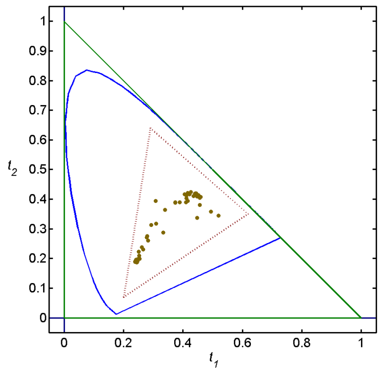
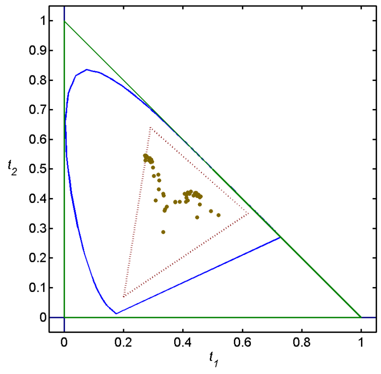
COLORLAB 1.0:
A COLOR PROCESSING TOOLBOX FOR MATLAB 5.X





VNIVERSITAT DE VALÈNCIA

DEPARTAMENT D'ÒPTICA. VISION SCIENCE GROUP

Coordinated by Jesús Malo and M.J. Luque

COLORLAB 1.0: User's Guide

Contents

1. Installing COLORLAB. ***5***

1.1. Contents of the CD or files to download. 5

1.2. Minimum requirements. 5

1.3. Installation procedure. 5
- 1.3.1. Get the installation file 6
- 1.3.2. Decompress *COLORLAB*2.zip 6
- 1.3.3. Update the *MATLAB* path 7

2. COLORLAB tutorial. ***9***

2.1. What is *COLORLAB*? 9

2.2. Who wrote *COLORLAB*? 10

2.3. How to skip reading the *Color Science in a nutshell* section 10

2.4. Color Science in a nutshell 10
- 2.4.1. Basic color description: tristimulus colorimetry 11
- 2.4.2. Advanced color description: color appearance models 11
- 2.4.3. Color processing 12
- 2.4.4. Color reproduction 12
- 2.4.5. Notation 14
  - 2.4.5.1. Tristimulus Colorimetry 14
  - 2.4.5.2. Advanced color description: color appearance models 20
  - 2.4.5.3. Color in computers 20

2.5. Overview of *COLORLAB* functions 22
- 2.5.1. Chromatic stimuli 22
- 2.5.2. Tristimulus representation of color 22
- 2.5.3. Advanced color description: color appearance models 25
  - 2.5.3.1. Cone excitation spaces. 27
  - 2.5.3.2. Single-opponent stage models. 30

2.5.3.3. Two opponent stage models. 35
2.5.3.4. Appearance models 37
2.5.4. Color and color images in *MATLAB* and *COLORLAB* 38

2.6. Data structures in *COLORLAB* 38
2.6.1. Spectral data (reflectances, radiances and color matching functions) 38
2.6.2. Color data (colors, trichromatic units and colormaps) 39
2.6.3. Color images 39
2.6.4. Monitor calibration data 39

2.7. *COLORLAB* databases 40
2.7.1. Illuminants and reflectances 40
2.7.2. Color matching functions and systems of primaries 41
2.7.3. Interesting colors 41
2.7.4. Images 43

2.8. *COLORLAB* demos 49
2.8.1. CRT monitor calibration 49
2.8.2. Color generation 53
2.8.3. Decomposition of color images in color channels 55
2.8.4. Changing the illuminant 56
2.8.5. Changing the color appearance of images 57
2.8.6. Color image compression 59
2.8.7. Color vector quantization 62
2.8.8. Getting information on the reference system. 66

3. A typical COLORLAB session 68

3.1. Some sample problems 68

3.2. Worked examples 68
3.2.1. Starting a *COLORLAB* session 68
3.2.2. Editing the color appearance of an image 72
3.2.3. Generating isoluminant grids. 79

4. Reference manual 90

5. Bibliography 224

Chapter 1

Installing *COLORLAB*

1.1 Contents of the CD or files to download.

Readme file:	`readme.txt` (basic information and installation procedure)
COLORLAB tutorial:	`usrguide.pdf` (this document)
COLORLAB folder:	`colorlab2.zip` (`*.m` routines and data bases)

1.2 Minimum requirements.

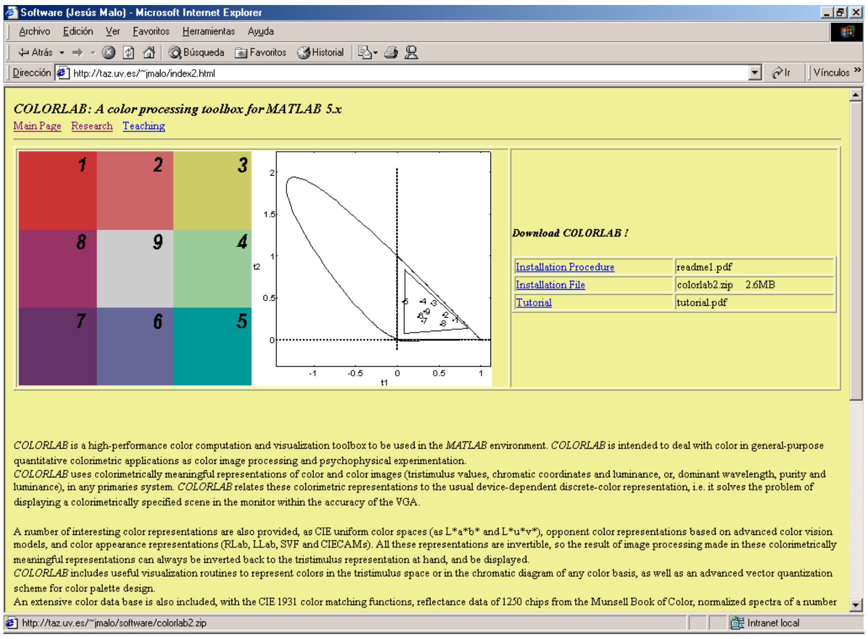
Hardware:	PC Pentium II, 64 MB RAM
Software:	Windows 95 MATLAB 5.X Image Processing Toolbox 2.2 Statistics Toolbox 2.2 Optimization Toolbox 2.0

1.3 Installation procedure.

Installing *COLORLAB* is easy!:
1.- Download the installation file.
2.- Unzip the installation file.
3.- Update the *MATLAB* path.

1.3.1. Get the installation file

Visit `http://taz.uv.es/~jmalo` and and select the software link on it: this page will show up:



Software (Jesús Malo) - Microsoft Internet Explorer

Dirección: http://taz.uv.es/~jmalo/index2.html

COLORLAB: A color processing toolbox for MATLAB 5.x

Main Page Research Teaching

Download COLORLAB !

Installation Procedure	readme1.pdf
Installation File	colorlab2.zip 2.6MB
Tutorial	tutorial.pdf

COLORLAB is a high-performance color computation and visualization toolbox to be used in the *MATLAB* environment. *COLORLAB* is intended to deal with color in general-purpose quantitative colorimetric applications as color image processing and psychophysical experimentation.
COLORLAB uses colorimetrically meaningful representations of color and color images (tristimulus values, chromatic coordinates and luminance, or, dominant wavelength, purity and luminance), in any primaries system. *COLORLAB* relates these colorimetric representations to the usual device-dependent discrete-color representation, i.e. it solves the problem of displaying a colorimetrically specified scene in the monitor within the accuracy of the VGA.

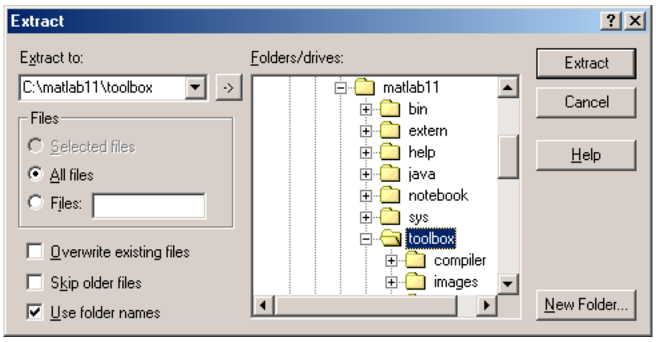
A number of interesting color representations are also provided, as CIE uniform color spaces (as L*a*b* and L*u*v*), opponent color representations based on advanced color vision models, and color appearance representations (RLab, LLab, SVF and CIECAMs). All these representations are invertible, so the result of image processing made in these colorimetrically meaningful representations can always be inverted back to the tristimulus representation at hand, and be displayed.
COLORLAB includes useful visualization routines to represent colors in the tristimulus space or in the chromatic diagram of any color basis, as well as an advanced vector quantization scheme for color palette design.
An extensive color data base is also included, with the CIE 1931 color matching functions, reflectance data of 1250 chips from the Munsell Book of Color, normalized spectra of a number

http://taz.uv.es/~jmalo/software/colorlab2.zip

Download the installation file (`colorlab2.zip`) to a temporal folder.

1.3.2. Decompress `colorlab2.zip`

Use winzip to decompress the file in the toolbox subfolder of your *MATLAB* folder (in this example `c:/matlab11/toolbox/`).
WARNING!: select the option *Use folder names*. In this way each file will go to the appropriate *COLORLAB* subfolder.

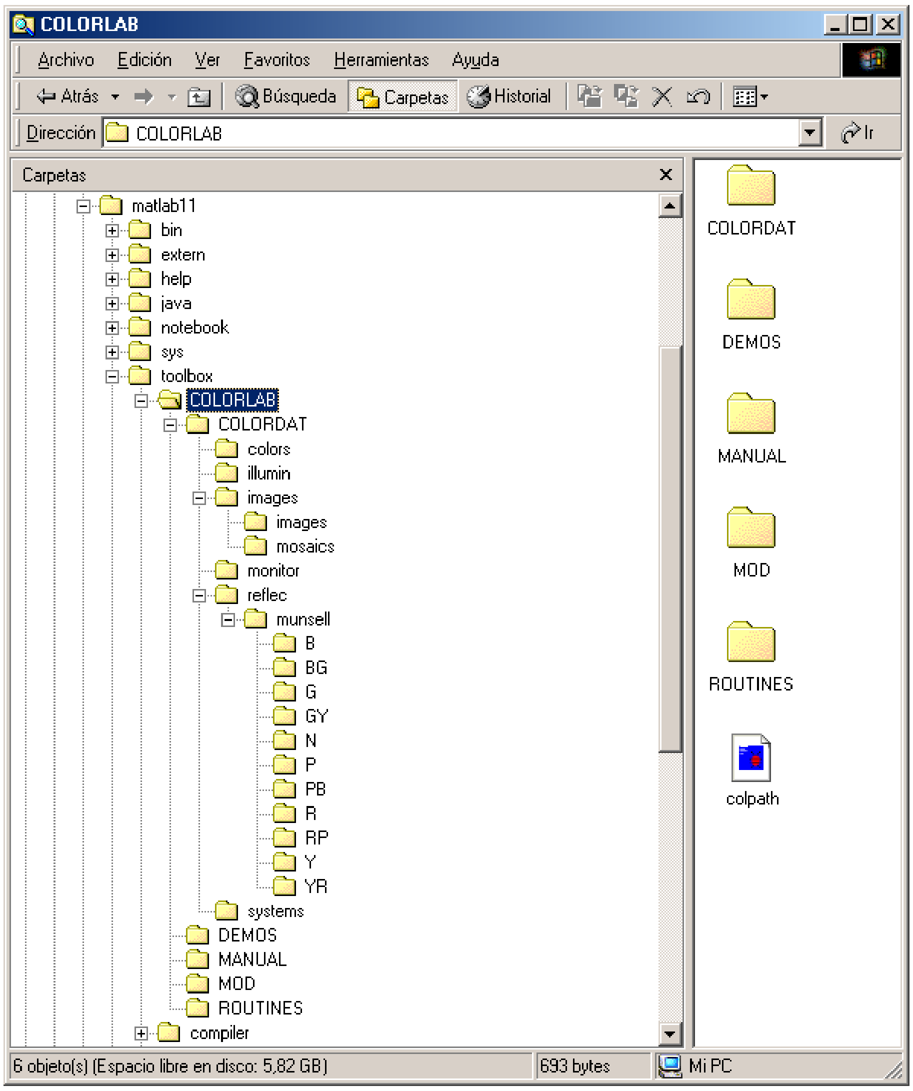


Extract to: C:\matlab11\toolbox

Files: Selected files [✓ All files] Files:

☐ Overwrite existing files
☐ Skip older files
☑ Use folder names

The decompression will generate the *COLORLAB* folder and associated subfolders:

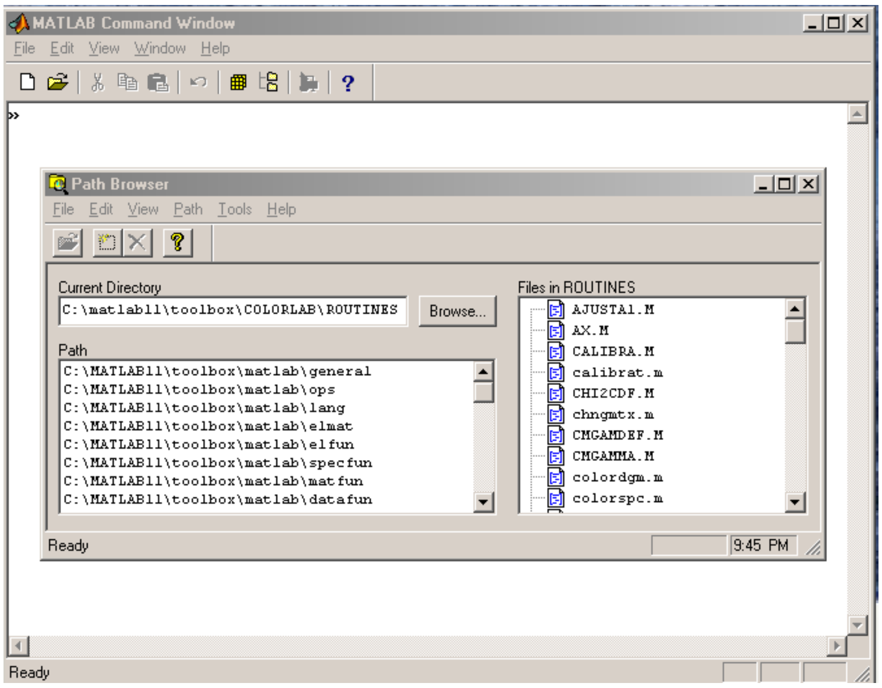


1.3.3. Update the *MATLAB* path

The *MATLAB* path is the set of folders where *MATLAB* looks for `*.m` functions.

You can do this once and forever using the standard procedure in *MATLAB*:

(a) Use the path browser (click on the folder icon in the *MATLAB* toolbar)



Current Directory

```
C:\matlab11\toolbox\COLORLAB\ROUTINES
```

Path

```
C:\MATLAB11\toolbox\matlab\general
C:\MATLAB11\toolbox\matlab\ops
C:\MATLAB11\toolbox\matlab\lang
C:\MATLAB11\toolbox\matlab\elmat
C:\MATLAB11\toolbox\matlab\elfun
C:\MATLAB11\toolbox\matlab\specfun
C:\MATLAB11\toolbox\matlab\matfun
C:\MATLAB11\toolbox\matlab\datafun
```

Files in ROUTINES

```
AJUSTA1.M
AX.M
CALIBRA.M
calibrat.m
CHI2CDF.M
chngmtx.m
CMGAMDEF.M
CMGAMMA.M
colordgm.m
colorspc.m
```

(b) Add the following five folders to the *MATLAB* path (select the *Path* menu and click on *Add to path...*)

```
...\colorlab\routines
...\colorlab\mod
...\colorlab\demos
...\colorlab\colordat\systems
...\colorlab\colordat\monitor
```

(c) Save the changes in the path using the *File* menu.

And that's it!

Now you can type `colorlab` to start the demo and get a flavour of the *COLORLAB* possibilities!

Chapter 2

COLORLAB tutorial

2.1 What is *COLORLAB*?

COLORLAB is a library of functions for processing, representing and reproducing color in a *MATLAB*Ò environment.

Using *COLORLAB* you will be able to:

- Describe chromatic scenes in physical terms (you can generate arbitrary distributions of reflectances and illuminate them with an arbitrary illuminant).
- Turn these chromatic scenes into color images and viceversa.
- Compute the tristimulus description of any image in any system of primaries.
- Represent the color content of any image in chromatic diagrams and tristimulus spaces in any system of primaries.
- Compute advanced color descriptions of any image using several color appearance models (CIELab, CIEluv, ATD, Rlab, LLab, SVF and CIECAM).
- Use your monitor as an accurate color reproduction device. After any color-processing task you will be able to show the results in the screen being sure that *what-you-see-is-what-you-put* (within the accuracy of the VGA!).

In the section 2.4, *Overview of COLORLAB functions*, we'll briefly introduce the different utilities and types of data on which they operate. For more information, see the chapter 4, *Reference Manual*, and the *on-line* helps of the functions listed in the tables and figures.

2.2 Who wrote *COLORLAB*?

COLORLAB was developed by the members of the Vision Science Group of the Dept.of Optics, at the School of Physics, Universitat de València (Spain).
The current educational and research interests of this group include colorimetry, color image coding and numerical models of human color vision.

Jesús Malo, coordinator of the project, came up with the *COLORLAB* idea and developed the general-purpose functions and the tristimulus colorimetry functions. *María José Luque* developed most of the color appearance functions and made an invaluable testing of the functions. Other members of the Vision Science Group involved in the project include: *María Dolores de Fez, Mara Díez, Juan Gómez and Alberto Palomares.*

For any comments or questions please write to:
`jesus.malo@uv.es`
`maria.j.luque@uv.es`
For *COLORLAB* upgrades please visit our web page:
`http://taz.uv.es/~jmalo`

2.3 How to skip reading the *Color Science in a nutshell* section

If you already heard of *tristimulus colorimetry*, *color appearance models* or *color reproduction*, you can skip most of the section 2.4 (i.e. 2.4.1-2.4.4) and go straight for subsection 2.4.5 where we introduce the notation used in *COLORLAB* and throughout this manual.

2.4 Color Science in a nutshell

The aim of Color Science is twofold: to characterize perceptions from the stimuli (*color measurement and description*) and, to synthesize stimuli from the perceptions (*color reproduction*).

Once you have solved these problems you may think on a third issue: *color processing*, i.e. modifying color information before color reproduction to get a new stimulus that gives rise to some desired perception.

In short, Color Science is *just* the following set of transforms:

$$\textit{Stimul.} \xrightarrow{\textit{descript.}} \textit{Percept.} \xrightarrow{\textit{process.}} \textit{Percept.}' \xrightarrow{\textit{reproduct.}} \textit{Stimul.}'$$

The rest of this section summarizes the set of transforms that constitute the core of Color Science as shown in the above diagram.

2.4.1. Basic color description: tristimulus colorimetry

The perception of stimuli viewed in complex backgrounds is more difficult to understand than the perception of isolated stimuli. This is why the first successful theory was the one developed for unrelated colors: the *tristimulus colorimetry* standardized by the *Commission International de l'Éclairage* (CIE) back in 1931.

Tristimulus colorimetry is based on the concept of color matching. In the paradigm of unrelated colors the perception of any isolated stimulus can be matched with an *additive mixture* of three lights, the so called *primaries*. Therefore the perception induced by a stimulus is completely characterized by a 3D vector, the *tristimulus vector*. The components of the vector (the *tristimulus values*) are related to the luminances of the primaries used by the observers to match the color.

The tristimulus colorimetry has become popular because it provides a framework for color measurement and reproduction:

- *Measurement*: the tristimulus values are related to the radiance coming from the stimulus and the color matching functions that characterize the linear behaviour of the observer.
- *Reproduction*: given a certain system of primaries (three *linearly independent* lights), the tristimulus vector in that system tells you the luminances you have to add of each primary to generate the color.

However, tristimulus colorimetry is not completely satisfactory not even for unrelated colors because the color descriptions it provides (*tristimulus vectors*, *chromatic coordinates* and *luminance*, or *dominant wavelength*, *purity* and *luminance*) are not linearly correlated with the perceptual dimensions of unrelated colors *hue*, *colorfulness* and *brightness*.

2.4.2. Advanced color description: color appearance models

The aim of recent color appearance models is to compute consistent descriptors of the perceptual dimensions of color for unrelated and related stimuli (*hue*, *colorfulness*, *chroma*, *brightness* and *lightness*). These representations have also to account for advanced color phenomena as *chromatic induction*, *chromatic adaptation* and *color constancy*.

While tristimulus colorimetry take the physical description of the test as input (in terms of radiances and reflectances), the color appearance models usually take a tristimulus description of the scene as input. In the most general case the tristimulus description of the spatio-temporal neighbourhood of the test must be provided.

2.4.3. Color processing

Color processing is any modification of the color content of a scene. In order to make the processing task intuitive it has to be done in a representation correlated with the perceptual dimensions of color.

2.4.4. Color reproduction

Computers provide an excellent way to reproduce colors according to the tristimulus colorimetry because you can control the luminance of the primaries of the monitor.

However, as in any color reproduction device, the parameters used to encode the color (and to control the luminances of the primaries of the monitor) are not colorimetrically meaningful values but a 3D array, *the digital values*, constrained by the physical properties of the device.
This array is not a vector in the color matching sense. Besides, the digital values description is device dependent because the same digital values on different monitors give rise to different colors.

Hence, in order to accurately reproduce colors in computers you must *calibrate* the color reproduction device to obtain the transform from a colorimetrically meaningful characterization (such as the tristimulus vectors) to the device dependent characterization (digital values).

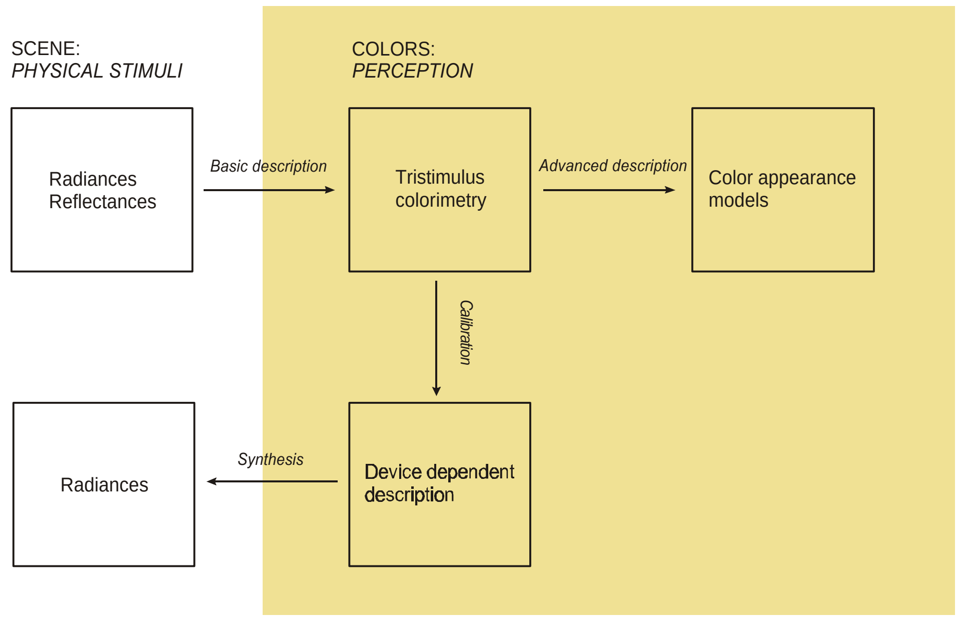


Figure 2.1. *Transfroms in Color Science (I).*

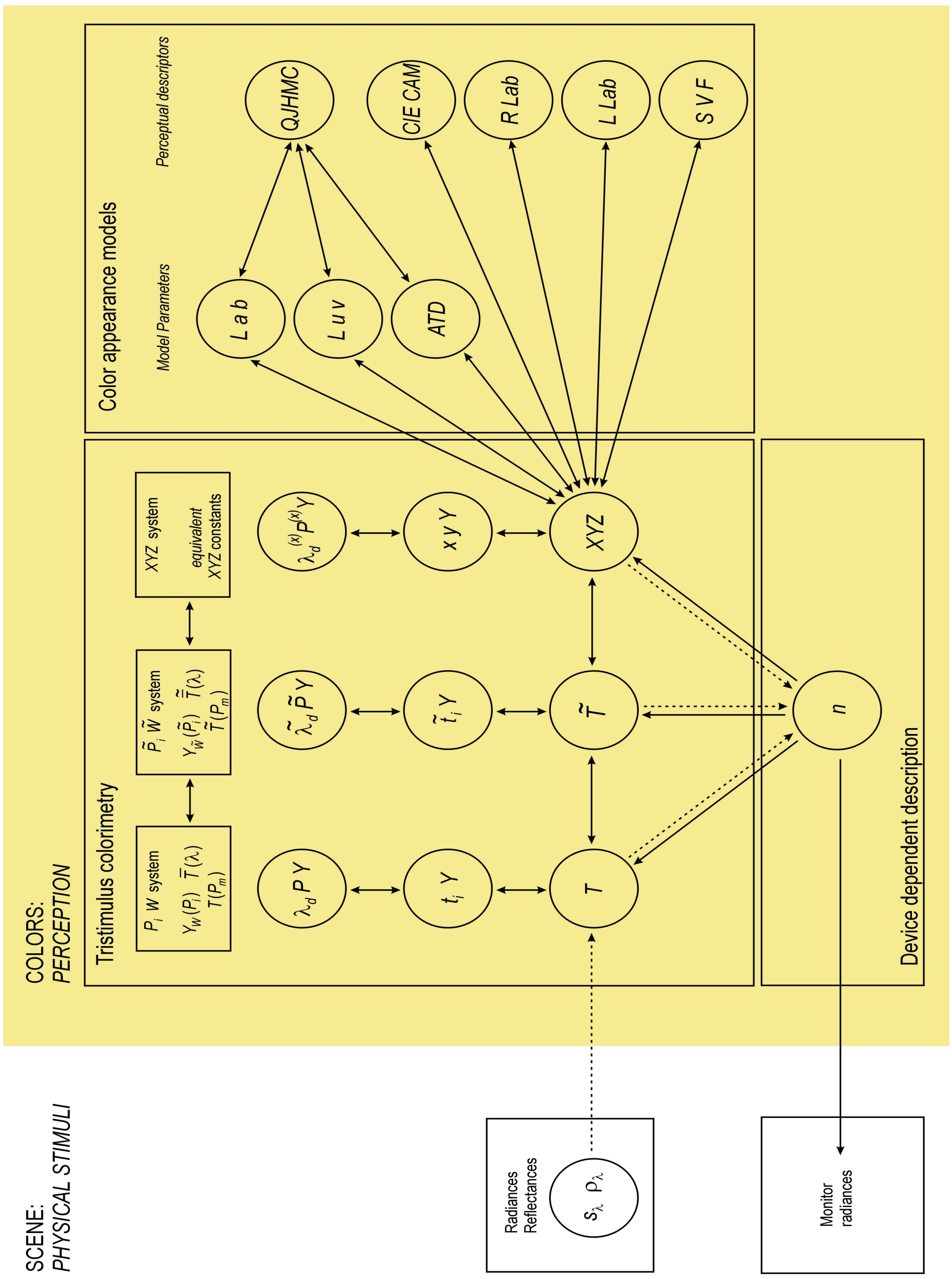


Figure 2.2. *Transforms in Color Science (II). See section 2.5 for details on the notation.*

2.4.5 Notation

The notation used in *COLORLAB* is the one shown in Figure 2.2.

A *chromatic stimulus* is just a set of objects under certain illumination. The physical description of the chromatic stimuli is given in terms of the spectral radiance of the light coming from the objects, s'_l.
Assuming lambertian objects and illuminants, the spectral radiance, s'_l, only depends on the spectral *reflectance of the objects*, $r_l \in [0\ 1]$, on the spectral *radiance of the illuminant*, s_l, given in W/m^{2}str, and on the illumination geometry (see Figure 2.3):

$$s_l' = \left[\frac{a \cdot \cos q}{p\, r^2}\right] \cdot r_l \cdot s_l \tag{2.1}$$

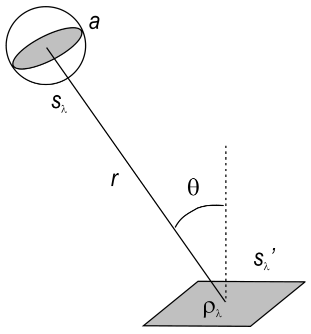


Figure 2.3: *Illumination geometry of a lambertian surface of reflectance* r_l. *The (constant) radiance of the surface,* s_l', *depends on the effective area,* a, *of the source, the distance to the source,* r, *the angle of incidence,* q , *and of course the radiance of the source,* s_l .

2.4.5.1 Tristimulus Colorimetry

In tristimulus colorimetry a *system of primaries* is defined by three independent light sources, the primaries P_i, i=1,2,3 and a white reference, *W*, used to scale the luminances of the primaries.
The *luminance* of a stimulus S, is referred to as *Y(S)*, measured in cd/m^2.

Tristimulus colorimetry is based on color matching mixing the primaries in certain proportions. The luminances of the primaries needed to match the color of a stimulus S are referred to as $Y_S(P_i)$.

The *tristimulus values* of a stimulus, S, are $T_i(S)$ given by,

$$T_i(S) = \frac{Y_S(P_i)}{Y_W(P_i)} \tag{2.2}$$

where the luminances of the primaries to match the white, $Y_W(P_i)$, are referred to as *trichromatic units*.
The color matching functions, $\bar{T}_i(l)$, are the tristimulus values of each monochromatic stimulus of spectral radiance 1/680 W/m^{2}str.

The tristimulus values are also related to the radiance, s'_l, through the color matching functions, $\bar{T}_i(l)$:

$$T_i(S) = k \cdot \int_{380}^{770} s'_l \cdot \bar{T}_i(l) \cdot dl \tag{2.3}$$

where the constant k = 680 lum m^2 str / W.
The color is characterized by the tristimulus vector $T(S) = (T_1(S), T_2(S), T_3(S))$.
Figure 2.4 shows the color matching functions in a system of arbitrary red, green and blue primaries scaled with an arbitrary white.

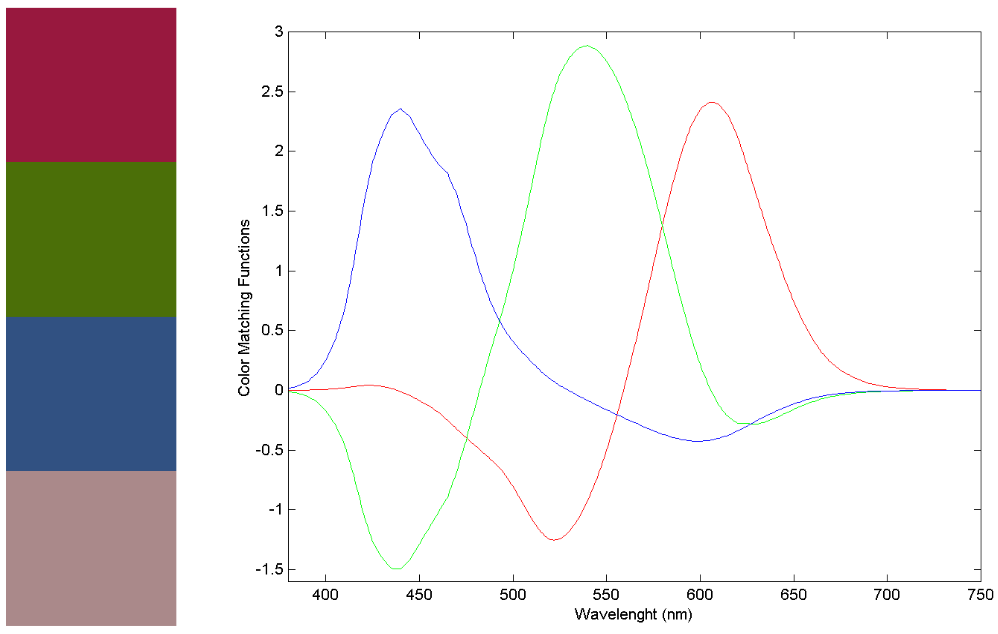


Figure 2.4: *Primaries (some arbitrary R,G and B), the scaling white and the corresponding color matching functions.*

Figure 2.5 shows the spectral radiance and the reflectance of an arbitrary illuminant and surface. Figure 2.6 shows the color, S, and its position in the tristimulus space given by the tristimulus vector, *T(S)*, in the basis of the primaries shown in Figure 2.5. The white and the primaries are also plotted as a useful reference. The *chromaticity diagram* (the plane going through the points (1,0,0), (0,1,0) and (0,0,1)) with the locus of chromaticities of the monochromatic stimuli has also been plotted.

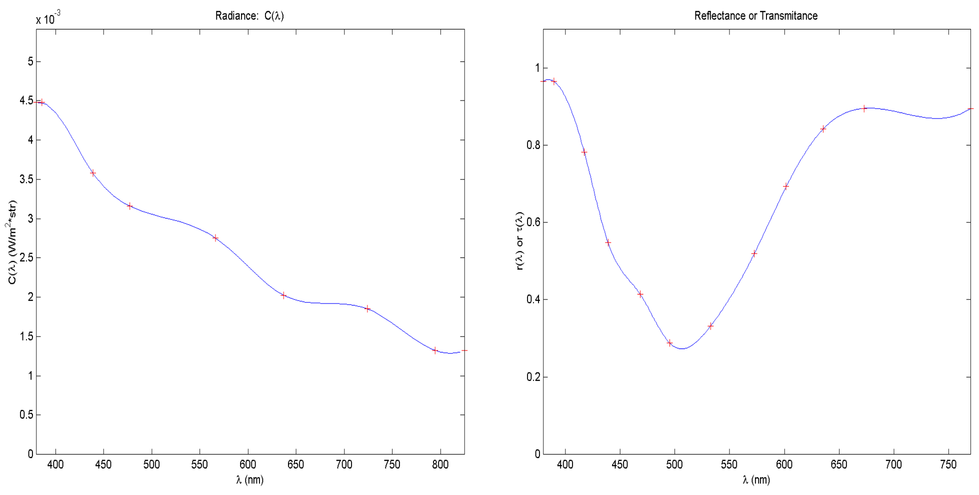


Figure 2.5: *Radiance of an illuminant (left) and reflectance of a surface (right).*

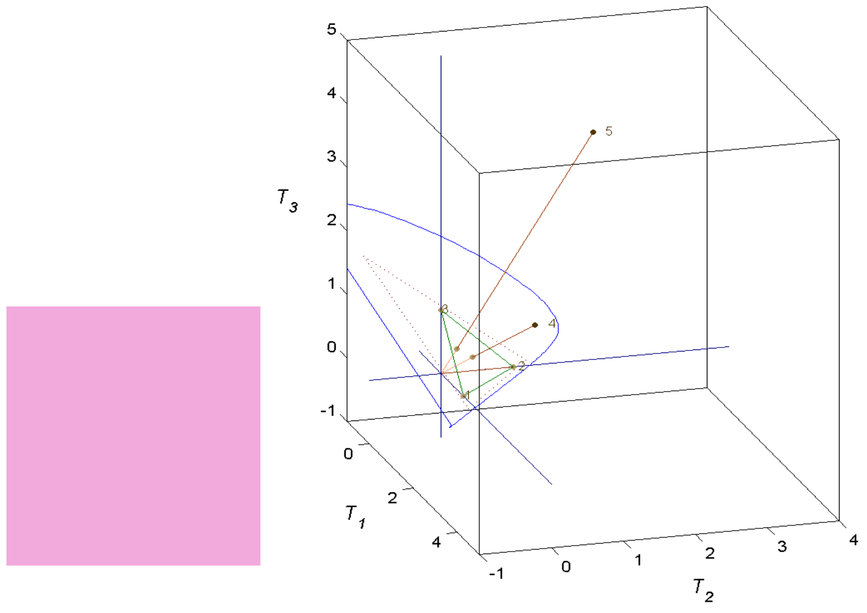


Figure 2.6: *The color of the surface and its representation its representation in the tristimulus space (labeled as 5). The primaries and the white reference (labeled as 1,2,3,4) are also plotted.*

Alternative descriptions of color have been defined to isolate the chromaticity information from the luminance: *chromatic coordinates and luminance* of a stimulus (*$t_1(S)$, $t_2(S)$, $Y(S)$*) and *dominant wavelength*, *purity* and *luminance* (*$l_d(S)$,$P(S)$,$Y(S)$*).

The *chromatic coordinates*, (*$t_1(S)$, $t_2(S)$*), are the intersection of the tristimulus vectors with the chromatic diagram (the orange dots in Figure 2.5). The *luminance*, *$Y(S)$*, can be computed from the tristimulus vector and the trichromatic units:

$$t_i = \frac{T_i}{\sum_{i=1}^{3} T_i} \qquad Y = \sum_{i=1}^{3} T_i \; Y_W(P_i) \tag{2.4}$$

Figure 2.7 shows the representation of these colors in the chromatic diagram. *$l_d(S)$* is the wavelength of the stimulus which is in the intersection of the line that goes through the white reference and the color. The (excitation) purity is the ratio between the distance from the white to the color over the distance from the white to the spectral locus.

