Real-time Colour Recognition for Machine Vision Systems

Bruce G. Batchelor
Department of Computing Mathematics
University of Wales
Cardiff, Wales, United Kingdom

Paul F. Whelan
School of Electronic Engineering
Dublin City University
Dublin, Ireland

Abstract: It is impossible to collect more than a tiny proportion of all of the possible examples of a given hue, to form a training set for a machine that learns to discriminate colours. In view of this, it is argued that colour generalisation is essential. Three mechanisms for learning colours as defined by a human being are described. One of these is based upon an idea developed by AP Plummer and is implemented in a commercial device known as the Intelligent Camera. This implementation can learn the characteristics of coloured scenes presented to it, and can segment a video image in real-time. This paper presents four procedures which allow the range of colours learned by such a system to be broadened, so that recognition is made more reliable and less prone to generating noisy images, which are difficult to analyse. Three of the procedures can be used to improve colour discrimination, while a fourth procedure is used when a single and general colour concept has to be learned. Several experiments were devised to demonstrate the effectiveness of colour generalisation. These have shown that it is indeed possible to achieve reliable colour discrimination / recognition for such tasks as inspecting packaging and fruit. A practical system based upon the Intelligent Camera and controlled by software written in Prolog has been developed by the authors and is being used in a study of methods for declarative programming of machine vision systems for industrial applications.

1. Introduction

Consider the task of designing a machine to inspect printed cardboard and plastic cartons, such as those used for food products, household goods, toiletries, etc. These are often printed in several colours. Inspecting such items could be achieved by first isolating the different colours and then applying conventional (i.e. monochrome) image analysis procedures to each of the colour separations. In this article, we shall discuss electronic filters that are capable of performing such a separation of colours. (Plummer 1991; Intelligent Camera 1990) A *Programmable Colour Filter (PCF)* might, for example, isolate the yellow streak on a margarine tub, so that it can be examined in detail. Once a yellow streak has been inspected, other coloured features can be treated in the same way. Of course, such a filter should be able to tolerate wide variations in the brightness of the scene being examined; it should be sensitive to colour, not intensity. Other possible applications include: reading resistor colour codes, identifying coloured wires, inspecting iconic displays on car dash boards, etc.

Previous work by the authors has shown that colour recognition can be used to good effect in a symbolic programming environment (Prolog has been used for several years as a medium for programming vision systems. See Batchelor (1991; 1992), Batchelor and Whelan (1993; 1994)). This approach to colour recognition, requires machines that can quickly learn to recognise colours such as *"margarine-tub yellow", "margarine-tub red"*, given a few examples of each. The samples on which such an inspection machine is to be designed would most conveniently be obtained by examining a small number of margarine tubs on a production line.

1.1 The Naming of Colours

The axiom on which this article is based is that the names of colours cannot be defined mathematically. The standard CIE (1931) Chromaticity Diagram should properly be regarded as a conceptual aid, since it cannot form a precise basis for discriminating between colours. The position of the boundary between any two named colours in the Chromaticity Diagram is plotted for an hypothetical *standard observer*, working in carefully controlled lighting conditions. In a practical situation, however, an industrial machine vision system is likely to be taught by a person untrained in colour science, working in a factory environment, where the lighting is highly variable. Schettini (1993) points out that camera, lighting and filter combinations affect the RGB values measured by a video camera.

Several authors, have represented the colours of ordinary everyday objects as points plotted on the Chromaticity Diagram. (Chamberlain and Chamberlain 1980). However, it should be noted that each point represents just one instance of a broad class of objects. The set of all ripe tomatoes, for example, is represented more accurately by a cluster of points, while ripening tomatoes generate a broad serpentine curve in the Chromaticity Diagram. The Chromaticity Diagram does not include definitions for the range of such colours as *"margarine-tub yellow"* or even *"sky blue"*. The fact that people recognise colours by some mental process that is not fully understood simply has to be accepted. (Chamberlain and Chamberlain 1980; Optical Society of America 1953). The authors suggest that a colour recognition filter used when inspecting artefacts such as food packaging, household good and pharmaceutical cartons could be designed using the principle of teaching-by-showing.

1.2 Notation

The set notation introduced in this section allows us to define colour generalisation process in formal mathematical terms. Generalisation is seen as being an essential function in any learning system. Let <X> denote the set of colours of objects in that class defined by human beings and which is called X. A machine that is designed for colour recognition, might well use the conventional RGB colour separations. To take account of this fact, we shall therefore take {X} as being the set of all of those (R,G,B)-vectors that can be associated with the label X. Notice that in this notation, <X> is defined by a person, while {X} is a set of 3-element (R,G,B)-vectors, derived by a machine.

