MAIA - MedicAl Imaging and Applications

Advanced Image Analysis

Lectures on Advanced Color Image Processing

B3 – Color Spaces - Applications

March/2018

Adrian Galdran - Post-Doctoral Researcher
Biomedical Imaging Lab – INESC TEC Porto (Portugal)
http://bioimglab.inesctec.pt/

Color Spaces - Applications

- Color Spaces – Applications
  1. Color Spaces in OpenCV
  2. Color Transfer in OpenCV:
  3. Opponent Color Spaces: Colorfulness
  4. Color Clustering
  5. Color+Location Clustering: Superpixels

Color Spaces in OpenCV

Remember we discussed on Thursday that RGB was not a **perceptually uniform** color representation.

In addition, it distributes the brightness and color information among the three channels, making it difficult to manipulate color alone, for instance.

We mentioned several color spaces that do not suffer from this inconveniencies. Today's lecture will be more practical, let us see how can we perform color conversions in OpenCV.

Open color_spaces.ipynb

Color Transfer

Color Transfer

There are several ways of transferring the color distribution of a given image to another one. One of them is histogram specification.

However, these techniques typically involve heavy computations to match the histogram of the source image to that of the target image.

Color Transfer

There are simpler ways for transferring colors between images. We are going to analize this work:

Applied Perception

Color Transfer between Images

Erik Reinhard, Michael Ashikhmin, Bruce Gooch, and Peter Shirley
University of Utah

What is interesting about Reinhard's technique is that it operates in the **Lab color space**.

Color Transfer

Color Transfer

- **Step 1**: Input a source and a target image. The source image contains the color distribution that we want the target image to mimic.
- **Step 2**: Convert both the source and the target image to the Lab color space. The Lab color space models perceptual uniformity, where a small change in an amount of color value should also produce a relatively equal change in color importance. The Lab color space does a substantially better job mimicking how humans interpret color than the standard RGB color space, and thus it works very well for color transfer.
- **Step 3**: Split the channels for both the source and target.
- **Step 4**: Compute the mean and standard deviation of each of the Lab channels for the source and target images.

Color Transfer

- **Step 5**: Subtract the mean of the Lab channels of the target image from target channels.
- **Step 6**: Scale the target channels by the ratio of the standard deviation of the source divided by the standard deviation of the target.
- **Step 7**: Add in the means of the Lab channels for the source.
- **Step 8**: Merge the channels back together.
- **Step 9**: Convert back to RGB color space from the Lab space.

What this paper showed was that simply transferring basic Lab statistics of one image to another is enough. Let us see it.

Open color_transfer.ipynb

Opponent Color Spaces: Colorfulness

The RGB color space is said to be an **additive** color space, because colors are obtained by adding a certain amount of R, G, and B components.

A different color representation is achieved by considering as primaries the [**red-green**] and the [**blue-yellow channel**]:

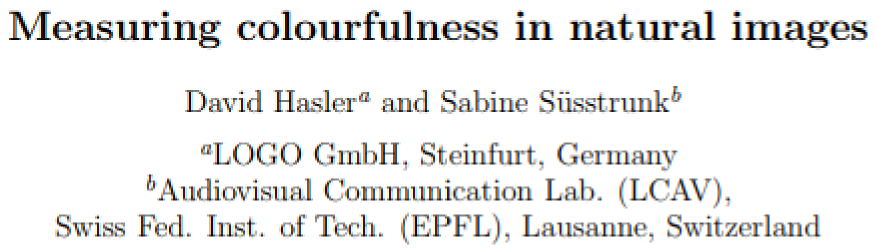
$$Opp_1 = Luminance$$

$$Opp_2 = rg = R - G$$

$$Opp_3 = yb = \frac{1}{2}(R + G) - B$$

Opponent Color Spaces: Colorfulness

We are going to apply opponent colors to measure the degree of colorfulness of an image, following this paper:



Measuring colourfulness in natural images

David Hasler[a] and Sabine Süsstrunk[b]

[a]LOGO GmbH, Steinfurt, Germany
[b]Audiovisual Communication Lab. (LCAV),
Swiss Fed. Inst. of Tech. (EPFL), Lausanne, Switzerland

Here, the authors asked a set of observers to describe how colorful several images were. Through a series of experimental calculations, they derived a simple metric that *correlated* with the results of the viewers.
They found through these experiments that a simple opponent color space representation along with the mean and standard deviations of these values correlates to 95.3% of the survey data.

Opponent Color Spaces: Colorfulness

In their paper, Hasler and Süsstrunk first asked 20 non-expert participants to rate images on a 1-7 scale of colorfulness. This survey was conducted on a set of 84 images. The scale values were:

- Not colorful
- Slightly colorful
- Moderately colorful
- Averagely colorful
- Quite colorful
- Highly colorful
- Extremely colorful

In order to set a baseline, the authors provided the participants with 4 example images and their corresponding colorfulness value from 1-7.

$$\sigma_{rg,yb} = \sqrt{\sigma_{rg}^2 + \sigma_{yb}^2}$$

$$\mu_{rg,yb} = \sqrt{\mu_{rg}^2 + \mu_{yb}^2}$$

Open colorfulness.ipynb

$$C = \sigma_{rg,yb} + 0.3 \cdot \mu_{rg,yb}$$

Color Transfer

Color Clustering

We are going to use the K-means is a clustering algorithm for this purpose. In K-means, the goal is to partition **n data points** into **k clusters**. Each of the **n** data points will be assigned to a cluster with the nearest mean. The mean of each cluster is called its “centroid” or “center”.

Applying K-means yields *k* separate clusters of the original *n* data points. Data points inside a particular cluster are considered to be “more similar” to each other than data points that belong to other clusters. In our case, our data points will be the RGB intensities of our images.

Open color_clustering.ipynb

SLIC superpixels

SLIC stands for Simple Linear Iterative Clustering, quite a descriptive name!

Simple to use: only one parameter (number of wanted superpixels)

Perceptually meaningful: it clusters pixels in the xy-Lab color space

Memory-efficient: The complexity of SLIC is linear in the number of pixels in the image, O(n)

Let's see the algorithm:

$$d_c = \sqrt{(l_j - l_i)^2 + (a_j - a_i)^2 + (b_j - b_i)^2}$$

$$d_s = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2},$$

$$D' = \sqrt{\left(\frac{d_c}{N_c}\right)^2 + \left(\frac{d_s}{N_s}\right)^2}.$$

Open slic.ipynb

```
Algorithm 1. SLIC superpixel segmentation
  /* Initialization */
  Initialize cluster centers $C_k = [l_k, a_k, b_k, x_k, y_k]^T$ by sampling
  pixels at regular grid steps $S$.
  Move cluster centers to the lowest gradient position in a $3 \times 3$
  neighborhood.
  Set label $l(i) = -1$ for each pixel $i$.
  Set distance $d(i) = \infty$ for each pixel $i$.

  repeat
    /* Assignment */
    for each cluster center $C_k$ do
        for each pixel $i$ in a $2S \times 2S$ region around $C_k$ do
          Compute the distance $D$ between $C_k$ and $i$.
          if $D < d(i)$ then
            set $d(i) = D$
            set $l(i) = k$
          end if
      end for
    end for
    /* Update */
    Compute new cluster centers.
    Compute residual error $E$.
  until $E \leq$ threshold
```