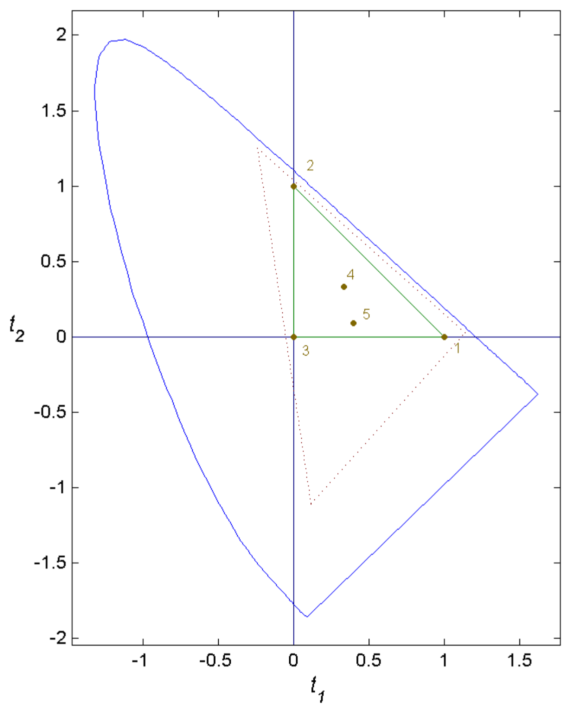


Figure 2.7: *The color (5), the primaries (1,2,3) and the wite (4) in the chromatic diagram.*
Of course, other tristimulus representations are possible because vectors in the tristimulus space can be represented in any basis of the space. The basis of the

primaries in Figure 2.4 is just one possible basis (of primaries P_i) but we can define some other basis (of primaries P_i') in the space.
The tristimulus values in the new basis (and hence the new color matching functions) will be given by the change-of-basis relation:

$$T'(S) = Mpp' \cdot T(S) \tag{2.5}$$

where Mpp' is the *change-of-basis* matrix. As you may know from elementary algebra, this matrix can be computed from the tristimulus vectors of the new primaries in the old basis (see the help of `chngmtx.m`).

To do so we have to select the chromaticities of the new basis and the new white reference (with its luminance). For instance, lets take the new basis with the colors shown in Figure 2.8 and a luminance of 32 cd/m^2 for the new white (these values have no particular meaning!).

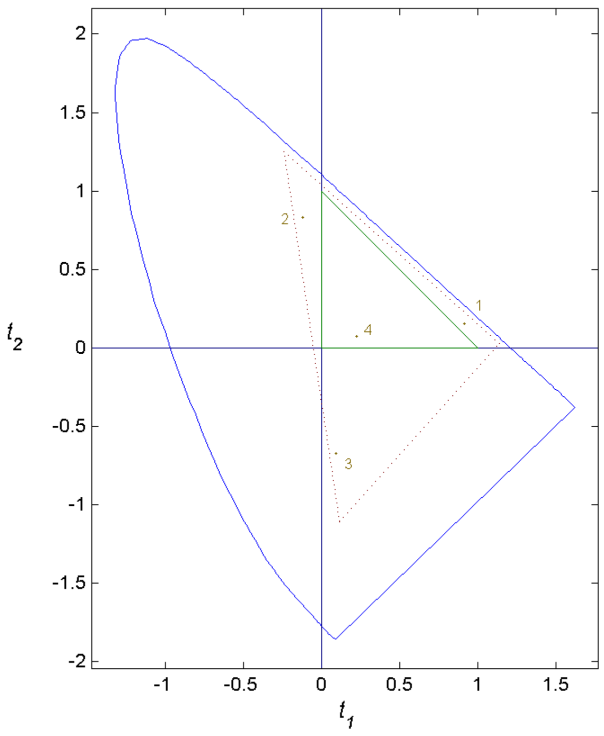


Figure 2.8: *New primaries and white in the chromatic diagram of the basis of Fig. 2. 4.*

The new primaries, white and color matching functions are shown in Figure 2.9. The color in the new representation is shown in Figures 2.10 (tristimulus space) and 2.11 (chromatic diagram).

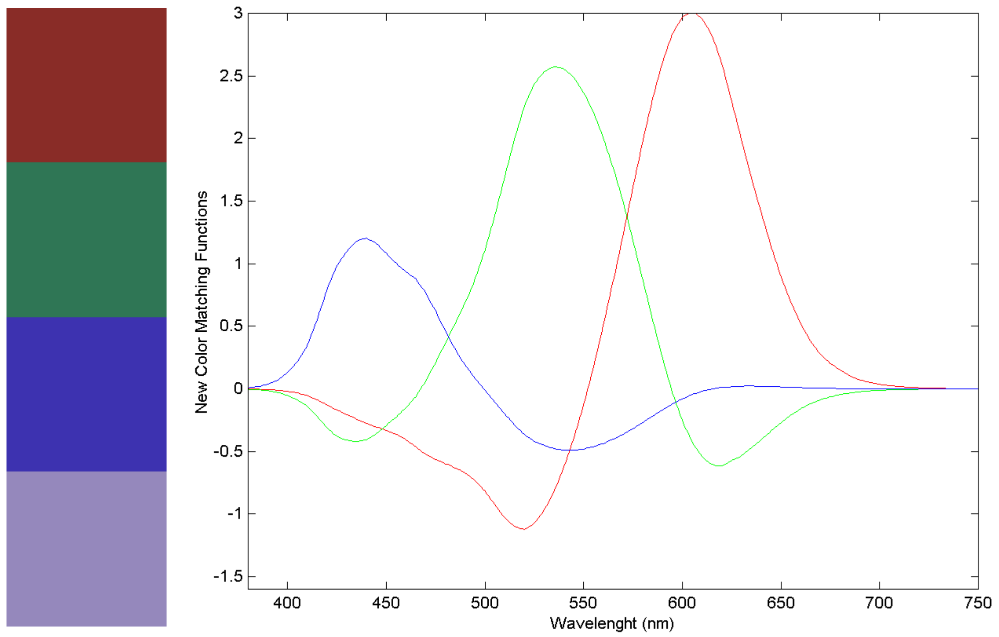


Figure 2.9: *New primaries (and new white) and new color matching functions.*

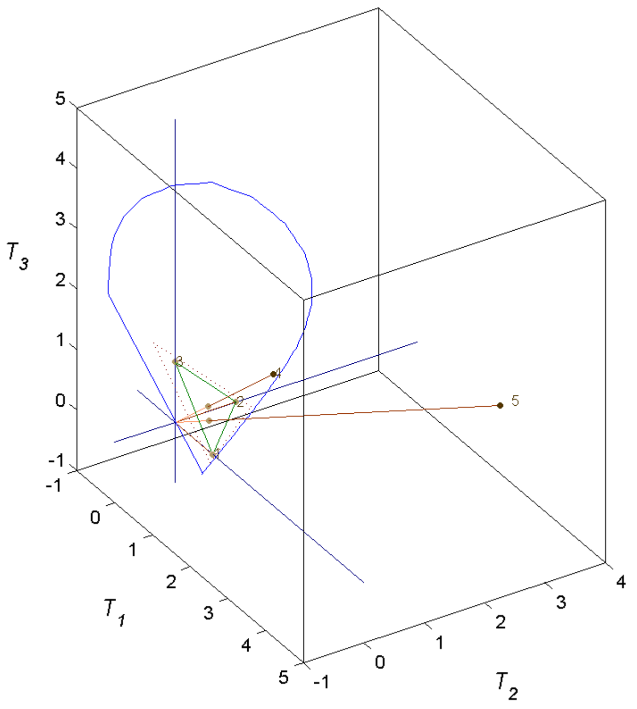


Figure 2.10: *The color (5) in the new basis. Note that the new primaries (1,2,3) and the new white (4) have again the same value (because they are expressed in their own basis).*

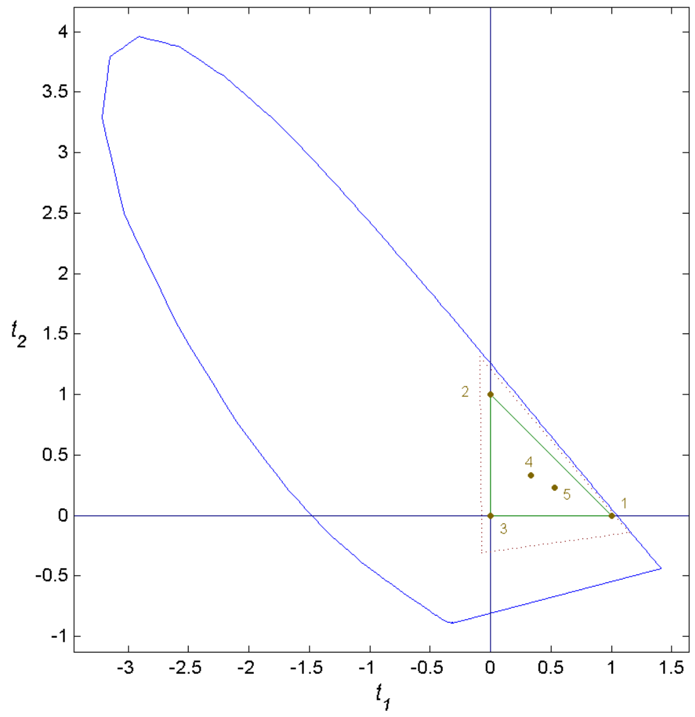


Figure 2.11: *The color (5) in the new chromatic diagram. Note that the new primaries (1,2,3) and the new white (4) have again the same value (because they are expressed in their own basis).*

2.4.5.2. Advanced color description: Color Appearance Models

COLORLAB includes several color representations using different Color Appearance models: CIE Lab, CIE Luv, Llab, Rlab, CIE CAM and. SVF. Also, it includes several physiological models giving rise to ATD (Achromatic, Tritanopic, Deuteranopic) representations (see subsection 2.5.3 for details).
These representations are accessible from a particular tristimulus basis (the CIE XYZ system). From some of these representations you can compute numerical values for the CIE perceptual descriptors: brightness, Q, lightness, J, hue, h, colorfulness, M, and chroma, c.

2.4.5.3. Color in computers

In computer systems color is not defined in any colorimetrically meaningful way but described by a 3D array, *the digital values*, $n = [n_1, n_2, n_3]$, constrained by the physical properties of the device. In *MATLAB*, the values in this array may take values in the range [0,1] (see the help of the *MATLAB* function `colormap.m`).

This array is not a vector in the color matching sense. Besides, this array has two basic problems: (1) the digital values description is device dependent because the same digital values on different monitors give rise to different colors, and (2) the monitor has a finite resolution so not any different value n_i is considered as different by the monitor.

Hence, in order to accurately reproduce colors in computers you must *calibrate* the color reproduction device to obtain the transform from a colorimetrically meaningful characterization (such as the tristimulus vectors) to the device dependent characterization (digital values).

The calibration process includes measuring the tristimulus vectors of the primaries of the monitor P_m (the phosphors of the monitor) as a function of the digital value (see the help of `calibrat.m`).

In *COLORLAB* these data are stored using the variation of the chromatic coordinates of the primaries in the variable `tm`, and fitting the variation of the three luminances with exponential functions of the form:

$$Y_i = a_i n_i^{g_i} \tag{2.6}$$

The chromatic coordinates of the phosphors, tm, and the values of the values of fitting curves, $a = [a_1, a_2, a_3]$, and $g = [g_1, g_2, g_3]$, are necessary to transform any tristimulus representation to the digital value representation prior to color reproduction.

This is why whenever we change of tristimulus representation, we also need to transform these data to the new system (as we do with intrinsically colorimetric parameters as the color matching functions or the trichromatic units).

In fact only t_m changes because luminance (and hence a and g) is invariant under tristimulus change of basis.

2.5 Overview of *COLORLAB* functions

The *COLORLAB* functions inherit the structure of the Color Science transforms shown in figures 1 and 2.

2.5.1. Chromatic stimuli

Loading and defining chromatic stimuli

To obtain	From	Use
A user-defined illuminant		`defillu`
A user-defined reflectance		`defrefl`
A *COLORLAB* format file with the data defining an illuminant		`saveillu`
A *COLORLAB* format file with the data defining a reflectance		`saverefl`
Stored data defining an illuminant	A *COLORLAB* format file selected with a dialog box	`loadillu`
Stored data defining an illuminant	A *COLORLAB* format file defined by its name	`loadilum`
Stored data defining a reflectance	A *COLORLAB* format file selected with a dialog box	`loadrefl`
Stored data defining a reflectance	A *COLORLAB* format file identified by its name or by Munsell descriptors	`loadrefm`

2.5.2. Tristimulus representation of color

Chromatic stimulus to tristimulus values transforms

To obtain	From	Use
Tristimulus values	Reflectances and illuminants	`spec2tri`
Reflectances	Tristimulus values	`tri2spec`

Tristimulus color representations

To obtain	From	Use
Chromaticity coordinates and luminance	Tristimulus values	`tri2coor`
Tristimulus values	Chromaticity coordinates and luminance	`coor2tri`
Dominant, purity and luminance	Chromaticity coordinates and luminance	`coor2lp`
Chromaticity coordinates and luminance	Dominant, purity and luminance	`lp2coor`

Graphic representation of stimuli

To obtain	From	Use
Color plot in the chromaticity diagram	Any tristimulus color description	`colordgm`
Color plot in the tristimulus space	Any tristimulus color description	`colorspc`
The spectral locus	Color matching functions and trichromatic units	`replocus`

Color definition

To obtain	From	Use
A user-defined set of color or set of colors	A color diagram	`defcolor`
A user-defined set of color or set of chromaticities	A color diagram	`defcroma`
A *COLORLAB* format file with the data defining a color or set of colors		`savecol`
Stored data defining a set of colors	A *COLORLAB* format file	`loadcol`

Colorimetric reference systems definition

To obtain	From	Use
A new colorimetric reference system	Tristimulus values of the primaries or Chromaticity coordinates of primaries and tristimulus values of white	`defsysm`
A new colorimetric reference system	Chromaticity coordinates of primaries and tristimulus values of white graphically defined	`defsys`
A *COLORLAB* format file with the data defining a reference system		`savesysm`
Stored data defining a reference system	A *COLORLAB* format file selected with dialog box	`loadsys`
Stored data defining a reference system	A *COLORLAB* format file	`loadsysm`
Data defining a reference system and a CRT display	A dialog box	`startcol`
The change-of-basis matrix between two systems	Tristimulus values of the primaries or Chromaticity coordinates of primaries and tristimulus values of white	`chngmtx`
Tristimulus values in system S’	Tristimulus values in system S	`newbasis`
Matching functions, trichromatic units and data defining a CRT display in system S’	Matching functions, trichromatic units and data defining a CRT display in system S	`newconst`

2.5.3 Advanced color description: color appearance models

Tristimulus colorimetry is incapable of explaining many color appearance phenomena. For instance, hue changes with luminance (Bezold-Brücke effect) while chromaticity coordinates don't. On the other hand, chromaticity coordinates are modified by changes in the spectral distribution of the illuminant but, if the observer is presented with a complex scene, color appearance isn't.

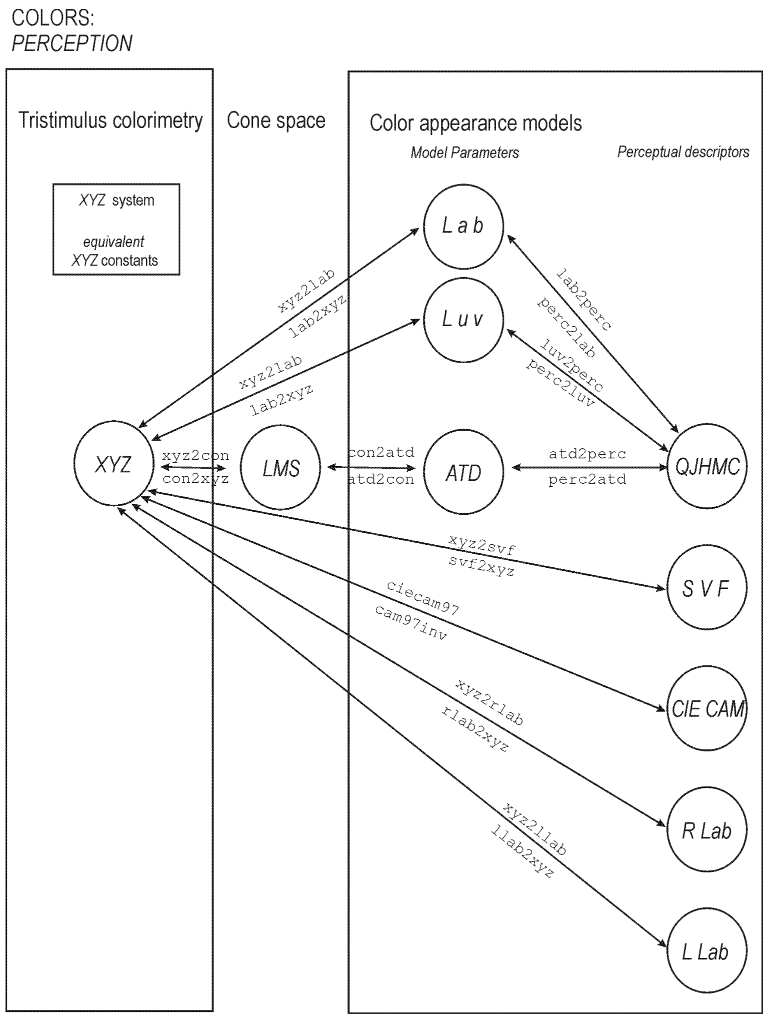


Figure 2.12: *Color spaces implemented in COLORLAB. The name of the COLORLAB functions above the lines correspond to the direct transforms; those below are the inverses.*

Tristimulus spaces do not describe correctly the appearance changes induced in isolated colors by the colorimetric parameters. Besides, they do not contemplate mechanisms to introduce the influence of the surround or of the illuminant.

Finally, tristimulus spaces are not well correlated with physiological mechanisms (photoreceptors and neural visual pathways.) More advanced representation spaces try to modelize these aspects of the visual system. For comprehensive reviews, see Capilla et al, 2002.

COLORLAB offers the possibility of working with some of the most recent color vision models and color spaces. Figure 2.12 shows the available models and the functions that allow to transform tristrimulus values into descriptors of each model. Tables 2.1 and 2.2 show which function must be used for a given transform.

To obtain	**From**	**Use**
Cone responses	XYZ tristimulus values	`xyz2con`
XYZ tristimulus values	Cone responses	`con2xyz`
Achromatic and chromatic opponent responses (*ATD*)	XYZ tristimulus values	`xyz2atd`
XYZ tristimulus values	*ATD* responses	`atd2xyz`
ATD responses	Cone responses	`con2atd`
Cone responses	First stage *ATD* responses	`atdf2con`
Second-stage *ATD* responses	First-stage *ATD* responses	`atd1atd2`
First-stage *ATD* responses	Second-stage *ATD* responses	`atd1atd2`
Perceptual descriptors (*B, S, h)*	Final stage *ATD* responses	`perc2atd`
Final stage *ATD* responses	Perceptual descriptors (*B, S, h)*	`atd2perc`

Table 2.1.- Possible transforms in COLORLAB between CIEXYZ, cone spaces, and ATD spaces

To obtain	From	Use
CIELAB coordinates and lightness	XYZ tristimulus values	`xyz2lab`
XYZ tristimulus values	CIELAB coordinates and lightness	`lab2xyz`
CIELAB lightness, hue and chroma	CIELAB coordinates and lightness	`lab2perc`
CIELAB coordinates and lightness	CIELAB lightness, hue and chroma	`perc2lab`
CIELUV coordinates and lightness	XYZ tristimulus values	`xyz2luv`
XYZ tristimulus values	CIELUV coordinates and lightness	`luv2xyz`
CIELUV lightness, hue and chroma	CIELUV coordinates and lightness	`luv2perc`
CIELUV coordinates and lightness	CIELUV lightness, hue and chroma	`perc2luv`
SVF coordinates and lightness	XYZ tristimulus values	`xyz2svf`
XYZ tristimulus values	SVF coordinates and lightness	`lab2svf`
LLAB coordinates and perceptual descriptors	XYZ tristimulus values	`xyz2llab`
XYZ tristimulus values	LLAB perceptual descriptors	`llab2xyz`
RLAB coordinates and perceptual descriptors	XYZ tristimulus values	`xyz2rlab`
XYZ tristimulus values	RLAB perceptual descriptors	`rlab2xyz`
CIECAM97 perceptual descriptors	Relative XYZ tristimulus values	`ciecam97`
Relative XYZ tristimulus values	CIECAM97 perceptual descriptors	`ciecam97inv`

Table 2,2.- Possible transforms in COLORLAB between CIEXYZ and color appearance spaces.

2.5.3.1. Cone excitation spaces.

In these spaces, cone responses or *excitations* are assumed to be linear. Therefore, the transform between any tristimulus space and a space where the stimulus descriptors are cone excitations is just a change of basis, as in Equation 2.5:

$$\begin{pmatrix} T_1(C) \\ T_2(C) \\ T_3(C) \end{pmatrix} = \begin{pmatrix} T_1([L]) & T_1([M]) & T_1([S]) \\ T_2([L]) & T_2([M]) & T_2([S]) \\ T_3([L]) & T_3([M]) & T_3([S]) \end{pmatrix} \begin{pmatrix} L(C) \\ M(C) \\ S(C) \end{pmatrix} \qquad (2.6)$$

L, M and *S* are, respectively, the responses of the long-, medium- and short-wavelength sensitive cone types and [L], [M] and [S] the cone primaries, that is, stimuli whose descriptors in the cone space are [1 0 0], [0 1 0] and [0 0 1].

The main problem is to determine the tristimulus values of the cone primaries. If we assume that dichromacy arises from lack of one of the three cone types (long, medium or short for protanopes, deuteranopes and tritanopes, respectively) the cone space primaries are the confusion points of the three kinds of dichromats. This is known as *König's hypothesis* (Wyszecki and Stiles, 1982, pp. 582-689). With this hypothesis, Equation (2.6) is written as follows in the CIEXYZ space, which is the usual starting space:

$$\begin{pmatrix} L(C) \\ M(C) \\ S(C) \end{pmatrix} = \begin{pmatrix} \dfrac{y([L])}{Y_{W'}([L])} & 0 & 0 \\ 0 & \dfrac{y([M])}{Y_{W'}([M])} & 0 \\ 0 & 0 & \dfrac{y([S])}{Y_{W'}([S])} \end{pmatrix} \begin{pmatrix} x([L]) & x([M]) & z([S]) \\ y([L]) & y([M]) & y([S]) \\ z([L]) & z([M]) & z([S]) \end{pmatrix}^{-1} \begin{pmatrix} X(C) \\ Y(C) \\ Z(C) \end{pmatrix} \quad (2.7)$$

The chromaticity coordinates of the cone primaries are obtained from the confusion lines of dichromats (see Table LA QUE SEA). The trichromatic units are traditionally determined by assuming a given functional dependence between the cone responses and the luminance. For instance, in the Vos and Walraven cone space (1971)

$$Y=L+M+S \qquad (2.8)$$

Therefore, the trichromatic units are equal to one and then:

$$\begin{pmatrix} L \\ M \\ S \end{pmatrix} = \begin{pmatrix} 0.15516 & 0.54308 & -0.03702 \\ -0.15516 & 0.45692 & 0.02969 \\ 0 & 0 & 0.00732 \end{pmatrix} \begin{pmatrix} X' \\ Y' \\ Z' \end{pmatrix} \qquad (2.9)$$

where (X',Y',Z') are the XYZ tristimulus values including Judd's modification (Judd, 1951).

In the more popular Smith and Pokorny space (1996), the luminance is the sum of the *L* and *M* responses:

$$Y=L+M \qquad (2.10)$$

With this, $Y_{W'}([L]) = Y_{W'}([M]) = 1$ but the matrix element $\dfrac{y([S])}{Y_{W'}([S])}$ is not well defined since both members of the quotient are zero:

$$\begin{pmatrix} L \\ M \\ S \end{pmatrix} = \begin{pmatrix} 0.15514 & 0.54312 & -0.03286 \\ -0.15514 & 0.45684 & 0.03286 \\ 0 & 0 & k \end{pmatrix} \begin{pmatrix} X' \\ Y' \\ Z' \end{pmatrix} \qquad (2.11)$$

The scaling constant k is free, and must be specified in each case. The procedure for determining this constant is often based on the existence of unique hues (a blue that is neither purplish nor greenish, a green that is neither bluish nor yellowish, a yellow that is neither greenish nor reddish and a red that is neither yellowish nor bluish). The uniqueness of these hues is assumed to be due to the fact that they arouse a response in only one of the two perceptual opponent mechanisms (red-green or blue-yellow) while eliciting no response in the other one. By assuming a given functional dependence between the cone responses and the opponent responses, a equation may be reached to compute k. For instance Wyszecki and Stiles, following Vos and Walraven (Vos and Walraven, 1971) use the condition (Wyszecki and Stiles, 1982, pp. 582-689) :

$$L(475.5)+M(475.5)= 16 \cdot S(475.5) \qquad (2.12)$$

This condition is based on knowledge of the unique hues, blue and yellow, 475.5 and 570 nm, respectively. From (2.11) and (2.11)

$$\begin{pmatrix} L \\ M \\ S \end{pmatrix} = \begin{pmatrix} 0.15514 & 0.54312 & -0.03286 \\ -0.15514 & 0.45684 & 0.03286 \\ 0 & 0 & 0.00801 \end{pmatrix} \begin{pmatrix} X' \\ Y' \\ Z' \end{pmatrix} \qquad (2.13)$$

COLORLAB includes a set of functions to transform *XYZ* values to *LMS* responses in different linear cone spaces (Vos and Walraven, Wyszecki and Stiles and MacLeod-Boynton, Stockman and Sharpe). For more details, go to the *Reference* section and consult the helps for `xyz2con` and `con2xyz`. Note that more advanced representation spaces and color vision models include a cone stage which can be also used as a non-linear cone space.

2.5.3.2. Single-opponent stage models.

COLORLAB functions `con2atd`, `atd2con`, `atd1atd2`, `xyz2atd` and `atd2xyz` can be used to transform data in CIEXYZ or in cone spaces to different single-opponent stage models. A brief summary is made below.

Linear ATD-spaces. In the XIX century, Herring noticed that although any color may be described by combining just four color names (blue, yellow, red and green), certain name combinations were not possible. In fact, no normal human observer would describe a color as bluish-yellow, yellowish-blue, reddish-green or greenish-red. Herring explained this fact by assuming the existence of two *opponent mechanisms,* that is, mechanisms whose response, unlike cones, can be positive (excitatory) or negative (inhibitory). One mechanism (D) would signal for changes in the blue-yellow direction and the other (T) in the green-red direction. For example, violet, which can be described as a reddish-blue, would give positive responses in both T and D. Orange, which can be described as reddish-yellow, would give positive response in T and negative in D. No color could be described as reddish-green, for instance, because that would imply that the T response ought to be simultaneously positive and negative. The existence of the unique hues, introduced in the discussion of LMS spaces is also explained. Unique blue, for instance, is a color that is perceived neither reddish nor greenish, and therefore would give response only in one of the two mechanisms (D).

ATD spaces are the mathematical formulation of this opponent mechanism theory. Colors are described by three numbers, A, T and D, that are interpreted as the responses of a luminance mechanism and two opponent color mechanisms. Besides the psychophysical mechanisms described by Herring, it has been proved that certain cells in the visual systems have also opponent behaviour. Some color vision models aim to reproduce the performance of the opponent and non-opponent cells with linear combinations of the LMS cone responses. These are the LGN-based models (Ingling and Tsou, 1977; Boynton, 1986). Nevertheless, other models (Jameson and Hurvich, 1955; Hurvich and Jameson, 1957; Guth et al., 1980), define cone transformations that, under certain constraints, adequately predict discrimination thresholds and color appearance properties.

If the ATD responses are assumed to be linear, they can be obtained as a linear transform of the cone responses:

$$\begin{pmatrix} A \\ T \\ D \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} L \\ M \\ S \end{pmatrix} \qquad (2.14)$$

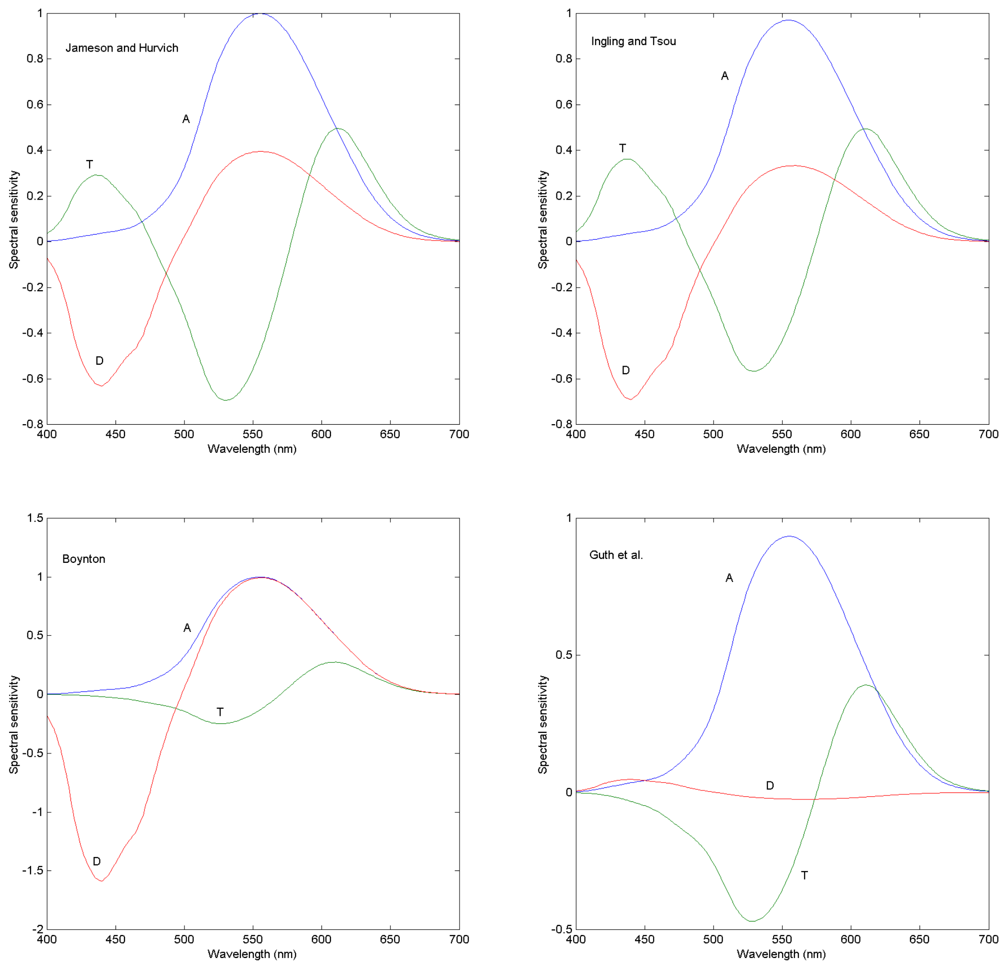


Figure 2.13: *Spectral sensitivities of the ATD mechanisms of the linear ATD models implemented in COLORLAB.*

Some models try to include the chromatic adaptation mechanisms of the visual system. These mechanisms are partially responsible for the capability of the visual system of keeping color appearance approximately constant with large illuminant variations. In linear models, it is assumed that a change in the illuminant modifies the relative scaling of the visual mechanism. With this idea, Equation 2.14 can be written as follows:

$$\begin{pmatrix} A \\ T \\ D \end{pmatrix} = \begin{pmatrix} k_A & 0 & 0 \\ 0 & k_T & 0 \\ 0 & 0 & k_D \end{pmatrix} M_{LMS \to ATD} \begin{pmatrix} L \\ M \\ S \end{pmatrix} \qquad (2.15)$$

where the constants k_i depend on the illumination conditions and $M_{LMS \rightarrow ATD}$ is a matrix valid for certain reference conditions. The particular form of Equation (2.14) changes with the model. The spectral sensitivities of the *A*, *T* and *D* channels for the different models implemented in *COLORLAB* are shown in Figure 2.13. The main differences between these models are (1) the contribution of *S* to luminance and to the *T* channel, (2) the contribution of *M* to the *D* channel and (3) the existence of an adaptation mechanism. Perceptual descriptors are computed as non-linear functions of the *ATD* responses. The different models implemented in *COLORLAB* are summarily described below.

In the model of Jameson and Hurvich (Jameson and Hurvich, 1955; Hurvich and Jameson, 1957), Equation (2.14) is written as:

$$\begin{pmatrix} A \\ T \\ D \end{pmatrix} = \begin{pmatrix} k_1 & 0 & 0 \\ 0 & k_2 & 0 \\ 0 & 0 & k_3 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 \\ 1 & -2 & 1 \\ -2 & 1 & 1 \end{pmatrix} \begin{pmatrix} L \\ M \\ S \end{pmatrix} \qquad (2.16)$$

where k_1, k_2 and k_3 are adaptation-dependent normalizing constants and *LMS* is given by

$$\begin{pmatrix} L \\ M \\ S \end{pmatrix} = \begin{pmatrix} 0 & 6.5341 & 0.1336 \\ -0.3368 & 7.0009 & 0.0020 \\ 0.3329 & 6.4671 & -0.1347 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \qquad (2.17)$$

Perceptual descriptors brightness (*B*), hue (*H*) and saturation (*S*) in this model are rather singular:

$$\begin{aligned} B &= |A| + |T| + |D| \\ H &= \frac{|T|}{|T| + |D|} \\ S &= \frac{|T| + |D|}{B} \end{aligned} \qquad (2.18)$$

Note that with these definitions, hue does not depend on luminance, contrary to experimental evidence. This is one of the main drawbacks of all ATD linear models.

The model proposed by Ingling and Tsou (Ingling and Tsou, 1977), gives the following form for Equation (2.14):

$$\begin{pmatrix} A \\ T \\ D \end{pmatrix} = \begin{pmatrix} 0.60 & 0.40 & 0 \\ 1.20 & -1.60 & 0.40 \\ 0.24 & 0.105 & -0.70 \end{pmatrix} \begin{pmatrix} L \\ M \\ S \end{pmatrix} \qquad (2.19)$$

$$\begin{pmatrix} A \\ T \\ D \end{pmatrix} = \begin{pmatrix} 0.60 & 0.40 & 0 \\ 1.20 & -1.60 & 0 \\ 0.048 & -0.039 & -0.042 \end{pmatrix} \begin{pmatrix} L \\ M \\ S \end{pmatrix} \qquad (2.20)$$

Equation 2.19 is valid for supra-threshold and white-light-adapted stimuli and Equation 2.20 for threshold and dark-adapted stimuli (Ingling, 1977). *LMS* are again the Smith-Pokorny fundamentals normalized to unity, but in a slightly modified version differing in the short-wavelength limbs of the *L*- and *M*- curves. In this model, and in the rest of the ATD linear models described, the perceptual descriptors are defined as follows:

$$\begin{gathered} B = \sqrt{A^2 + T^2 + D^2} \\ H = atan\frac{D}{T} \qquad (2.21) \\ S = \frac{\sqrt{T^2 + D^2}}{A} \end{gathered}$$

In the model proposed by Guth and co-workers (Guth et al., 1980), which considers ATD as neural transformations at an unspecified neural site, Equation (2.14) takes the following form:

$$\begin{pmatrix} A \\ T \\ D \end{pmatrix} = \begin{pmatrix} k_1 & 0 & 0 \\ 0 & k_2 & 0 \\ 0 & 0 & k_3 \end{pmatrix} \begin{pmatrix} 0.5967 & 0.3654 & 0 \\ 0.9553 & -1.2836 & 0 \\ -0.0284 & 0 & 0.0483 \end{pmatrix} \begin{pmatrix} L \\ M \\ S \end{pmatrix} \qquad (2.22)$$

where k_1, k_2 and k_3 are adaptation-dependent weightings (at absolute threshold conditions, their value is unity) and *LMS* are the Smith-Pokorny fundamentals, normalized to unity at their peaks:

$$\begin{pmatrix} L \\ M \\ S \end{pmatrix} = \begin{pmatrix} 0.2434 & 0.8524 & -0.0516 \\ -0.3954 & 1.1642 & 0.0837 \\ 0 & 0 & 0.6225 \end{pmatrix} \begin{pmatrix} X' \\ Y' \\ Z' \end{pmatrix} \qquad (2.23)$$

The space more used lately is Boynton's, which is also based on the Smith and Pokorny fundamentals, with the condition that *L* and *M* sum up to luminance (Boynton, 1986). The achromatic channel is, therefore, *L+M*. *T* and *D* are required to have neutral points at 570 and 498 nm respectively, assuming that these wavelengths correspond to the neutral points of the opponent red-green and yellow-blue LGN cells (Wiesel and Hubel, 1966; Derrington et al., 1984). The scaling condition on luminance automatically implies that $L=2M$ at 570 nm. So, if a neutral point is wanted at that wavelength, the T-channel must be defined as k_1L+k_2M with k_1=1 and k_2=-2. The D-channel is assumed to have a neutral point at 498 nm and to receive inputs of the form $k_1((L+M)-S)$. This determines the scaling constant for the S cones. All these consideration determine the new form taken by Equation (2.14):

$$\begin{pmatrix} A \\ T \\ D \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & -2 & 0 \\ 1 & 1 & -1 \end{pmatrix} \begin{pmatrix} L \\ M \\ S \end{pmatrix} \quad (2.24)$$

Boynton's ATD channels so defined (see Fig. 2.13) are consistent with the cardinal directions measured by Krauskopf et al. (Krauskopf et al., 1982) in LGN cells. Note that the main characteristic of this space is that the T channel does not re-emerge in the violet end of the spectrum, in clear disagreement with the spectral sensitivity determined by hue cancellation. This poses a strong problem for some color appearance predictions.

