SHINE_color: controlling low-level properties of colorful images

Rodrigo Dal Ben[1]

1 Concordia University

Summary

Many experiments in Psychology and Cognitive Neuroscience require precise control of visual stimuli properties, either to precise manipulate variables of interest or to avoid experimental confounds. One way to control low-level properties of images is to use the SHINE toolbox (Willenbockel et al., 2010), which has been used and cited hundreds of times across a wide range of research topics. Here we describe an adaptation of the SHINE toolbox for controlling low-level properties of colorful images, dubbed SHINE_color.

Statement of need

One powerful way to control low-level properties of visual experimental stimuli is to use the SHINE toolbox for MATLAB (Willenbockel et al., 2010). This toolbox contains a set of functions that allows users to precisely specify luminance and contrast, histogram, and Fourier amplitude spectra of visual stimuli. These parametric manipulations minimize potential low-level confounds when investigating higher-level processes (e.g., cognitive effort, recognition). However, SHINE only works with greyscale images. Whereas this serves well to many research purposes (e.g., Lawson et al., 2017; Rodger et al., 2015), other research goals might benefit from colorful images (e.g., Cheng et al., 2019; Hepach & Westermann, 2016; Zhang et al., 2019). Here, we describe the SHINE_color, an adaptation of SHINE that allow users to perform all operations from SHINE toolbox on colorful images. Such adaptation can be useful for a wide array of research topics that rely on the precise low-level properties of colorful visual stimuli, such as developmental pupillometry studies (Hepach & Westermann, 2016; Sirois & Brisson, 2014; Tsukahara & Engle, 2020; Zhang et al., 2019; Zhao et al., 2019).

Implementation

The SHINE_color toolbox works in an intuitive way (Figure 1; complete flowchart available at OSF). Once called in the command window of MATLAB, by typing SHINE_color, the script guides the user through a series of questions that specify the input files characteristics (either a set of images or a video), the color space to be used (i.e., HSV or CIELab), and the SHINE operations to be performed (luminance, histogram, Fourier amplitude spectra specification Figure 1). We strongly recommend referring to Willenbockel and colleagues (2010) for a detailed description of each operation. Following, the input RGB images are transformed to either HSV or CIELab color space (see Ruedeerat (2018) for a similar approach that normalize RGB images directly). If a video is provided, its frames are first extracted, then they are transformed to either HSV or CIELab color space. From RGB images, the HSV color space creates Hue, Saturation, and Value (luminance) channels; likewise the CIELab color space creates lightness (l*), red and green (a*), and blue and yellow (b*) channels. Following this transformation, Hue and Saturation or a* and b* (HSV or CIELab respectively) channels are held in memory and are not manipulated. On the other hand, the luminance channel (either Value or l* channel) is rescaled (from 0 to 1 or 0 to 100, HSV and CIELab respectively) to grayscale range (from 0 to 255). Then, all operations from SHINE (Table 1) can be performed in the scaled luminance channel. For instance, Figure 2 displays an example of histogram matching. Following, the luminance channel is rescaled to its original range (from 0 to 1 to HSV or 0 to 100 to CIELab) and is combined with its original Hue and Saturation or a* and b* channels (HSV or CIELab respectively). These HSV or CIELab images are then transformed back to RGB images. For videos, there is an additional step in which frames are recombined back into a video.

Finally, for both images and videos, the mean and standard deviation of the luminance channel before and after manipulations are calculated for all images or frames. These statistics provide a quick summary of the modifications in luminance and are stored in a `.txt` file in the folder `SHINE_color_OUTPUT/DIAGNOSTICS`. In addition, for images (but not for videos), users can choose to plot diagnostic plots (luminance histogram, spatial frequency, or spectrum) to compare luminance properties before and after manipulations.

