Can we mimic the human ability for color-naming?

Carles Escrig Royo
Universitat Jaume I
Castelló de la Plana, País Valencià
al066522@uji.es

Abstract

An application that computes the approximate description of the perceived color of an image has been developed. Some experiments have been conducted to prove the validity of the descriptions obtained. Finally, some conclusions about the accuracy of the image analysis and the validity of the used QCD model have been draft.

1 Introduction

In order to make perception usable by consciousness, humans attach labels to remarkable features of stimulus. We bring out essential characteristics of things so that a classification can be performed and further abstract work can be developed. Colour is one of these essential characteristics. Any technology which goal is to reproduce some level of human intelligence, or plainly involves human-machine communication regarding the environment (for instance, assistive technology), must be able to successfully recognize and categorize colours.

A great effort has been done during recent years to mimic the color naming capabilities of humans. [Falomir *et al.*, unpublished] reviewed at least ten approaches to color-naming using different colour models. These authors have focused on reproducing a cognitive centered colour-naming, in order to achieve a description of colours as close as possible to that of humans. Their approach uses the HSL colour space and defines a model for Qualitative Colour Description (QCD) by dividing the former into intervals.

The goal of this project is to implement an easy to use application to test extensively the validity of the proposed QCD model. The architecture of the software should be in accordance with principles of the model: *"designed to be generally adaptable and kept as simple as possible*". Making it available for mobile devices would be a plus.

2 Development

2.1 The application

Overview

The application must take an image as its input and analyze it to extract the most common color. Then apply the QCD colour-naming model of [Falomir *et al.*, unpublished] to obtain a *human understandable* linguistic description of the color. The most common color and the *obtained description* are shown to the user and she is asked to do a review of the result. If the description is not right, user is asked (1) to input her own freely expressed description, and (2) to pick one of the descriptions allowed by this QCD model. Review of the result for a given image can be performed by any number of subjects. The information of the reviews is used to generate basic statistics about the adequateness of the obtained description.

Technology

An application that meets this requirements has been developed using *web* technologies. The most solid reason to choose this technology is its widespread use and ubiquity, comparable to that of a virtual machine like Java. Almost any machine has a web browser installed, and usually it is a fundamental tool kept up to date.

Commonly the web architecture involves a client web-browser and a server dispensing web pages. With the newest HTML version –five–, the standard has been augmented with new powerful features that enrich the client-side and can be used to operate it independently, like a classical desktop application. We have taken advantage of this features to build a web application that can be read both from a local filesystem or a web server and operated without access to the Internet.

User interface design is done by HTML/CSS, and the logic programming with ECMAScript (mostly known as JavaScript). We have reused libraries and code from other free projects in order to fasten the development:

- PureCSS[1], a set of responsible CSS modules. Helpful for mobile web development.
- Font awesome[2], a web font that provides scalable vector icons.
- jQuery[3], the most popular JavaScript library for client-side DOM manipulation.
- Underscore.js[4], a library that provides functional programming support.

[1] http://purecss.io/
[2] http://fortawesome.github.io/Font-Awesome/
[3] http://jquery.com/
[4] http://underscorejs.org/

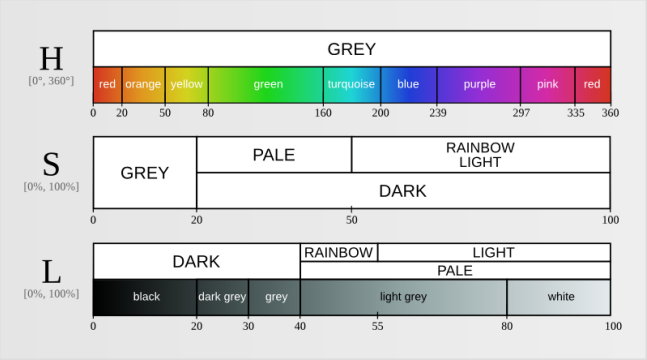

Figure 1: Ranges for color categories (uppercase) and some specific color descriptions (lowercase) of the QCD model.

- Backbone.js[5], a library with a RESTful JSON interface based on the model–view–presenter (MVP) application design paradigm.

On top of those components we build all the necessary parts for our application. In the next subsections we give some details about the work done.