Opponent modulation space. The opponent modulation space is designed to describe incremental responses on a background. Let us consider a background stimulus defined by its (L_0, M_0, S_0) components in the cone excitation space. We aim to compute the amplitudes (*DA, DT, DD*) associated with an incremental stimulus (*DL, DM, DS*) on the background. The general solution to the problem can be written in a compact form as follows (Brainard, 1996):

$$\begin{pmatrix} \Delta A \\ \Delta T \\ \Delta D \end{pmatrix} = \begin{pmatrix} k_A & 0 & 0 \\ 0 & k_T & 0 \\ 0 & 0 & k_D \end{pmatrix} \begin{pmatrix} W_{A,L} & W_{A,M} & W_{A,S} \\ W_{T,L} & W_{T,M} & W_{T,S} \\ W_{D,L} & W_{D,M} & W_{D,S} \end{pmatrix} \begin{pmatrix} \Delta L \\ \Delta M \\ \Delta S \end{pmatrix} \quad (2.25)$$

The (*DA, DT, DD*) amplitude space or *opponent modulation space* is also known in the literature as *DKL-space,* after Derrington, Krauskopf and Lennie (Derrington et al., 1984). To compute the coefficients of the matrixes in Equation 2.25, the ATD channels are defined by the properties they must verify. For the A channel we will assume:

1. $A=L+M$
2. T does not respond to pure luminance changes and do not receive input from S cones.
3. D does not respond to pure luminance changes nor to changes leave both the S cone response and the achromatic mechanism response unchanged.

With these conditions, Equation 2.25 takes the following form:

$$\begin{pmatrix} \Delta A \\ \Delta T \\ \Delta D \end{pmatrix} = \begin{pmatrix} k_A & 0 & 0 \\ 0 & k_T & 0 \\ 0 & 0 & k_D \end{pmatrix} \begin{pmatrix} 1 & 1 & 0 \\ 1 & -\frac{L_0}{M_0} & 0 \\ -1 & -1 & \frac{L_0 + M_0}{S_0} \end{pmatrix} \begin{pmatrix} \Delta L \\ \Delta M \\ \Delta S \end{pmatrix} \qquad (2.26)$$

where k_i usually depends on the background. For an equal-energy white background, the *DKL* space has the same form as Boynton's except for a global minus sign in *D* and the scaling constants. The choice of the sign reflects that the majority of the opponent blue-yellow cells are blue-on (Zrenner and Gouras, 1981).

Different criteria, all equally arbitrary and therefore disputable, have been proposed to scale *DA*, *DT* and *DD*. When using the *DKL-space* in *COLORLAB*, the unscaled responses are given. No perceptual descriptors are computed in this space.

2.5.3.3. Two opponent stage models.

As we have seen in the previous section, linear ATD spaces focus either in the responses of an intermediate physiological level (Boynton) or in the perceptual mechanisms (Jameson and Hurvich, Ingling and Tsou, Guth et al.). Two opponent stage models consider two consecutive opponent stages after the cone stage: the first one has usually a physiological correlate with the LGN opponent cells and the second one simulates the final output of the visual process. *COLORLAB* functions `con2atd, atd2con, atd1atd2, xyz2atd` and atd2xyz can be used to transform data in CIEXYZ or in cone spaces to second stage opponent models.

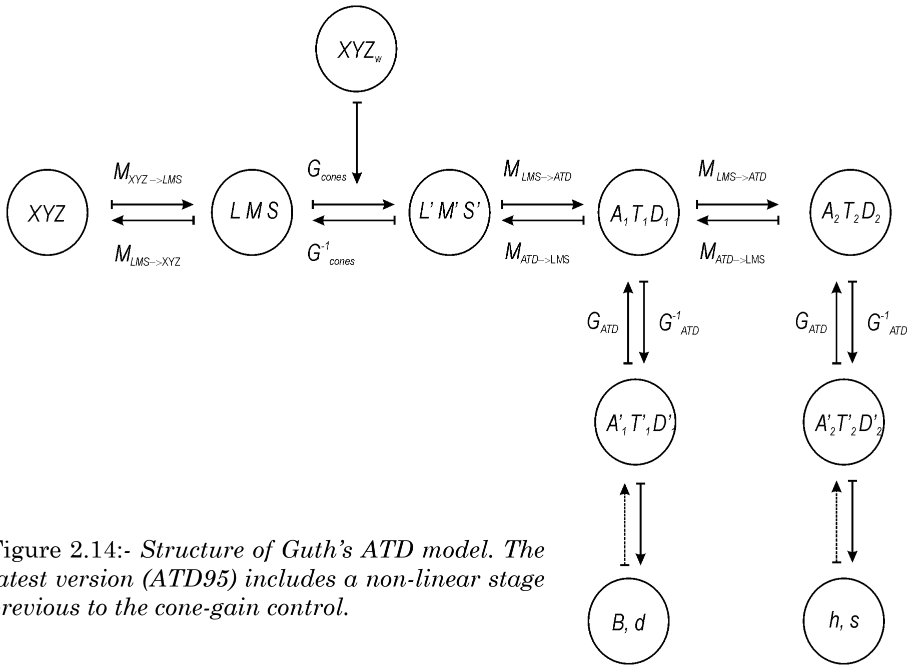


Figure 2.14:- *Structure of Guth's ATD model. The latest version (ATD95) includes a non-linear stage previous to the cone-gain control.*

Although some models with this structure are linear (for example, the model by De Valois and De Valois, 1993), *COLORLAB* includes only a well-tested non-linear model (Guth 1991-1995), whose structure is summarized in Figure 2.14. The tristimulus values of a color are transformed into linear cone responses, computed with the change of basis matrix corresponding to Smith and Pokorny's fundamentals normalized to one in their peaks. The cone responses undergo a set of non-linear transforms that include information about the illuminant. Linear first-stage opponent responses, (A_1 T_1 D_1), are computed from the non-linear cone outputs. After going through a non-linear stage, the first-stage opponent responses are combined to yield the brightness:

$$B = \sqrt{A_1^2 + T_1^2 + D_1^2} \qquad (2.27)$$

Perceptual distances are computed as Euclidean distances between the non-linear ATD responses. A linear transform on (A_1 T_1 D_1) yields the second-opponent stage responses, (A_2 T_2 D_2). After going through the same non-linearity as the first-stage responses, the remaining perceptual parameters (hue and saturation) are computed as:

$$H = atan\frac{D'_2}{T'_2} \qquad (2.28)$$

$$S = \frac{\sqrt{T'^{2}_{2} + D'^{2}_{2}}}{A'_2}$$

This non-linear color vision model reproduces correctly the basic perceptual phenomena of isolated colors, in particular the luminance dependence of the perceptual parameters. Its performance for related colors is not so good.

2.5.3.4. Appearance models

Although most of the models we have described above give some numerical quatization of color appearance, the name of *color appearance models* is usually applied to a set of models whose first objective is to describe the color appearance of related colors. The general structure of these models reflect the different processes of the visual system but, unlike ATD models, the different intermediate stages do not necessarily have physiological of psychophysical correlates.

Basically (Figure 2.15) the inputs of a color appearance model are the tristimulus values of the problem color and those of the illuminant. Information about the surround of the problem color can also be included in the most advanced models (*CIECAM97*, for instance). There follows a cone stage, with some sort of adaptation mechanism. The cone responses are transformed into an achromatic, a red-green and a blue-yellow opponent response that are finally combined to yield the perceptual descriptors (brightness, *Q*, colorfulness, *M,* saturation, *s*, and hue angle *h*, for any color and for related colors also lightness *J* or *L*,* and chroma, *C*)

COLORLAB includes two simple color appearance models, CIELAB and CIELUV, and more recent models such as LLAB, RLAB and CIECAM97 (see Capilla et al, 2002ab and Fairchild 1998). The older models, particularly CIELUV, differ considerably from the general structure described in Figure 2.15. Although it has an adaptation stage, the opponent stage is missing. CIELAB has a linear adaptation stage and an opponent transform. More recent models include non-linear adaptation mechanisms.

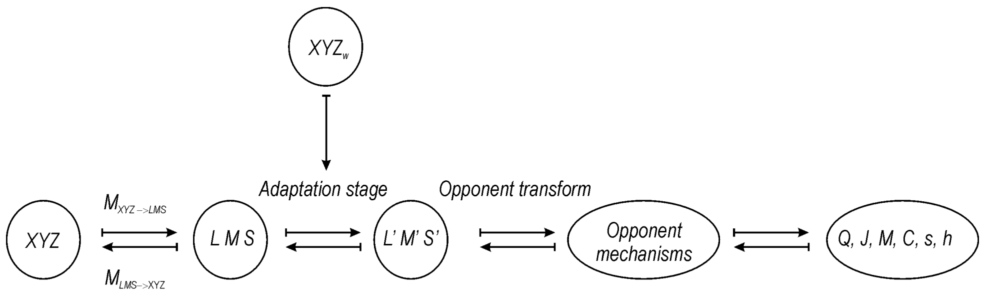


Figure 2.15: *General structure of a color appearance model.*

2.5.4. Color and color images in *MATLAB* and *COLORLAB*

Color reproduction in CRTs.

To obtain	**From**	**Use**
CRT phosphors chromaticities and luminance vs. digital level curves	Measurements with a fotocolorimeter	`calibrat`
A *COLORLAB* format file with the data defining a CRT display		`savemon`
Digital values	Tristimulus values and CRT data	`tri2val`
Tristimulus values	Digital values and CRT data	`val2tri`
An indexed image and palette	A true-color image	`true2pal`
A true-color image	An indexed image and palette	`pal2true`

2.6 Data structures in *COLORLAB*

COLORLAB has fixed formats for the more usual classes of variables: spectral and color data, color images and monitor calibration data. These formats must be scrupulously respected when calling a *COLORLAB* function. Otherwise, scalar products, matrix products and other internal operations of the functions shall not work properly.

2.6.1. Spectral data (reflectances, radiances and color matching functions)

Any magnitude depending solely on wavelength is included under the epigraph *spectral data*. Spectral transmittances and reflectances of objects, spectral radiances of light sources and color matching functions are typical examples.

Spectral data corresponding to *M* objects, measured for *N* wavelengths of the visible spectrum are entered as *N*(M+1)* matrixes. The first column contains the *N* wavelengths. Column *i+1* contains the data corresponding to object *i*. With this criterion, color matching functions are *N*4* matrixes. The first column contains the wavelengths whereas each of the remaining columns corresponds to one of the three primaries of the color space.

2.6.2. Color data (colors, trichromatic units and colormaps)

Sets of *N* colors in a tristimulus color space are described by *N*3* matrixes where each row contains a different color, each column is a different descriptor. Many advanced appearance models use more than three parameters to describe color appearance. In these models, colors are *N*P* matrixes, *P* being the number of appearance descriptors of the model. If a *N*3* matrix is meant to be a colormap to be used with `imshow` or `image`, remember that it must contain digital values between 0 and 1.

Trichromatic units (the luminance of each of the primaries of the color system used) are *1*3* vectors and therefore conform to the same structure as colors, although their physical meaning is different.

2.6.3. Color images

Images are represented discreet arrays of *N*M* numbers. To possible formats are available. In the first, the tristimulus values of each of the *N*M* points of the image are specified. The image is described by a *N*M*3* three-dimensional matrix. The number stored at the position (*i,j,k*) of the matrix represents color descriptor *k* (*k=1,2,3*) of the stimulus placed at the spatial position (*i,j*) of the image.

The second option is to specify a list of colors of the image (a *C*3* matrix called *palette*) and a *N*M* matrix (*indexed image*) indicating which of the *C* colors of the palette appears at the position of coordinates (*i,j*) in the image.

2.6.4. Monitor calibration data

To transform colorimetric descriptors of a stimulus into digital values, the following information about the current CRT monitor is needed:

- Chromaticities of the gun, *i*, as a function of the digital step, n_i.
- Parameters (a_i and g_i) of the gamma relation between the luminance of the gun, *i*, and the digital value, n_i (see Equation 2.6).

These data are obtained by measuring the chromaticity and the luminance of each of the three monitor guns at *N* different digital levels. The chromaticities of the guns are stored in *7*N* matrixes. The first row contains the values n_i, the second and third row contain the chromaticities of the red gun for those digital steps in the color reference system specified, the fourth and fifth those of the green gun and the sixth and seventh those of the blue gun. By default, these variables are called *tm* The parameters of the gamma relation are scalars; their default names are *a* and *g*.

2.7 *COLORLAB* databases

COLORLAB includes different useful data sets, comprising standard illuminants, the Munsell Atlas, color matching functions and trichromatic units of several standard colorimetric systems, color sets and complex images.

By default, these data sets are in the directory `\\colorlab\colordat`.

2.7.1. Illuminants and reflectances

Folder `\\colorlab\colordat\illumin` contains a set of standard illuminants and some non-standard cut-off and broadband illuminants. All illuminants are stored in mat files in *spectral data format*.

The standard illuminants are stored in files with the following name-structure:

`illuminan.name`

The extension `name` (three characters) is the standard name of the illuminant (A, B, C, D+color temperature, etc.). Both daylight and fluorescent illuminants are available.

The non-standard illuminants are stored in `*.mat`. A list of all available illuminants appears in the previous table.

Standard illuminants	**Non-standard illuminants**
iluminan.a	amar573.mat
iluminan.b	azul482.mat
iluminan.c	cyan500.mat
iluminan.d10	e2.mat
iluminan.d40	m_amar.mat
iluminan.d50	m_azul.mat
iluminan.d55	m_rojo.mat
iluminan.d65	m_verde.mat
iluminan.d75	naran585.mat
iluminan.f11	purp500.mat
iluminan.f2	purp559.mat
iluminan.f7	rojo672.mat
	tubo.mat
	unamaril.mat
	unazul.mat
	unblanco.mat
	uncyan.mat
	unmorao.mat
	unrojo.mat
	unverde.mat
	verde526.mat
	verde555.mat

COLORLAB also comes with a database of reflectances corresponding to the color samples of the Munsell Book of Color. In this book the samples are classified according to Hue, Chroma and Value (Lightness).

The files are stored in folder `\\colorlab\colordat\reflect\munsell`. The notation used in the files is as follows:

* The files are organized in hue-named folders:

```
r,  yr,  y,  gy,  g,  bg,  b,  pb,  p,  rp
```

* Inside each folder, the name of the files is made of 8 numbers:

```
HHHHVVCC
```

The first 4 numbers (`HHHH`) mean HUE. They may have values in the range `HHHH`=0000..0250....0500....1000.

Going from 0000 to 1000 means fine change from one coarse hue to the next one (for example from 'green' -g- to green-bluish -bg-).

The next 2 numbers (`VV`) mean VALUE. They may have values in the range `VV`= 00, 10, 20, 30, 40, ...,90.

The last 2 numbers (`CC`) mean CHROMA. They may have values in the range `CC`= 00, 01, 02 ... 16.

2.7.2. Color matching functions and systems of primaries

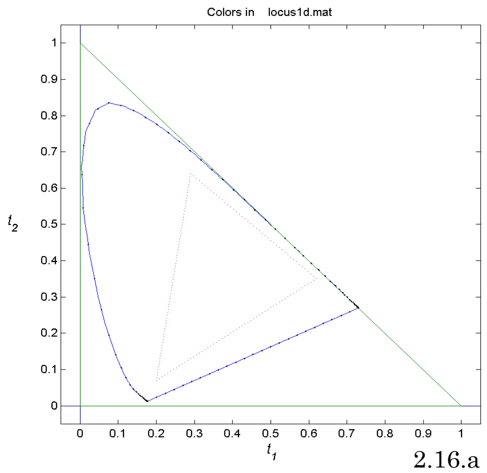
The `*.mat` files in directory `\\colorlab\colordat\systems` contain the color matching functions, the trichromatic units and a change-of-basis matrix of different standard color reference systems, including the CIERGB and the CIEXYZ systems, and some non-standard ones.

2.7.3. Interesting colors

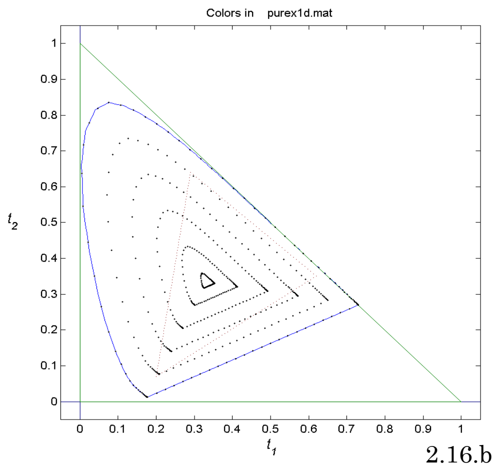
The folder `\\colorlab\colordat\colors` contains these files:

- `locus1cd.mat` contains the locus of spectral colors with uniform sampling in wavelength (not in the purple region). The luminance is 1 cd/m^2. (Fig. 2.16.a)
- `purex1cd.mat` contains locus of colors with different excitation purities [0.05 0.2 0.4 0.6 0.8 1] and uniform sampling in wavelength (not in the purple region). The luminance is set to 1 cd/m^2. (Fig. 2.16.b)

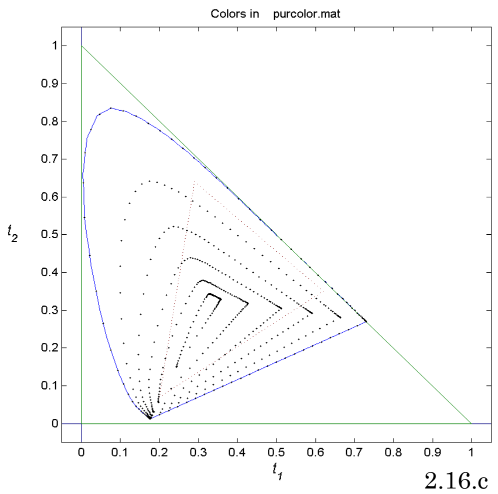
- `purcolor.mat` contains locus of colors with different colorimetric purities [0.05 0.2 0.4 0.6 0.8 1] and uniform sampling in wavelength (not in the purple region). The luminance is set to 10 cd/m^2. (Fig. 2.16.c)
- `macadam.mat` contains the MacAdam ellipsoids of chromatic discrimination from 25 selected colors in the chromatic diagram. The luminance is set to 50 cd/m^2 according to the experimental conditions (Fig. 2.16.d).



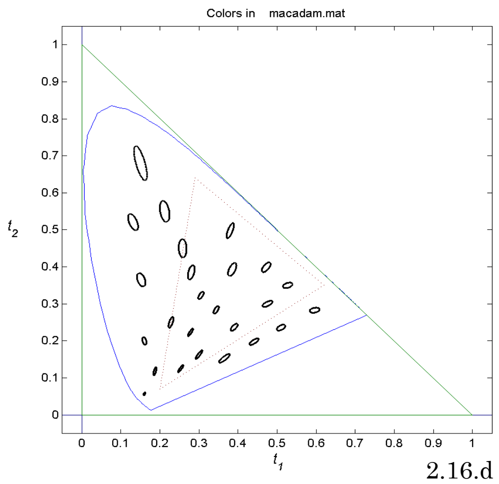
2.16.a



2.16.b



2.16.c



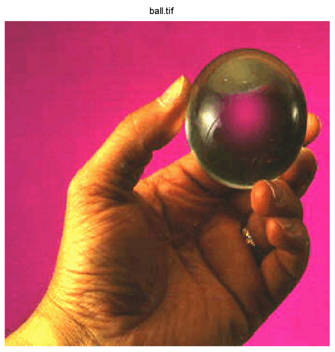
2.16.d

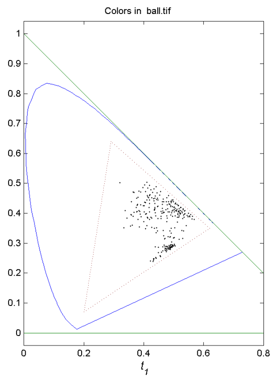
2.7.4. Images

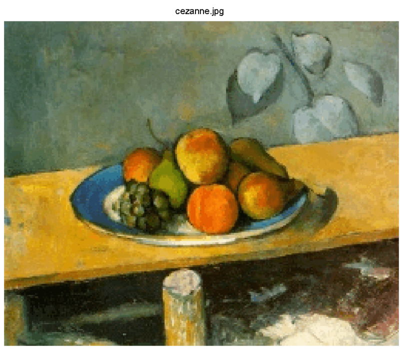
COLORLAB comes with two kinds of images:

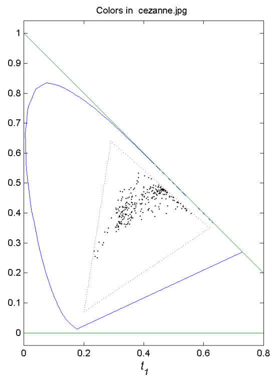
- Standard digital images that you can read with `imread.m` and display with `image.m` (`*.tif` and `*.jpg` files in).
- Mosaics (`*.mat` files in). The mosaics are just *MATLAB* files containing two variables: `palre` and im. The first one, `palre`, is a spectral-like variable containing N reflectances (is a palette of reflectances). The second, `im`, is the *mosaic*. It is just as an indexed image, but here the indices refer to the reflectances in `palre`. Mosaics don't have fixed tristimulus values because they depend on how they are illuminated. Once you have loaded the data (using `load.m`) you have to obtain the tristimulus values using some illuminant and `spec2tri.m`. Then you have to obtain the device dependent characterization (`tri2val.m`) before using `image.m`.

Standard images:

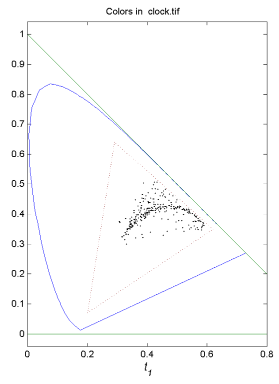




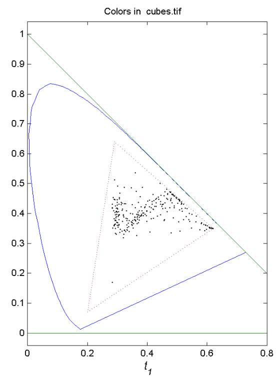




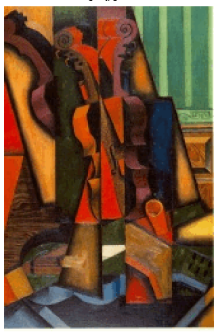
clock.tif

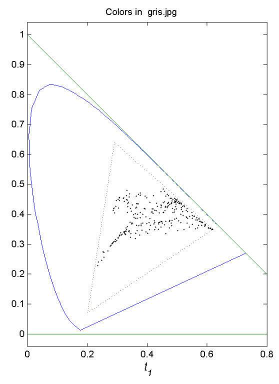


cubes.tif



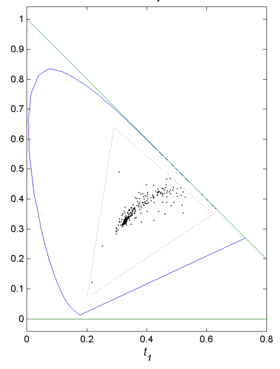
gris.jpg





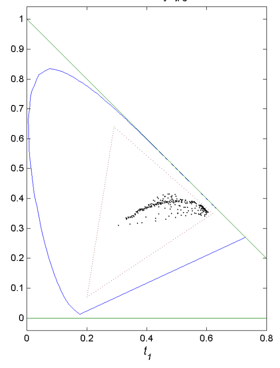
keys.tif

Colors in keys.tif



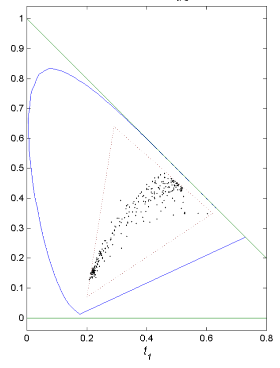
marilyn.jpg

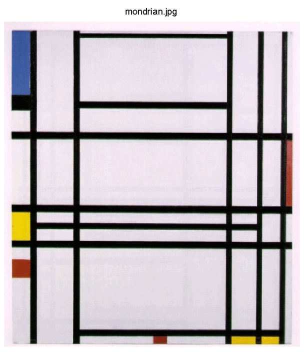
Colors in marilyn.jpg

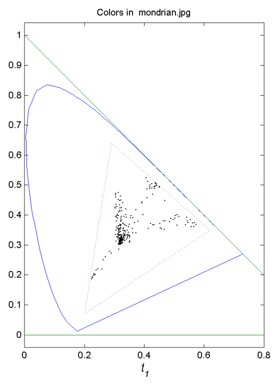


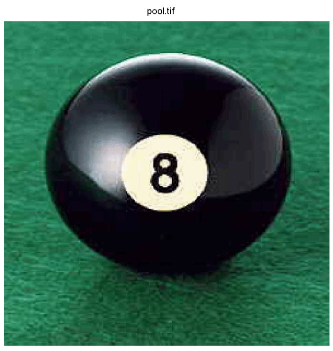
matisse.jpg

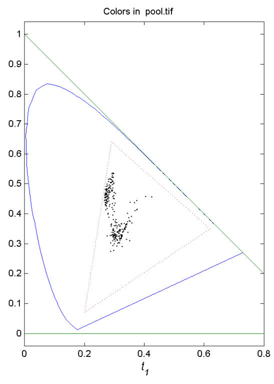
Colors in matisse.jpg

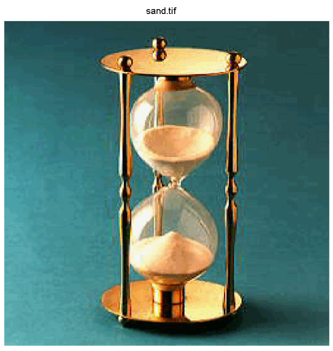


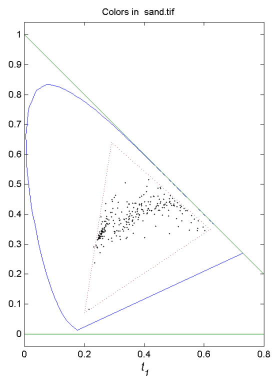

mondrian.jpg

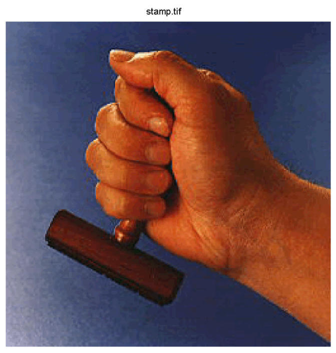

Colors in mondrian.jpg
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.2
0.4
0.6
0.8
t_1

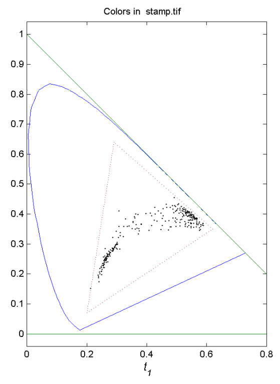

pool.tif

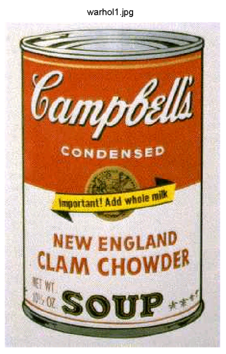

Colors in pool.tif
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.2
0.4
0.6
0.8
t_1

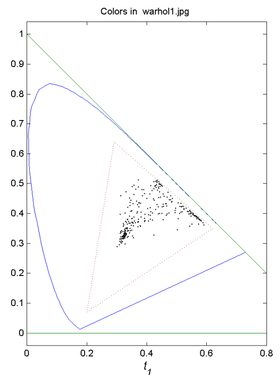

sand.tif

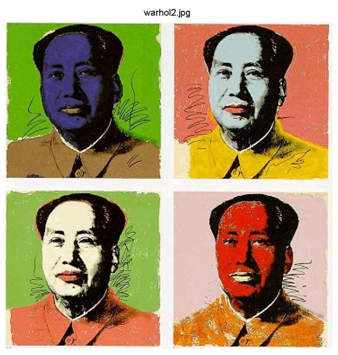

Colors in sand.tif
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.2
0.4
0.6
0.8
t_1

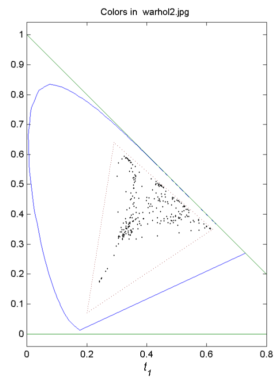

stamp.tif


Colors in stamp.tif
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.2
0.4
0.6
0.8
t_1


warhol1.jpg
Campbell's
CONDENSED
Important! Add whole milk
NEW ENGLAND
CLAM CHOWDER
NET WT 10¾ OZ
SOUP

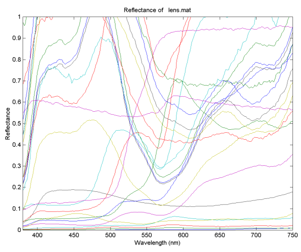

Colors in warhol1.jpg
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.2
0.4
0.6
0.8
t_1

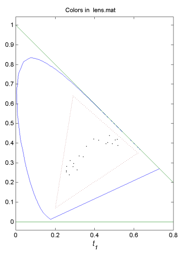

warhol2.jpg

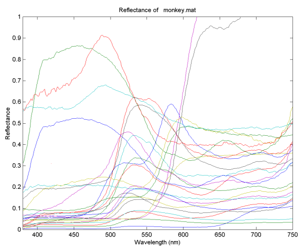

Colors in warhol2.jpg
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.2
0.4
0.6
0.8
t_1

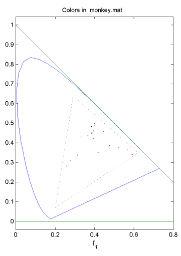
Mosaics:

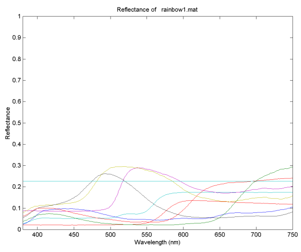
These mosaics are illuminated with a equienergetic white of 150 cd/m2.



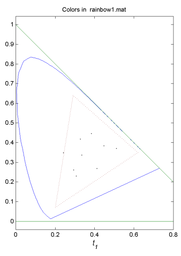


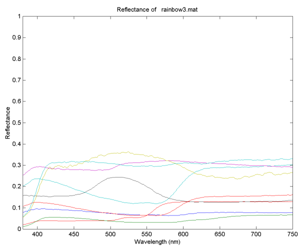




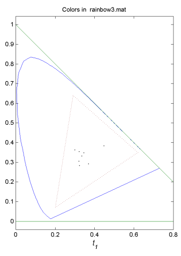


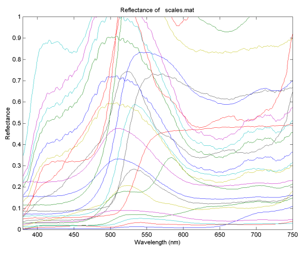
Mosaic: rainbow1.mat

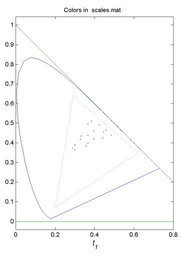


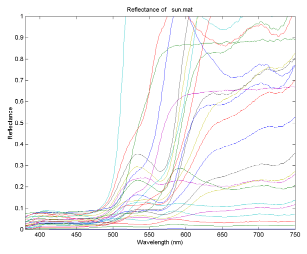


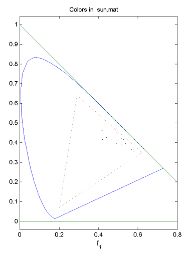
Mosaic: rainbow3.mat



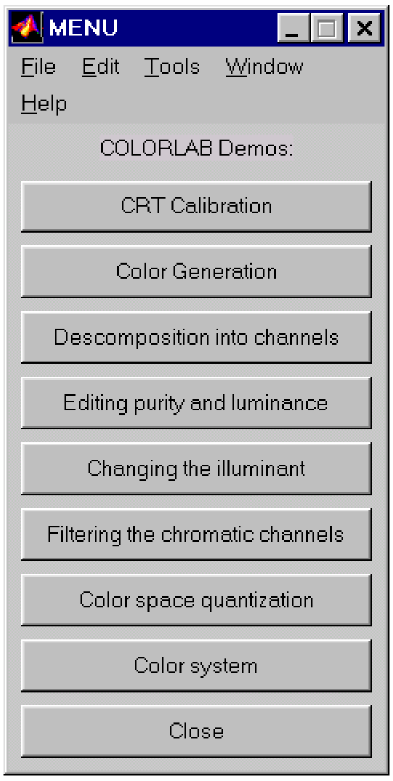








2.8 *COLORLAB* demos



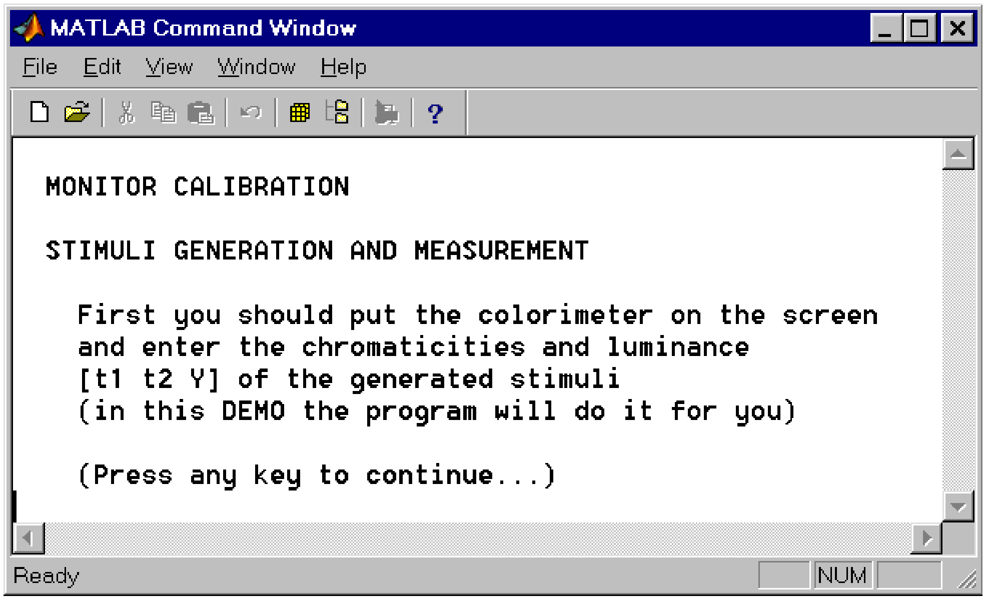
COLORLAB includes a set of demos to help gain a feeling of the possibilities of the library to solve complex color problems. Type `colorlab` at the command line. This function displays a menu with the available demos.

2.8.1. CRT monitor calibration

If you wish to obtain colorimetrically precise stimuli, you need to obtain information about your CRT monitor. The basis of the procedure is reviewed in Sections 2.4.4. and 2.4.5. Function `calibrate` generates the stimuli necessary for monitor calibration, stores and processes the measured data. This demo shows how this function works.

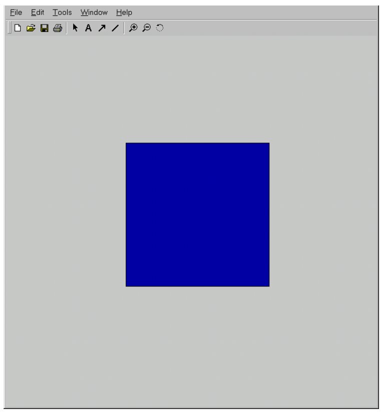
After selecting *CRT Calibration* from the main menu, a submenu appears offering two possibilities: simulating the calibration process (*Calibration Demo)* or viewing the monitor data obtained in such a process (*Monitor Data*).