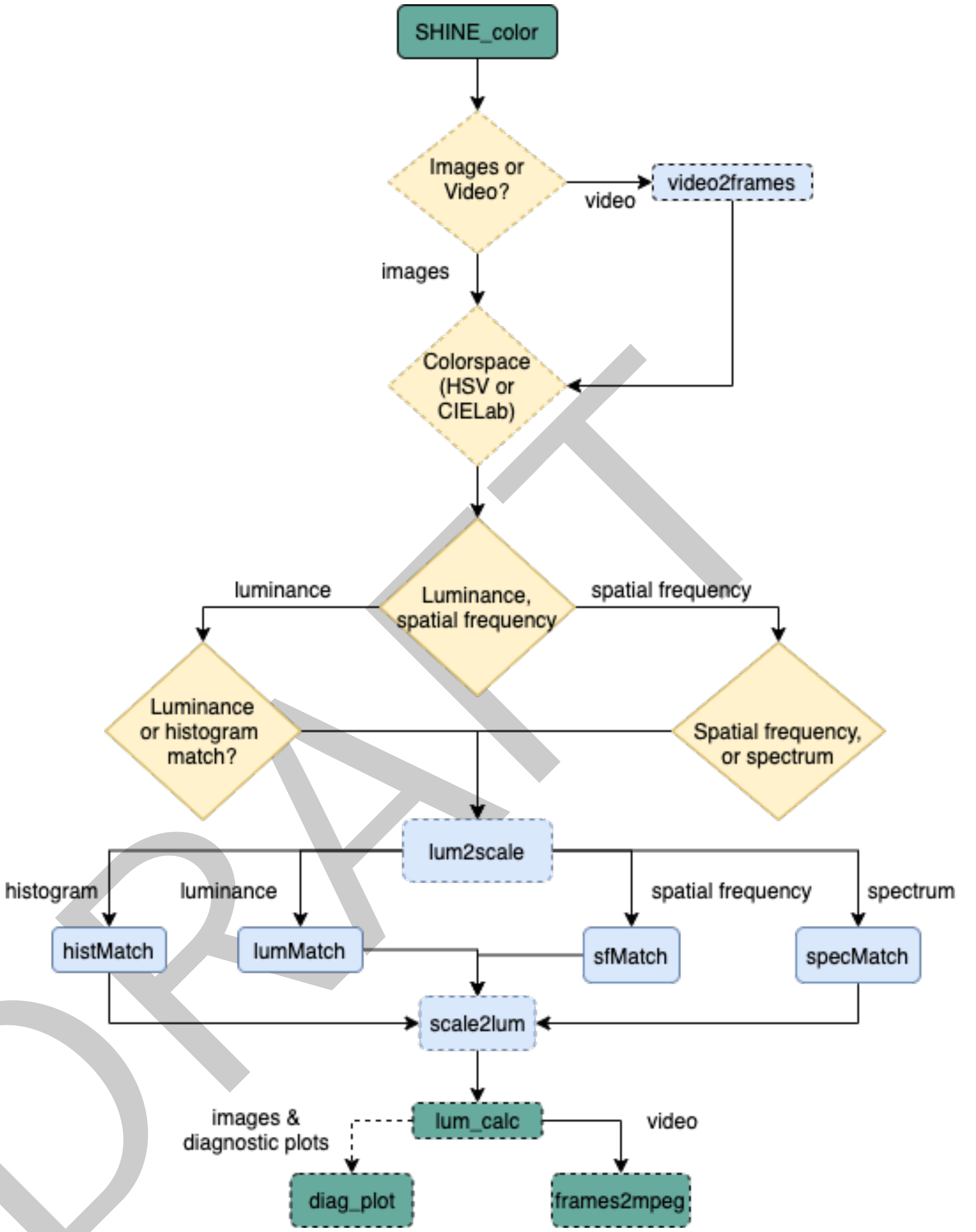

Figure 1: SHINE_color condensed flowchart. Functions (rounded rectangle) and decisions (diamonds) with dashed borders are introduced by SHINE_color (e.g., `video2frames`, `lum2scale`, `scale2lum`, `lum_calc`, `diag_plot`, `frames2mpeg`). They allow SHINE operations to be performed on colorful images.

Table 1. Overview of the functions from SHINE_color. Most functions come from the SHINE toolbox, and their descriptions are also available on Willenbockel et al. (2010). Single stars (*) denotes functions that have been adapted from SHINE, double stars (**) indicates new functions from SHINE_color. Functions are listed in alphabetical order.

Function	Description
avgHist	computes average histogram

Function	Description
diag_plot**	creates diagnostics plots for manipulated images (luminance histogram, sfPlot, spectrumPlot)
frames2mpeg**	creates a mpeg (.mp4) video from a sequence of frames
getAllFilesInFolder**	read all frames from a folder
getRMSE	computes root mean square error
hist2list	transforms histogram into a sorted (darker-to-brigther) list
histMatch	exact histogram matching across images
imstats	computes image statistics
lum2scale**	converts RGB to HSV or CIELab color spaces, extracts the luminance channel, and scale it to grayscale range
lum_calc**	computes the luminance channel average and standard deviation
lumMatch	scales mean luminance and contrast
match	histogram specification
readImages*	read input images and apply the lum2scale function (see below)
rescale	luminance rescaling
scale2lum**	scales the luminance channel (either V channel from HSV, or L channel from CIELab) from grayscale range to HSV or CIELab range
separate	foreground-background segregation
sfMatch	equates the rotational average of the amplitude spectra
sfPlot	plots the energy at each spatial frequency
SHINE_color*	main function for loading, equating, and saving grayscale and colorful images
specMatch	matches amplitude spectrum
spectrumPlot	plots amplitude spectrum
ssim_index	computes Structural Similarity index
ssim_sens	computes SSIM gradient
tarhist	computes a target histogram
video2frames**	extracts all frames from a video

