Intelligent system for color difference recognition in an industrial color copy process

Intelligent Systems, Department of Information Engineering, University of Pisa

Federico Rossi

2017/2018

1 Part I

1.1 Introduction

In the first part of this work we'll need to:

Step 1 Generate reasonable noisy spectra from the master spectra, convert them into CIE $L^*a^*b^*$ format and, for now, compute the target differences between a master and the corespondent copy ΔE using the following formula (by means of `de(LAB1,LAB2)` function included in the `optprop` MATLAB library):

$$\Delta E^* = \sqrt{(L_1^* - L_2^*)^2 + (a_1^* - a_2^*)^2 + (b_1^* - b_2^*)^2} \tag{1}$$

Step 2 Take the output of the previous step - that is a matrix of dimension $[421 * 2; 1269 * n]$ where n is the number of copies we have generated for each master sample - and extract features from it (to reduce reasonably the number of `rows` into a set that best represents the original spectrum).

Step 3 Train a shallow neural network to solve a *function fitting problem* with the previous subsampled spectra as inputs and the differences computed at step 1. as targets.

1.2 Step 1 - Generation of copies

To generate copies from a master patch we must introduce some kind of noise to the spectrum, in order to slightly change the final color.

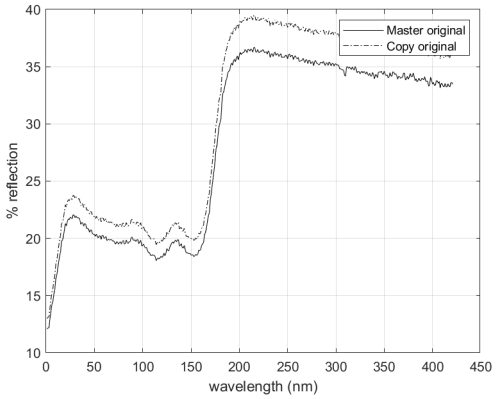

Figure 1: Master spectrum compared to its copy spectrum

To generate a random, yet repeatable, noise the following snippet of code has been used in MATLAB:

```
rng(12); %set the seed
for i=1:numcopies
    noise=random('unif',1,1.13);
    copy=spectra*noise;
end
```

Parameters of uniform distributions are chosen empirically to obtain a maximum difference of 5.

At this point, if we convert the noisy spectrum back to the CIE $L^*a^*b^*$ format using `roo2lab` function from `optprop`, we can compute the target difference between the master patch and the copy patch using the `de` function.

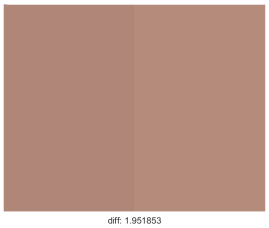

Figure 2: Master color compared to its copy

Example An example of a master spectrum and the noisy copy is shown in Figure 1. Figure 2 shows the master color and its copy color.

Output Then the output of this step is composed as follow:

- `masterCopyExpectedDiff` a $[1; 1269 * n]$ matrix containing the target differences computed as before.
- `masterCopyPairs` a $[421 * 2; 1269 * n]$ matrix containing, for each column, the pair:

$$< \text{master}[1, 421]; \text{copy}[422, 842] > \quad (2)$$

1.3 Step 2 - Feature extraction and selection

The output of the previous step is clearly too big in terms of a single sample to be feed to a shallow neural network. Therefore we need to reduce the quantity of information provided per sample, maintaining quite the same quality.

The idea is to divide the spectrum in k wavelength ranges of size M, and internally aggregate the information. For our purposes, an aggregation function used is shown in Equation 3. Using this formula, as a side, we also manage to remove some noise from original data.

$$m_i = \frac{1}{M} \sum_{j=i-\frac{M}{2}}^{i+\frac{M}{2}} s_j, i = \frac{(2l+1) * M}{2}, \forall l \in [0, \frac{S}{M} - 1] \quad (3)$$

Where s_j is the original signal's sample, m_i is the result of the aggregation window and S is the number of original samples.