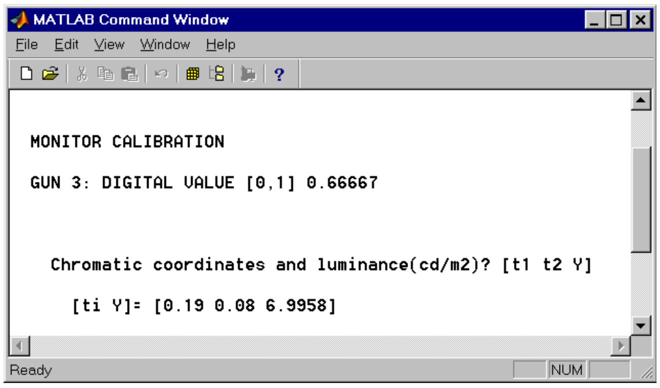
If the *Calibration Demo* option is entered, the following message appears at the Matlab Command Window:



```
MONITOR CALIBRATION

STIMULI GENERATION AND MEASUREMENT

  First you should put the colorimeter on the screen
  and enter the chromaticities and luminance
  [t1 t2 Y] of the generated stimuli
  (in this DEMO the program will do it for you)

  (Press any key to continue...)
```

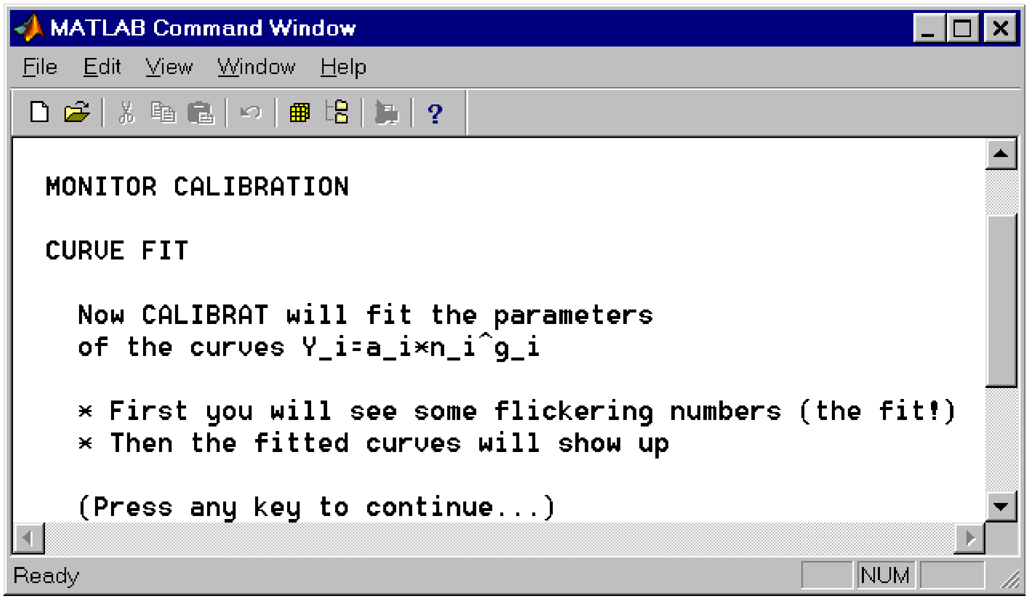
After any a key is pressed, the measurement configuration used by `calibrat`, a colored square against a grey background appears. The color of the square, obtained with a single phosphor (red, green and blue, in succession) is controlled by the program. In a real calibration process, the user should measure the chromaticity and the luminance of the stimulus using a telecolorimeter or a chromaticity probe. The data should be entered in the command line of the program and the next chromaticity presented. The demo simulates the entire process, presenting four stimuli for each phosphor and other four for white stimuli generated with equal digital levels of the three phosphors.





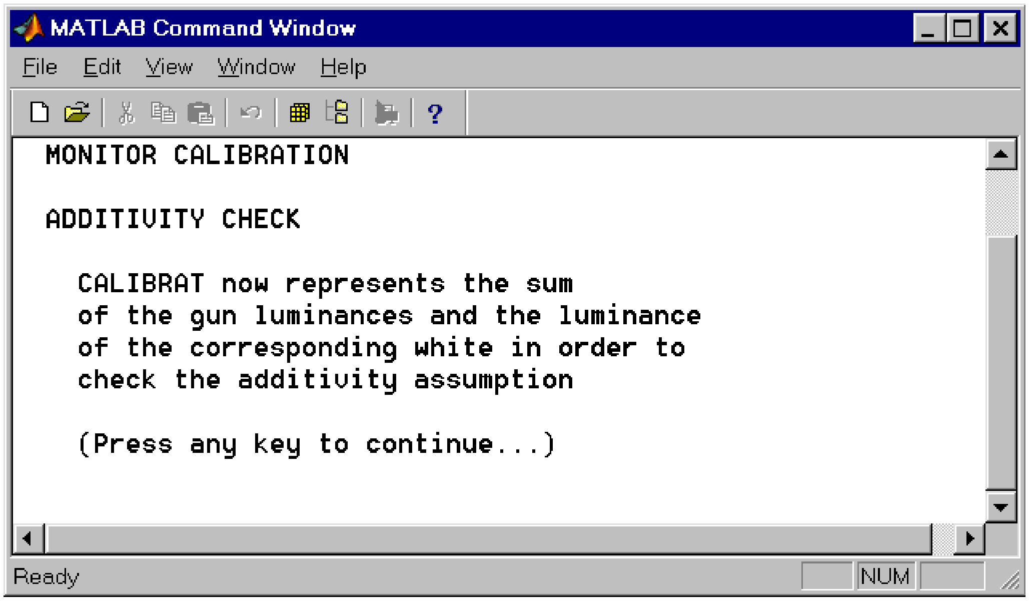
```
MONITOR CALIBRATION

GUN 3: DIGITAL VALUE [0,1] 0.66667


  Chromatic coordinates and luminance(cd/m2)? [t1 t2 Y]

    [ti Y]= [0.19 0.08 6.9958]
```

Once the measurement stage is over, the parameter of the gamma curves fitting the experimental data are obtained:



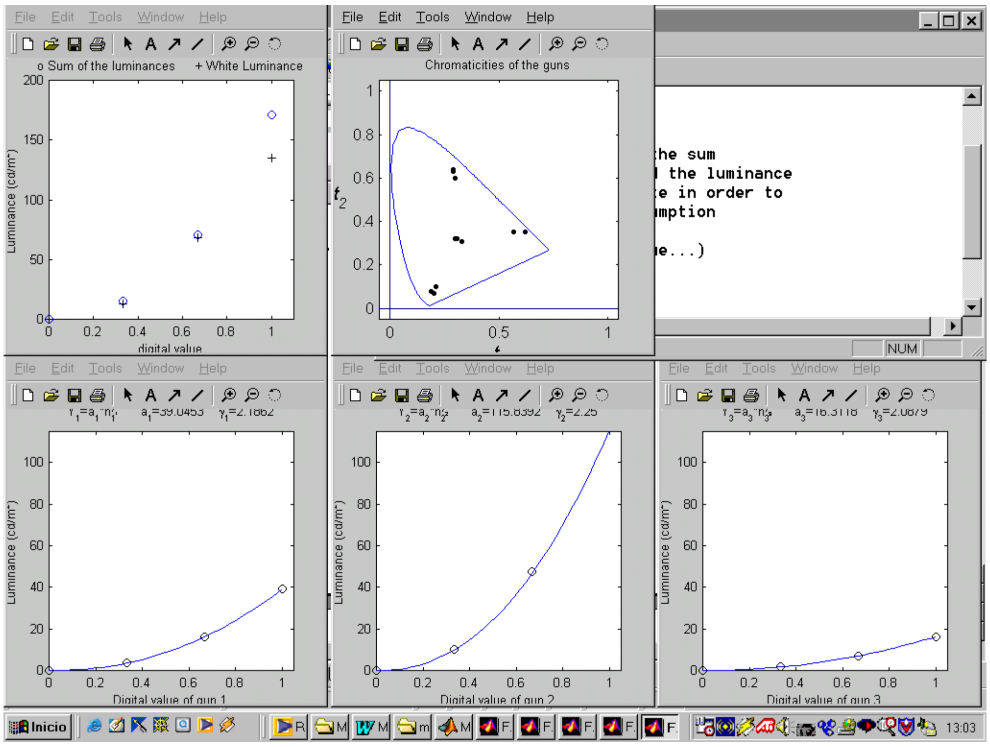
```
MONITOR CALIBRATION

CURVE FIT

  Now CALIBRAT will fit the parameters
  of the curves Y_i=a_i*n_i^g_i

  * First you will see some flickering numbers (the fit!)
  * Then the fitted curves will show up

  (Press any key to continue...)
```

The process ends with an additivity check.



```
MONITOR CALIBRATION

ADDITIVITY CHECK

  CALIBRAT now represents the sum
  of the gun luminances and the luminance
  of the corresponding white in order to
  check the additivity assumption

  (Press any key to continue...)
```

The program displays five figures in all. The three first ones show the luminances of the CRT guns as a function of the digital values. The circles represent the data measured by the user. The blue lines are fits the result of fitting these data to the Equation 2.6. The fourth figure plots as circles the predicted luminance of the white stimulus (sum of the luminances of the three isolated guns) and as crosses the measured luminance as a function of the digital level. The better the agreement between these two point sets, the more additive the CRT is.



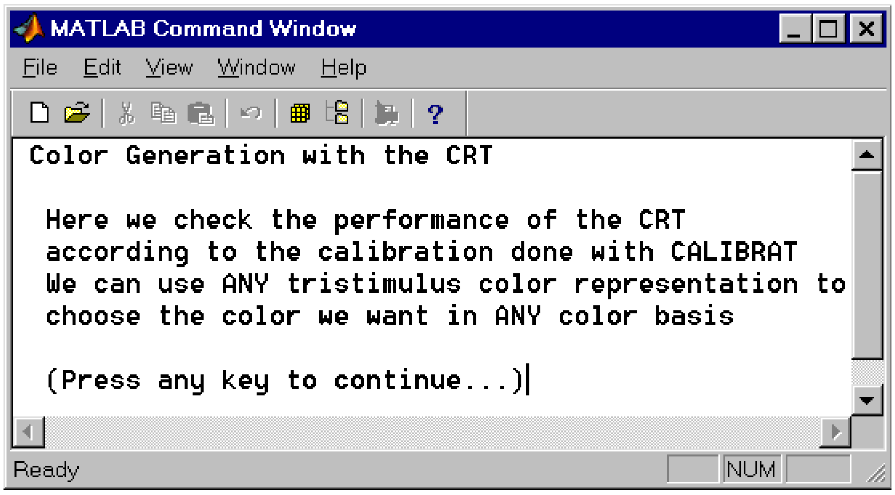
```
he sum
 the luminance
e in order to
mption

e...)
```

The last figure is concerned with the dependence between chromaticity and digital level. In ordinary monitors, the chromaticity of the guns depends strongly on luminance, and therefore in digital level.

This set of figures (except for the additivity curve and the experimental points in the gamma curves) are shown if the option *Monitor data* is selected in the demo menu.

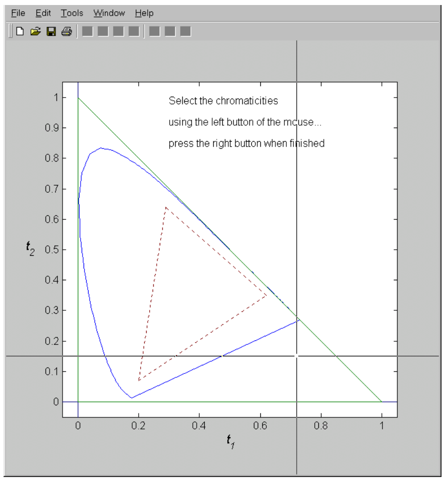
2.8.2. Color generation

Once the CRT is calibrated, it is useful to test the accuracy with which we can generate stimulus. This is the aim of the *Color generation* demo.



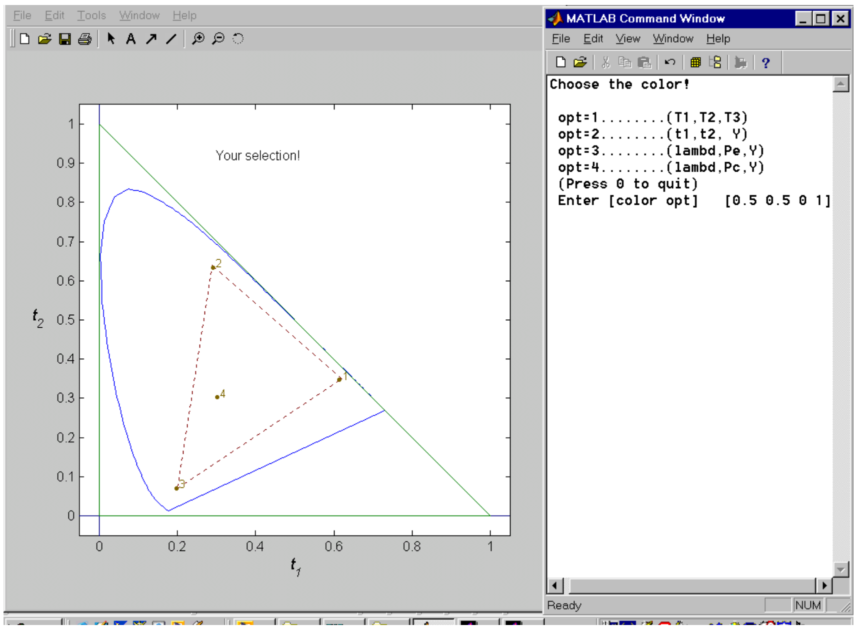
```
Color Generation with the CRT

 Here we check the performance of the CRT
 according to the calibration done with CALIBRAT
 We can use ANY tristimulus color representation to
 choose the color we want in ANY color basis

 (Press any key to continue...)
```

The first step consists in defining the basis (primaries and white) of the color reference system we are going to use to describe the stimuli we want to reproduce. The primaries are graphically defined:



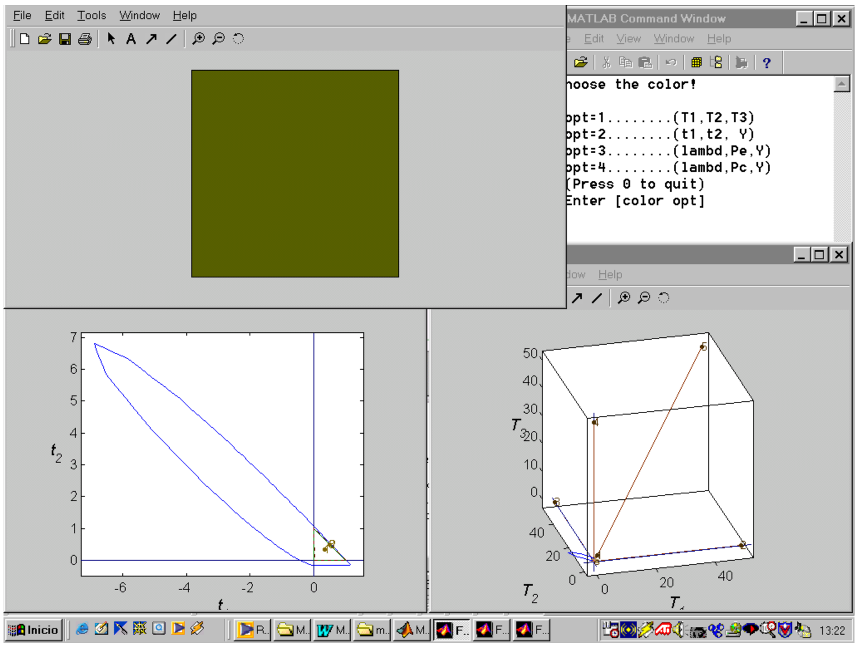
Select the chromaticities
using the left button of the mouse...
press the right button when finished

Now it only remains to enter the description of the stimulus we want to reproduce, using any set of descriptors among those available in tristimulus colorimetry.



```
Choose the color!

 opt=1........(T1,T2,T3)
 opt=2........(t1,t2, Y)
 opt=3........(lambd,Pe,Y)
 opt=4........(lambd,Pc,Y)
 (Press 0 to quit)
 Enter [color opt]   [0.5 0.5 0 1]
```

The program then shows the stimulus and represents it in the chromaticity and the tristimulus space. The chromaticity and luminance of the stimulus can be measured and the accuracy of the reproduction process assessed.

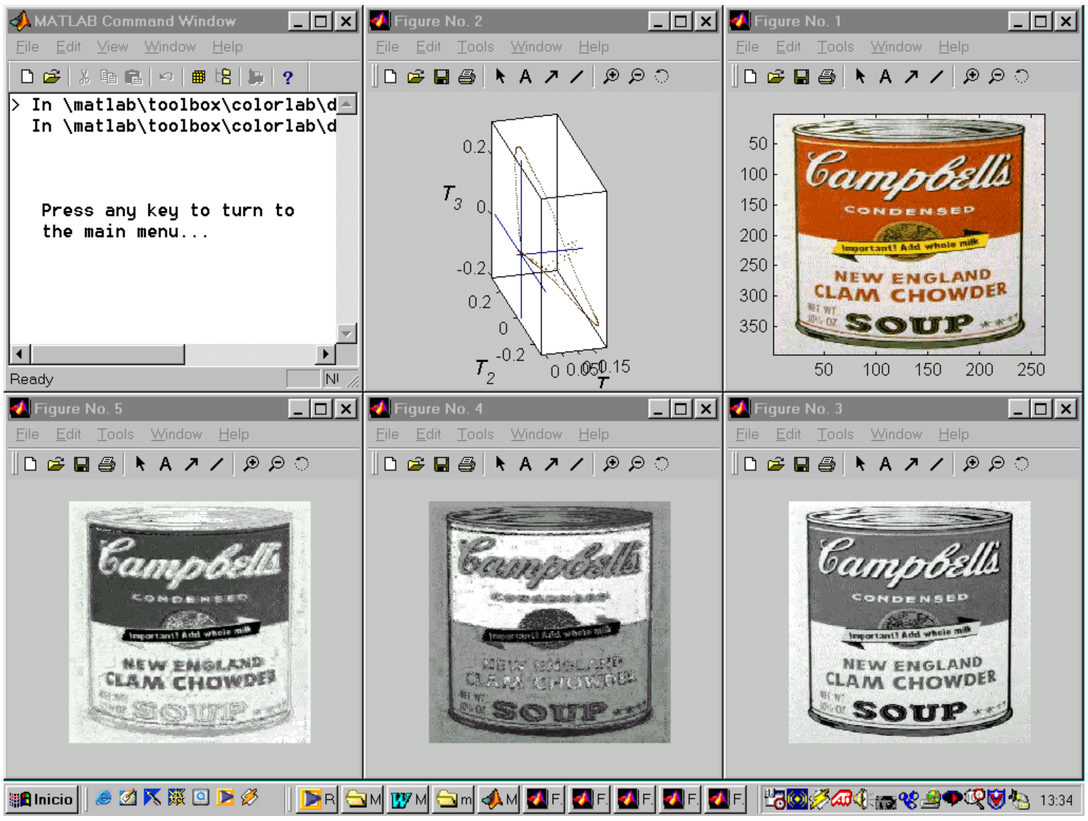


```
hoose the color!

opt=1........(T1,T2,T3)
opt=2........(t1,t2, Y)
opt=3........(lambd,Pe,Y)
opt=4........(lambd,Pc,Y)
(Press 0 to quit)
Enter [color opt]
```

2.8.3. Decomposition of color images in color channels

The *Editing purity and Luminance* demo shows how to decompose an image into color channels. First, we select an image in TIFF or JPEG formats. The image is described in a color representation space whose primaries we may define graphically or numerically among a set of standards. The descriptors of the image in the chosen representation are computed, coded as grey levels and displayed as three separate images. The image colors are displayed in the color space for reference.

The example shown below corresponds to the ATD descriptors in Guth's ATD95 model is used. Note that *A* is the luminance, *T* is the red-green mechanism and *D* the blue-yellow mechanism. The orange strip in the image, for instance, is coded as having low luminance (grey in Figure 3 of the mosaic), red component (white in Figure 4 of the mosaic) and yellow component (grey in Figure 5 of the mosaic).

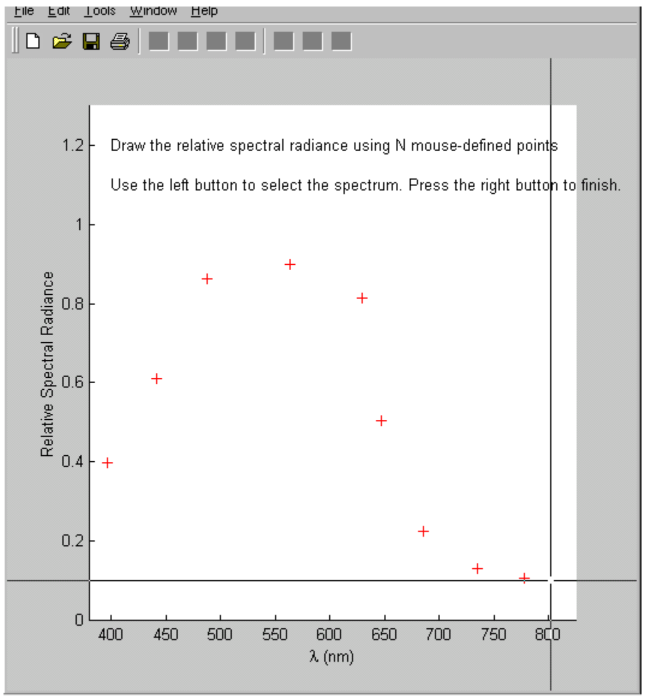


```
> In \matlab\toolbox\colorlab\d
  In \matlab\toolbox\colorlab\d


   Press any key to turn to
   the main menu...
```

2.8.4. Changing the illuminant

In this demo, an image is changed into a mosaic, where each pixel has a reflectance. The scene is illuminated with the chosen illuminant and the result displayed.

The illuminant is graphically defined by the user. The luminance is first numerically entered. A *relative spectral radiance vs. wavelength* plot appears, where the user may define the relative shape of the spectral radiance of the illuminant by selecting *N* points. The radiances at the remaining points are obtained by interpolation and the entire curve scaled to yield the desired luminance.



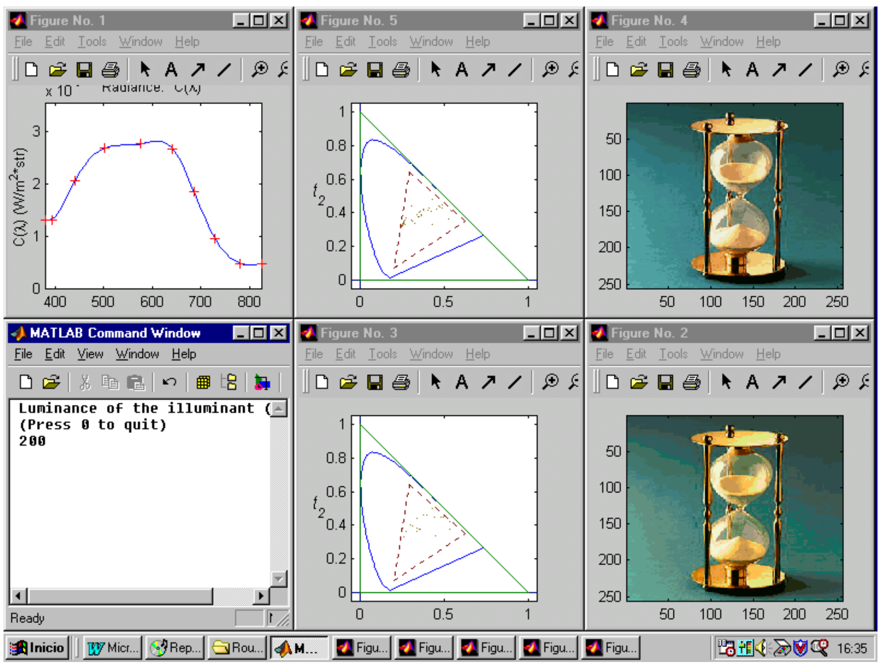
Draw the relative spectral radiance using N mouse-defined points

Use the left button to select the spectrum. Press the right button to finish.

Equation 2.3 is used to compute the tristimulus values of the image under the new illuminant. The resulting image is displayed and the colors plotted in a chromaticity diagram.

Note that a human observer would not always perceive the image as displayed with this method. The phenomenon of color constancy, the mechanism of the visual system that keeps color appearance approximately constant under illuminant changes, is not included in the algorithm used.

In the example figure below, for instance, the illuminant is greenish light and the chromaticity of the image is shifted towards the chromaticity coordinates of the illuminant. A human observer adapted to the illuminant would perceive the image approximately equal to the original one.



```
Luminance of the illuminant (
(Press 0 to quit)
200
```

2.8.5. Changing the color appearance of images

This program demonstrates how the final appearance of an image may be altered in the desired direction by manipulating its colorimetric parameters, particularly its colorimetric purity and luminance.

In particular, once an image has been loaded, its mean luminance and colorimetric purities are computed. The colors in the image are represented in the xy chromaticity diagram and in the XYZ color space. Two sliders allow to change separately these parameters.

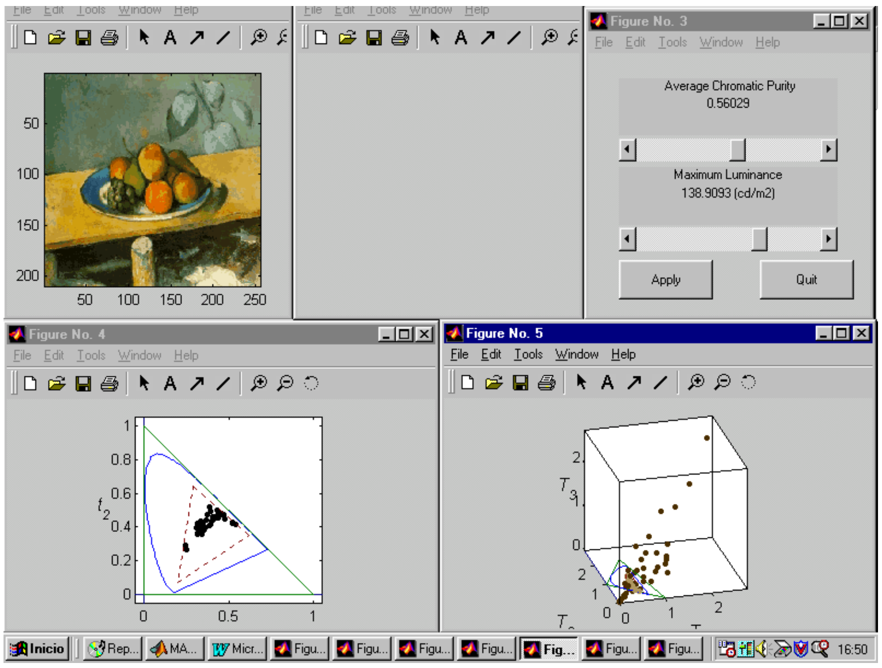


Figure No. 3

Average Chromatic Purity
0.56029

Maximum Luminance
138.9093 (cd/m2)

The resulting image is displayed, along with the new mean colorimetric purity and luminances and the new chromaticities in the color diagram and the color space. The original image is included for comparison.

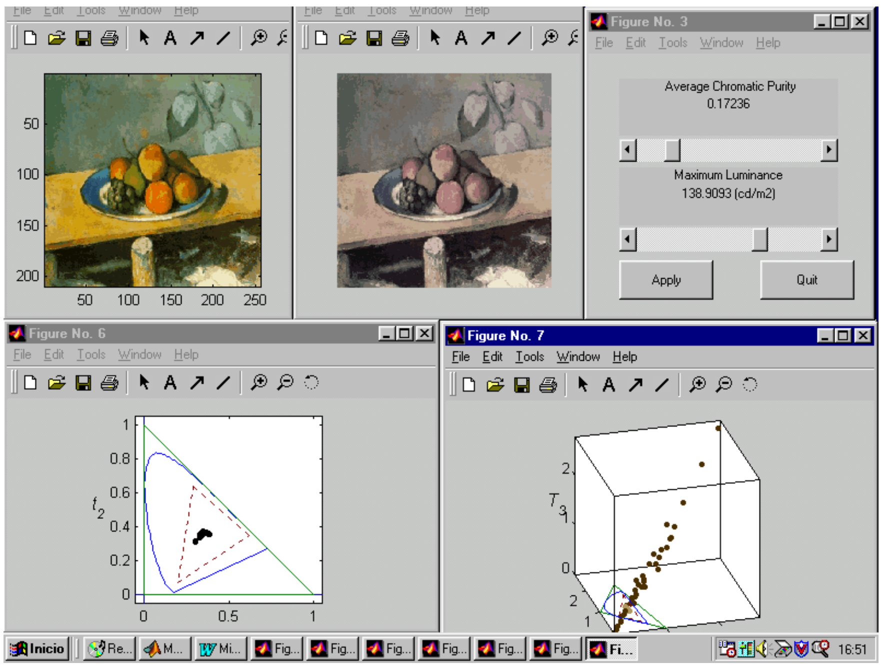


Figure No. 3

Average Chromatic Purity
0.17236

Maximum Luminance
138.9093 (cd/m2)

In the example, we have kept the luminance and decreased the colorimetric purity. The result is a far less saturated image than the original.

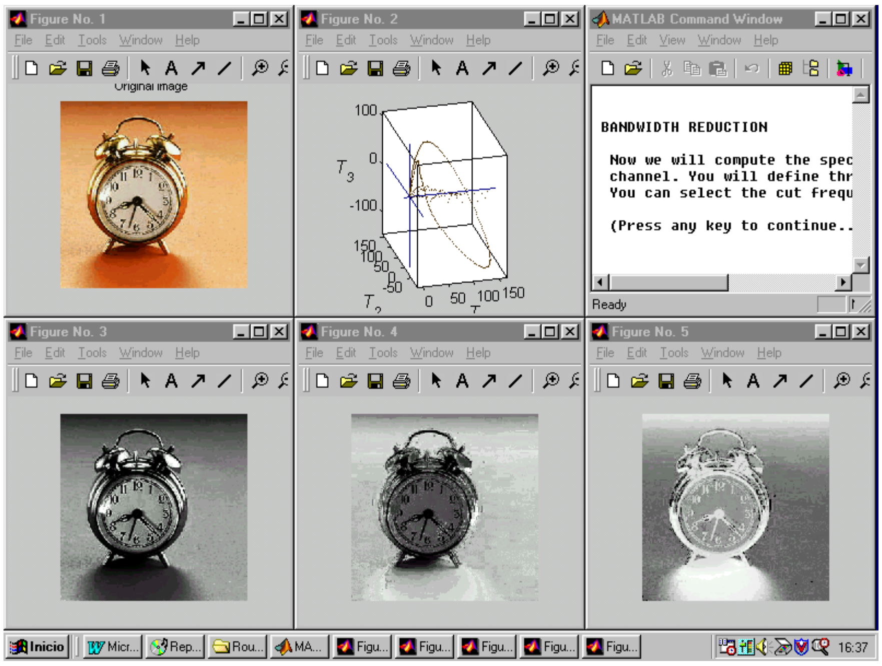
Although the computations are performed in the CIEXYZ space, the procedure may be easily used in any color space.

2.8.6. Color image compression

The *Filtering the chromatic channels* demo offers an example of how an efficient procedure for image compression can be reached using an color system or model that can separate the chromatic information from the achromatic.

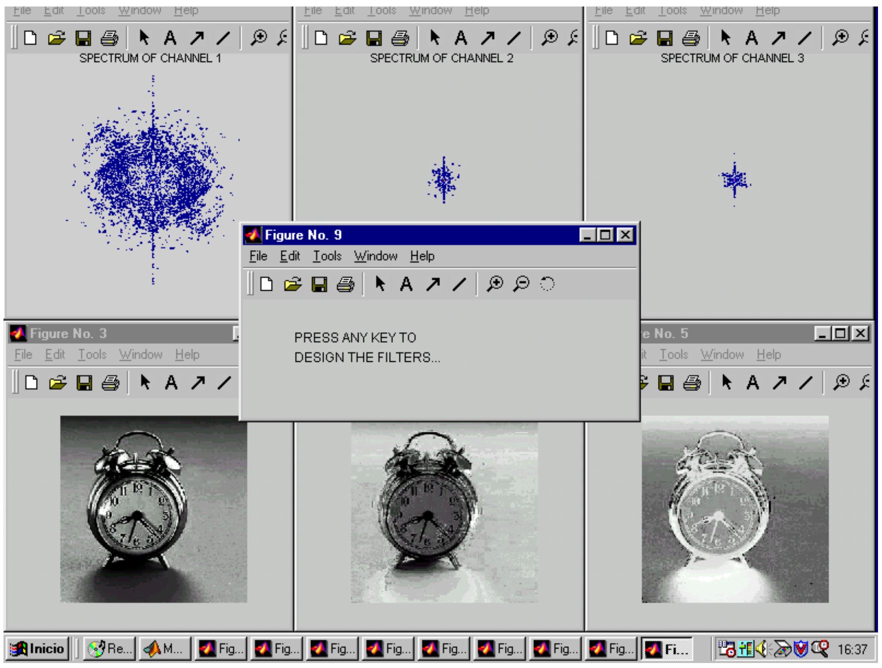
It is a known psychophysical fact that contrast discrimination is poorer at the chromatic mechanisms than in the achromatic one. Therefore, more chromatic information can be removed from the image without serious distorting effects that is the case with achromatic information.

The procedure is as follows. The problem image is loaded and described in the chosen color space (YIQ in the following example). The image is then decomposed in the three channels of the model or color space used.



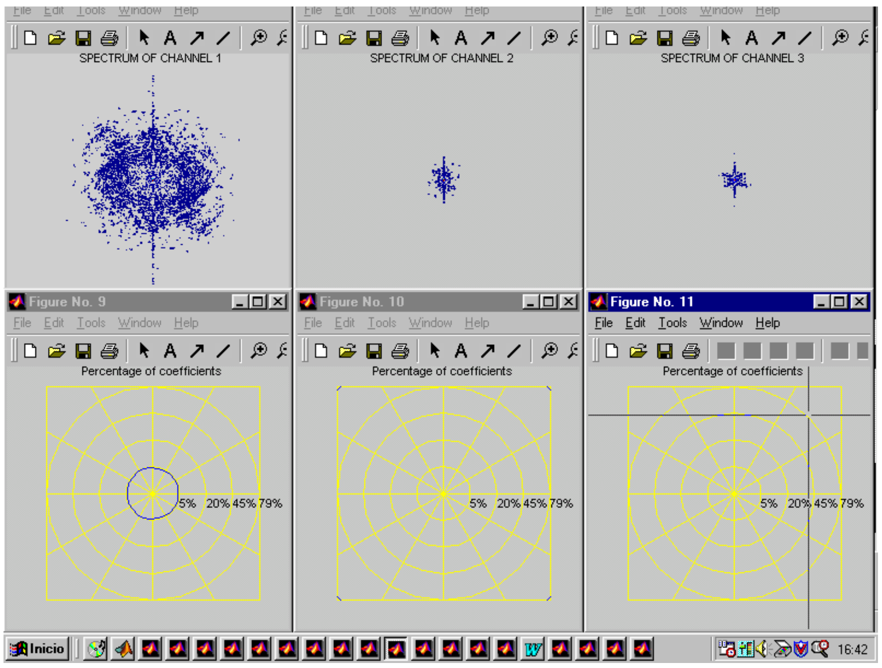
```
BANDWIDTH REDUCTION

 Now we will compute the spec
 channel. You will define thr
 You can select the cut frequ

 (Press any key to continue..
```

The Fourier Transform of each of the three images is computed:

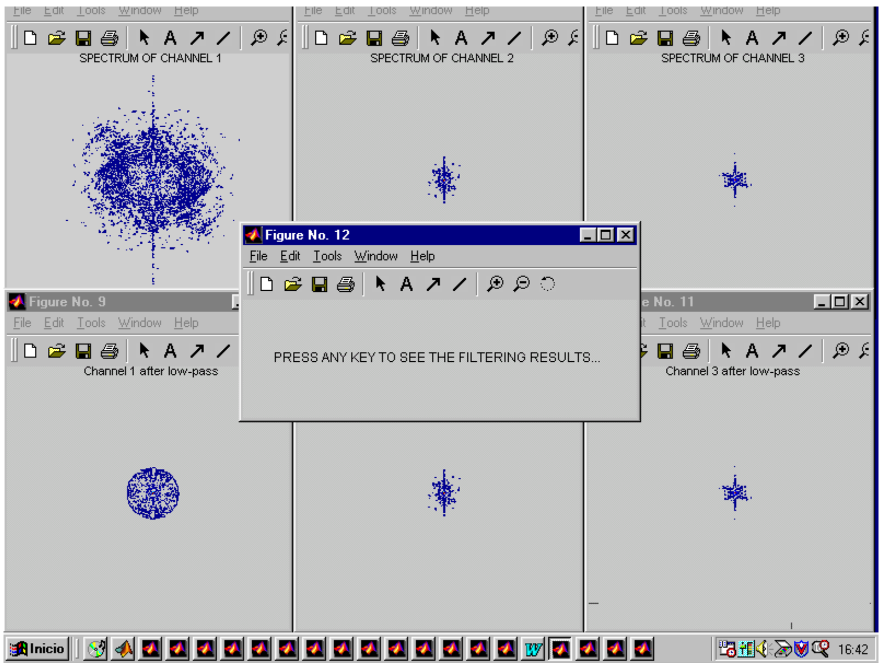


PRESS ANY KEY TO
DESIGN THE FILTERS...

and low-pass filters defined for each channel

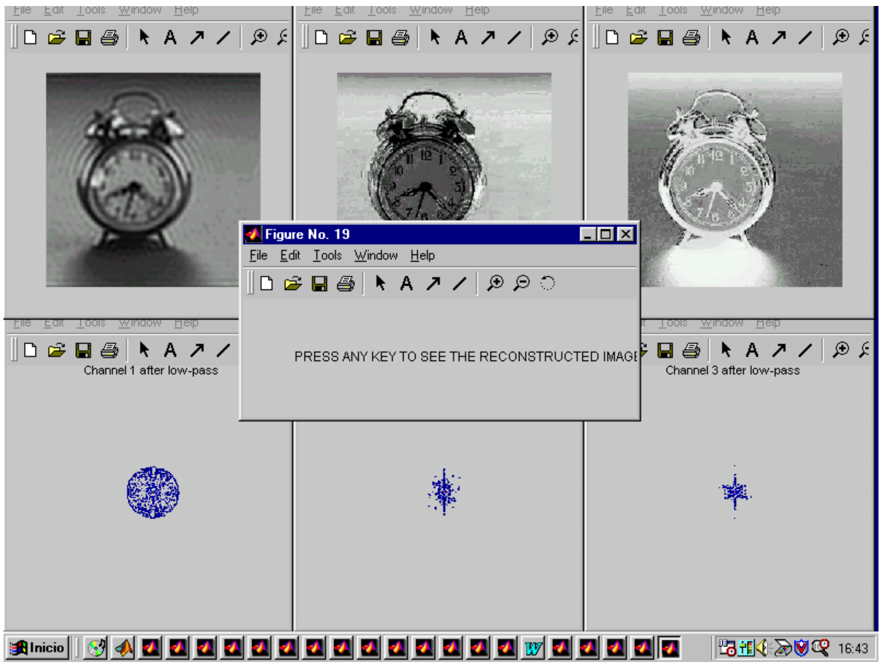


The resulting filtered spectra are computed and displayed:



PRESS ANY KEY TO SEE THE FILTERING RESULTS...

and the three filtered images shown



PRESS ANY KEY TO SEE THE RECONSTRUCTED IMAGE

The three filtered chromatic components of the image are recombined to yield the final filtered image, which is compared with the original. In the figure below, the results of two different filtering processes are shown. In the first, only 5% of the frequency domain is preserved in the achromatic Y image, whereas the two chromatic images are not filtered. In the second, both chromatic images are filtered, and only 2.5% of the frequency domain is preserved.