1.3 Recognition and Generalisation of Colours

Implicit in our approach to colour recognition is the concept of teaching by showing. It is important, of course, to make the maximum use of each colour sample, since they may be difficult and/or expensive to collect. It is impossible, in practice, to obtain more than a very small proportion of all the colours of a class such as <yellow>, so we must teach our machine using a few well chosen samples and leave it to generalise. Generalisation is universally accepted as being essential in *all* pattern recognition machines, of which the PCF is an example.

Given that <daffodil> $\cup$ <canary> $\cup$ <banana> $\cup$ <lemon> $\subseteq$ <yellow> it is reasonable to expect that {daffodil} $\cup$ {canary} $\cup$ {banana} $\cup$ {lemon} $\subseteq$ {yellow}. Now, we want to find some operation upon the set {daffodil} $\cup$ {canary} $\cup$ {banana} $\cup$ {lemon} which will generate an enlarged set E, such that E $\supseteq$ {daffodil} $\cup$ {canary} $\cup$ {banana} $\cup$ {lemon} and $\forall$ X: X $\in$ E $\rightarrow$ X $\in$ {yellow}. An important but ill-defined condition is that the set E should be as small as possible, thereby avoiding over-generalisation.

This is one of the two types of colour generalisation we discuss in this paper. It is appropriate for those situations in which we are interested in *colour recognition* (single colour class), as distinct from *colour discrimination* (more than one colour class). We shall present one procedure for generalisation in colour recognition. (*Procedure 4* defined below) A different type of colour generalisation is needed when we have to discriminate between colours. For reasons of economy, we might, for example, need to use a small data set to learn to distinguish between {apple} and {tomato} and wish to make the discrimination more reliable, so that colours in these sets that were not represented in the training data are classified appropriately.

2. Colour Recognition

The inspection of coloured objects and surfaces by machine has, of course, been studied by numerous researchers over many years. Particular attention has been paid to the characterisation and matching of subtle colouring of fabrics, paint, paper, printing and automobiles. Since very precise colour measurement is needed in these areas, non-imaging techniques have been widely used. It should be understood that the approach that we have taken is quite different, since we are concerned with relatively coarse, high-speed recognition and discrimination processes. A typical application for the techniques we shall discuss is the inspection of packages and containers for food, domestic goods and pharmaceutical products on a factory production line. A review of previous work in this area can be found in Batchelor and Whelan (1994).

2.1 Real-time Recognition of Colours in Electronic Hardware

Figure 1 shows the block diagram of a colour recognition system designed by Plummer (1991). This is built into a small self-contained commercially available image processing unit, called the Intelligent Camera (1990). The authors used the Intelligent Camera, in conjunction with control software written in Prolog (Batchelor 1991; 1992) in the experiments reported below. Another implementation using a real-time RGB/HSI converter chip (Umbaugh et al 1992) is suggested in Figure 2. A third implementation relies upon the use of the *xy* parameters used to define the standard Chromaticity Diagram, see Figure 3. These last two configurations have not yet been implemented.

Notice that in Figures 1-3, the output from the Look-Up Table (LUT) is a stream of 8-bit values, which may be regarded as forming intensities in a monochrome image. This image can be analysed in a conventional monochrome image processing sub-system. All of these hardware systems can be fully simulated in a software environment, but not necessarily in real-time. While it is the accepted wisdom that the HSI representation is better able than RGB to discriminate colours as we perceive them, this hardware arrangement is, in fact, quite general, since the LUT in Figure 1 can be programmed to generate H, S and I, given R, G and B. Hence, Figure 1 is able to implement any functions which Figures 2 and 3 can. A further advantage of Figure 1 is that it relies upon cheap standard memory devices, rather than custom ICs or real-time divider circuits. Our discussion hereafter is based upon the system using a LUT with RGB inputs, as illustrated in Figure 1.

The LUT forms the heart of the *Programmable Colour Filter*. The use of high-speed random access memory (RAM) to form a look-up table, together with "flash" analogue-to-digital converters, makes the PCF very fast indeed. It is well able to perform transformations upon a digitised video signal, in real time. Training the PCF consists of calculating appropriate values for each of the LUT's 2^{18} storage cells. (See Batchelor (1992), Batchelor and Whelan (1993; 1994) for details on the programming of the PCF). Once the PCF has been programmed, the colour recognition process takes place in real-time and does not increase the computational load needed for image analysis in any way.