Colours

The QCD model is defined for the HSL colour space, thus the image colours must be converted from RGB(A) to HSL color space. This can be done easily since HSL is just *"a rearrangement* [of] *the geometry of RGB in an attempt to be more intuitive and perceptually relevant..."* [6]. A fast search on Internet comes up with some conversion methods[7]. The QCD model defines valid ranges for each color description, figure 1 shows an overview. Detailed ranges can be found on the relevant paper. In brief, we defined data structures for both colour models, methods for converting between them, and a method to match a colour against a qualitative color description determined by at least 1 range of values for each dimension or channel.

All this has been done as a jQuery plug-in which exports some functions into the jQuery global object. Details can be seen in the source code file `app/js/jquery.color.js`:

- `Range(ini, end, opts)` Constructor function to build a range object.
- `RGBColor(mixed)` Constructor function to build an RGB color model object. Basically contains the values for each channel, validity ranges, and methods for printing and converting to other models.
- `HSLColor(mixed)` Constructor function to build an HSL color model object, analogous to previous function.
- `color_definitions` An array with all the qualitative color objects defined by the QCD model. Each qualitative color is an object of type `QColor`, which contains a label –the *description*– and an array of valid values for each channel of the color model.
- `perception(color)` A function that receives a HSLColor instance as the input parameter and outputs the label of the qualitative color (from `color_definitions`) which ranges validate against all the input color channels. Given a HSL colour, only one qualitative colour must match, otherwise an error is thrown.

Image analysis

When browser receives the image from the user, we use an HTML5 canvas element to draw the image an gain access to the pixel information. Pixel information is stored in unidimensional array where groups of 4 positions (channels R,G,B,A) represent a single pixel. The array is reorganized to explicitly express its 3 dimensions (alpha is discarted) and then apply the k-means clustering algorithm to find out the k centroids and the pixel assignments. Parameter k is configurable in the user interface and represents the number clusters to ultimately be found on the image. The adequacy of this parameter depends on each image, but a safe default of 4 has been established. We use the Euclidean distance as a measure of proximity. The implementation of the k-means algorithm itself comes from project Figue[8]; we felt it was better to rely on optimized and well tested code. Moreover, all the work is done on a downscaled version of the original image in order to speed up the clustering. We rely on the native scaling algorithm present in the web browser. It is very good in current Mozilla Firefox 22, but might be different for other vendors. The size for the downscaling might be configured as well.

Methods regarding the analysis of the image have been packaged onto a jQuery plug-in (source is at `app/js/jquery.canvasimage.js`). The plugin adds a function `createAnalyzer(resize, k)` to the jQuery prototype, so that this function might be called from any jQuery wrapped DOM element. Nevertheless calling this function only makes sense when the wrapped DOM element is an `img` tag. In that case, the function will return a new `CanvasImage` object. This object emits two events:

- `'load'` when the image data has been loaded and resized into the internal canvas, and the object is ready to perform an analysis. When this event triggers, it is save to call `analyze()` on the object.
- `'analyzed'` when k-means has ended. Once this event has triggered, results of the k-means algorithm can be found on property `result` of the object.

Property `result` contains the original k-means result (`assignments` and `centroids`) and an additional `rank` property. `rank` is an array of perceptions sorted by relevance. First object in the array is the most relevant perception, with the percentage of pixels assigned to this perception and the centroids that where classified as belonging to this perception, also in descending relevance.

The result of an image analysis is presented to the user side by side to the analyzed image. It consists of the most relevant perception label (for instance "Dark red"), shown over a box filled with the color of the winner centroid (the one that got the most number of assignments among all the centroids that

[5] http://backbonejs.org/

[6] *https://en.wikipedia.org/wiki/HSL_and_HSV*

[7] For instance, http://www.rapidtables.com/convert/color/hsl-to-rgb.htm#doc (07/11/2013)

[8] http://code.google.com/p/figue/

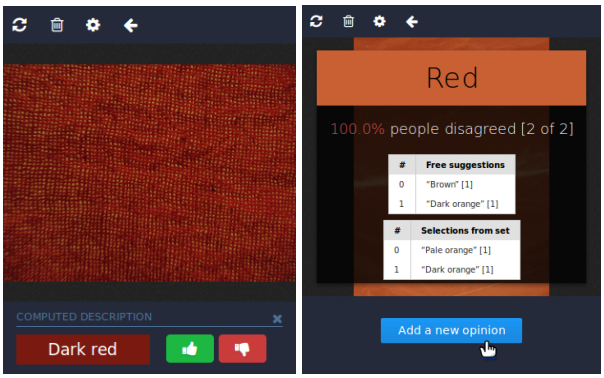

Figure 2: (left) Result of an image analysis presented to the user. (right) Result of image analysis alongside statistics about acceptance, and rank of user free suggestions and user selections from the QCD model available options.

fullfill this very same perception label). Figure 2 (left) illustrates this.