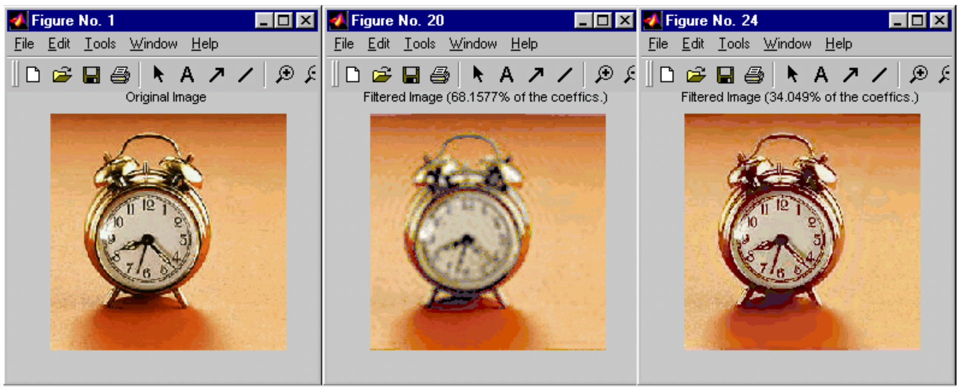
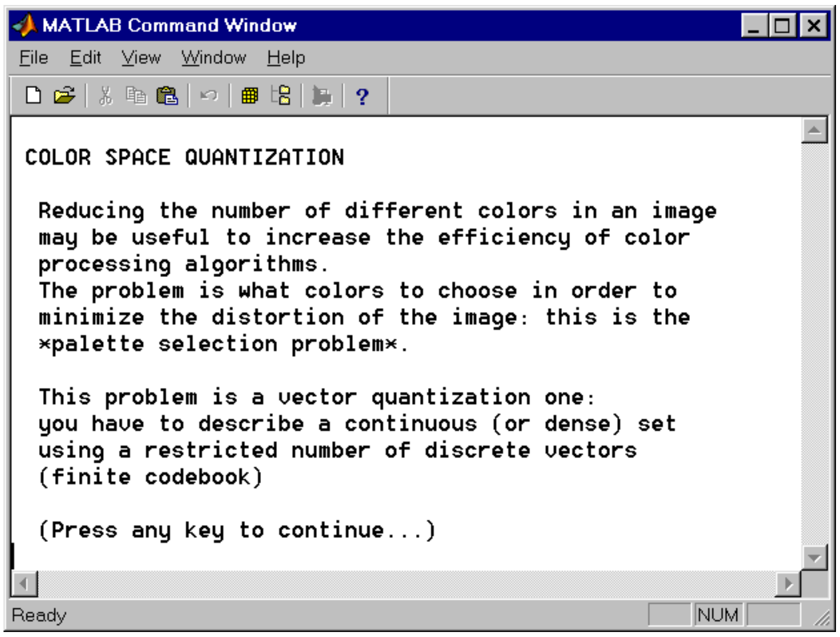


Figure No. 1
File Edit Tools Window Help
Original Image
Figure No. 20
File Edit Tools Window Help
Filtered Image (68.1577% of the coeffics.)
Figure No. 24
File Edit Tools Window Help
Filtered Image (34.049% of the coeffics.)

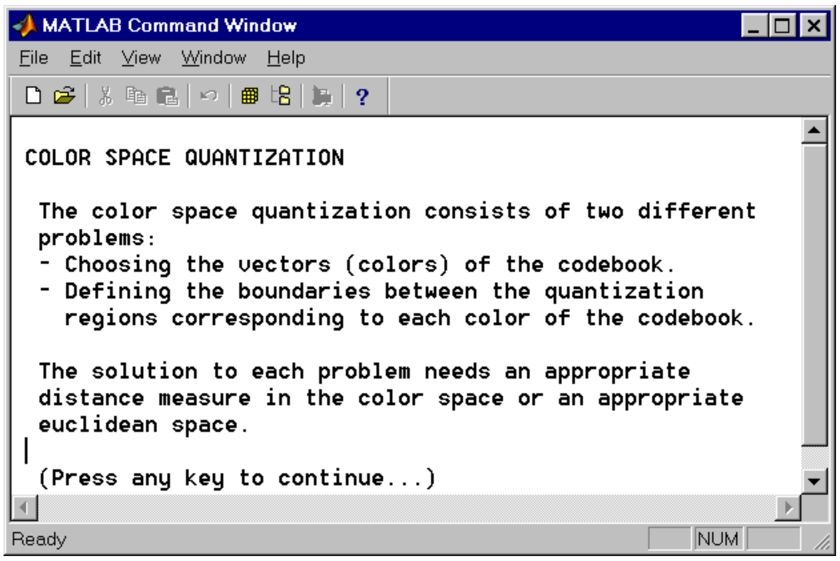
Note that although both filtered images differ from the original (on the left), the effect of quite a radical filtering in the chromatic components distorts the image less than a low-pass filter acting on the achromatic component. In this case, the image appears too defocused to seem acceptable.

2.8.7. Color vector quantization (palette selection)

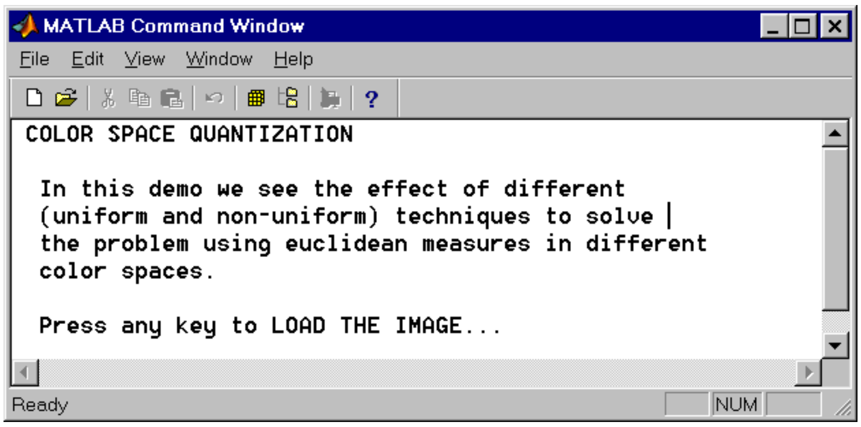
In this demo, the number of colors in an image is reduced in different ways (*minimum variance method* or *uniform sampling*) and in different color spaces (almost uniform spaces such as CIELab or standard tristimulus spaces such as CIE XYZ). We will use two extreme cases (CIELab and variance minimization versus CIE XYZ and uniform sampling) to stress the differences that may be found.



```
COLOR SPACE QUANTIZATION

 Reducing the number of different colors in an image
 may be useful to increase the efficiency of color
 processing algorithms.
 The problem is what colors to choose in order to
 minimize the distortion of the image: this is the
 *palette selection problem*.

 This problem is a vector quantization one:
 you have to describe a continuous (or dense) set
 using a restricted number of discrete vectors
 (finite codebook)

 (Press any key to continue...)
```

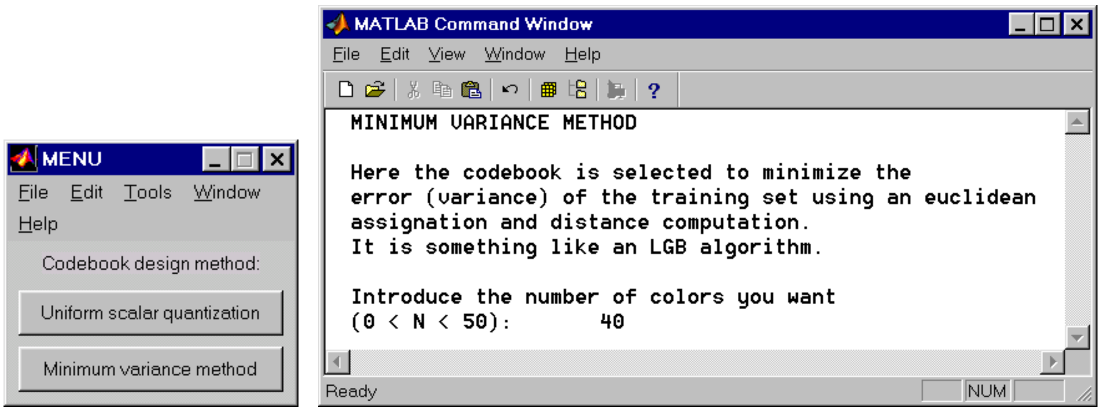


```
COLOR SPACE QUANTIZATION

 The color space quantization consists of two different
 problems:
 - Choosing the vectors (colors) of the codebook.
 - Defining the boundaries between the quantization
   regions corresponding to each color of the codebook.

 The solution to each problem needs an appropriate
 distance measure in the color space or an appropriate
 euclidean space.

 (Press any key to continue...)
```



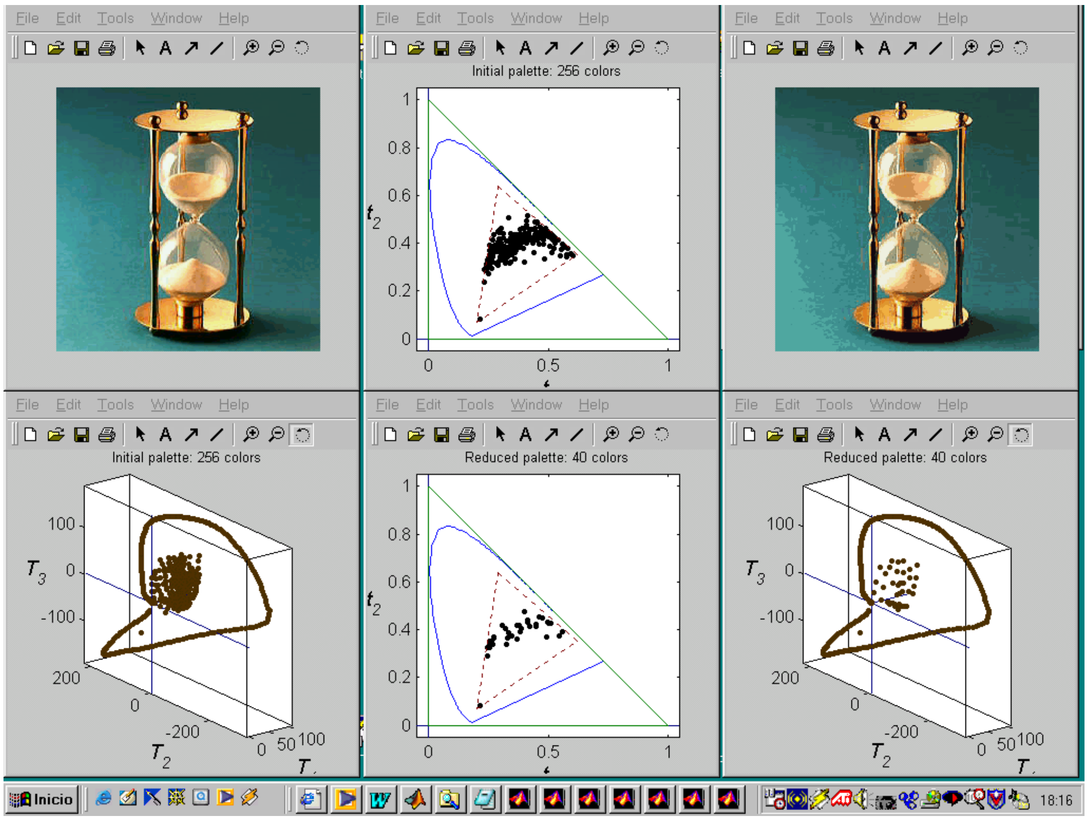
```
COLOR SPACE QUANTIZATION

 In this demo we see the effect of different
 (uniform and non-uniform) techniques to solve
 the problem using euclidean measures in different
 color spaces.

 Press any key to LOAD THE IMAGE...
```

At this point we select the image `sand.tif` using the dialog box. Then we are prompted to select the color space we want to use. In this first example we choose CIE Lab.

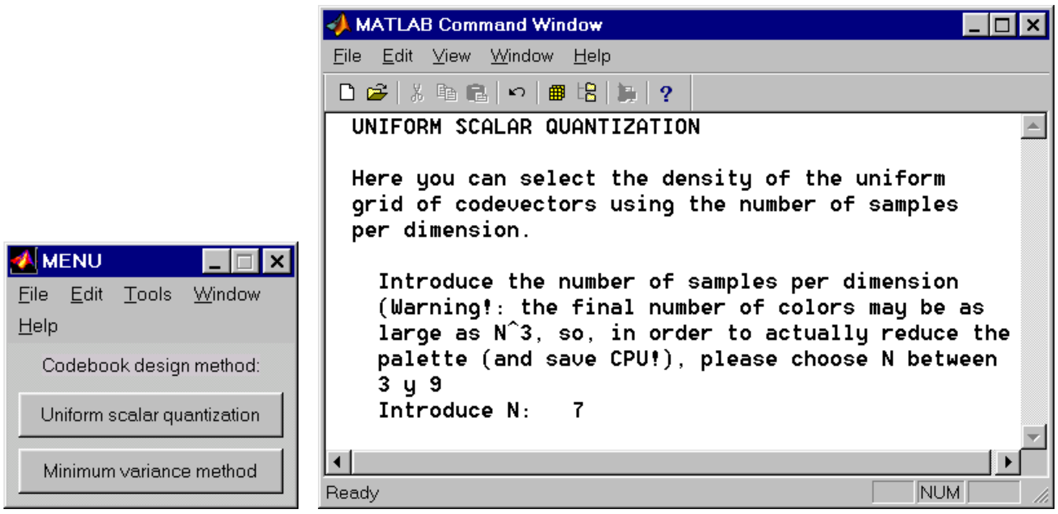

MENU
File Edit Tools Window Help
Codebook design method:
Uniform scalar quantization
Minimum variance method
MATLAB Command Window
File Edit View Window Help
Ready
NUM

```
MINIMUM VARIANCE METHOD

Here the codebook is selected to minimize the
error (variance) of the training set using an euclidean
assignation and distance computation.
It is something like an LGB algorithm.

Introduce the number of colors you want
(0 < N < 50):       40
```

Then we choose the minimum variance method with 40 colors. The results that are obtained (using the same space for assignation of the colors) are shown in the next picture:

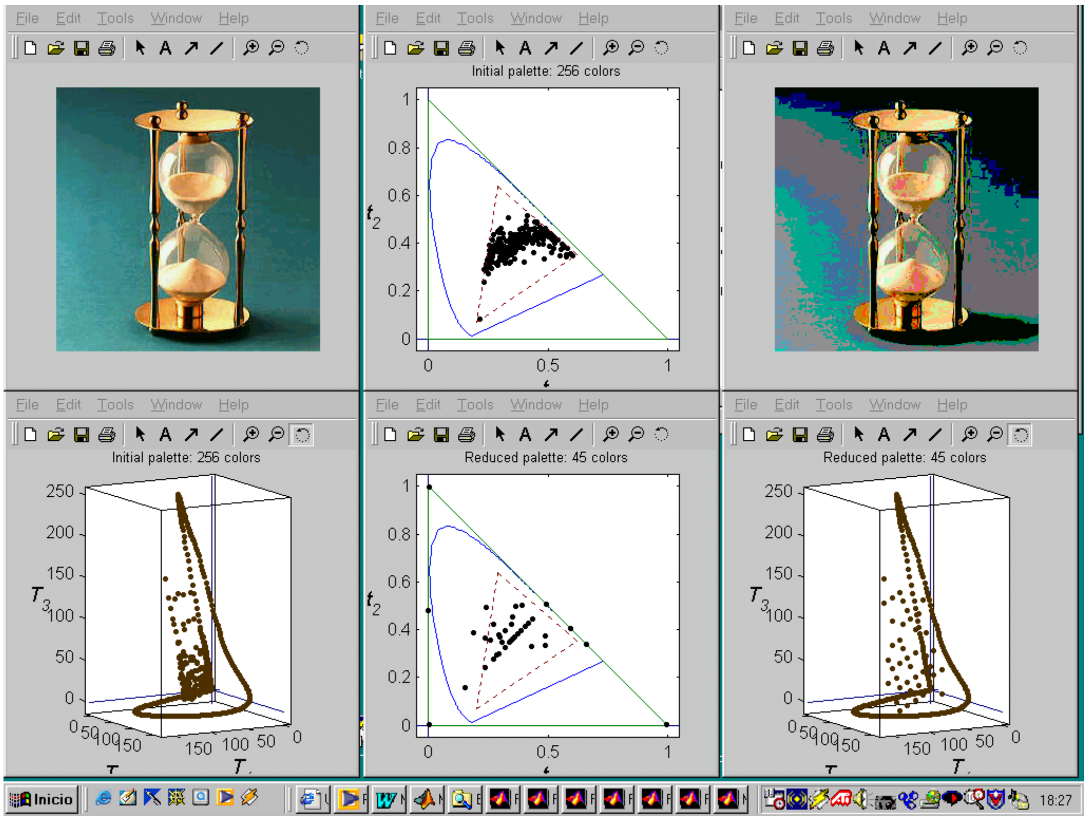

Initial palette: 256 colors
Reduced palette: 40 colors
Initial palette: 256 colors
Reduced palette: 40 colors
t_2
T_3
T_2
Inicio
18:16

Now we try another experiment with the same image. In this case we choose CIE XYZ and the uniform sampling method.


MENU
File Edit Tools Window Help
Codebook design method:
Uniform scalar quantization
Minimum variance method
MATLAB Command Window
File Edit View Window Help
Ready
NUM

```
UNIFORM SCALAR QUANTIZATION

Here you can select the density of the uniform
grid of codevectors using the number of samples
per dimension.

  Introduce the number of samples per dimension
  (Warning!: the final number of colors may be as
  large as N^3, so, in order to actually reduce the
  palette (and save CPU!), please choose N between
  3 y 9
  Introduce N:    7
```

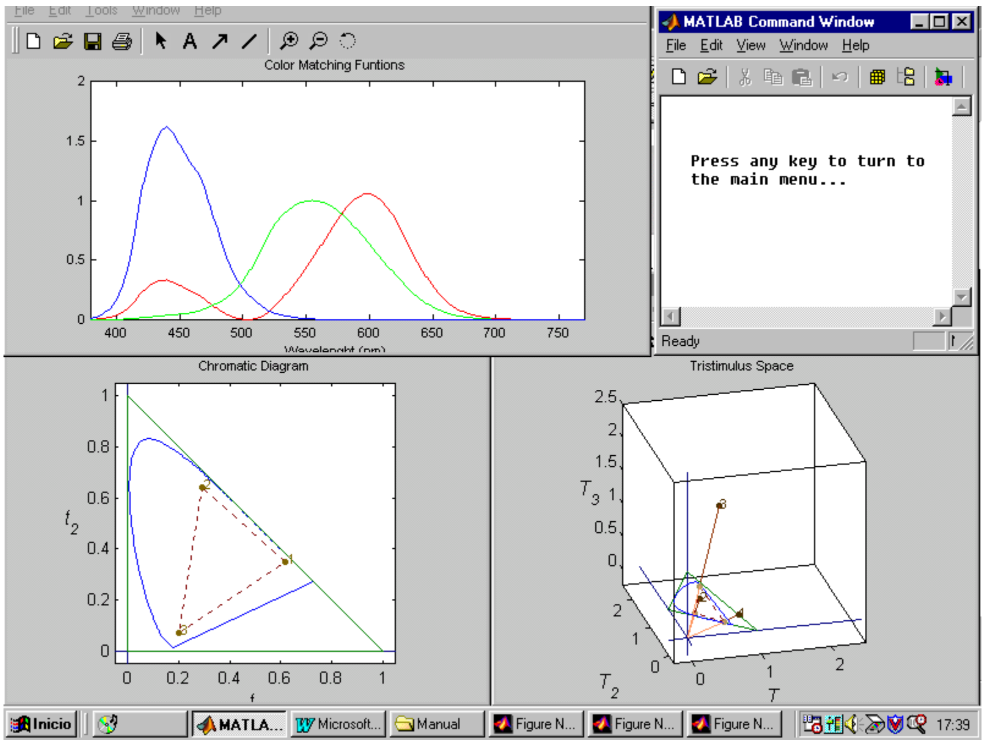
In this case we choose a grid of 7 colors per dimension, and we get this result using 45 colors:


File Edit Tools Window Help
Initial palette: 256 colors
Reduced palette: 45 colors
t_2
T_3
Inicio
18:27

It is clear that is not the number of colors what matters but the rationale to choose them!

2.8.8. Getting information on the reference system.

The *Color System* demo displays information on the reference system (CIEXYZ in the example below). The color matching functions are shown (upper left), as well as the chromaticity diagram (lower left) and the tristimulus space (lower right).



```
Press any key to turn to
the main menu...
```

The color matching functions are the tristimulus values of spectral colors with unity energy. The chromaticity coordinates of these colors define the *spectral locus*. Mixtures of the first visible blue and the last visible red are the *pure purples*. The blue line in both the chromaticity diagram and the color space represent the spectral and pure purple loci.

In the chromaticity diagram, the chromaticities of the CRT guns are displayed (colors 1, 2 and 3) and the triangle enclosing all the chromaticities that can be generated by the CRT is shown.

In the color space diagram, the chromaticity diagram is also shown. The maximum tristimulus values of the CRT guns are plotted.

Chapter 3

A typical *COLORLAB* session

3.1 Some sample problems

In this chapter we describe in detail how to use *COLORLAB* to solve some colorimetric problems.

Imagine you want to solve these problems:
- Analyze and edit the color appearance of an image.

--Generate an isoluminant grid using a particular color appearance model.

3.2 Worked examples

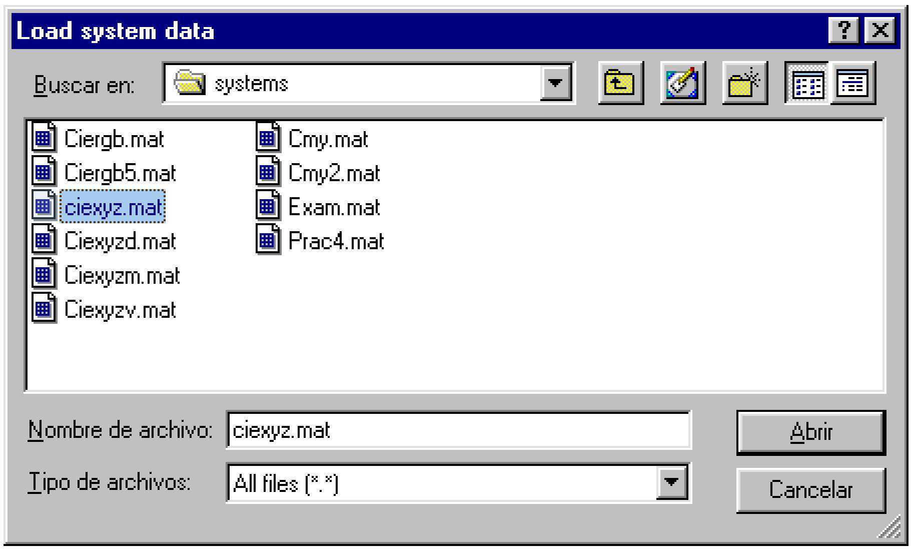
3.2.1. Starting a *COLORLAB* session

There is certain information that is constantly used during a *COLORLAB* session. The first decision you must make is the color reference system where you will describe your stimuli. The information about the reference system is explicitly required by many *COLORLAB* functions. For instance, if you want to transform chromaticity coordinates and luminances to tristimulus values, the trichromatic units of the system are needed. If you want to plot the chromaticity coordinates of a set of colors in a chromaticity diagram, the color matching functions of the system are needed to plot the spectral locus that serves as a reference.

On the other hand, if you wish to simulate a stimulus, that is, if you want to generate a metamer of your stimulus in your CRT display, you need colorimetric information about your monitor. This information must be described in the same color reference system you are using to describe your stimuli.

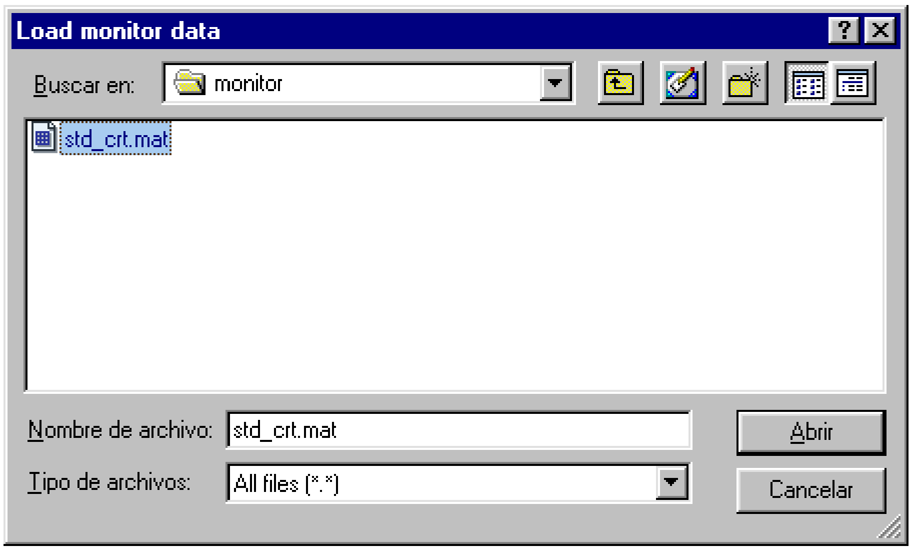
Therefore, the first thing to do when starting a *COLORLAB* session is to load all this relevant information. Two procedures are available.

1. Type `startcol`. This function successively opens to dialog boxes. The first is labelled *Load system data* and displays by default the contents of the `//colorlab/colordat/systems` folder, although any other folder can be selected with the mouse. The file corresponding to the desired color system may be selected and open.



Three new variables, `T_l`, `Yw` and `Msx`, appear in the workspace. They contain, respectively, the color matching functions, the trichromatic units and the change-of-basis matrix from the extant color space to CIEXYZ.

The second dialog box is labelled *Load monitor data* and displays by default the contents of the `//colorlab/colordat/systems` folder. The desired file may be then selected and opened.

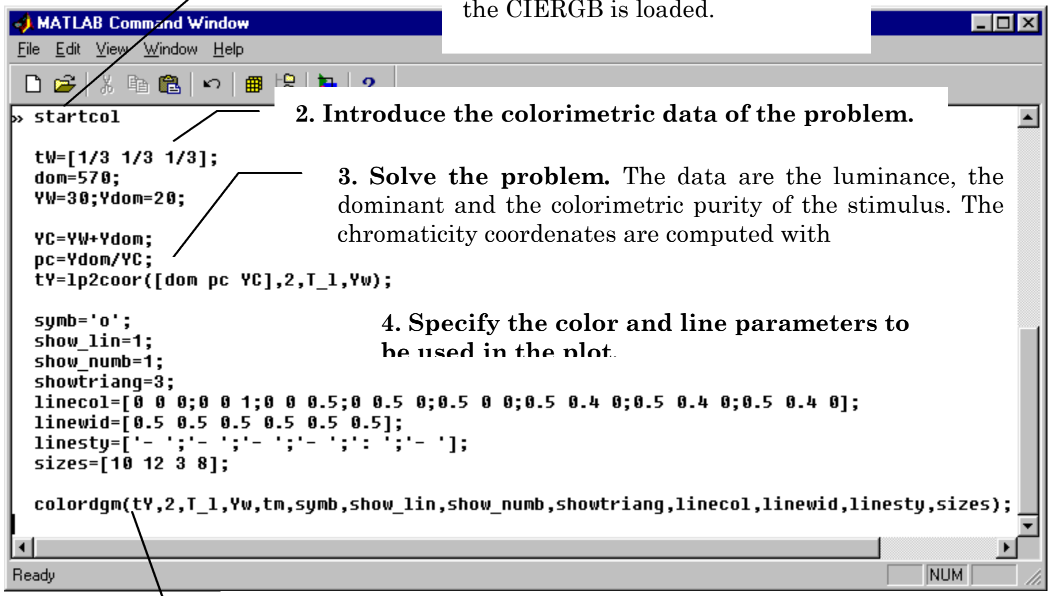


Monitor data are stored in three variables of the workspace, `tm`, `a` and `g`, containing, respectively, the monitor chromaticities and the parameters of the gamma function.

2. If you foresee that you will need to work with more than one color reference system in the same session, you may prefer to use functions allowing to rename the variables stored in the data files. In such case, use `loadsys` and `loadmon` (these two functions are called by `startcol`) or `loadsysm` and `loadmonm` if you prefer to specify the file name instead of selecting it with a dialog box. These last two functions are particularly useful within M-files.

As an example, let us assume that we wish to plot in a RGB chromaticity diagram the result of adding an equal energy white of 30 cd/m^2 and 20 cd/m^2 of the spectral stimulus 570 nm. The result is shown in Figure 3.1.

1. Start the *COLORLAB session.* The file corresponding to the CIERGB is loaded.



```
» startcol

  tW=[1/3 1/3 1/3];
  dom=570;
  YW=30;Ydom=20;

  YC=YW+Ydom;
  pc=Ydom/YC;
  tY=lp2coor([dom pc YC],2,T_l,Yw);

  symb='o';
  show_lin=1;
  show_numb=1;
  showtriang=3;
  linecol=[0 0 0;0 0 1;0 0 0.5;0 0.5 0;0.5 0 0;0.5 0.4 0;0.5 0.4 0;0.5 0.4 0];
  linewid=[0.5 0.5 0.5 0.5 0.5 0.5];
  linesty=['- ';'- ';'- ';'- ';': ';'- '];
  sizes=[10 12 3 8];

  colordgm(tY,2,T_l,Yw,tm,symb,show_lin,show_numb,showtriang,linecol,linewid,linesty,sizes);
```

2. Introduce the colorimetric data of the problem.

3. Solve the problem. The data are the luminance, the dominant and the colorimetric purity of the stimulus. The chromaticity coordenates are computed with

4. Specify the color and line parameters to be used in the plot.

5. Display the result in the appropriate chromaticity diagram.

Even in this simple problem, we have made use of the information about the reference system twice: to compute the chromaticity coordinates from the dominant, colorimetric purity and luminance of the stimulus and to plot the spectral locus in the chromaticity diagram. The information on the monitor, has served only in the chromaticity plot. However, it plays a useful role, since, from Figure 1, we learn that a metamer of the problem color might be generated with our monitor, as it plots inside the triangle defined by the chromaticities of the three monitor guns.

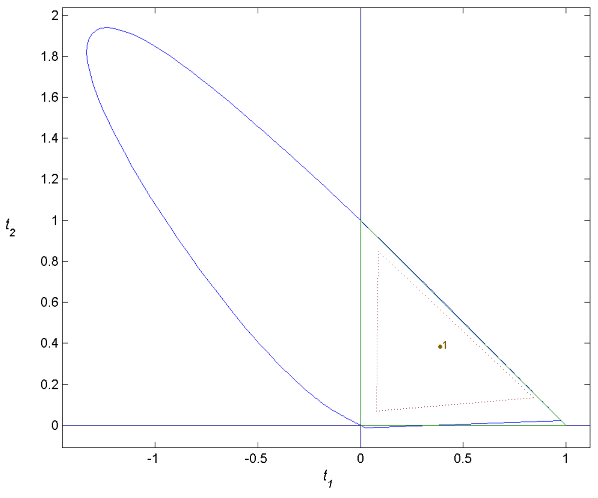


Figure 3.1.

3.2.2. Editing the color appearance of an image

In this example we are going to change the appearance of a poorly scanned color image in 2 ways:

(1) We want to increase the luminance of dark colors and increase the saturation of the scene.

We will use a simple approach:
- Express the palette in dominant wavelength, purity and luminance.
- Equalize the luminance (using an exponential non-linearity).
- Equalize the saturation (using an exponential non-linearity).

(2) We want to change the hue of the background.
We will change the dominant wavelength of the colors in the blue region (adding some constant to the values in the region [380-500]nm).