3. Procedures for Colour Generalisation

The colour scattergram (generated by projecting all RGB vectors onto the colour triangle, this is a convenient representation of the distribution of colours within the input (Batchelor and Whelan 1993; 1994)), may be displayed as a grey-scale image, in which intensity indicates the number of pixels with the same values of hue and saturation. (See Figure 4 for some results.) The colour scattergram must be simplified before any further processing takes place. An obvious step is to threshold it, This process will generate a compact blob for each region of similar colours.

Our approach to colour generalisation consists of adjusting the sizes of the blobs created by thresholding the colour scattergram. It will be necessary to do this in such a way that blobs which were distinct when the (multi-cluster) scattergram was first thresholded, remain separate. In a typical application, a number of coloured scenes are used to design the PCF. As each scene is being viewed, a scattergram is generated in the colour triangle. Noise is then removed from the scattergram, using common image processing operations, such as low-pass filtering. This is followed by thresholding. If the input scene consists of a single colour, such as *<margarine-tub yellow>*, thresholding the scattergram, after clean-up, creates a single blob, B_i. This process is repeated for each colour we wish to use to design the PCF. As each new blob (B_i, $i = 1,\ldots,n$) is generated, it is superimposed on the colour triangle. Therefore, prior to generalisation, the colour triangle consists of a number of blob regions, each of which corresponds to one of the trained colours. The aim of the generalisation procedures is to expand these regions, forming the regions C_i, $i = 1,\ldots,n$. By projecting the C_i, back onto the colour cube, we generate the contents of the PCF look-up table. If the C_i have been generated appropriately, the resulting colour recognition process is more reliable, than it would have been if the smaller blobs B_i had been used instead (Batchelor and Whelan 1993; 1994). Here are the definitions of four suggested procedures for colour generalisation.

Procedure 1 - Simple Dilation: Each blob, B_i, in the colour triangle is dilated (expanded) by single a pixel for a fixed number of iterations. The number of iterations is denoted by the variable N. The resultant blob is C_i.

Procedure 2 - Dilation with Preservation of Connectivity: A single layer of background pixels is stripped from the (binary) image in the colour triangle. Unlike the previous approach, pixels critical for connectivity are retained. The number of iterations in this 'onion peeling' operation is denoted by the variable N.

Procedure 3 - Watershed: This approach involves finding the watershed for each of the blobs B_i. The watershed is generated by finding the medial axis transformation of the image background.

Procedure 4 - Convex Hull Generalization: The convex hull is drawn around the set of blobs B_i $(i = 1,\dots,n)$ in the colour triangle. It is reasonable to expect that points within this convex hull will correspond to a generalisation of the observed colours, $\{A_i\}$, $i = 1, \dots,n$.

4. Demonstration of Colour Recognition

Our experiment is concerned with the analysis of a multi-colour pattern, similar to those found on a number of product logos. In Figure 4(a) we have the original artwork of the multi-colour pattern. This was produced using Photoshop image processing software, running on a Macintosh computer, and a Kodak ColorEase laser printer. (When the image was used in the experiments, a narrow white border around this pattern was included in the cameras field of view). In Figure 4(b) we see the result of RGB colour separations. The top left image is red, top right is green and the bottom left is blue. Figure 4(c) represents the resultant colour scattergram. Notice that there are 7 blobs, corresponding to the six colour bands in Figure 4(a). The narrow white border mentioned in Figure 4(a) is represented by the central spot in the colour triangle.

Figure 4(d) is a pseudo-colour display of the colour scattergram. Notice that the dark regions in Figure 4(c) are more clearly visible here, as purple cloud-like structures. The colour scattergram is then thresholded, and all minor blobs removed, Figure 4(e). The remain major blobs have be assigned colours in an arbitrary manner. Figure 4(f) is the output of the programmable colour filter when trained on the blobs of Figure 4(e) (when the camera was again focused on the image in Figure 4(a)). Notice that the colour bands contains dark spots and are separated by dark streaks. Both of these effects are caused by the blobs in Figure 4(e) being too small. (The white border of Figure 4(a) is visible here as a yellow edge).

