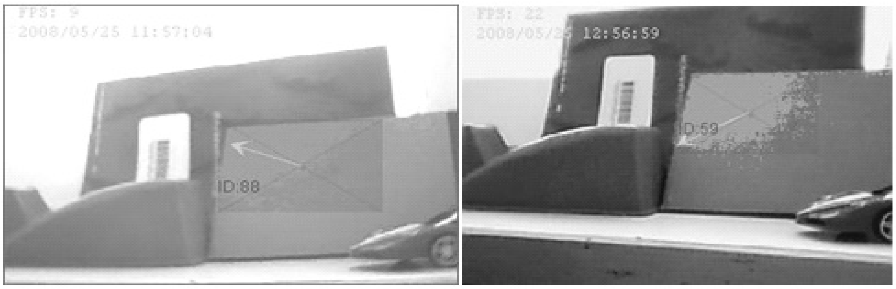
Example tests are shown an output image and algorithm speed measured in FPS.

A. Resolution 320x240



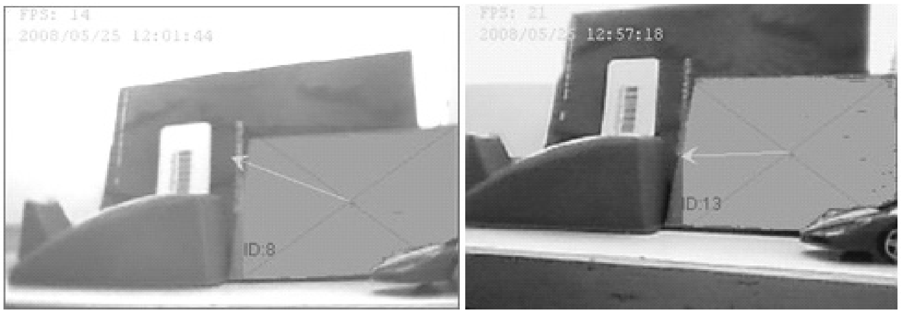
(a) (b)

Fig. 2. Screen without detection – (a) Trust 150 SpaceC@m Portable, (b) Logitech QuickCam Express



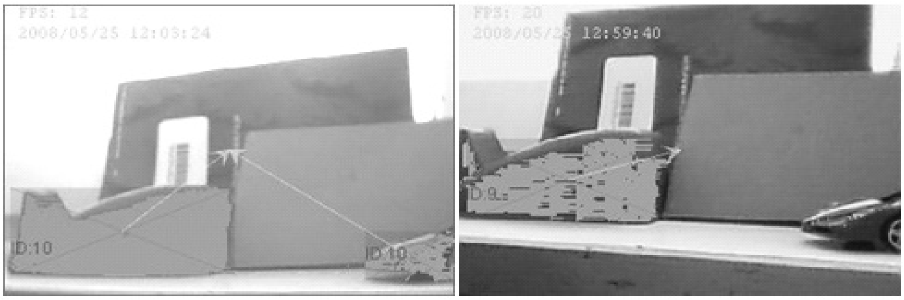
(a) (b)

Fig. 3. Green colour detected with distance vector, tolerance 118 (ξ = 118, χ=0) Yellow arrow points the centre of the image. (a) Trust 150 SpaceC@m Portable screen. (b) Logitech QuickCam Express



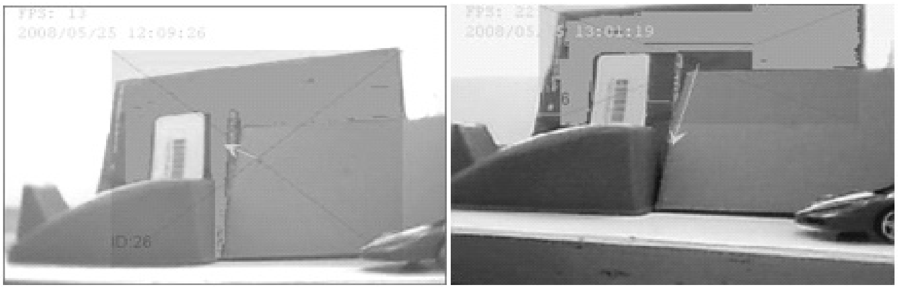
(a) (b)

Fig. 4. Green colour detected (ξ = 118, χ=9). (a) Trust 150 SpaceC@m Portable screen. (b) Logitech QuickCam Express



(a) (b)

Fig. 5. Red colour detected (ξ = 50, χ=8). (a) Trust 150 SpaceC@m Portable screen. (b) Logitech QuickCam Express



(a) (b)

Fig. 6. Blue colour detected (ξ = 50, χ=8). (a) Trust 150 SpaceC@m Portable screen. (b) Logitech QuickCam Express