Our image analysis presented here is very simple and does not account for color constancy or spatial relations among colors. First reason for this is the size of this project, and second, the expected target of images is going to be fairly plain (closeups of fabrics). To adapt the application to be used with any kind of image, a correction method described by [Mojsilovic, 2005] on section IV could be applied, and then obtain a more real color composition of the image. This might be important if images are coming from low quality mobile pictures (for instance, taken by a visually impaired person).

Layout

The application layout is tied up and controlled by the *backbone.js* framework. The home page (or main screen) lists all the images in the database in the form of a grid (figure 4 on the next page), ordered by descending creation time. Some very minimal statistics are shown. Management of each individual image is done by clicking on it. All the information of an image (color analysis, user reviews, etc.) is managed as a backbone model. All models are contained into a backbone collection. The collection is permanently saved in the local storage of the web browser.

2.2 The tests

The design of tests has been very simple. It consisted on the selection of 37 images, each one apparently fitting on one of the 37 color names considered by the QCD model. Each image analysis has been repeated with different parameters, to observe the variation on the results, and avoid saving an uncommon result (a "hand-operated" cometee of machines). Finally results have been reviewed by two independent subjects.

3 Experiments

The application successfully has output *one* perception for all tested images. In general this proves the correct operation

Figure 3: Six of the thirteen not accepted perceptions.

of the analysis, and the correct definition of ranges for each qualitative color.

Table 1 summarizes the experiment results. 16 perceptions got a 100% of acceptance among the reviewers. 8 perceptions got a 50% of acceptance. Finally 13 perceptions got a 0% of acceptance. Most of the time perception was accepted by humans, but why was it rejected 35% of the time?

Eventhough any conclusion based on tests done to only two individuals is far from getting any credibility, in our opinion flaws on the QCD model have been revealed. Figure 3 shows 6 representative cases where computed perception was not accepted by the reviewers.

We consider (a), (c), and (d) to fall into a common case of a "fuzzy frontier" problem in the hue scale. It is quite difficult to solve, and divergences may arise due to cultural differences, display conditions, and subjective perception. It is the less serious of the errors.

We would call case (a) and (e) plainly erroneus, and a bit alarming. Case (a) `hsl(130, 88%, 32%)` is an example of how values in the saturation and lightness scales can modify the established intervals of each other. Based on this

User's perception acceptance	Num. of images
100%	16 (43%)
50%	8 (21%)
0%	13 (35%)

Table 1: Summary of calculated perceptions acceptance.

case we could argue that higher values of saturation influence downward the DARK - RAINBOW lightness frontier. Case (e) `hsl(300, 12%, 48%)` is a painfull example of how same values of saturation and lightness are not enough evidence to classify a color in the grayscale category. For some values of hue, for instance `200`, we would consider the grayscale category acceptable. In this case, whether we are too much influenced by our own perception, or the values of the model are wrong.

We consider case (f) to be more of an analysis error than a model error. In this case most of the image is very dark, an just obtaining the most common color is not enough to find out the right perception. The reason lies on the fact that the color of the few light areas influences greatly the perception of the whole picture, because humans rely on the light areas to infer which the dark color might be. In this case, a previous correction of the image might pose a solution.

Finally, some subjects might think that some variations of a pure rainbow color should deserve their own specific name, like case (d) where the "dark red" is suggested to be callled "brown".

4 Conclusions and further work

The few experiments done reveal some failures on the analysis of the image and the color description approximation defined by the QCD model. Further work on the analysis should involve previous treatment and correction of the image. We feel the QCD model is a bit harsh trying to keep color-naming as simple as possible. Some color names could be added. Also it would be nice to give a try to a more dynamic interval system, in which frontiers among categories would be influenced by values of the other scales. Anyhow it is crystal clear that more testing should be done to draw any solid conclusion.