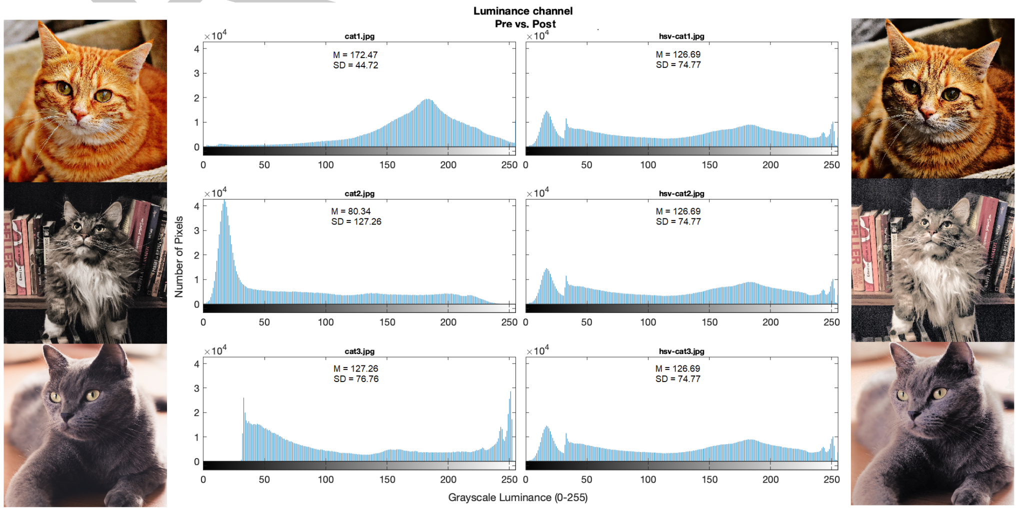

Figure 2: An example of the histogram matching by SHINE_color using the HSV color space. On the left there are images (from pexels), luminance histograms, and summary statistics before the operation. On the right, we have the same elements after the operation.

SHINE_color allow users to take full advantage of the powerful functions from the SHINE toolbox (Willenbockel et al., 2010) for controlling low-level properties of colorful images and videos. It is worth noting that the accurate display of SHINE_color output for experimental research ultimately depends on several factors. Of particular importance are the assumption of linearity between the manipulated luminance values and the display luminance (see the the monitor calibration section from Willenbockel et al., 2010), and room lightning conditions (for a detailed discussion see Tsukahara & Engle, 2020).

The SHINE_color toolbox is openly available at OSF and GitHub. Plans for future development include a MATLAB guided user interface and an adaptation to Python language, for integration with experimental packages such as PsychoPy (Peirce et al., 2019).

Acknowledgments

I am thankful for my former supervisors, Jessica Hay, Ph.D., and Debora de Hollanda Souza, Ph.D., for their support. This work was partially funded by grants from FAPESP (#2015/26389-7, #2018/04226-7) and CAPES (#001). The funders had no role in study design, data collection, analysis and interpretation of the data, decision to publish, or preparation of the manuscript.

References

Cheng, C., Kaldy, Z., & Blaser, E. (2019). Focused attention predicts visual working memory performance in 13-month-old infants: A pupillometric study. *Developmental Cognitive Neuroscience*, *36*(January), 100616. https://doi.org/10.1016/j.dcn.2019.100616

Hepach, R., & Westermann, G. (2016). Pupillometry in Infancy Research. *Journal of Cognition and Development*, *17*(3), 359–377. https://doi.org/10.1080/15248372.2015.1135801

Lawson, R. P., Mathys, C., & Rees, G. (2017). Adults with autism overestimate the volatility of the sensory environment. *Nature Neuroscience*, *20*(9), 1293–1299. https://doi.org/10.1038/nn.4615

Peirce, J., Gray, J. R., Simpson, S., MacAskill, M., Höchenberger, R., Sogo, H., Kastman, E., & Lindeløv, J. K. (2019). PsychoPy2: Experiments in behavior made easy. *Behavior Research Methods*, *51*(1), 195–203. https://doi.org/10.3758/s13428-018-01193-y

Rodger, H., Vizioli, L., Ouyang, X., & Caldara, R. (2015). Mapping the development of facial expression recognition. *Developmental Science*, *18*(6), 926–939. https://doi.org/10.1111/desc.12281

Ruedeerat. (2018). *RGBshine*. https://github.com/Ruedeerat/RGBshine

Sirois, S., & Brisson, J. (2014). Pupillometry. *Wiley Interdisciplinary Reviews: Cognitive Science*, *5*(6), 679–692. https://doi.org/10.1002/wcs.1323

Tsukahara, J. S., & Engle, R. W. (2020). Is baseline pupil size related to cognitive ability? Yes (under proper lighting conditions). *Cognition*, *211*(March), 104643. https://doi.org/10.1016/j.cognition.2021.104643

Willenbockel, V., Sadr, J., Fiset, D., Horne, G. O., Gosselin, F., & Tanaka, J. W. (2010). Controlling low-level image properties: The SHINE toolbox. *Behavior Research Methods*, *42*(3), 671–684. https://doi.org/10.3758/BRM.42.3.671

Zhang, F., Jaffe-Dax, S., Wilson, R. C., & Emberson, L. L. (2019). Prediction in infants and adults: A pupillometry study. *Developmental Science*, *22*(4), 1–9. https://doi.org/10.1111/desc.12780

Zhao, S., Bury, G., Milne, A., & Chait, M. (2019). Pupillometry as an Objective Measure of Sustained Attention in Young and Older Listeners. *Trends in Hearing*, *23*, 233121651988781. https://doi.org/10.1177/2331216519887815