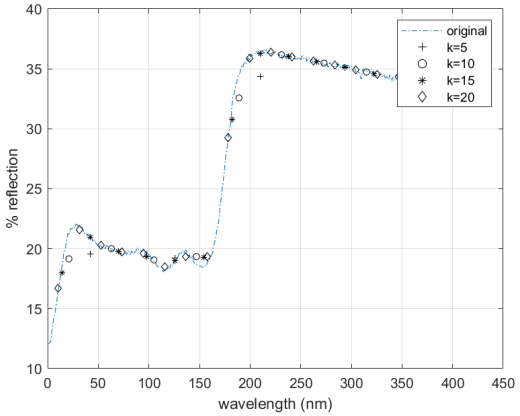

Figure 3: Master spectrum compared to the various aggregate versions

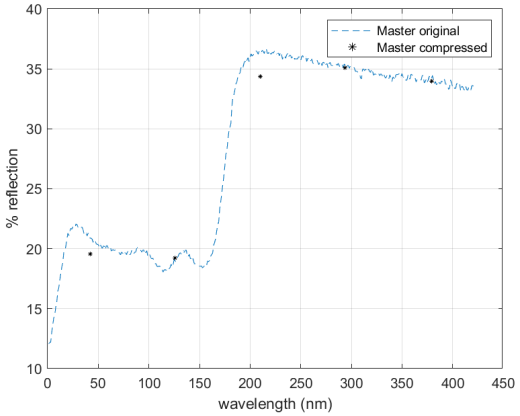

Figure 4: Master spectrum compared to the aggregate version with k=5

How to choose k If k is too small we are loosing too much information from the aggregation; on the other hand, if we sub-sample the spectrum with an high number of ranges we are not compressing the information enough. Figure 3 shows how aggregation varies increasing k.

Trying some different parameters for k, the ones that are a good trade-off between information compression and information loss are $k \in [5, 20]$. From now on, we will go for the higher aggregation and see if performances are good enough.

Recalling the previous example, the aggregation result for $k = 5$ is shown in 4, thus obtaining 10 features for each pair.

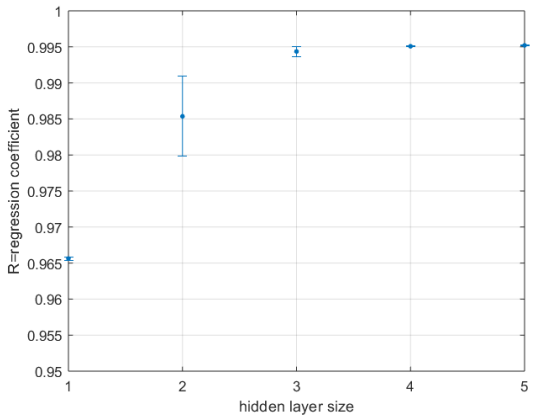

Figure 5: Values of R vs number of neurons

Selection We set up `sequentialfs` from MATLAB to extract the 4 best features that approximate Formula 1, using the `mse` from a fitting operation as choice criterion.

Output The output from this step is:

- `masterCopyPairsMean` a $[4; 1269 * n]$ containing the selected features.

1.4 Step 3 - Fitting network

Now that we have our *inputs* and *targets* we can feed them into a shallow neural network for function fitting.

We create the network in MATLAB using the `fitnet` function, with size of hidden layer equal to n:

```
hiddenLayerSize=n
net=fitnet(hiddenLayerSize);
net.divideParam.trainRatio=70/100;
net.divideParam.valRatio=15/100;
net.divideParam.testRatio=15/100;
net=train(net,inputs,targets);
```

Performance At the end of the training phase, we measure the performances of the neural network in terms of *mean squared error* and *regression coefficient*.

Hidden layer size	Execution time (s)
1	1 ± 0
2	2 ± 0.1
3	3 ± 0.2
4	3.2 ± 0.2
5	3.4 ± 0.3

Table 1: Profiling results from MATLAB profile tool (10 reps, 95% CI)

Evaluation Performance evaluation consists in repeating the training a certain number of times (10 in our case, since performance metrics are not too jittery).

The randomness of different repetitions is introduced by the `dividerand` option used by MATLAB to subdivide the provide dataset into *training,validation* and *testing* folds.