To begin with, we have to load the system data and monitor data to start working with *COLORLAB*. In particular we are going to choose the CIE XYZ system. To do so we use,

```
startcol
```

In the same way as explained before. Then we load the image:

Compute a palette of 50 colors (in digital values)

```
[imi,palette]=true2pal(im,50);
```

Obtain the tristimulus vectors from the digital values of the image:

```
T=val2tri(palette,Yw,tm,a,g);
```

Display the image and plot the colors:

```
figure(1),imshow(imi,palette),title('Original Image')
figure(2),colordgm(T,1,T_l,Yw,tm);title('Original Colors')
symb='o';
lim_axis=[-0.3 4.5 -0.3 4.5 -0.3 4.5];
showvectors=0;
show_lin=0;
show_numb=0;
showdiag=1;
showtriang=3;
linecolors=[0 0 0;0 0 1;0 0 0.5;0 0.5 0;0.5 0 0;...
0.3 0.2 0;0.6 0.2 0;0.3 0.2 0;0.3 0.2 0];
linewidths=[0.5 0.5 0.5 0.5 0.5 0.5 0.5];
```

```
linestyles=['- ';'- ';'- ';'- ';': ';'- ';'- '];
sizes=[10 12 2 8];
```

It is didactic to see the chromatic diagram in the tristimulus space representation. However as the values of the tristimulus vectors are far from the chromatic diagram the colors T are normalized by 5/110 in order to put them in the range [0,5]. This is also done in every tristimulus space representation in this example.

```
figure(3)
colorspc(5*T/110,1,T_l,Yw,tm,symb,lim_axis,showvectors,...
show_lin,show_numb,showdiag,showtriang,linecolors,...
linewidths,linestyles,fontsizes);
title('Original Colors')
```

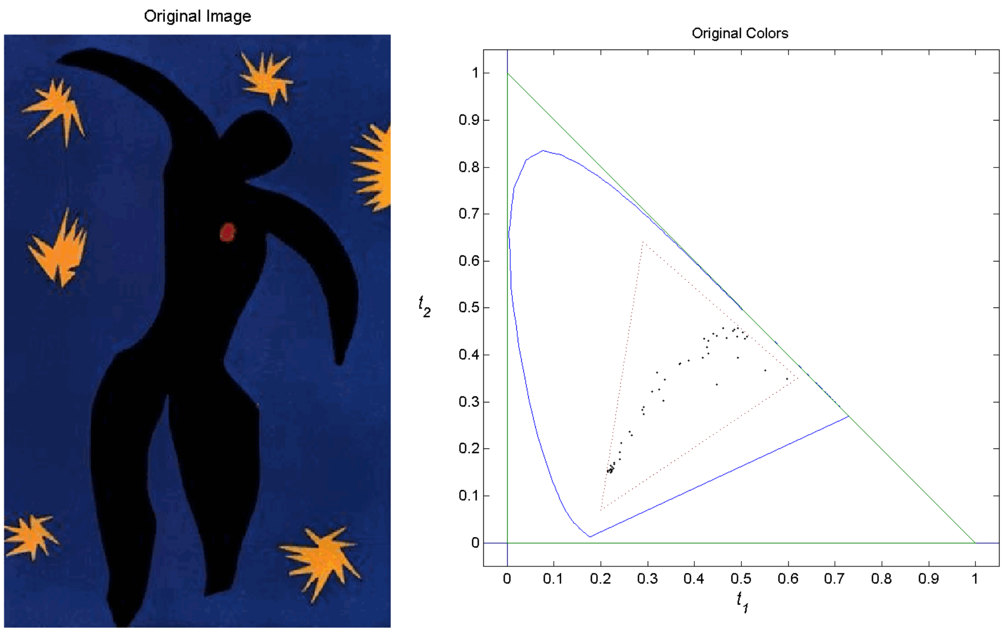


Figure 3.2.a. *Original image and colors in the chromatic diagram CIE x y*

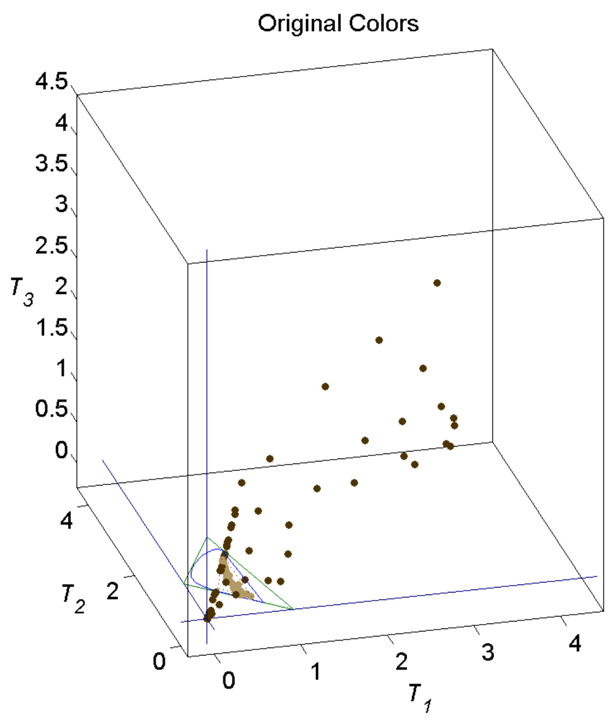


Figure 3.2.b *Original colors in the tristimulus space CIE XYZ*

Now transform to dominant wavelength, purity and luminance to isolate the luminance and purity:

```
tY=tri2coor(T,Yw);
lpY=coor2lp(tY,1,T_l,Yw);
```

Now we set the parameters of the non-linearities to process the luminances and the purities:

Exponent for luminance equalization:

```
ex=0.65;
```

Exponent for purity equalization:

```
pf=0.6;
```

Equalize the luminance:

```
lpY2=lpY;
lpY2(:,3)=max(lpY(:,3))*(lpY(:,3).^ex)/(max(lpY(:,3)).^ex);
```

Increase the purity:

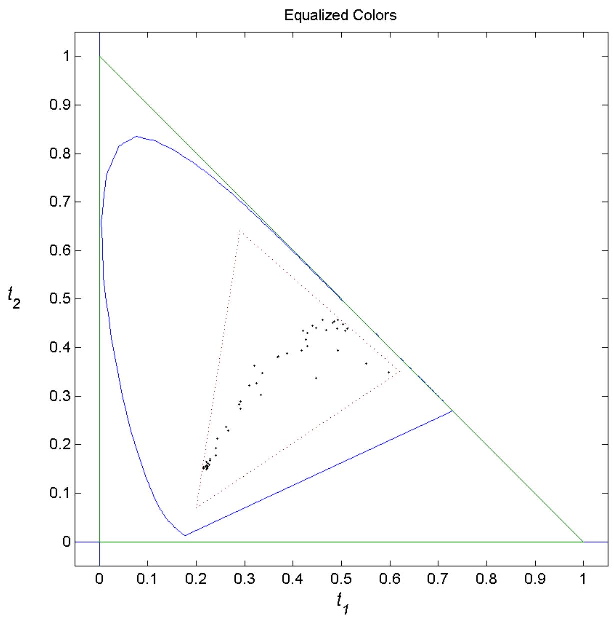
Obtain the new palette:

```
tY2=lp2coor(lpY2,1,T_l,Yw);
T2=coor2tri(tY2,Yw);
palette2=tri2val(T2,Yw,tm,a,g,8);
```

Display the new image:

```
figure(4),imshow(imi,palette2),title('Equalized Image')
figure(5),colordgm(T,1,T_l,Yw,tm);title('Equalized Colors')
symb='o';
lim_axis=[-0.3 4.5 -0.3 4.5 -0.3 4.5];
showvectors=0;
show_lin=0;
show_numb=0;
showdiag=1;
showtriang=3;
linecolors=[0 0 0;0 0 1;0 0 0.5;0 0.5 0;0.5 0 0;...
0.3 0.2 0;0.6 0.2 0;0.3 0.2 0;0.3 0.2 0];
linewidths=[0.5 0.5 0.5 0.5 0.5 0.5 0.5];
linestyles=['- ';'- ';'- ';'- ';': ';'- ';'- '];
sizes=[10 12 2 8];

figure(6)
colorspc(5*T2/110,1,T_l,Yw,tm,symb,lim_axis,showvectors,...
show_lin,show_numb,showdiag,showtriang,linecolors,...
linewidths,linestyles,fontsizes);
title('Equalized Colors')
```

Equalized Image



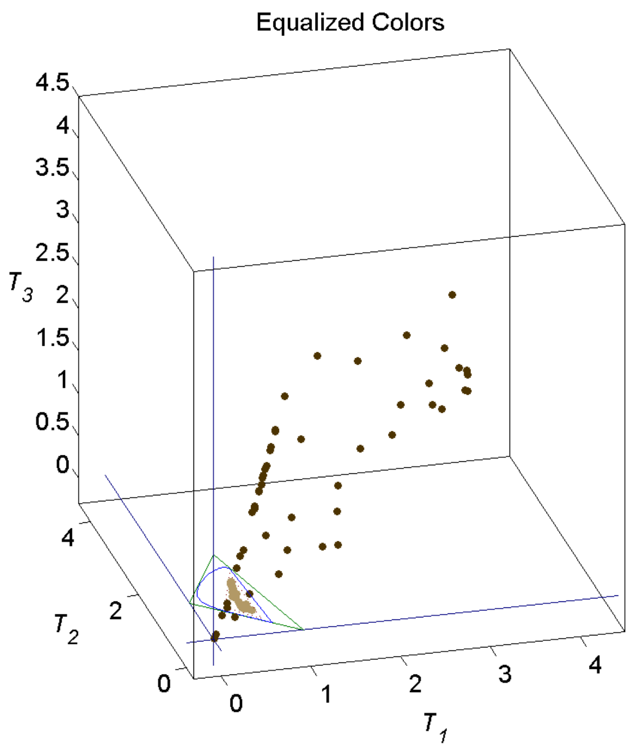


Figure 3.3 *Equalized image and equalized colors in the chromatic diagram CIE xy and the tristimulus space CIE XYZ*

Now we are going to turn the blues into greens increasing the dominant wavelength.

First we are going to set a certain increment in dominant wavelength and apply it to the colors with dominant wavelengths in the range [380,500] nm:

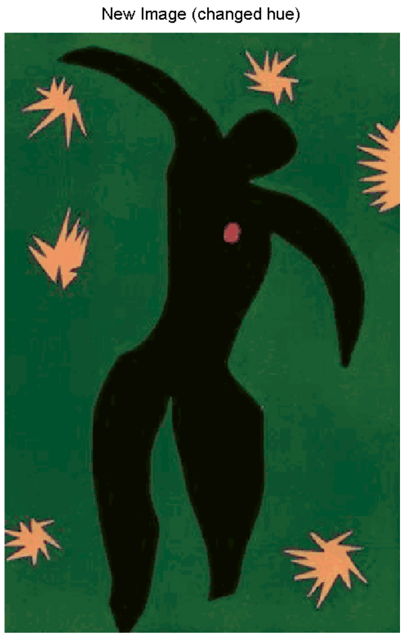
Obtain the new palette (first tristimulus and then digital values):

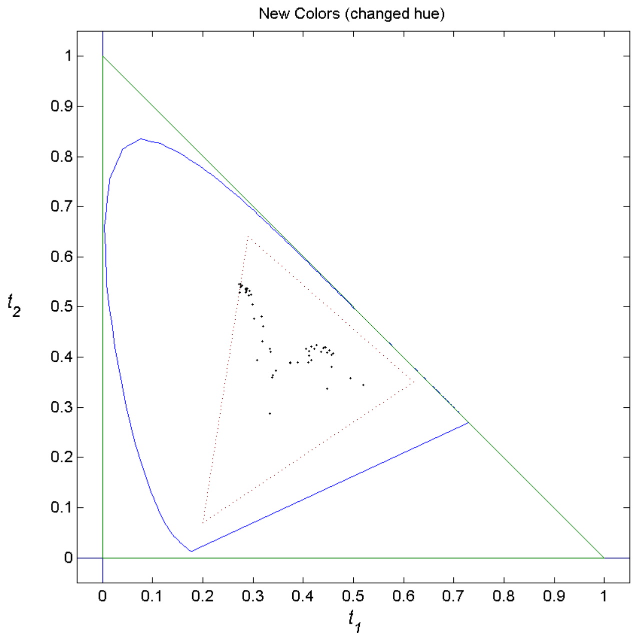
```
tY3=lp2coor(lpY3,1,T_l,Yw);
T3=coor2tri(tY3,Yw);
palette3=tri2val(T3,Yw,tm,a,g,8);
```

Finally, display the new image and colors:

```
figure(7),imshow(imi,palette3),
title('New Image (changed hue)')
```

```
symb='o';
show_lin=0;
show_numb=0;
showtriang=3;
linecolors=[0 0 0;0 0 1;0 0 0.5;0 0.5 0;0.5 0 0;...
            0.5 0.4 0;0.5 0.4 0;0.5 0.4 0];
linewidths=1.5*[1 1 1 1 1 1];
linestyles=['- ';'- ';'- ';'- ';': ';'- '];
fontsizes=[14 14 4 8];

figure(8),colordgm(T3,1,T_l,Yw,tm,symb,show_lin,...
show_numb,showtriang,linecolors,linewidths,linestyles,...
fontsizes);

figure(8),colordgm(T3,1,T_l,Yw,tm);
title('New Colors (changed hue)')
figure(9),clf;colorspc(T3,1,T_l,Yw,tm);
title('New Colors (changed hue)')
```





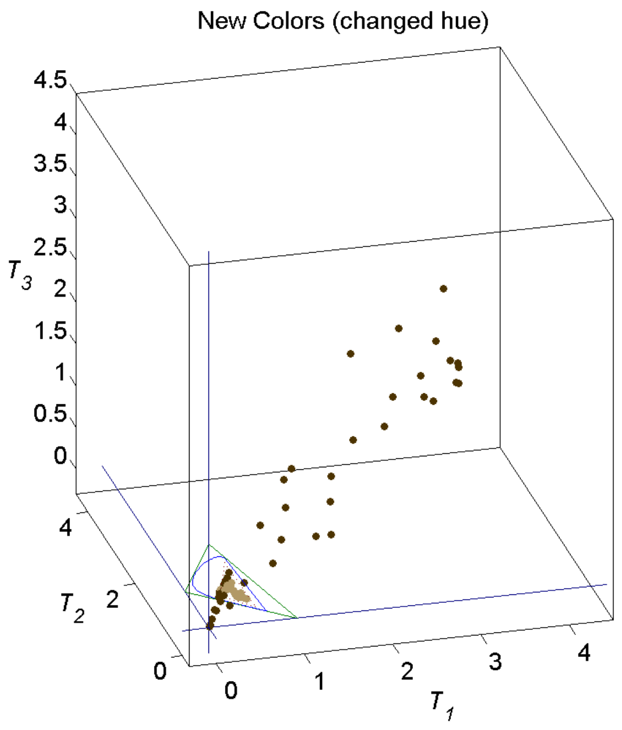


Figure 3.4 *Image after the hue change and the corresponding colors in the chromatic diagram CIE xy and the tristimulus space CIE XYZ*

3.2.3 Generating isoluminant grids.

Luminance gratings. Most readers will be familiar with the problem of generating pure luminance gratings. Let us consider, for instance, the simple case of a sinusoidal spatial variation. The luminance vs. position function, *Y(x,y)* is defined as:

$$Y(x,y) = Y_m\left(1 + C sin 2\pi (f_x x + f_y y)\right) \qquad (3.1)$$

where f_x and f_y are the spatial frequencies of the grating in the *x* and *y* Cartesian axis, Y_m is the mean luminance of the grating and C is Michelson's contrast, defined as:

$$C = \frac{Y_{max} - Y_{min}}{Y_{max} + Y_{min}} = \frac{\Delta Y}{Y_m} \qquad (3.2)$$

ΔY, the amplitude of the grating is the difference between the maximum luminance value of the grating and the mean value (see Figure 3.5 for a one-dimensional example). The chromaticity is kept constant for all (*x,y*) values.

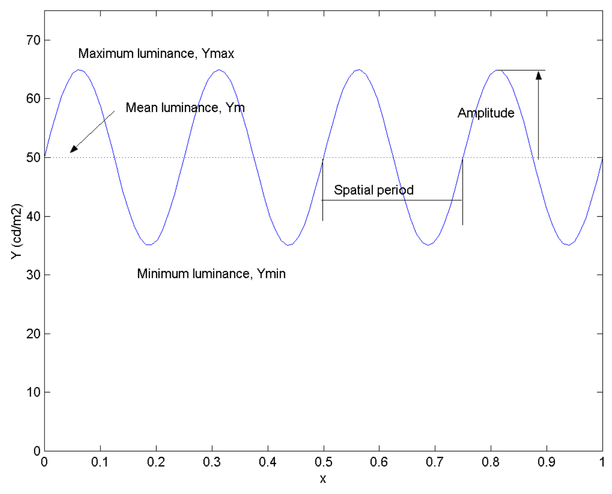


Figure 3.5.*Basic parameters of a one-dimensional grating. The frequency is the inverse of the period.*

Generating sinusoidal gratings with *COLORLAB* is very easy. The procedure is detailed below.

1. Start a *COLORLAB* session. A colorimetric reference system must be defined and the colorimetric characterization of the CRT monitor used specified.

Although other choices are perfectly possible, we shall chose the CIEXYZ system.

2. Choose the chromaticity of the grating.

Our grating will be white. Therefore,

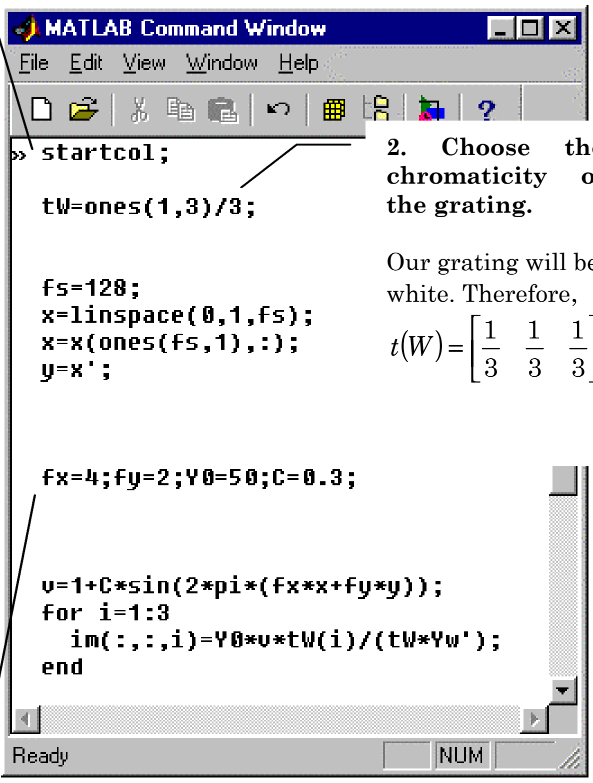
$$t(W) = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix}$$

3. Define the spatial domain (x,y)

Both the range of x and y values and the sampling fequency, f_s must be defined.

In this example, $x, y \in [0,1]$ and f_s=128 cpg.

4. Define the parameters of the grating: spatial frequencies, f_x and f_y mean luminance, Y_m, and contrast, C.



```
>> startcol;

   tW=ones(1,3)/3;

   fs=128;
   x=linspace(0,1,fs);
   x=x(ones(fs,1),:);
   y=x';

   fx=4;fy=2;Y0=50;C=0.3;

   v=1+C*sin(2*pi*(fx*x+fy*y));
   for i=1:3
     im(:,:,i)=Y0*v*tW(i)/(tW*Yw');
   end
```

5. Define a tristimulus image of the grating.

The tristimulus values of each point of the grating are defined by the equation:

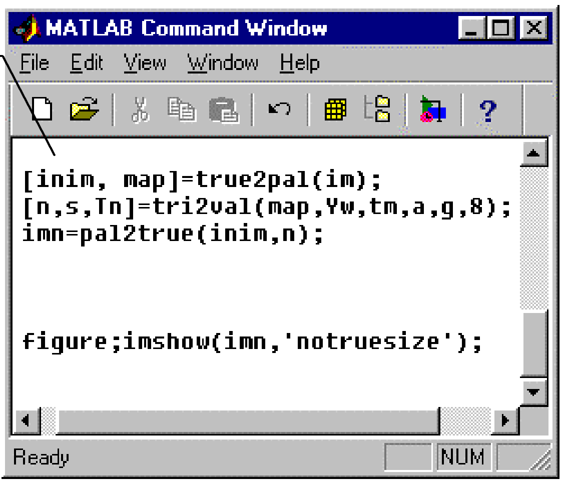
$$T(x,y) = Y_m\left(1 + C\sin 2\pi(f_x x + f_y y)\right)\frac{t(W)}{\sum_j t_j(W) Y_W(P_j)}$$

6. Convert the tristimulus image into digital values. This is done in three steps:

6.1. Convert the f_x*f_x*3 image into an indexed image and a tristimulus colormap (*inim* and *map*, respectively).

6.2. Convert the tristimulus colormap into digital values (*n*).

6.3. Convert the indexed image *inim* and the digital value colormap *n* into a true-color image *imn*.



```
[inim, map]=true2pal(im);
[n,s,Tn]=tri2val(map,Yw,tm,a,g,8);
imn=pal2true(inim,n);


figure;imshow(imn,'notruesize');
```

7 Display the image. For the frequencies f_x and f_y to be correct, the image must be viewed at such a distance that it subtends one degree from the observer's eye.

The results obtained are displayed in Figure 3.6. The mesh on the left represents the luminance spatial profile of the image, which, since we are working in the CIEXYZ system, is just `im(:,:,2)`. The grating generated with `imshow` can be seen on the right.

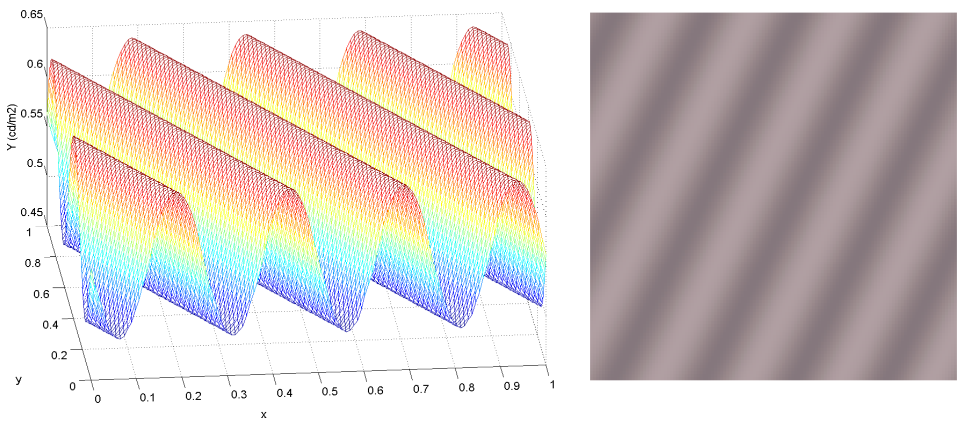


Figure 3.6.- *Luminance profile (left) of a luminance spatial grating (right).*

Gratings in the tristimulus space. The problem of generating sinusoidal spatial variations of color and not just luminance, poses some theoretical problems. To begin with, color is defined by at least three parameters. Therefore, while with a luminance grating the direction of the spatial change is clear -the stimulus can be made to appear brighter or darker-, with a color grating infinite directions of change are possible. Once a particular direction has been chosen, it remains the problem of how defining the amplitude of the change. In other words, how is the grating contrast defined in this case?

Let us consider the problem of producing a sinusoidal variation in the tristimulus values in a particular color reference system. Taking luminance sinusoidal gratings as our model, we must determine the following parameters:

1. The point of the color space around which we are going to create the color variation. This parameter, analogous to the mean luminance of the luminance gratings, shall be denoted by T_m
2. The maximum increment vector, *DT*. This vector determines both the direction of variation and its module: $\Delta T = |\Delta T|\vec{u}$, where $\vec{u}$ is a unit vector in the direction of change.

With all this, the tristimulus values vs. spatial position function is defined by the following equation:

$$T(x,y) = T_m + \Delta T \sin 2\pi(f_x x + f_y y) \qquad (3.3)$$

If we consider separately each component of the previous equation:

$$T_i(x,y) = T_{m_i} + \sin 2\pi(f_x x + f_y y)\Delta T_i \qquad (3.4)$$

we may reach a definition of contrast for each component, analogous to the Michelson's contrast:

$$C_i = \frac{T_{max_i} - T_{min_i}}{T_{max_i} + T_{min_i}} = \frac{T_{m_i} + \Delta T_i - (T_{m_i} - \Delta T_i)}{T_{m_i} + \Delta T_i + (T_{m_i} - \Delta T_i)} = \frac{\Delta T_i}{T_{m_i}} \qquad (3.5)$$

Note that the contrast in each direction of the color space is different. Unlike luminance contrast, this color contrast is not always well defined. In linear ATD spaces, for instance, a white stimulus is characterized by [*A* 0 0]. Therefore, contrast along the *T* and *D* directions would be infinite for any increment [*DA DT DD*], against all logic.

As an example, let us solve the problem of generating a sinusoidal spatial color grating in the CIEXYZ space, with T_m= [30 30 30] and *DT*=[4 0 4]. This is a constant luminance grating.

1. Start a *COLORLAB* session. We shall chose the CIEXYZ system.

2. Define the spatial domain (*x*,*y*)

Both the range of x and y values and the sampling fequency, f_s must be defined.

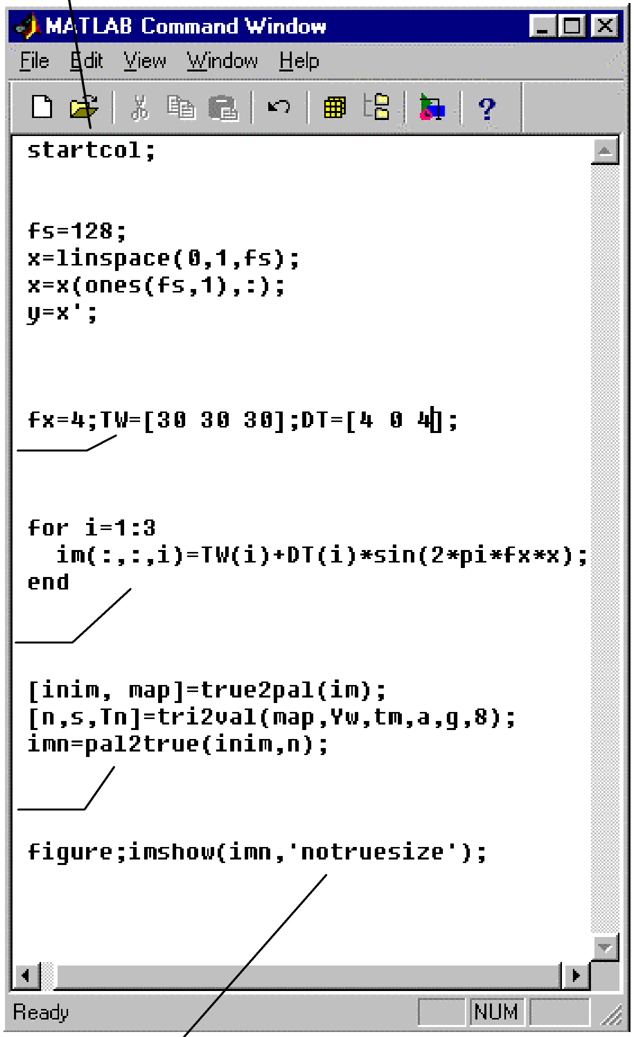
In this example, $x, y \in [0,1]$ and f_s=128 cpg.

3. Define the parameters of the grating: spatial frequencies, f_x origin of the variation, T_m, and amplitude *DT*.

4. Define a tristimulus image of the grating.

5. Convert the tristimulus image into digital values.

6. Display the image



```
startcol;


fs=128;
x=linspace(0,1,fs);
x=x(ones(fs,1),:);
y=x';


fx=4;TW=[30 30 30];DT=[4 0 4];


for i=1:3
  im(:,:,i)=TW(i)+DT(i)*sin(2*pi*fx*x);
end


[inim, map]=true2pal(im);
[n,s,Tn]=tri2val(map,Yw,tm,a,g,8);
imn=pal2true(inim,n);


figure;imshow(imn,'notruesize');
```

The results are displayed in Figure 3.7. The grating (bottom right) represents a modulation from white in a reddish-greenish direction at constant luminance.

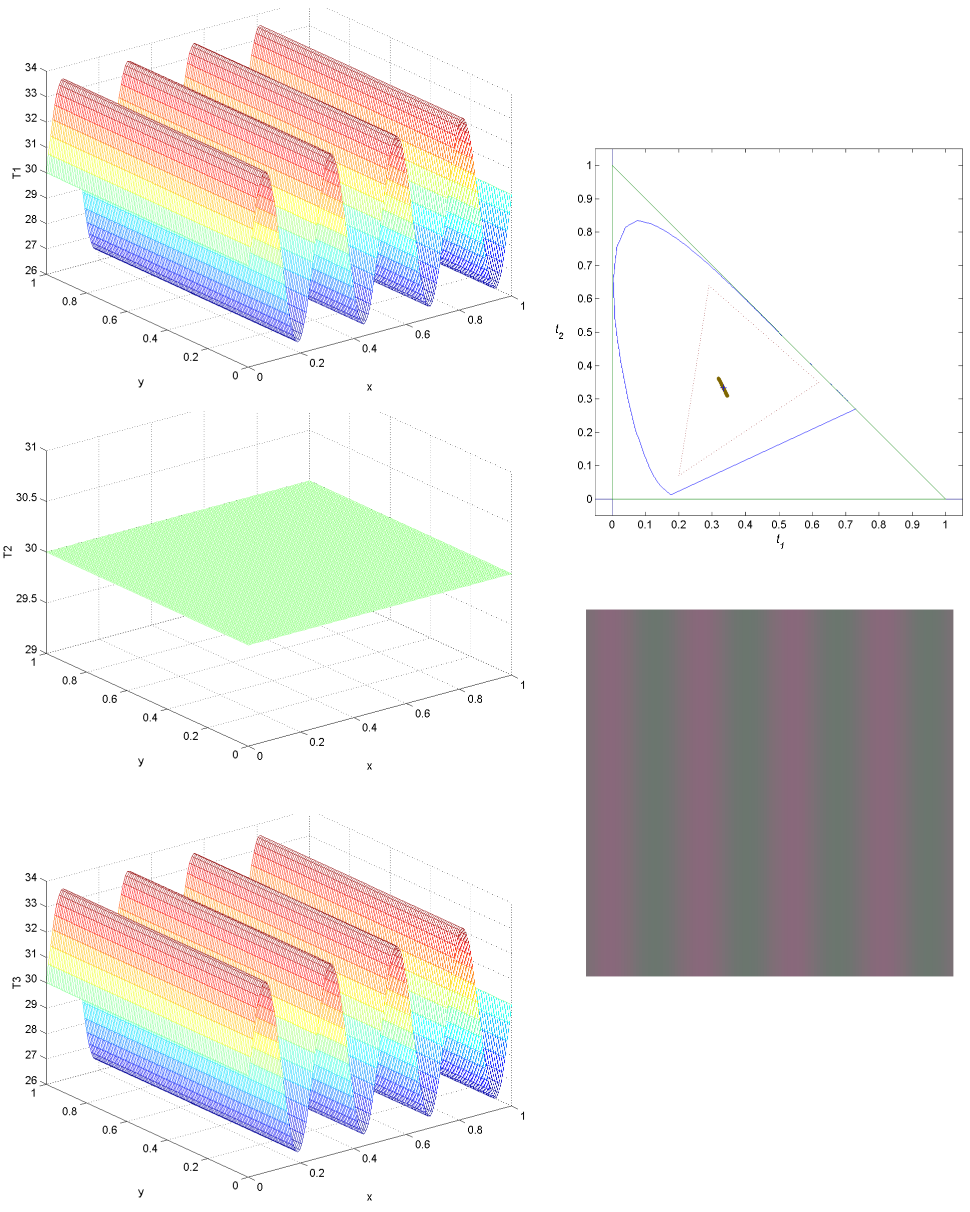


Figure 3.7: *Sinusoidal spatial variation from stimulus XYZ=[30 30 30], with amplitude [5 0 5]. The meshes on the left show the spatial variation of each of the tristimulus components. The range of colors is plotted in a chromaticity diagram on the right. The grating is also displayed (bottom right).*

Isoluminant gratings. In the grating we have just generated only the chromaticity changes with position. This does not mean, however, that the only change an observer perceives is a brightness change. The green bars appear considerably darker than the red ones. While the problem of generating spatial changes of the chromaticity coordinates is relatively simple, as we have seen before, generating changes color changes at constant brightness is a difficult experimental problem. We shall discuss two of the more common solutions.

In the first place, let us consider two sinusoidal luminance-only gratings of the same spatial frequency but in counterphase. The color of each grating is chosen depending of along which direction of the color space we pretend to obtain a chromatic modulation. For example, let us assume that one of the gratings is red and the other green. Thus, for the red grating:

$$T(R,x,y) = Y_m(R)\left(1 + C(R) sin 2\pi(f_x x + f_y y)\right)\frac{t(R)}{\sum_j t_j(R) Y_W(P_j)} \tag{3.6}$$

and for the green grating:

$$T(G,x,y) = Y_m(G)\left(1 - C(G) sin 2\pi(f_x x + f_y y)\right)\frac{t(G)}{\sum_j t_j(G) Y_W(P_j)} \tag{3.7}$$

Let us sum up both gratings, assuming that the two contrasts are equal. We obtain:

$$T(R+G,x,y) = T_m + \Delta T sin 2\pi(f_x x + f_y y) \tag{3.8}$$

where

$$T_m = Y_m(R)\frac{t(R)}{\sum_j t_j(R) Y_W(P_j)} + Y_m(G)\frac{t(G)}{\sum_j t_j(G) Y_W(P_j)} \tag{3.9}$$

and

$$\Delta T = C\left(Y_m(R)\frac{t(R)}{\sum_j t_j(R) Y_W(P_j)} - Y_m(G)\frac{t(G)}{\sum_j t_j(G) Y_W(P_j)}\right) \tag{3.10}$$

Although we have reached the same functional dependence of the tristimulus values with position than in the previous section, two differences are noteworthy. First, since the chromaticities of the two gratings we are summing up are constant, to change the contrast of the resulting grating it suffices to change the

Michelson contrast of the red and green grating. This provides a finite definition of contrast. On the other hand, it is possible to manipulate the mean luminances of the red and green grating to obtain psychophysical isoluminance. To this end, flicker photometry is used. An practical example is shown below.

1. Start a *COLORLAB* session. We shall chose the CIEXYZ system.

2. Define the spatial domain (*x*,*y*)

Both the range of *x* and *y* values and the sampling fequency, f_s must be defined.

In this example, $x, y \in [0,1]$ and f_s=128 cpg.

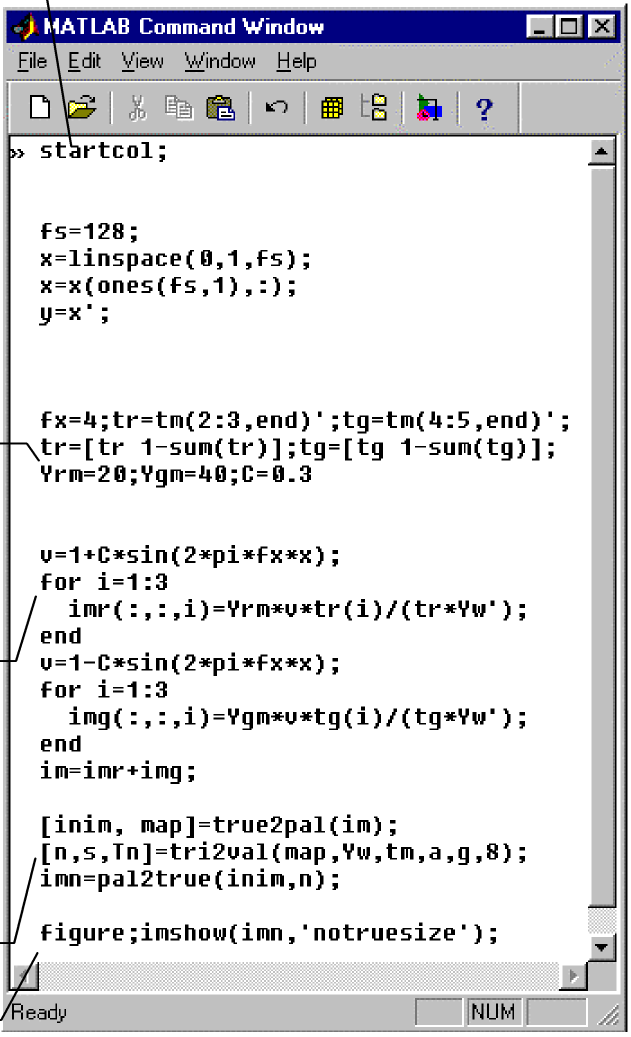
3. Define the parameters of the luminance gratings: spatial frequencies, f_x, chromaticities *t*, mean luminance Y_m and contrast *C*.

4. Define a tristimulus image of the grating.

Define first the tristimulus images of the red and green luminance gratings (*imr* and *img*) in counterphase. The desired image, *im*, is the sum of these two gratings.

5. Convert the tristimulus image into digital values.

6. Display the image



```
» startcol;


 fs=128;
 x=linspace(0,1,fs);
 x=x(ones(fs,1),:);
 y=x';


 fx=4;tr=tm(2:3,end)';tg=tm(4:5,end)';
 tr=[tr 1-sum(tr)];tg=[tg 1-sum(tg)];
 Yrm=20;Ygm=40;C=0.3


 v=1+C*sin(2*pi*fx*x);
 for i=1:3
   imr(:,:,i)=Yrm*v*tr(i)/(tr*Yw');
 end
 v=1-C*sin(2*pi*fx*x);
 for i=1:3
   img(:,:,i)=Ygm*v*tg(i)/(tg*Yw');
 end
 im=imr+img;

 [inim, map]=true2pal(im);
 [n,s,Tn]=tri2val(map,Yw,tm,a,g,8);
 imn=pal2true(inim,n);

 figure;imshow(imn,'notruesize');
```

The grating we have obtained in this example is shown in Figure 3.8. The two luminance gratings used and the chromaticity range of the final grating are also shown. Note that the mean luminances chosen (20 and 40 cd/m^2) do not provide equal brightness, but the luminosity difference between the orange and the greenish hemiperiods is reduced. In an experiment, the condition of equal brightness must be determined for each observer.

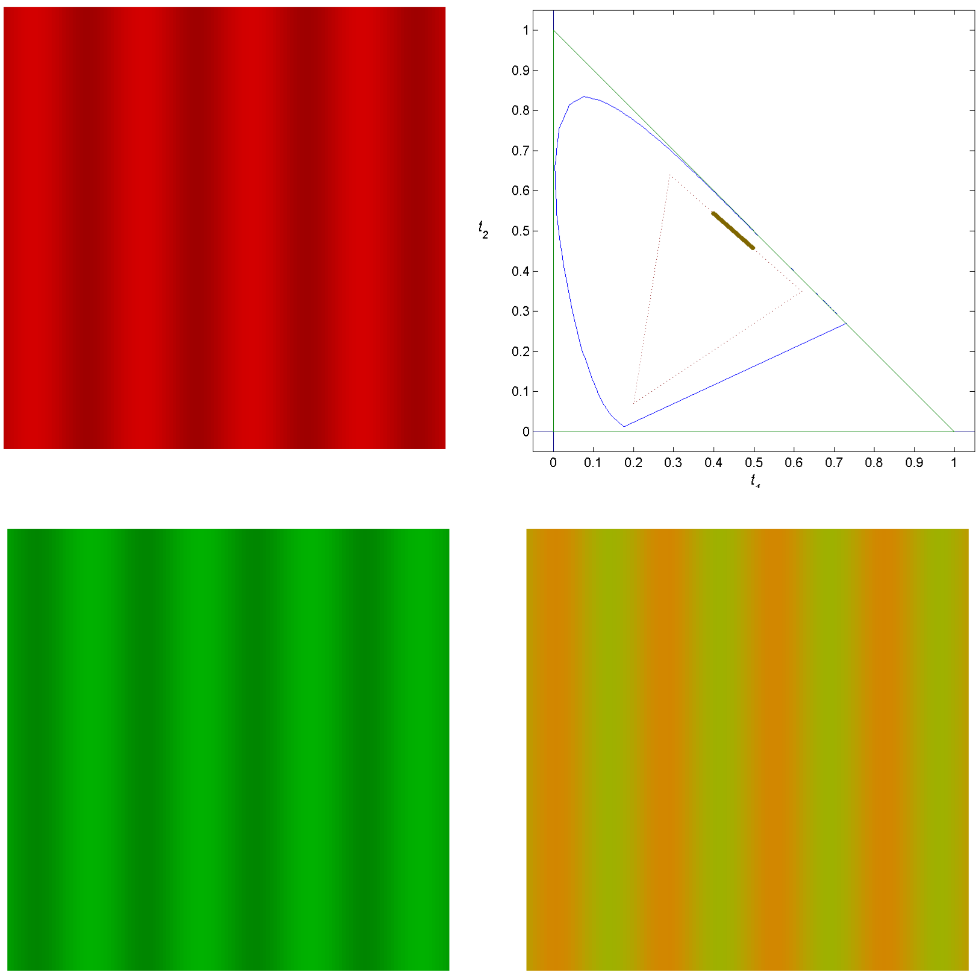


Figure 3.8- *Sinusoidal chromaticity grating obtained (bottom right) as the sum of a red and a green luminance grating in counterphase (left).*

If equal brightness is not required, an alternative for generating pure chromaticity gratings is to define the stimulus in a color space where the color descriptors have perceptual significance. In the example below, Boyton's ATD space is used, and it is assumed that when *A* is constant the grating is isoluminant. The modulation created would isolate the response of the red-green cells. The results are shown in Figure 3.9.