Code

The web application can be executed loading the `app/index.html` file in your web browser.

Thanks

Project title, proposal and ideas are a direct suggestion of Lledo Museros, from Computer Science and Artificial Intelligence department at University Jaume I. This work has been done as a final course project of *Cognitive processes* (SIE019), of Master in Intelligent Systems.

References

[Falomir *et al.*, unpublished] Zoe Falomir, Lledo Museros, and Luis Gonzalez-Abril. A model for cognitive color description and comparison using conceptual neighbourhood diagrams. *IEEE Transactions on Image Processings*, unpublished.

[Mojsilovic, 2005] Aleksandra Mojsilovic. A computational model for color naming and describing color composition of images. *IEEE Transactions on Image Processing*, 14(5):690–699, 2005.

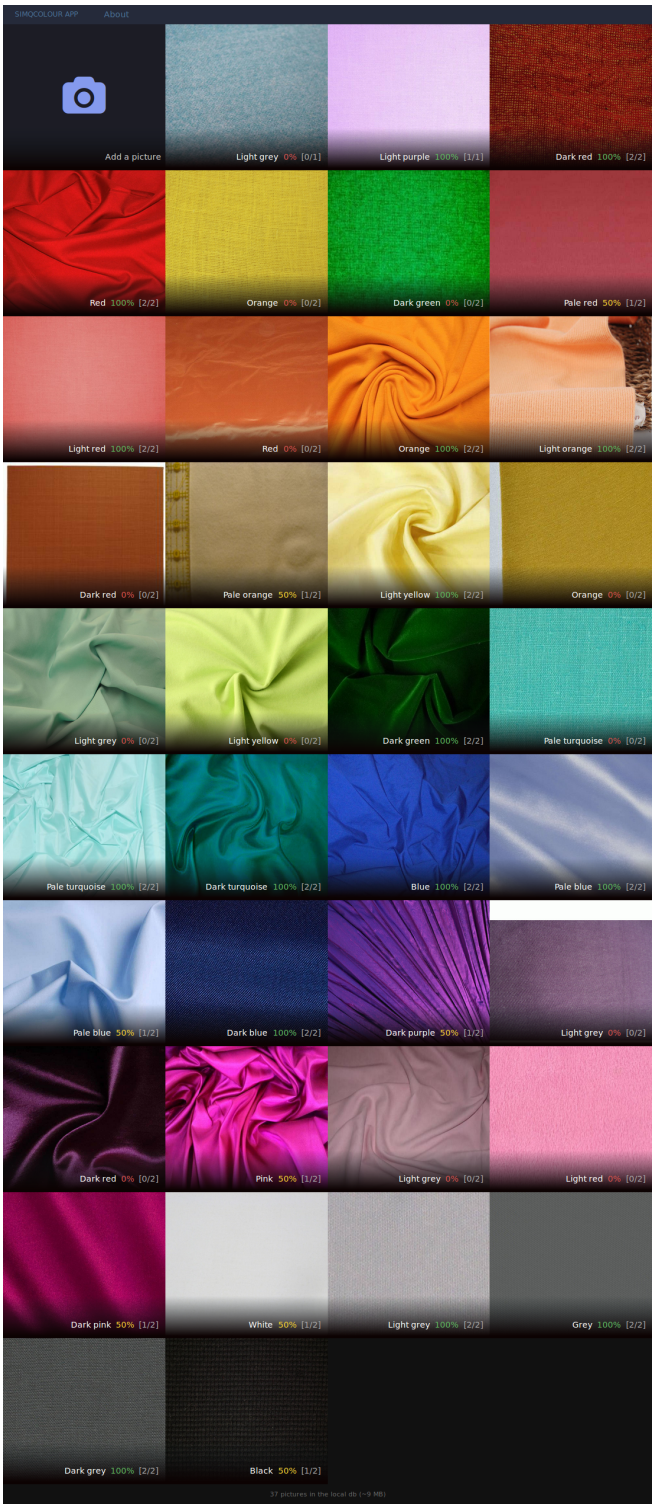

Figure 4: Overview of the 37 tested images.