How to choose n High values of n will require more training samples and time but, at the same time, will produce more accurate results in terms of *regression coefficient*. Lower values will visibly reduce the amount of time and samples required to train the network but will produce less accurate results. Figure 5 shows the behavior of R compared to number of neuron in hidden layer (values reported are the average of 10 repetitions with a 95% confidence interval).

Training profiling Moreover profiling the training script with the MATLAB tool produced results in Table 1.

So, from previous considerations, the choice of 4 hidden neurons in the hidden layer of the network is a good trade-off between network performances and training time. Results are the following (considered 95% confidence interval):

$$\text{regression} = 0.995076 \pm 0.000057 \tag{4}$$

$$\text{mse} = 0.016 \pm 0.0001 \tag{5}$$

Figure 6 shows one repetition's regression plot.

As we can see from Equation 4, the regression coefficient is very close to 1, thus the fitting is good.

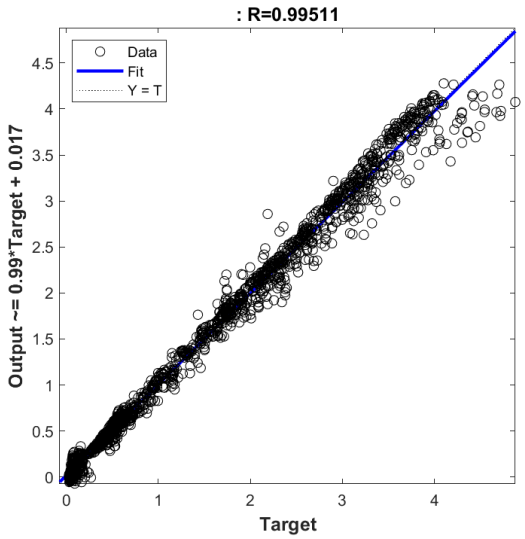

Figure 6: Regression plot

Note The *mse*'s order of magnitude must be compared to a meaningful granularity from the target sets, that in our case is around 0.1.

2 Part II

In the second part we need to take into account some imprecisions that might occur in Equation 1. We'll make use of another color space, called CIE $L^*c^*h^*$ - that, in this case, is a cylindric coordinates system.

In particular, in the second part, we'll need to:

- Find out the areas of the CIE $L^*a^*b^*$ sphere that lead to imprecisions in the formula w.r.t an external observer.
- Express those areas by means of a set of *fuzzy rules* on *fuzzy input sets* that we'll explain later on.
- Adjust the output of Equation 1 using a *fuzzy system* that works on previous defined *fuzzy rules* and use the new adjusted values as the *targets* of the neural network built in Subsection 1.4.

2.1 Step 1 - ΔE imprecisions

Figure 7: Dark red and dark blue patches

Dark colors First set of imprecisions occurs when we have colors which are very dark. For example, in Figure 7 are reported, respectively, a dark red patch and a dark blue one. If we use 1 to compute color difference we get $\Delta E^* \sim 10$ but, if we apply the definition for color difference from CIE, that pair of colors should have a difference between 0 and 1.5, so the output of the formula must be adjusted in this direction.[1]

The blue-violet area In this area the reported ΔE^* is typically smaller than perceived difference, so we could think of increasing its value.[2].

The yellow area In this area instead we experience the opposite behavior - that is, the perceived difference is typically smaller than the one reported by ΔE^* (also in Note 2).

Unsaturated colors Another area is defined by colors that are unsaturated and not too much dark. Here the formula will compress our perceived difference, so we'll need to adjust it by increments.