1. Start a *COLORLAB* session. We shall chose the CIEXYZ system.

2. Define the spatial domain (x,y)

Both the range of x and y values and the sampling fequency, f_s must be defined.

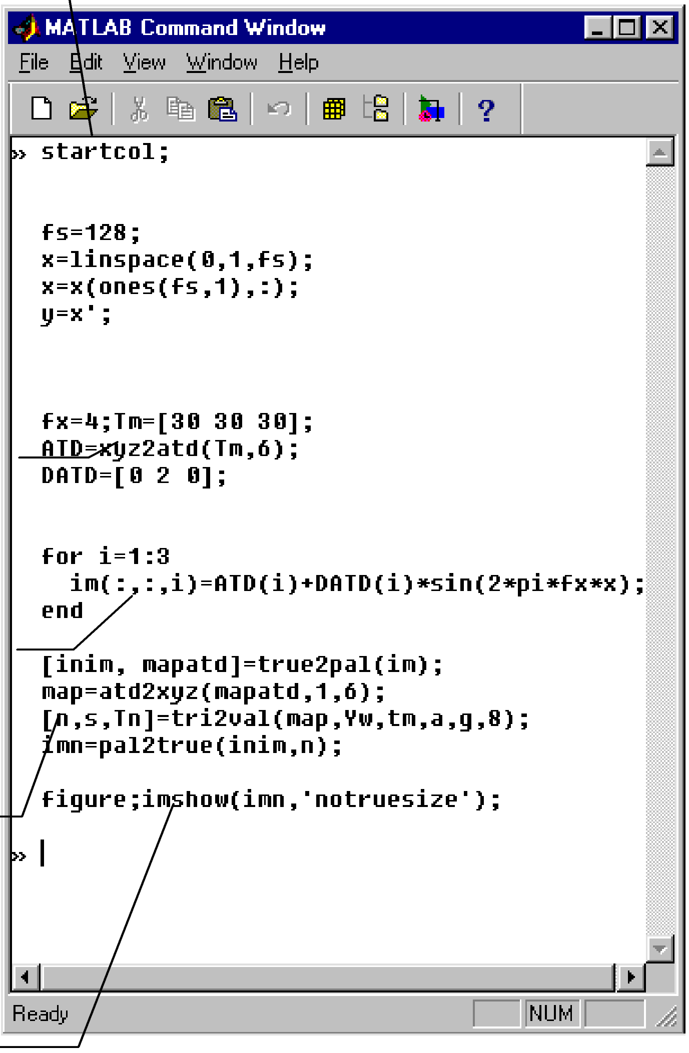
In this example, $x, y \in [0,1]$ and f_s=128 cpg.

3. Define the parameters of the grating. The amplitude and mean of the grating are specified in Boynton's model.

4. Define an ATD image of the grating.

5. Obtain an image with the digital values. It is necessary to transform the ATD palette into XYZ tristimulus values for this to be feasible.

6. Display the image



```
>> startcol;

 fs=128;
 x=linspace(0,1,fs);
 x=x(ones(fs,1),:);
 y=x';


 fx=4;Tm=[30 30 30];
 ATD=xyz2atd(Tm,6);
 DATD=[0 2 0];

 for i=1:3
   im(:,:,i)=ATD(i)+DATD(i)*sin(2*pi*fx*x);
 end

 [inim, mapatd]=true2pal(im);
 map=atd2xyz(mapatd,1,6);
 [n,s,Tn]=tri2val(map,Yw,tm,a,g,8);
 imn=pal2true(inim,n);

 figure;imshow(imn,'notruesize');

>> |
```

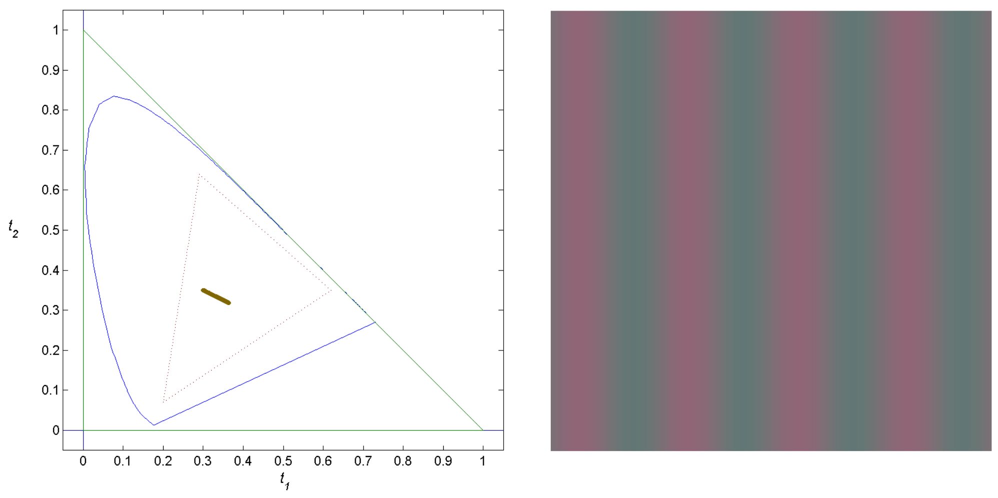


Figure 3.9.- *Reg-green isoluminant grating generated with Boynton's ATD color mo*

Chapter 4

Reference manual

In this section we list the *COLORLAB* functions in alphabetical order giving a detailed account of their syntax and some examples of use.

atd1atd2

Purpose

Compute first opponent stage responses from second stage responses and vice versa with a given model.

Description

Although most models implemented in this library have a single opponent stage, with an achromatic (*A*) and two chromatic descriptors (red-green *T* and blue-yellow *D*), more advanced models consider a first opponent stage at the Lateral Geniculate Nucleus level and a second cortical opponent stage. This function computes the responses of one of these stages from the responses of the other.

If in the model used the *ATD* values go through a non-linear stage before entering the next opponent stage or the final perceptual stage, the input of `atd1atd2` must include the non-linearities. The output *ATD* values also include non-linearities.

Syntax

```
ATD=atd1atd2(ATD0,stage0,model)
```

`ATD0`	*ATD* responses at the input stage.
`stage0`	Input stage (1 or 2)
`model`	Integer identifying the model (1-13). See `xyz2atd` for details.
`ATD`	Output *ATDs* in the desired stage. If `stage0`=1, `ATD` are second stage responses. If `stage0`=2, `ATD` are first stage responses.

Related Functions

`xyz2atd, atd2xyz, xyz2con, con2xyz, con2atd, atd2con`

Required Functions

`compcop, icompcop`

atd2perc

Purpose Compute perceptual descriptors with an ATD model.

Description Perceptual descriptors in ATD models are Brightness (*B*), Hue (*H*) and Saturation (*S*). Although the meaning of these parameters is always the same, the dependence with the *ATD* values changes with the model.

-Jameson and Hurvich model

$$B = |A| + |T| + |D|$$

$$H = \frac{|T|}{|T| + |D|}$$

$$S = \frac{|T| + |D|}{B}$$

- In the rest of linear first-stage models:

$$B = \sqrt{A^2 + T^2 + D^2}$$

$$H = atan\frac{D}{T}$$

$$S = \frac{\sqrt{T^2 + D^2}}{A}$$

- In ATD95

$$B = \sqrt{A_1{}^2 + T_1{}^2 + D_1{}^2}$$

$$H = atan\frac{D_2}{T_2}$$

$$S = \frac{\sqrt{T_2{}^2 + D_2{}^2}}{A_2}$$

`atd2perc` computes *BHS* from the output of the last stage of the model, including the non-linearities.

Syntax

`BHS=atd2perc(ATDE,model)`

`ATDE` *ATD* descriptors of the last stage of the model.

`model` Number identifying the model (1-13). See `xyz2atd` for details.

`BHS` For N colors, Nx3 matrix containing in the first column the Brightness (*B*), in the second the hue angle ($0 \leq H \leq 2*\pi$) or hue coefficient and in the third the saturation (*S*) of each input stimulus.

Related Functions

`perc2atd, xyz2atd, atd1atd2`

Required Functions

atd2xyz

Purpose Compute *ATD* from *XYZ*.

Description `atd2xyz` computes the tristimulus values *XYZ* from *ATD* opponent responses of the specified stage of a given model, in the specified observation conditions.

ATD must include the non-linearities of the model.

See `xyz2atd` for a list of available models.

Syntax `XYZ=atd2xyz(ATD,stage,model,XYZB,adap,weights,mode,sig)`

`ATD` Responses of the achromatic (A), red-green (T) and blue-yellow (D) mechanisms to the test stimuli.
For N stimuli, ATD is a Nx3 matrix.
If the model has non-linearities, ATD must include them.

`stage` 1 and 2, for first and second opponent stage *ATDs*, respectively.

`model` Integer (1-14) identifying the model. See `xyz2atd` for details.

`XYZB` Background stimulus (3x1)

`adap` 3x3 or 3x1 matrix, depending on the model, including adaptation effects or adaptation conditions. See below.

`weights` Only for models 8-11. Contributions of test stimuli and the background to the adaptation stimulus.

`mode` If ATD are increments on XYZB, mode=1. Otherwise, mode=2.
Use only with models 8-11.

`sig` Scalar parameter controlling the non-linearity at the cone stage.

Use only with models 8-11.

`XYZ` Tristimulus values of the test stimuli. The color observer may be the CIE-1931 standard observer, the observer modified by Judd or by Judd and Vos, depending on the model.

First stage models need, at most, the information about the background or the adaptation matrix, so the function may be used as

`XYZ=atd2xyz(ATD,1,model,[],adap)` (models 4, 5 and 7)

`XYZ=atd2xyz(ATD,1,model)` (model 6)

`XYZ=atd2xyz(XYZ,1,model,XYZB)` (model 13).

If a model does not use a given parameter, enter []. To use the default values of a parameter, use [] if any parameter with a user-defined value comes behind, or just leave them out if they are the last inputs of the function.

Related Functions

`xyz2atd, atd2perc, perc2atd, atdf2con, con2xyz, incomcop, ingancon, xyzn2xyz`

Required Functions

`xyz2con, con2atd, atd1atd2, gancon, compcop, icompcop, xyz2xyzn`

atdf2con

Purpose

Compute cone responses from opponent responses.

Description

`atdf2con` computes the gain-controlled cone responses from the compressed first-stage *ATD* responses of a given model, under the specified adaptation conditions. See `con2atd` for details.

Syntax

`LMS=atdf2con(ATD1,model,adap)`

`ATD1`	Compressed first-stage responses. For N colors, this is a Nx3 matrix.
`model`	1-13. See `xyz2atd` for details.
`adap`	3x3 or 3x1 matrix with the adaptation conditions. See `xyz2atd` for details.
`LMS`	Cone responses, including the non-linearities of the model. For N colors, this is a Nx3 matrix.

Related Functions

`xyz2atd, atd2xyz, xyz2con, con2atd, atd1atd2, atdf2con, con2xyz, compcop, gancon, ingancon, xyz2xyzn, xyzn2xyz`

Required Functions

`incomcop`

calibrat

Purpose Procedure for (manual and tedious) monitor calibration.

Description Computers do not use an appropriate (colorimetrically meaningful) color description, but 3D arrays of parameters $n=[n_1\ n_2\ n_3]$ that control the voltages of each gun of the CRT. In most standard image formats these n_i values (digital values) are given using 8 bit integers (uint8 variables ranging from 0 to 255). This is a device-dependent characterization because the color obtained from a given array, n, depends on the particular color reproduction device.

In *MATLAB* colormaps, the digital values that describe the colors are real numbers with values in [0,1] (ranging from minimum luminance to maximum luminance of the gun).

Some calibration data are needed to relate these device-dependent values to colorimetrically meaningful (device-independent) values. In this way we will be able to generate accurately colors in our CRT monitor. Besides, we will be able to estimate the tristimulus components of the digital images from their digital encoding.

The *'COLORLAB format for CRT calibration data storage'* is just a name convention for the variables involved in the CRT calibration. Such a convention is convenient for future automatic transform of the data to the required color system when using `loadmon`

COLORLAB assumes additivity and independence between the guns of the CRT monitor. *COLORLAB* also assumes an exponential relation (gamma relation) between the luminance of the gun, i, and the corresponding digital value, n_i (i=1,2,3). No assumption is made on the chromaticities of the guns, which may depend on the digital value in any complex way.

With these assumptions, the calibration data include:

- Chromaticities of the gun, i, as a function of the digital step, n_i.

- Parameters (a_i and g_i) of the gamma relation between the luminance of the gun, i, and the digital value, n_i:

$$Y_i = a_i n_i^{g_i}$$

A typical VGA-like card allows 256 steps in each gun.

The calibration process consists in measuring samples of these curves (generating a number of stimuli and measuring the corresponding colors) and using these samples to fit the curve parameters a and g.

`calibrat` generates the stimuli in each gun, asks for the chromaticity and luminance values and fits the curves. This procedure is repeated for each gun. The process is also done for white (gray) stimuli in order to check the additivity of the guns.

The program returns the parameters of the luminance curves (*a,g*), their errors (*ea,eg*) and the luminances of the guns and the white stimuli. Besides, `calibrat` returns two matrices with the data of the chromaticities of the guns and the white stimuli:

* t_m 7*N matrix with the chromaticities of the monitor guns (primaries).
  This 7*N format is referred to as calibration data in other routines help.

* t_w 3*N matrix with the chromaticities of the white stimuli.

As an example, if we choose n_i to be ni=0:0.1:1 we will have:

$$t_m = \begin{pmatrix} 0 & 0.1 & \cdots & 0.9 & 1 \\ t_1(P_1(0)) & t_1(P_1(0.1)) & \cdots & t_1(P_1(0.9)) & t_1(P_1(1)) \\ t_2(P_1(0)) & t_2(P_1(0.1)) & \cdots & t_2(P_1(0.9)) & t_2(P_1(1)) \\ t_1(P_2(0)) & t_1(P_2(0.1)) & \cdots & t_1(P_2(0.9)) & t_1(P_2(1)) \\ t_2(P_2(0)) & t_2(P_2(0.1)) & \cdots & t_2(P_2(0.9)) & t_2(P_2(1)) \\ t_1(P_3(0)) & t_1(P_3(0.1)) & \cdots & t_1(P_3(0.9)) & t_1(P_3(1)) \\ t_2(P_3(0)) & t_2(P_3(0.1)) & \cdots & t_2(P_3(0.9)) & t_2(P_3(1)) \end{pmatrix}$$

and

$$t_W = \begin{pmatrix} 0 & 0.1 & \cdots & 0.9 & 1 \\ t_1(W(0)) & t_1(W(0.1)) & \cdots & t_1(W(0.9)) & t_1(W(1)) \\ t_2(W(0)) & t_2(W(0.1)) & \cdots & t_2(W(0.9)) & t_2(W(1)) \end{pmatrix}$$

If low digital values are below the sensitivity of the colorimeter, introduce [0 0] for these chromaticities. For those digital values, we will interpolate the chromaticities between the first measured value and the chromaticity of zero ([0 0 0]). The chromaticity assigned to [0 0 0] is that of the first measurable white.

Syntax

```
[a,g,ea,eg,yr,yg,yb,yw,tm,tw]=calibrat(fig,ni,Msx,...
'path',look_for_data);
```

INPUT variables:

`fig` Figure (number) where the stimuli will be generated.

`ni` 1*N vector with the digital values that will be measured.
`ni` MUST include 0 and 1.
For example, if you want to measure the colors in N equally spaced points for each gun, you may enter
`ni=linspace(0,1,N);`

`Msx` 3*3 change-of-basis matrix relating the system S (where the data are described) to the CIEXYZ system.

`'path'` String with the path to the file that shall contain the data. For example:
`'c:\`*MATLAB*`\toolbox\`*COLORLAB*`\colordat\monitor\...`
`monito.mat'`

`look_for_data` Informs about the existence of previously measured data.
Calibration is a tedious process (typically you will have to measure and type 24 to 32 colors -6 to 8 per gun plus the whites-). `calibrat` automatically saves each measure in the specified file, so if something happens you don't have to re-start measuring again.
If `look_for_data==1, calibrat` looks for the previously measured data (in `'path'`), and only generates the stimuli you need to finish the series of measurements.
If `look_for_data==0`, a new (complete) series of measurements is begun.

OUTPUT variables:

`a` 1*3 vector including the constants a_i of the gamma relation for each gun.

`g` 1*3 vector including the constants g_i of the gamma relation for each gun.

`ea` 1*3 vector including the errors of the constants a_i of the gamma relation.

`eg` 1*3 vector including the errors of the constants g_i of the gamma relation.

`yr` 1*N vector containing the N luminance measures for the Red Gun.

`yg` 1*N vector containing the N luminance measures for the Green Gun.

`yb`	1*N vector containing the N luminance measures for the Blue Gun.
`yw`	1*N vector containing the N luminance measures for the white stimuli.
`tm`	7*N matrix that contains the chromaticities of the R, G and B guns for N calibration points of the digital value. The chromaticities are given in the system at hand even though they are stored in the CIEXYZ system (this is why `Msx` is needed).
`tw`	3*N matrix that contains the chromaticities of the white stimuli for N calibration points of the digital value. The chromaticities are given in the system at hand even though they are stored in the CIEXYZ system (this is why `Msx` is needed).

Related Functions

`savemon, loadmon, loadmonm, tri2val, val2tri`

Required Functions

`coor2tri, sacagama*, ajusta1*, error1*, funcion*, newbasis, justa1*, grafic1*, derivad2*`

Routines with * are required for the fit only.

chngmtx

Purpose Compute the change-of-basis matrix that relates the systems *P* and *P.'*

Description `chngmtx` solves the problem in two different cases (from different initial data):

Case 1. We know the tristimulus values of the new color primaries in the old basis: $T_i(P_j')$.

Case 2. We know the chromatic coordinates of the new color primaries in the old basis: $t_i(P_j')$ and the characterization of the new reference white, *W'*: Chromatic coordinates (in the old basis), $t_i(W')$, and luminance, $Y(W')$.

Besides, `chngmtx` also computes the new trichromatic units from the old ones.

Syntax `[M12,Yw2]=chngmtx(DATA,Yw1,option);`

`M12` The 3*3 change-of-basis matrix that relates the systems P and P' (the systems 1 and 2).

`Yw2` Trichromatic units of the new basis (1*3 vector).

`Yw1` Trichromatic units of the old basis (1*3 vector).

`DATA` 3*3 matrix with the data to compute the matrix `M12`.
The interpretation of `DATA` (case 1 or case 2) depends on the value of the variable option. See below for details on how to introduce `DATA` in each case.

`option` Variable that controls the meaning of `DATA`.
Using `option=1` we mean we know the tristimulus values of the new color primaries $T_i(P_j')$. In this case, you have to

introduce `DATA` in this way:

$$\mathtt{DATA} = \begin{pmatrix} T_1(P_1') & T_2(P_1') & T_3(P_1') \\ T_1(P_2') & T_2(P_2') & T_3(P_2') \\ T_1(P_3') & T_2(P_3') & T_3(P_3') \end{pmatrix}$$

Using `option=2` we mean we know the chromaticities of the new color primaries and the white. In this case, you have to introduce `DATA` in this way:

$$\mathtt{DATA} = \begin{pmatrix} t_1(P_1') & t_2(P_1') & t_1(W') \\ t_1(P_2') & t_2(P_2') & t_2(W') \\ t_1(P_3') & t_2(P_3') & Y(W') \end{pmatrix}$$

Related Functions `newconst, newbasis, defsys, defsysm`

Required Functions `coor2tri`

ciecam97

Purpose Compute color descriptors with the CIECAM model.

Description CIECAM97 computes the hue (*h*), the hue quadrature (*H*), the contributions of red, yellow, green and blue to the hue, *HC*, the lightness, *J*, the brightness, *Q*, the saturation, *s*, the chroma, *C* and the colorfulness, *M*, of a set of stimuli with the CIECAM97 model.

The input may comprise different stimuli, each with its own observation conditions. The backgrounds cover completely a 10deg-field around the test and are embedded in the surround. Both fields constitute the adapting field. Different adaptation degrees can be considered .

Syntax

```
[h,H,HCR,HCY,HCG,HCB,J,Q,s,C,M]=CIECAM97(xyY,xyYW,...
xyYB,YA,view,adap);
```

`xyY` Nx3 matrix with the chromaticity coordinates and the luminance factor of the test stimuli.

`xyYW` Chromaticity coordinates and luminance factor of the reference white.
Either a single white for all stimuli, or a different white for each stimulus may be introduced.

`xyYB` Chromaticity coordinates and luminance factor of the background.
Either a single background for all stimuli, or a different background for each stimulus may be introduced.

`YA` Luminances of the adapting fields. A single number indicates that the same value holds for all stimuli.
Normally, this value equals 20% of the luminance of the white in the adapting field.

`view` describes the viewing conditions, determining certain constants in the model:

`view=1`	Average surround with samples > 4º.
`view=2`	Average surround with samples < 4º.
`view=3`	Dim surround.
`view=4`	Dark surround.
`view=5`	Cut-sheet transparencies.

A surround with luminance factor above 20 is considered "average". If the luminance factor is below 20, it is dim, and when near zero is dark.

`adap` describes the degree of chromatic adaptation.

`adap=1`	There is no chromatic adaptation.
`adap=2`	Normal level of chromatic adaptation.
`adap=3`	Strong chromatic adaptation (the illuminant is partially discounted)
`adap=4`	Total chromatic adaptation (the illuminant is totally discounted)

`xyY` Nx3 matrix with the chromaticity coordinates and the luminance factor of the test stimuli.

`[h,H,HCR,HCY,HCG,HCB,J,Q,s,C,M]` CIECAM97 descriptors.

Related Functions `cam97inv`

Required Functions

cam97inv

Purpose Compute XYZ from CIECAM97 perceptual descriptors.

Description The function computes the chromaticity coordinates *xy* and the luminance factor *Y,* either of a set of unrelated or related stimuli, described by their brightness (*Q*), colorfulness (*M*) and hue quadrature (*H*) or of a set of related stimuli described by their lightness, *J*, chroma *C* and hue quadrature, *H*. The observation and adaptation conditions for each stimuli must be specified. The conditions for each stimulus can be different.

Syntax `xyY=cam97inv(test,rel,xyYW,xyYB,YA,view,adap)`

`test`	For N colors, Nx3 matrix containing the descriptors indicated by `rel`.
`rel`	If `rel`=0, `test`=[Q1 M1 H1;Q2 M2 H2;Q3 M3 H3;...] If `rel`=1, `test`=[J1 C1 H1;J2 C2 H2;J3 C3 H3;...]
`xyYW`	Chromaticity coordinates and luminance factor of the reference white. Either a single white for all stimuli, or a different white for each stimulus may be introduced.
`xyYB`	Chromaticity coordinates and luminance factor of the background. Either a single background for all stimuli, or a different background for each stimulus may be introduced.
`YA`	Luminances of the adapting fields. A single number indicates that the same value holds for all stimuli. Normally, this value equals 20% of the luminance of the white in the adapting field.

`view` describes the viewing conditions, determining certain constants in the model:

`view=1`	Average surround with samples > 4º.
`view=2`	Average surround with samples < 4º.
`view=3`	Dim surround.
`view=4`	Dark surround.
`view=5`	Cut-sheet transparencies.

A surround with luminance factor above 20 is considered "average". If the luminance factor is below 20, it is dim, and when near zero is dark.

`adap` describes the degree of chromatic adaptation.

`adap=1`	There is no chromatic adaptation.
`adap=2`	Normal level of chromatic adaptation.
`adap=3`	Strong chromatic adaptation (the illuminant is partially discounted)
`adap=4`	Total chromatic adaptation (the illuminant is totally discounted)

`xyY` Nx3 matrix with the chromaticity coordinates and the luminance factor of the test stimuli.

Related Functions

`ciecam97`

Required Functions

colordgm

Purpose Plot colors in the chromatic diagram.

Description `colordgm` represents a set of colors together with the spectral colors in the chromatic diagram of the current color space.

`colordgm` accepts colors in any tristimulus-based representation: tristimulus vectors, *T*, chromatic coordinates and luminance, (*t,Y*), or dominant wavelength, purity and luminance, (*l,P,Y*).

`colordgm` may also plot two convenient 'all positive' triangles:

- The limits of colors with positive tristimulus values in the current basis.
- The limits of the color gamut reproducible in the current monitor (given by the chromaticities of the guns with maximum saturation).

`colordgm` allows the user to control many parameters of the plot:

- the color and width of the lines.
- the font size in the axis and labels.
- the symbols representing the colors, its size and color.
- a line may be plotted through the color points.
- numbers may be given to the plotted colors.
- the auxiliary color regions (the all positive triangles) may be enabled or disabled.

This function may be used in two ways:
- Easy. Only 5 colorimetric parameters and default graphic parameters.
- Comprehensive. 5 colorimetric parameters and 8 graphic parameters.

If you don’t specify the optional graphic parameters the easy way is taken.

Syntax

Easy: `colordgm(C,characteriz,T_l,Yw,tm);`

Comprehensive:

```
colordgm(C,characteriz,T_l,Yw,tm,symb,show_lin,...
show_numb,showtriang,colors,linewidths,...
linestyles,sizes);
```

`C` N*3 matrix (color-like variable) containing the set of N colors

`characteriz` Number indicating the color characterization used in the matrix C.
`characteriz` = 1 ⇒ Tristimulus vectors
`characteriz` = 2 ⇒ Chromatic coordinates and luminance
`characteriz` = 3 ⇒ dom. wavelength, excitation purity and luminance
`characteriz` = 4 ⇒ dom. wavelength, chromatic purity and luminance

`T_l` Color Matching Functions in the current basis (M*4 spectral-like matrix).

`Yw` Trichromatic units (1*3 vector).

`tm` Chromaticities of the monitor. 7*N matrix with the calibration data (see `calibrat` or `loadmon`).

`symb` String containing the symbol to be used to represent the colors.

`show_lin` Parameter to enable/disable the plot of a line through the color points.
If `show_lin` = 1 ⇒ Plots the line
If `show_lin` = 0 ⇒ No line

`show_numb` Parameter to enable/disable the plot of numbers to identify the colors.
If `show_numb` = 1 ⇒ Plots the numbers
If `show_numb` = 0 ⇒ No number

`showtriang` Parameter to enable/disable the plot of the 'all positive' triangles

If `showtriang` = 0

`replocus` doesn't plot any triangle at all

If `showtriang` = 1

It ONLY plots the triangle of the colors with all positive tristimulus values.

If `showtriang` = 2

It ONLY plots the triangle of generable colors.

If `showtriang` = 3

It plots BOTH triangles.

`colors` 8*3 matrix containing the *MATLAB* notation of the color for the following objects:

1- Lines of the bounding box
2- Locus of spectral colors
3- Axis of the plot
4- Triangle of primaries of the system
5- Triangle of 'primaries' of the monitor
6- Line through the colors.
7- Symbols representing the colors
8- Numbers associated to the colors.

`linewidths` 1*6 vector containing the widths of the lines (in the order given above).
The LineWidth of the symbols and the numbers cannot be selected because they are not lines!

`linestyles` 6*2 matrix containing the strings that define the style of the lines (in the order given above) according to the convention used in plot.
The LineStyle of the symbols and the numbers cannot be selected because they are not lines!

`sizes` 1*4 vector containing the sizes of:
1- the numbers in the axis.
2- the labels of the axis.
3- the symbols used to represent the colors.

4- the numbers used to represent the colors.

If you don’t want to waste your time thinking on aesthetics, here you have an example you can use to begin with (cut, paste and explore!):

```
symb='o';
show_lin=1;
show_numb=1;
showtriang=3;
linecolors=[0 0 0;0 0 1;0 0 0.5;0 0.5 0;0.5 0 0;0.5 0.4
0;0.5 0.4 0;0.5 0.4 0];
linewidths=[0.5 0.5 0.5 0.5 0.5 0.5];
linestyles=['- ';'- ';'- ';'- ';': ';'- '];
fontsizes=[10 12 3 8];

colordgm(C,characteriz,T_l,Yw,tm,symb,show_lin,...
show_numb,showtriang,linecolors,linewidths,...
linestyles,fontsizes);
```

This means that it will plot the locus and both triangles, the bounding box will be black, the locus will be blue, the axis will be dark blue, the triangle of primaries will be dark green and the monitor triangle will be dark red. All the lines will be solid but the monitor line and all of them will be 0.5 width. The numbers of the axis will be size 10 and the labels size 12. The colors will be represented with brown circles of size 3, numbered with brown numbers of size 8 and interpolated with a solid brown line of width 0.5.

Related Functions

`colorspc, replocus`

Required Functions

`tri2coor, niv2coor, mini, ganadora, lp2coor, replocus, maxi`

colorspc

Purpose

Plot colors in the tristimulus space.

Description

`colorspc` accepts colors in any tristimulus-based color description: tristimulus vectors, *T*, chromatic coordinates and luminance, (*t,Y*), or dominant wavelength, purity and luminance, (*l,P,Y*). `colorspc` may also plot the chromatic diagram in the tristimulus space.

`colorspc` allows the user to control many parameters of the plot:

- the color and width of the lines.
- the font size in the axis and labels.
- the symbols representing the colors, its size and color.
- vector lines may be associated to the color points.
- a line may be plotted through the color points.
- numbers may be given to the plotted colors.
- the different elements of the chromatic diagram may be enabled or disabled.

This function may be used in two ways:
- Easy. Only 5 colorimetric parameters and default graphic parameters.
- Comprehensive. 5 colorimetric parameters and 11 graphic parameters.

If you don't specify the optional graphic parameters the easy way is taken.

Syntax

Easy: `colorspc(C,characteriz,T_l,Yw,tm);`

Comprehensive:

```
colorspc(C,characteriz,T_l,Yw,tm,symb,lim_axis,...
showvectors,show_lin,show_numb,showdiag,...
showtriang,colors,linewidths,linestyles,sizes);
```

`C` N*3 matrix (color-like variable) containing the set of N colors

`characteriz` Number indicating the color characterization used in the matrix C.
`characteriz` = 1 ⇒ Tristimulus vectors
`characteriz` = 2 ⇒ Chromatic coordinates and luminance
`characteriz` = 3 ⇒ dom. wavelength, excitation purity and luminance
`characteriz` = 4 ⇒ dom. wavelength, chromatic purity and luminance

`T_l` Color Matching Functions in the current basis (M*4 spectral-like matrix).

`Yw` Trichromatic units (1*3 vector).

`tm` Chromaticities of the monitor. 7*N matrix with the calibration data (see `calibrat` or `loadmon`).

`symb` String containing the symbol to be used to represent the colors.

`lim_axis` Automatic/manual axis limits
If `lim_axis`=0 ⇒ Automatic determination of the limits
else, `lim_axis` must be a 1*6 vector containing the limits of the 3D axis as in the `axis` function.

`showvectors` Parameter to enable/disable the plot of a vector line for each color
If `showvectors` = 1 ⇒ Plot vector lines
If `showvectors` = 0 ⇒ No vector lines

`show_lin` Parameter to enable/disable the plot of a line through the color points.
If `show_lin` = 1 ⇒ Plots the line
If `show_lin` = 0 ⇒ No line

`show_numb` Parameter to enable/disable the plot of numbers to identify the colors.

If `show_numb` = 1 ⇒ Plots the numbers
If `show_numb` = 0 ⇒ No number

`showdiag` Parameter to enable/disable the plot of the chromatic diagram
If `showdiag` = 1 ⇒ Plots the diagram
If `showdiag` = 0 ⇒ Doesn't plot the diagram

`showtriang` Parameter to enable/disable the plot of the 'all positive' triangles
If `showtriang` = 0
`replocus` doesn't plot any triangle at all
If `showtriang` = 1
It ONLY plots the triangle of the colors with all positive tristimulus values.
If `showtriang` = 2
It ONLY plots the triangle of generable colors.
If `showtriang` = 3
It plots BOTH triangles.

`colors` 8*3 matrix containing the *MATLAB* notation of the color for the following objects:
1- Lines of the bounding box
2- Locus of spectral colors
3- Axis of the plot
4- Triangle of primaries of the system
5- Triangle of 'primaries' of the monitor
6- Line through the colors.
7- Symbols representing the colors
8- Numbers associated to the colors.

`linewidths` 1*6 vector containing the widths of the lines (in the order given above).
The LineWidth of the symbols and the numbers cannot be selected because they are not lines!

`linestyles` 6*2 matrix containing the strings that define the style of

the lines (in the order given above) according to the convention used in plot.
The LineStyle of the symbols and the numbers cannot be selected because they are not lines!

`sizes` 1*4 vector containing the sizes of:
1- the numbers in the axis.
2- the labels of the axis.
3- the symbols used to represent the colors.
4- the numbers used to represent the colors.

If you don't want to waste your time thinking on aesthetics, here you have an example you can use to begin with (cut, paste and explore!):

```
symb='o';
lim_axis=0;
showvectors=1;
show_lin=1;
show_numb=1;
showdiag=1;
showtriang=3;
linecolors=[0 0 0;0 0 1;0 0 0.5;0 0.5 0;0.5 0 0;0.3 0.2
0;0.6 0.2 0;0.3 0.2 0;0.3 0.2 0];
linewidths=[0.5 0.5 0.5 0.5 0.5 0.5 0.5];
linestyles=['- ';'- ';'- ';'- ';': ';'- ';'- '];
sizes=[10 12 1.5 8];

colorspc(C,characteriz,T_l,Yw,tm,symb,lim_axis,...
showvectors,show_lin,show_numb,showdiag,showtriang,...
linecolors,linewidths,linestyles,fontsizes);
```

Related Functions `colordgm, replocus`

Required Functions `tri2coor, lp2coor, mini, maxi, niv2coor, ganadora`

compcop

Purpose

Non-linearity acting on the opponent mechanisms.

Description

`compcop` applies a compressive non-linearity to the linear *ATD* responses of the specified stage and model.

Among all the models implemented in *COLORLAB*, only Guth's models have this mechanism. The non-linearity is of Naka-Rushton type:

$$F' = \frac{F}{F + k}$$

where F=A, T or D. In ATD80, k=0.008. In the revisions of this model k=200.

This function is called by `con2atd`.

Syntax

`ATDG=compcop(ATD,stage,model)`

`ATD`	Linear *ATD* responses to a set of stimuli. For N stimuli, this is a Nx3 matrix.
`stage`	1 or 2, depending on whether `ATD` correspond to the first or the second opponent stage of the model.
`model`	Integer (1-13) identifying the model. See `xyz2atd` for details. Only models 8-11 have this mechanism.
`ATDG`	Non-linear *ATD* responses. For N stimuli, this is a Nx3 matrix.

Related Functions `incomcop, con2atd, xyz2atd`

Required Functions

con2atd

Purpose

Compute *ATD* responses from cone responses.

Description

`con2atd` computes the responses of the first and second opponent stages of the specified color vision model from the cone responses *LMS*, as obtained by `xyz2con`, under certain adaptation conditions.

COLORLAB includes the following models with opponent stages:

Jameson and Hurvich (1957)
: One opponent linear stage with adaptation matrix. A second linear transform includes adaptation conditions. At threshold, this linear transform is the identity.

Ingling and Tsou (1977)
: One opponent linear stage, with different matrixes for threshold suprathreshold

Boynton (1986)
: One opponent linear stage, without adaptation.

Guth (1980)
: One opponent linear stage with adaptation matrix.

Guth (1990, 1993, 1994 and 1995)
: Two opponent non-linear stages. The adaptation mechanism is already included in the model.

DKL
: Opponent modulation space. LMS must be increments on a background.

This functions is used by `xyz2atd`

Syntax

```
[ATD1,ATD2]=con2atd(LMS,model,adap)
```

`LMS`	Cone responses. Color format.
`model`	Integer (4-11, 13) describing the model.

`models` 1:3 and 12 do not have opponent stages.
`model`=4 Jameson and Hurvich (1957)
`model`=5 Ingling and Tsou (1977)
`model`=6 Boynton (1986)
`model`=7 Guth (1980)
`model`=8 Guth (1990)
`model`=9 Guth (1993)
`model`=10 Guth (1994)
`model`=11 Guth (1995)
`model`=13 DKL

`adap` In models 4 and 7, `adap` is a 3x3 matrix scaling the ATD responses. In model 5, is a scalar, and in model 13 a 1x3 vector with the white tristimulus values.

`model`=4	At threshold, `adap=eye(3)`. This is the default value.
`model`=5	At threshold `adap`=1. Otherwise, `adap`=2. By default, `adap`=2;
`model`=7	At threshold, `adap=eye(3)`. This is the default value.
`model`=13	`adap=XYZB`;

When this parameter is not necessary or when the default parameters are valid, use `[ATD1,ATD2]=con2atd(LMS,model).`

`ATD1` and `ATD2` First and second opponent stage descriptors of the stimuli.
With one-stage models, use `ATD1=con2atd(...)`.