Figure 4(g) illustrates the result of colour generalisation on the blobs of the colour scattergram. The blobs have been greatly enlarged. When the PCF is now trained on these new enlarged blobs, the result PCF output is greatly improved, see Figure 4(h). Notice the absence of dark spots and streaks when compared to Figure 4(f). Recognising the colour pattern in Figure 4(a) as the anticipated logo can be performed quite easily by measuring the proportions of each of the component colours and noting their position and shape. Thus, colour pattern recognition reduces to logically combining the results of a series of very simple binary image analysis operations.

5. Discussion and Conclusions

The idea of using a look-up table to perform colour recognition is not new but has considerable appeal for such tasks as recognising (ripe) fruit on a tree, recognising resistor colour codes, tracing wiring, inspecting food products, cartons and pharmaceutical packaging. It lends itself to implementation in fast electronic hardware. Commercial equipment has been available for colour recognition, for some years. The colour scattergram is a useful tool, which allows the user to associate areas of the colour triangle with colours that he/she can recognise and name. Once a colour scattergram has been generated, the user can think about colour in convenient terms, using the concepts of blob position, shape and size. He/she can also apply a wide range of image processing operators to the colour triangle. This is possible because the colour triangle is an image, like any other. Colour recognition is achieved in real time, although the subsequent procedures for image analysis may not be. Four techniques for colour generalisation have been described and have been studied extensively by ourselves, using an interactive image processing system. As a result of their experience, the authors are convinced that the techniques described above provide a useful addition to the range of facilities available for recognising colours. It is possible to extend the range of colours recognised by the PCF to any extent desired. In *Procedure 2*, for example, the parameter N can be adjusted at will; if N is increased, the degree of generalisation will become higher. It is possible for a person, working with an interactive system, to experiment with the colour generalisation parameter, to obtain the best results for a given application. On the other hand, a program can be written which chooses a suitable value for N, according to some pre-defined criterion.

References

Batchelor BG (1991) *Image Processing in Prolog*. Springer Verlag, Berlin & New York.

Batchelor BG (1992) Colour Recognition in Prolog. *Proc. SPIE conf Machine Vision Applications, Architectures and Systems Integration* **SPIE 1823**: 294-305.

Batchelor BG, Whelan PF (1993) Generalisation Procedures for Colour Recognition. *Proc. SPIE conf. on Machine Vision Applications, Architectures and Systems Integration II* **SPIE 2064**: 36-46.

Batchelor BG, Whelan PF (1994) Real-time Colour Recognition in Symbolic Programming for Machine Vision Systems. In Press.

Chamberlain GJ, Chamberlain DG (1980) *Colour: Its Measurement, Computation and Application*. Heyden & Son Ltd., London, :18-45.

Intelligent Camera (1990), Image Inspection Ltd., Unit 7, First Quarter, Blenheim Road, Kingston, Surrey, KT19 9QN, U.K.

Optical Society of America (1953) *The Science of Colour*. Optical Society of America Committee on Colorimetry : 99-144.

Plummer AP (1991) Inspecting Coloured Objects Using Grey Scale Vision Systems. *Proc. SPIE conf. Machine Vision Systems Integration* **SPIE CR36**: 78-92.

Schettini R (1993) A segmentation algorithm for color images. *Pattern Recognition Letters* **14**: 499-506.

Umbaugh SE, Moss RH, Stoecker WV (1992) Automatic color segmentation algorithm with application to identification of skin tumor borders. *Computerized Medical Imaging and Graphics* **16**(3): 227-235. Refers to Part no. DT 2871, Data Translation Ltd.

Figure 1. Hardware structure of the Programmable Colour Filter, based upon RGB inputs to the LUT. This is the block diagram of the implementation of this technique in the Intelligent Camera (1990; Plummer 1991) and was used by the authors in the experiments reported here.

Figure 2. Proposed hardware structure of a Programmable Colour Filter, based upon HSI inputs to the LUT, using a real-time RGB/HSI converter chip (Umbaugh et al 1992).

Figure 3. Proposed hardware structure of a Programmable Colour Filter, based on the *xy* parameters used in defining the standard Chromaticity Diagram.

Figure 4. Analysing a multi-colour pattern. (a) Original artwork. (b) RGB colour separations. (c) Colour scattergram. (d) Pseudo-colour display of the colour scattergram. (e) Processed colour scattergram. (f) PCF output when trained on image (e). (g) Application of colour generalisation. (h) PCF output when trained on image (g). (i) Pseudo colour display of an intensity wedge, black on the left and white on the right.