2.2 Step 2 - Fuzzy sets and rules

2.2.1 Fuzzy sets

We want to model a *linguistic variable* for each L,c,h and original ΔE^* *crisp set* that represents concepts exploited in defining imprecision areas in Sub-section 2.1. Membership functions for sets are shown is following figures:

- *Luminosity* figure 9. Range is $[0, 100]$ according to CIE standards.
- *Hue* figure 10. Range is $[0°, 360°]$ according to CIE standards.
- *Chroma* figure 11. Chroma range depends on L^* value, and it is known to span from 0 to $C^*_{max} = 127$ at $L^* = 50$. So the maximum saturation may happen slightly before that value for values of L^* far from 50. So we may think to approximate the the relation between L^* and C^* as an ellipsis shape:
$$\frac{c^{*2}}{C^{*2}_{max}} + \frac{l^2}{50^2} = 1$$
(where $l = |L^* - 50|$). Then we compute the $C^{*'}_{max}$ as in Figure 8. Then we scale the c^* value as
$$c^*_{\%} = 100 * \frac{c^*}{C^{*'}_{max}}$$
Where $C^*_{max} = 127$. Therefore, the range we assume for c^* is $[0, 100]$.
- *Original* ΔE^* figure 12. Range is $[0, 8]$ according both to project specifications and empirical outcomes.

2.2.2 Fuzzy rules

According to considerations in Sub-section 2.1,some of the rules inserted into the *rule base* are shown in Table 2

[1] Project specifications

[2] https://opentextbc.ca/graphicdesign/chapter/4-4-lab-colour-space-and-delta-e-measurements/

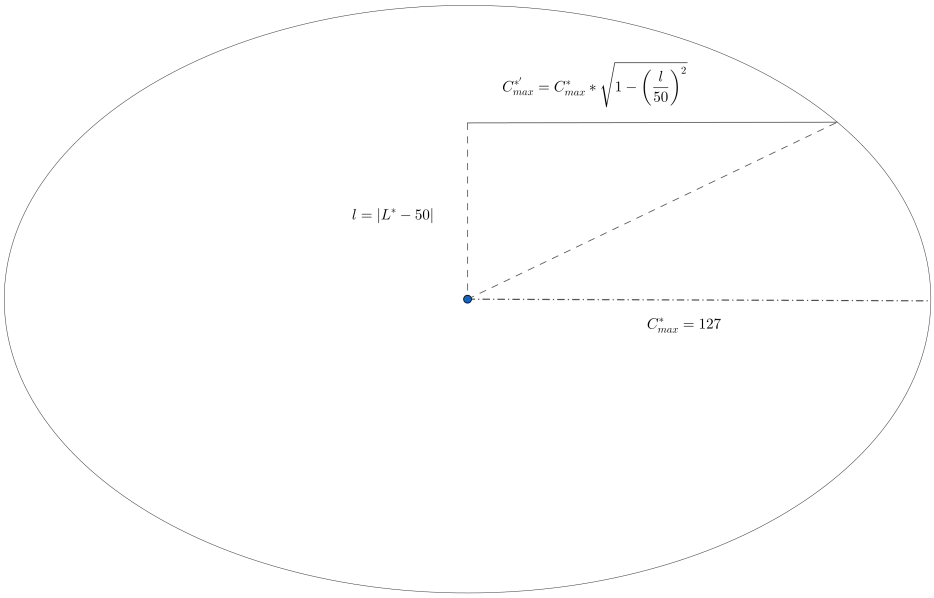

Figure 8: $L^*c^*h^*$ sphere approximation

L	c	h	ΔE^*_{orig}	ΔE^*_{adj}
dark	-	-	-	no_diff
not dark	saturated	yellow	unexpdiff	expdiff
not dark	saturated	yellow	cleardiff	unexpdiff
not dark	saturated	blue	expdiff	unexpdiff
not dark	saturated	blue	unexpdiff	cleardiff
not dark	saturated	violet	expdiff	unexpdiff
not dark	saturated	violet	unexpdiff	cleardiff
not dark	unsaturated	-	expdiff	unexpdiff
not dark	unsaturated	-	unexpdiff	cleardiff