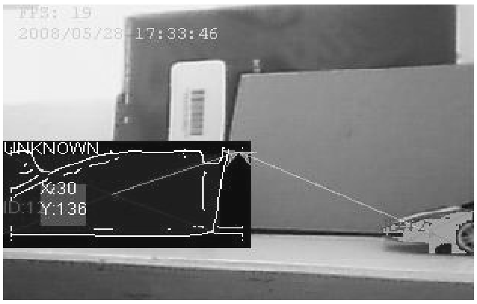
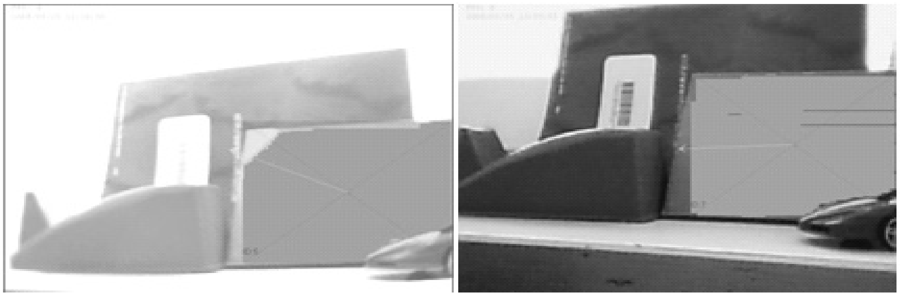


Fig. 7. An example of further processing only within specified sub-images created during blob detection (The Canny Edge detector)

B. Resolution 640x480



(a) (b)

Fig. 9. Green colour detected (ξ = 50, χ=10). (a) Trust 150 SpaceC@m Portable screen. (b) Logitech QuickCam Express.

C. Speed Comparison

TRUST 150 SpaceC@m Portable			LOGITECH Quickcam Express	
Blob Size	320x240	640x480	320x240	640x480
0%	21	5	37	14
20%	12	4	22	6
40%	10	3	20	5

Fig. 9. Comparison of algorithm Speed for different blob sizes, resolution and camera. All values are the outcome from counting FPS

The result collation table presents acquired results. It is shown that the speed of the algorithm varies depending on the size of blob and resolution. The interesting part of this collation is that different camera models can increase or decrease the speed of calculations. Outcome pictures present that the Trust camera achieved more accurate results but was much slower in comparison to Logitech camera. On the other hand, the results of the Logitech camera were less accurate.

Conclusions

The presented algorithm aims at practical use in cybernetics; therefore, it was designed to detect and extract blobs efficiently. It was shown that the popular Euclidean distance vector approach can fail in real-life cases, and the above solution successfully corrected the inaccuracy. The χ parameter is sensitive to noise. Overall algorithm detection efficiency can be further improved by convolving the image with Gaussian blur mask at the beginning of the detection process. Another possible improvement is performing erosion and dilatation before extracting blobs. The simplicity of the proposed approach should make it an attractive tool for cybernetic implementations.

References

1. Luong Q.-T.: “Color in computer vision” in Handbook of Pattern Recognition and Computer Vision, C.H. Chen, L.F. Pau and P.S.P. Wang, editors, 311–368, World Scientific Publishing Company, 1993.
2. Amini A., Weymouth T., Jain R.: Using dynamic programming for solving variational problems in Vision. PAMI, 12(9), 1990.
3. Healey G.: Segmenting images using normalized color" IEEE T. on Sys., Man, and Cyb., 22, 1992.
4. Swain M. Ballard D.: Color indexing. IJCV, 7(1), 1991.
5. Terzopoulos D.: On matching deformable models to images: Direct and iterative solutions. In OSA Technical Digest Series, vol. 2, 1987.
6. Hedley M., Yan H..: Segmentation of color images using spatial and color space information. Journal of Electronic Imaging, 1, 374–380, 1992.

Reviewer:
Jerzy FRĄCZEK

Szybkie rozpoznawanie kolorów w robotyce

Słowa kluczowe

Przetwarzanie obrazu, rozpoznawanie celu, interface maszyna-człowieka.

Streszczenie

Opracowanie szybkich, a zarazem dokładnych algorytmów przetwarzania obrazów 2D jest istotnym zagadnieniem dla praktycznego zastosowania w mobilnej robotyce. W niniejszej pracy prezentowany jest algorytm szybkiego oraz dokładnego wykrywania obszarów zainteresowania. Ponadto przedstawiona jest ich przykładowa ekstrakcja z obrazu oparta o użycie dwóch parametrów: ξ i χ. Algorytm ma za zadanie działać w przestrzeni barw w celu uchronienia przed utratą ewentualnych informacji, jednakże może być również wykorzystany w przestrzeni skali szarości. Uzyskane obszary zainteresowań mogą być dalej przetwarzane w celu uzyskania opisu na wyższym poziomie abstrakcji. Algorytm jest zaimplementowany w języku programowania Java w celu uzyskania łatwo wyników niezależnie od urządzeń wideo, czy systemów operacyjnych.