Table 2: Main rules applied in the mamdani fuzzy inference system

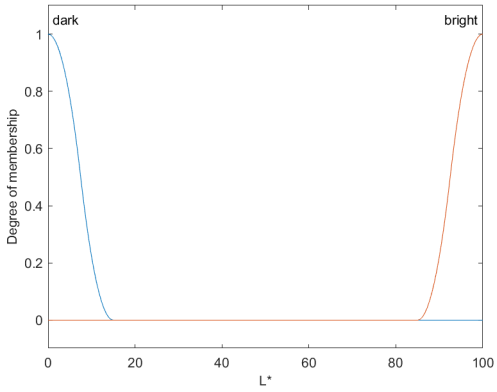

Figure 9: Luminosity membership function

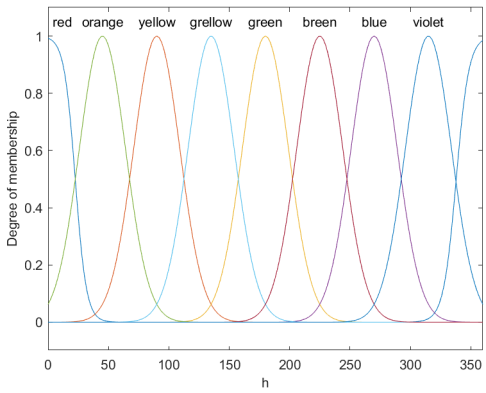

Figure 10: Hue membership function

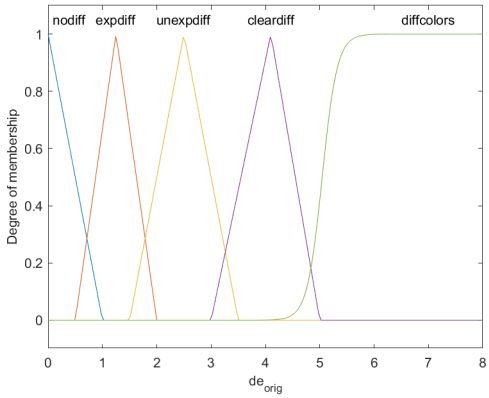

Figure 12: ΔE^* membership function both for input and output

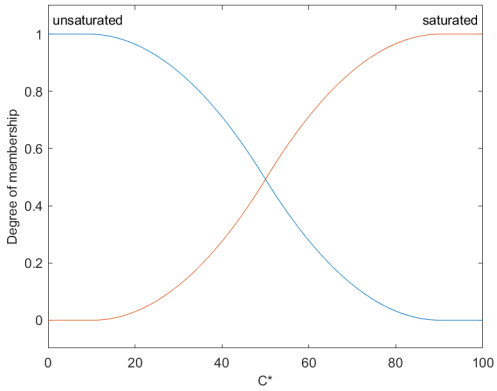

Figure 11: Chroma membership function

2.3 Step III - Fitting network with new data

At this point of the work we have again inputs represented by pairs of masters and copies and targets represented by corrected ΔE^*_{adj}. We submit these data to the network defined in Sub-section 1.4 for training.

Performance metrics over 10 repetitions are:

$$R = 0.984801 \pm 0.002223 \tag{6}$$

$$\text{mse} = 0.051570 \pm 0.007305 \tag{7}$$

NOTE We can obtain a regression coefficient similar to the first part of the project is we reasonably increase the number of neurons (~ 50) and as a consequence, the time required to train the network. Therefore we obtain, on 10 repetitions of training these testing performances, that are quite comparable to the ones obtained in the first part, but now with differences recognized closer to the perceived ones.

$$R = 0.993290 \pm 0.001360 \tag{8}$$

$$\text{mse} = 0.018770 \pm 0.003756 \tag{9}$$

3 Conclusion and examples

At the end we apply the whole system to a pair of selected master and copy patches (not provided during training phase), one belonging to the violet area,one belonging to the unsaturated area and one belonging to the yellow area

Figure 13 shows how the original difference was too way different from the perceived one, reporting quite no differences between the two samples while having a visible difference.

Figure 14 shows the same concept but with saturated colors. Note that this pair of samples behaves in different ways according to the view angle. However, for sure, the perceived difference is not the one reported by the original formula.

Figure 15 shows the opposite concept applied to the yellow area. As we can see, the reported difference was too higher for that pairs of colors, reporting two completely different colors instead of a clear difference between them.

Figure 13: Violet area difference and correction

Figure 14: Unsaturated area difference and correction

Figure 15: Yellow area difference and correction