Related Functions `xyz2con, con2xyz, atd2con, xyz2atd, atd2xyz`

Required Functions `compcop`

con2xyz

Purpose Compute *XYZ* from cone responses.

Description `con2xyz` computes the *XYZ* tristimulus values of a set of stimuli from the cone responses in a cone-space or in a color-vision model using linear and non-linear transforms.

This transform needs information about the model or set of fundamentals used (`model`), the background (`XYZB`), and the non-linear mechanisms of the model (`weights`, `mode` and `sigma`).

FUNDAMENTALS AND MODELS

`model`=1 Vos and Walraven's fundamentals
Y=L+M+S
XYZ are computed with the CIE-1931 observer modified by Judd (1951).

`model`=2 Wyszecki and Stiles's fundamentals
L(475.5)+M(475.5)=16S(475.5)
XYZ are computed with the CIE-1931 observer modified by Judd (1951).

`model`=3 MacLeod-Boynton's fundamentals
L(400)+M(400)=S(400)
XYZ are computed with the CIE-1931 observer modified by Vos (1978).

`model`=4 Jameson and Hurvich (1957)
XYZ are described with the CIE-1931 observer.

`model`=5 Ingling and Tsou (1977)

`model`=6 Boynton (1986)
L(498)+M(498)=S(498)

`model`=7 Guth (1980)
L, M and S normalized to one

`model`=8 Guth (1990)
XYZ is normalized so that Y=0.0015 equals 100 Td
Smith and Pokorny fundamentals, normalized to one
L:M:S0.66:1:0.55
Cone gain-control:
XYZB represents the background or the adapting stimulus. Both the test and the background may contribute to the adapting stimulus. This is controlled by `weight`:

`weight`=[Test contribution, Background contribution]. These values sum to one. In the different updates of the model, they may take any value.
The meanings of XYZB, mode and weight in models 8-11 are the same.
Defaults: XYZB=[0 0 0], mode=2, weight=[1 0]. These values apply also to models 9-11.

`model`=9 Guth (1993): XYZ are transformed to trolands.
Smith and Pokorny fundamentals, normalized to one
L:M:S0.66:1:0.45
Gain-control

$$LMS' = LMS \frac{sigma}{sigma + LMSA}$$

`sigma`=400 but to compute MacAdam's ellipses `sigma`=220.
By default, `sigma`=400.

`model`=10 Guth (1994):XYZ are transformed to trolands.
Smith and Pokorny fundamentals, normalized to one
L:M:S0.66:1:3
Gain-control

$$LMS' = LMS \frac{sigma}{sigma + LMSA}$$

`sigma`=400 but to compute MacAdam's ellipses

`sigma`=220.
By default, `sigma`=400.

`model`=11 Guth (1995):XYZ are transformed to trolands.
Smith and Pokorny fundamentals, normalized to one
L:M:S0.66:1:0.43
Gain-control

$$LMS' = LMS \frac{sigma}{sigma + LMSA}$$

`sigma`=300 but to compute MacAdam's ellipses
`sigma`=220.
By default, `sigma`=300.

`model`=12 Stockman and Sharpe's fundamentals (2000)

Except for `model`=1 and `model`=12, the Smith and Pokorny fundamentals, with different normalization conditions, are used.

Syntax

`XYZ=con2xyz(LMS,model,XYZB,weight,mode,sigma)`

`LMS`	Cone responses. Color format.
`model`	Integer (1-13) identifying the model. A brief description of each model and the necessary parameters can be seen below.
`XYZB`	Background stimulus (3x1)
`weigth`	Contributions of test stimuli and the background to the adaptation stimulus. Use only with models 8-11.
`mode`	If `LMS` are increments on `XYZB`, mode=1. Otherwise, mode=2. Use only with models 8-11.
`sigma`	Scalar parameter controlling the non-linearity at the cone stage.

Use only with models 8-11.

`XYZ` Output tristimulus values.
The tristimulus space is either CIE-1931 or this space as modified by Judd or by Judd and Vos.

With `con2xyz(LMS, model)`, the rest of the parameters are set to zero in `model`≤7 and `model`=12 and to the default parameters in the rest.

Related Functions

`xyz2con, xyz2atd, atd2xyz, atd2perc, perc2atd, con2atd, atdf2con, atd1atd2`

Required Functions

`xyz2xyzn, ingancon, xyzn2xyz, cambial, xyztd2l`

coor2lp

Purpose Compute dominant wavelength, purity and luminance from chromatic coordinates and luminance.

Description We can choose chromatic or excitation purity:

P_excitat ⇒ `opt=1`
P_chromat ⇒ `opt=2`

Syntax `lpY=coor2lp(tY,opt,T_l,Yw);`

`tY` Input chromatic coordinates and luminances (color-like variable)

`opt` Selects the kind of output purity:
P_excitat ⇒ `opt=1`
P_chromat ⇒ `opt=2`

`T_l` Color matching functions

`Yw` Trichromatic units

`lpY` Output colors expressed in dominant wavelength in nm, purity and luminance in cd/m^2 [l P Y]
Purples are characterized by the complementary dominant wavelength with negative sign.

Related Functions `lp2coor, coor2tri, tri2coor`

Required Functions `coor2tri`

coor2tri

Purpose Compute tristimulus values from chromaticity coordinates.

Description The trichromatic units of the reference system used must be specified.

Syntax `T=coor2tri(t,Yw);`

`t` Output chromatic coordinates and luminance (color-like variable, [t_1 t_2 Y]).

`Yw` Trichromatic units [$Y_w(P_1)$ $Y_w(P_2)$ $Y_w(P_3)$]

`T` Input tristimulus vectors (color-like variable, colors in rows).

Related Functions `tri2coor, coor2lp, lp2coor`

Required Functions

defcolor

Purpose Graphic definition of colors.

Description `defcolor` opens a window where the user selects the chromaticity of an arbitrary number of colors. Then, the user is prompted to introduce a luminance value for each selected chromaticity.

`defcolor` returns the colors expressed in chromatic coordinates and luminance

Syntax `t=defcolor(T_l,tm);`

`T_l`	Color Matching Functions in the current basis (N*4 spectral-like matrix).
`tm`	Chromaticities of the monitor. 7*N matrix with the calibration data (see `calibrat` or `loadmon`).
`t`	N*3 matrix (color-like variable) containing the chromaticities and luminances of the selected colors.

Related Functions `savecol, loadcol, defcroma`

Required Functions `defcroma, colordgm, colord_c, tri2coor, lp2coor, replocus, mini, maxi, niv2coor, ganadora`

defcroma

Purpose Graphic definition of N chromaticities.

Description `defcroma` opens a window with the current chromatic diagram for the user to choose the chromaticities from.

`defcroma` returns the selected chromaticities.

Syntax `t=defcroma(T_l,tm);`

`T_l`	Color Matching Functions in the current basis (M*4 spectral-like matrix).
`tm`	Chromaticities of the monitor. 7*K matrix with the calibration data (see `calibrat` or `loadmon`).
`t`	Output chromaticities (N*2 matrix)

Related Functions `defcolor, savecol, loadcol`

Required Functions `colordgm, tri2coor, lp2coor, replocus, mini, maxi, niv2coor, ganadora`

defillu

Purpose Graphic definition of illuminants.

Description `defillu` opens an interactive window where the user can define (draw) the relative spectral radiance. This relative spectrum is scaled to give the desired luminance or radiance.

`defillu` returns a spectral-like variable (wavelength-magnitude) in nm and Wstr^{-1}m^{-2} respectively. The wavelength domain will be given by the limits of the color matching functions at hand.

Syntax `esp=defillu(Y,opt,lambda,T_l,Yw,fig);`

`Y`	Luminance (in cd/m^2) or radiance (in Wstr^{-1}m^{-2}) The meaning of `Y` (lum or rad) depends on the value of `opt`.
`opt`	Selects the meaning of `Y`. If `opt==1`, `Y` means luminance, else, `Y` means radiance.
`D_lambda`	Wavelength step (in nm) to sample the spectrum.
`T_l`	Color matching functions.
`Yw`	Trichromatic units (in cd/m^2).
`fig`	Figure number

Related Functions `loadillu, loadilum, saveillu`

Required Functions

defrefl

Purpose Graphic definition of spectral reflectances/transmittances.

Description `defrefl` opens an interactive window where the user can define (draw) the spectral reflectance or transmittance. The reflectance (transmittance) should always be in the [0 1] range.

`defrefl` returns a spectral-like variable (wavelength-magnitude). The reflectance (transmittance) is interpolated in 1-nm steps within the given range.

Syntax

```
refl=defrefl([l_min l_max],fig);
```

`[l_min l_max]`	Wavelength range (in nm)
`fig`	Figure number

Related Functions `loadrefl, loadrefm, saverefl`

Required Functions

defsys

Purpose Graphic definition of a new color system.

Description `defsys` opens a window for the user to choose the chromaticities of the new primaries and the new white.

The only numerical data about the new system required by `defsys` is the luminance of the new white.

`defsys` returns the data that defines a color system and other convenient data:

- The color matching functions in the new system, `T_l2.`
- The trichromatic units of the new system, `Yw2.`
- The change-of-basis matrix, `M2x`, that relates the new system, `S2,` to the CIE XYZ system.
- (Not necessary but convenient) the change-of-basis matrix, `M12`, that relates the 'new' system, `S2`, to the 'old' system, `S1.`
- (Also convenient in *COLORLAB* are) the chromaticities of the monitor, `tm2.`

Syntax

```
[M2x,M12,T_l2,Yw2,tm2]=defsys(YW2,T_l1,Yw1,M1x,tm1);
```

`YW2` Luminance (in cd/m^2) of the new white.

`T_l1` Color matching functions in the old basis (spectral-like variable, N*4 matrix).

`Yw1` Trichromatic units of the old basis (1*3 vector).

`M1x` The 3*3 change-of-basis matrix that relates the old system, S1, to the CIEXYZ system.

`tm1` Chromaticities of the monitor in the old basis (7*N matrix).

`M2x` The 3*3 change-of-basis matrix that relates the new system, S2, to the CIEXYZ system.

`M12` The 3*3 change-of-basis matrix that relates the old system, S1, to the new one, S2.

`T_l2` Color matching functions in the new basis (spectral-like variable, N*4 matrix).

`Yw2` Trichromatic units of the new basis (1*3 vector).

`tm2` Chromaticities of the monitor in the new basis (7*N matrix).

Related Functions

`defsysm, savesysm, lodadys, loadsysm`

Required Functions

```
chngmtx, coor2tri , newbasis, defcroma, colordgm,
colord_c, tri2coor, lp2coor, replocus, mini, maxi,
niv2coor, ganadora
```

defsysm

Purpose Define a new color system from (manually) given information.

Description The data that define a new color system, S2, are:

- The color matching functions, `T_l2`

- The trichromatic units, `Yw2`

- The change-of-basis matrix, `M2x`, that relates the system, S2, to the system CIE XYZ.

- (Not necessary but convenient) the change-of-basis matrix, `M12`, that relates the 'new' system, S2, to the 'old' system S1.

- (Also convenient in *COLORLAB* are) the chromaticities of the monitor, `tm2`.

`defsysm` solves the problem from the old data in two different cases (from different initial data defining the new system):

Case 1: We know,

- Tristimulus values of the new color primaries in the old basis: $T_i(P_j')$

Case 2: We know,

- Chromatic coordinates of the new color primaries in the old basis: $t_i(P_j')$

- The characterization of the new reference white, W':

- Chromatic coordinates (in the old basis), $t_i(W')$, and luminance, $Y(W')$.

Syntax

```
[M2x,M12,T_l2,Yw2,tm2]=defsysm(DATA,option,T_l1,Yw1,M1x,
tm1);
```

`M2x` The 3*3 change-of-basis matrix that relates the new system, s2, to the CIEXYZ system.

`M12` The 3*3 change-of-basis matrix that relates the old system, s1, to the new one, S2.

`T_l2` Color matching functions in the new basis.

`Yw2` Trichromatic units of the new basis (1*3 vector).

`tm2` Chromaticities of the monitor in the new basis (7*N matrix).

`T_l1` Color matching functions in the old basis.

`Yw1` Trichromatic units of the old basis (1*3 vector).

`M1x` The 3*3 change-of-basis matrix that relates the old system, s1, to the CIEXYZ system.

`tm1` Chromaticities of the monitor in the old basis (7*N matrix).

`DATA` 3*3 matrix with the data to compute the matrix `M12`.
The interpretation of `DATA` (case 1 or case 2) depends on the value of the variable `option`. See below for details on how to introduce `DATA` in each case.

`option` Variable that controls the case (the meaning of `DATA`).
If `opt==1` we know the tristimulus values of the new color primaries $T_i(P_j')$
In this case, you have to introduce `DATA` in this way:

$$\texttt{DATA} = \begin{pmatrix} T_1(P_1') & T_2(P_1') & T_3(P_1') \\ T_1(P_2') & T_2(P_2') & T_3(P_2') \\ T_1(P_3') & T_2(P_3') & T_3(P_3') \end{pmatrix}$$

i.e.:

```
DATA=[T1(P1')T2(P1')T3(P1');...
T1(P2')T2(P2')T3(P2');T1(P3')T2(P3')T3(P3')];
```

If `opt`==2 we know the chromaticities of the new color primaries and the white.
In this case, you have to introduce `DATA` in this way:

$$\mathtt{DATA} = \begin{pmatrix} t_1(P_1') & t_2(P_1') & t_1(W') \\ t_1(P_2') & t_2(P_2') & t_2(W') \\ t_1(P_3') & t_2(P_3') & Y(W') \end{pmatrix}$$

That is,

```
DATA=[t1(P1')t2(P1')t1(W');t1(P2')t2(P2')t2(W');
t1(P3')t2(P3')Y(W')];
```

Related Functions `defsys, savesysm, loadsys, loadsysm`

Required Functions `chngmtx,coor2tri,newconst,newbasis`

difsvf

Purpose

Compute perceptual distances in SVF.

Description

`difsvf` computes the color difference in the SVF space between two stimuli, under the specified illumination conditions.

The definition of distance used weights differently the achromatic and chromatic contributions:

$$d(C_1, C_2) = \sqrt{2.3\Delta^2 V + \Delta^2 F_1 + \Delta^2 F_2}$$

Syntax

`D=difsvf(XYZ1,XYZ2,XYZW)`

`XYZ1, XYZ2`	Tristimulus CIE-1931 values of the two set of stimuli whose color differences we want to compute. Both parameters are Nx3 matrixes, with N=number of colors.
`XYZW`	Tristimulus CIE-1931 values of the reference white.
`D`	Color difference (Nx1 matrix).

Related Functions

`xyz2svf, svf2xyz`

Required Functions

`xyz2svf, svf2xyz`

gancon

Purpose Compute non-linear cone responses.

Description Function used by xyx2con to compute the gain-controlled cone response within the specified model.

The gain control is of the form

$$LMS_{gc} = LMS\left(a + b\,\frac{sigma}{LMS(A) + sigma} \right)$$

where *LMS* are the linear cone-responses to the test stimulus, *LMS(A)* the cone-responses to the adapting stimulus and *a*, *b* and *sigma* constants of the model.

For linear models, *b*=0 and *a*=1. For Guth's ATD90, *a*=0.01, *b*=0.99, *sigma*=0.05. The subsequent revisions of this model take *a*=0 and *b*=1. For the possible values of *sigma*, see `xyz2con`.

Syntax `lms=gancon(lmst,lmsf,model,sigma)`

`LMST`	Tristimulus values of the tests in the lineal cone-space of the corresponding model (Nx3 matrix for N tests).
`LMSF`	Tristimulus values of the background in the lineal cone-space of the corresponding model. The size of `LMSF` must be the same than that of `LMST`. If all the stimuli in `LMST` are seen against the same background, it must be repeated the appropriate number of times.
`model`	Integer (1-13) identifying the model. See `xyz2atd` for details.
`sigma`	Scalar controlling the non-linearity. See `xyz2con` for details.

`LMS`	Gain-controlled cone responses to the test stimuli.

Related Functions

`ingancon`

Required Functions

icompcop

Purpose Undo the effect of opponent-stage non-linearities.

Description `icompcop` is the inverse of `compcop`. Returns uncompressed *ATD* responses from the compressed `ATDC` responses of the specified `stage` and `model`. See `compcop` for details.

This function is called by `atd1atd2` and `atd2con-`

Syntax

```
ATD=icompcop(ATDC,stage,model)
```

`ATDC`	Non-linear *ATD* responses. For N stimuli, this is a Nx3 matrix.
`stage`	1 or 2, depending on whether `ATD` correspond to the first or the second opponent stage of the model.
`model`	Integer (1-13) identifying the model. See `xyz2atd` for details. Only models 8-11 have this mechanism.
`ATD`	Linear *ATD* responses to a set of stimuli. For N stimuli, this is a Nx3 matrix.

Related Functions `compcop, atd1atd2, atdf2con`

Required Functions

ingancon

Purpose

Undo the effect of the cone-stage non-linearity.

Description

The function undoes the effect of the gain-control on the cone-responses.

`ingancon` is called by `con2xyz.`

Syntax

`LMS=ingancon(LMSTGAN,LMSBACK,model,weights,sigma);`

`LMSTGAN`	Gain-controlled cone-responses. For N colors, this is a Nx3 matrix.
`LMSBACK`	Tristimulus values of the background in the lineal cone-space of the corresponding model. The size of `LMSF` must be the same than that of `LMSTGAN`. If all the stimuli in `LMSTGAN` are seen against the same background, this must be repeated the appropriate number of times.
`model`	Integer (1-13) identifying the model. See `xyz2atd` for details.
`weights`	[wt wb]. Contributions of test and background to the adapting stimulus See `xyz2con` for details.
`sigma`	Scalar controlling the non-linearity. See xyz2con for details.

Related Functions

`gancon, xyz2con, con2xyz`

Required Functions

lab2perc

Purpose

Compute perceptual descriptors in CIELAB.

Description

`lab2perc` computes the lightness (L^*), chroma (C^*) and hue angle (h^*, in radians) of a set of colors characterized in the CIELAB space.

$$L^* = 116 f\left(\frac{Y}{Y_W}\right) - 16$$

$$C^* = \sqrt{a^{*2} + b^{*2}}$$

$$h^* = arctan\left(\frac{b^*}{a^*}\right), 0 \le h^* \le 2\pi$$

Syntax

`LhC=lab2perc(Lab)`

`Lab` Lightness and chromaticity coordinates of the stimuli in CIELAB. For N stimuli, this is a Nx3 matrix.

`LhC` For N stimuli, Nx3 matrix. The first column contains the lightness L^*, the second the hue angle h^*, in radians, and the third the chroma C^*.

Related Functions

`perc2lab, xyz2lab, lab2xyz`

Required Functions

lab2xyz

Purpose Compute *XYZ* from lightness and chromaticity coordinates in CIELAB space.

Description `lab2xyz` computes the tristimulus values in CIEXYZ of a set of colors from their lightness, *L**, and chromatic coordinates *a** and *b** in CIELAB.

CIELAB is a simple appearance model providing perceptual descriptors (lightness, hue and chroma) for related colors (colors in a scene).

In this representation, information about the illumination conditions or, alternatively, about the scene, is included in a reference stimulus. Using CIELAB in the standard conditions implies that the reference stimulus is a perfect diffuser illuminated as the test.

Syntax `XYZ=lab2xyz(Lab,XYZR)`

`Lab`	For N colors, Nx3 matrix, containing, in columns, the lightness *L**, and the chromaticity coordinates *a** and *b**.
`XYZR`	Tristimulus values of the reference stimulus. If the reference stimulus is the same for all the test stimuli, this is a 1x3 matrix. If the reference is different for each test stimulus `XYZR` is a Nx3 matrix.
`XYZ`	Tristimulus values of the test stimuli. For N colors, this is a Nx3 matrix.

Related Functions `xyz2lab, lab2perc, perc2lab`

Required Functions

llab2xyz

Purpose Transform *LCH* in LLAB space to *XYZ*.

Description `llab2xyz` computes the XYZ tristimulus values of stimuli whose perceptual descriptors Lightness (*L*), chroma (*C*) and hue angle (*H*), in degrees, are known in the LLAB space.

The tristimulus values of the input and ouput stimuli are normalized so that *Y*=100 for the equal-energy stimulus, illuminated as the test, which is the reference stimulus. The absolute luminance values of the reference white and the background are also required. The remaining observation conditions are specified by parameter `mo`.

Syntax `XYZ=llab2xyz(LCH,XYZW,YW,YB,mo)`

`LCH`	[L C H(º)] in LLAB space. For N colors, this is a Nx3 matrix.
`XYZW`	Relative tristimulus values of the reference white.
`YW`	Luminance (cd/m^2) of the reference white (1x1).
`YB`	Luminance factor of the background (1x1).
`MO`	Parameter describing the observation conditions: `M0`=1.....reflection samples, average surround, field>4º `M0`=2.....reflection samples, average surround, field<4º `M0`=3.....television and VDU, dim surround `M0`=4.....cut-sheet transparency in dim surround `M0`=5.....35 mm projection transparency in dark surround
`XYZ`	Relative tristimulus values of the samples. For N colors, this is a Nx3 matrix.

Related Functions	`xyz2llab`
Required Functions	`calcc2`

loadcol

Purpose Load a set of colors in *COLORLAB* format.

Description `loadcol` returns the colors in the appropriate color space and in the desired representation.

The meaning of a given color characterization depends on its nature -for instance, *T* versus (*t*,*Y*)- and on the color space where the color was defined -for instance CIERGB versus NTSCRGB-.

The *COLORLAB* format for color storage includes the necessary additional information to ensure an appropriate retrieval from any other tristimulus space and required representation.

A *COLORLAB* color file consists of three variables:

- `T`, a N*3 color-like matrix that specifies the N colors: .
- `Msx`, a 3*3 change-of-basis matrix that specifies the color space where `T` is defined. It relates the system with the CIEXYZ system.
- `characteriz`, a parameter that specifies the representation used in `T`.

Syntax `T=loadcol(T_l,Yw,Msx,characteriz,'file');`

`T_l`	Specification of the current color system (the color system where we want the output). Color matching functions.
`Yw`	Specification of the current color system (the color system where we want the output). Trichromatic units.
`Msx`	Specification of the current color system (the color system where we want the output): the variable, `Msx`, 3*3 change-of-basis matrix that relates the system with the CIEXYZ. (This matrix is given in the definition of the system. See `defsys)`

`characteriz` Specification of the representation: the variable, `characteriz`, describes the nature of the representation used in `T`.

`characteriz` = 1.....	Tristimulus vectors
`characteriz` = 2.....	Chromatic coordinates and luminance
`characteriz` = 3.....	Dominant wavelength, excitation purity and luminance
`characteriz` = 4.....	Dominant wavelength, colorimetric purity and luminance

`file` String with the path to `*.mat` file we want to load.

`T` Output specification of the N colors in the desired space and representation: N*3 color-like matrix.

Related Functions

`defcolor, savecol, defcroma`

Required Functions

`newbasis, coor2tri, lp2coor, tri2coor, coor2lp`

loadillu

Purpose Load illuminants using a dialog box.

Description `loadillu` opens a dialog box where the user can select the file to load the illuminant from.

This file should have been created by `saveillu` or contain a spectral-like variable named `iluminan`.

The relative spectrum in the file is scaled to give the desired luminance or radiance.

`loadillu` returns a spectral-like variable (wavelength-magnitude) in nm and Wstr^{-1}m^{-2} respectively. The wavelength domain will be given by the limits of the color matching functions at hand. You can select the sampling resolution of the wavelength domain.

Syntax `esp=loadillu(Y,opt,D_lambda,T_l,Yw);`

`Y` Luminance (in cd/m^2) or radiance (in Wstr^{-1}m^{-2})
The meaning of `Y` (lum or rad) depends on the value of `opt`.

`opt` Selects the meaning of `Y`.
If `opt`==1, `Y` means luminance; else, `Y` means radiance.

`D_lambda` Wavelength step (in nm) to sample the spectrum.

`T_l` Color matching functions.

`Yw` Trichromatic units (in cd/m^2).

Related Functions `loadilum, defillu, saveillu`

Required Functions

loadilum

Purpose Load illuminant from a (manually) given file.

Description This file should have been created by `saveillu` or contain a spectral-like variable named `iluminan`.

The relative spectrum in the file is scaled to give the desired luminance or radiance.

`loadilum` returns a spectral-like variable (wavelength-magnitude) in nm and Wstr^{-1}m^{-2} respectively. The wavelength domain is given by the limits of the color matching functions at hand. You can select the sampling resolution of the wavelength domain.

Syntax

```
esp=loadilum('path',Y,opt,D_lambda,T_l,Yw);
```

`'path'`	String containing the path to the file with the illuminant. Example: `'c:/MATLAB/toolbox/COLORLAB/colordat/,... illumin/iluminan.a'`
`Y`	Luminance (in cd/m^2) or radiance (in Wstr^{-1}m^{-2}) The meaning of `Y` (lum or rad) depends on the value of `opt`.
`opt`	Selects the meaning of `Y`. If `opt==1`, `Y` means luminance; else, `Y` means radiance.
`D_lambda`	Wavelength step (in nm) to sample the spectrum.
`T_l`	Color matching functions.
`Yw`	Trichromatic units (in cd/m^2).

Related Functions

`loadillu, defillu, saveillu`

Required Functions

loadmon

Purpose Open a dialog box to load the calibration data of the monitor.

Description The *'COLORLAB format for CRT calibration data storage'* is just a name convention for the variables involved in the CRT calibration. Such a convention is convenient for future automatic transform of the data to the required color system when using `loadmon`

COLORLAB assumes additivity and independence between the guns of the CRT monitor. *COLORLAB* also assumes an exponential relation (gamma relation) between the luminance of the gun, *i*, and the corresponding digital value, n_i (i=1,2,3). *COLORLAB* makes no assumption on the chromaticities of the guns, which may depend on the digital value in any complex way.

With these assumptions, the calibration data include:

- Chromaticities of the gun, *i*, as a function of the digital step, n_i.
- Parameters (a_i and g_i) of the gamma relation between the luminance of the gun, *i*, and the digital value, n_i:

$$Y_i = a_i n_i^{g_i}$$

`loadmon` loads these data from a (generic) file provided with *COLORLAB* (as `monito.mat`) or from a specific file generated with `calibrat` or `savemon` for your particular display (see the help of `calibrat` for details on the calibration).

Syntax `[tm,a,g]=loadmon(Msx);`

`Msx` 3*3 change-of-basis matrix that relates the system S (where the data are described) to the CIEXYZ system.

`tm` 7*N matrix that contains the chromaticities of the R, G and B

guns for N calibration points of the digital value (see `calibrat` for details).
The chromaticities are given in the system at hand even though they are stored in the CIEXYZ system (this is why `Msx` is needed).

`a` 1*3 vector including the constants a_i of the gamma relation for each gun.

`g` 1*3 vector including the constants g_i of the gamma relation for each gun.

Related Functions `savemon, loadmonm, calibrat, tri2val, val2tri`

Required Functions `coor2tri, newbasis`

loadmon

Purpose Load monitor calibration data from a (manually) given file.

Description The *'COLORLAB format for CRT calibration data storage'* is just a name convention for the variables involved in the CRT calibration. Such a convention is convenient for future automatic transform of the data to the required color system when using `loadmon`

COLORLAB assumes additivity and independence between the guns of the CRT monitor. *COLORLAB* also assumes an exponential relation (gamma relation) between the luminance of the gun, *i*, and the corresponding digital value, n_i (i=1,2,3). *COLORLAB* makes no assumption on the chromaticities of the guns, which may depend on the digital value in any complex way.

With these assumptions, the calibration data include:

- Chromaticities of the gun, *i*, as a function of the digital step, n_i.
- Parameters (a_i and g_i) of the gamma relation between the luminance of the gun, *i*, and the digital value, n_i:

$$Y_i = a_i n_i^{g_i}$$

`loadmonm` loads these data from a (generic) file provided with *COLORLAB* (as `monito.mat`) or from a specific file generated with `calibrat` or `savemon` for your particular display (see the help of `calibrat` for details on the calibration).

Syntax `[tm,a,g]=loadmonm('path',Msx)`

`'path'` String with the path to the file containing the data.
For example:
```
'c:\MATLAB\toolbox\COLORLAB\colordat\...
monitor\monito.mat'
```

`Msx`	3*3 change-of-basis matrix that relates the system S (where the data are described) to the CIEXYZ system.
`tm`	7*N matrix that contains the chromaticities of the R, G and B guns for N calibration points of the digital value (see `calibrat` for details). The chromaticities are given in the system at hand even though they are stored in the CIEXYZ system (this is why `Msx` is needed).
`a`	1*3 vector including the constants a_i of the gamma relation for each gun.
`g`	1*3 vector including the constants g_i of the gamma relation for each gun.

Related Functions

`savemon, loadmon, calibrat, tri2val, val2tri`

Required Functions

`coor2tri, newbasis`

loadrefl

Purpose Load a reflectance/transmittance using dialog box.

Description `loadrefl` opens a dialog box where the user can choose a file from which to load the reflectance/transmittance.

`loadrefl` returns a spectral-like variable (wavelength-magnitude).

NOTE (on the Munsell Reflectance Database)

COLORLAB comes with a database of reflectances corresponding to the color samples of the Munsell Book of Color. In this book the samples are classified according to Hue, Chroma and Value (Lightness).

The notation used in the files is as follows:

* The files are organized in hue-named folders:

```
r,  yr,  y,  gy,  g,  bg,  b,  pb,  p,  rp
```

* Inside each folder, the name of the files is made of 8 numbers:

```
HHHHVVCC
```

The first 4 numbers `(HHHH)` mean HUE. They may have values in the range `HHHH`=0000..0250....0500....1000.

Going from 0000 to 1000 means fine change from one coarse hue to the next one (for example from 'green' -g- to green-bluish -bg-).

The next 2 numbers `(VV)` mean VALUE. They may have values in the range `VV`= 00, 10, 20, 30, 40, ...,90.

The last 2 numbers `(CC)` mean CHROMA. They may have values in the range `CC`= 00, 01, 02 ... 16.

Syntax

```
reflec=loadrefl;
```

`reflec`	Reflectance matrix.

Related Functions

`defrefl, loadrefm, saverefl`

Required Functions

loadrefm

Purpose Load a reflectance/transmittance from explicit file.

Description This file should have been generated with `saverefl` or contain a spectral-like variable named `reflec`.

`loadrefm` returns a spectral-like variable (wavelength-magnitude).

`loadrefm` also provides an easy way to load sets of Munsell reflectances (which would be a pain otherwise!).

See **Syntax** and the note on Munsell reflectances below.

NOTE: The Munsell notation in *COLORLAB*.

COLORLAB comes with a database of reflectances corresponding to the color samples of the Munsell Book of Color. This color atlas contains colored and gray samples.

The gray samples are identified by a single parameter, *N*, describing their lightness. *N* has values between 0.5 and 9.5, in 0.25 steps. The lower *N*, the darker appears the sample when seen against the reference white.

The colored samples are classified according to Hue, Chroma and Value (Lightness).

The main hue, *h*, is a name identifying the color as blue, green-blue, green, green-yellow, yellow, yellow-red, red, red-purple, purple or purple blue. In this function, *h* is an integer between 1 and 10 (1 for blue, 2 for green-blue, 3 for green, and so on).

H identifies different shades of the same hue. The number of shades for each hue is not constant. Possible *H* values are between 1.25 and 10, in 1.25 steps.

The "value", *V*, is a measurement of the lightness of the sample. This descriptor takes integer values between 0 and 9.

C is the chroma of the sample, a parameter measuring the color content of the sample compared with that of the reference white in the same scene. Possible chroma values are 0, 1 and even values between 2 and 16.

`loadrefm` accepts as input a matrix of the form `[H1 V1 C1 h1;H2 V2 C2 h2;...]` describing N colors by their Munsell notation.

See `loadrefl` for a note on the file names of the Munsell reflectance.

Syntax

```
reflec=loadrefm(identif);
```

`identif` Parameter that identifies the reflectance/transmittance file(s) to be loaded.
This parameter can be a string indicating the path of the file or it can be a matrix, `[H V C h]` (nx4 for n colored samples) or `[N]` (nx1 for n grey samples), that specifies the Munsell reflectances to be loaded.

Not all the combinations of `H`, `V`, `C` and `h` are valid. With the options:

```
[reflec, vHVCh]=loadrefm([H V C h])
[reflec, vN]=loadrefm([N])
```

the function returns a matrix `(vHVCh` or `vN)` containing the subset of our original descriptors corresponding to real colors in the Munsell Atlas.

`reflec` Reflectance/Transmittance matrix.

Related Functions `loadrefl, defrefl, saverefl`

Required Functions

loadsys

Purpose Load the data of a color basis using a dialog box.

Description The data that defines a basis of the tristimulus space are:

1. Color matching functions
2. Trichromatic units
3. The matrix that relates this representation to the CIE XYZ representation

Syntax `[T_l,Yw,Mbx]=loadsys;`

`T_l` Color matching functions (spectral-like, N*4 variable)

`Yw` Trichromatic units (1*3 vector)

`Mbx` The change-to-CIEXYZ matrix (3*3 matrix)

Related Functions `loadsysm, defsys, defsysm, savesysm, startcol`

Required Functions

loadsysm

Purpose Load the data of a color basis from a (manually) given file.

Description The data that defines a basis of the tristimulus space are:

1. Color matching functions
2. Trichromatic units
3. The matrix that relates this representation to the CIE XYZ representation

Syntax

```
[T_l,Yw,Mbx]=loadsysm(['path']);
```

`['path']`	String with the path to a file with color basis data For example, `'c:/`*`MATLAB`*`/toolbox/`*`COLORLAB`*`/colordat,... /systems/ciergb.mat'`
`T_l`	Color matching functions (spectral-like, N*4 variable)
`Yw`	Trichromatic units (1*3 vector)
`Mbx`	The change-to-CIEXYZ matrix (3*3 matrix)

Related Functions `loadsysm, defsys, defsysm, savesysm, startcol`

Required Functions

lp2coor

Purpose Compute [t_1 t_2 Y] from [l_D P Y].

Description The color matching functions and the trichromatic units of the system are needed. As in `coor2lp`, the colorimetric or the excitation purities may be used.

Syntax `tY=lp2coor(lPY,opt,T_l,Yw);`

`lpY` Output colors expressed in dominant wavelength in nm, purity and luminance in cd/m^2 [l P Y]
Purples are characterized by the complementary dominant wavelength with negative sign.

`opt` Selects the kind of output purity:
P_excitat ⇒ `opt`=1
P_chromat ⇒ `opt`=2

`T_l` Color matching functions

`Yw` Trichromatic units

`tY` Input chromatic coordinates and luminances (color-like variable)

Related Functions `coor2lp, tri2coor, coor2tri`

Required Functions `tri2coor`

lum2td

Purpose Compute the retinal illuminance produced by a stimulus.

Description The retinal illuminance, *I*, in trolands, for an object of luminance *Y*, in cd/m^2, is defined as:

$$I = Y \cdot A(Y)$$

where *A(Y)* is the luminance-dependent pupil area, in mm^2.

Two different options for computing this area are available:

1. The pupil is assumed to be circular and Crawford’s formula is used to compute the pupil diameter, *d*:

$$d = 5 - 3tanh(0.4\,log(Y))$$

2. Guth’s formula for the pupil area:

$$A(Y) = 18Y^{-0.2}$$

Syntax

```
I=lum2td(Y,form)
I=lum2td(Y)
```

`I` Retinal illuminance (trolands)

`Y` Stimulus luminance (cd/m^2)

`form` Optional parameter determining the expression for computing I
If `form`=1, the pupilar diameter is computed with Crawford’s formula.
If `form`=2, Guth's formula is used. This is the default value.

Related Functions	`td2lum, xyzl2td, nguth`
Required Functions	

luv2perc

Purpose Compute perceptual descriptors in CIELUV.

Description `luv2perc` computes the lightness (L^*), chroma (C^*) and hue angle (h^*), in radians, of a set of colors characterized in the CIELUV space.

$$L^* = 116 f\left(\frac{Y}{Y_W}\right) - 16$$

$$C^* = \sqrt{u^{*2} + v^{*2}}$$

$$h^* = arctan\left(\frac{v^*}{u^*}\right), 0 \le h^* \le 2\pi$$

Syntax `LhC=lab2perc(Luv)`

`Luv` Lightness and chromaticity coordinates of the stimuli in CIELUV. For N stimuli, this is a Nx3 matrix.

`LhC` For N stimuli, Nx3 matrix. The first column contains the lightness L^*, the second the hue angle h^* and the third the chroma C^*.

Related Functions `perc2luv, xyz2luv, luv2xyz`

Required Functions

luv2xyz

Purpose

Compute *XYZ* from lightness and chromaticity coordinates in CIELUV space.

Description

`luv2xyz` computes the tristimulus values of a set of colors from their lightness, *L**, and chromatic coordinates *u** and *v** in CIELUV.

CIELUV is a simple appearance model providing perceptual descriptors (lightness, hue and chroma) for related colors (colors in a scene).

In this representation, information about the illumination conditions or, alternatively, about the scene, is included in a reference stimulus. Using CIELUV in the standard conditions implies that the reference stimulus is a perfect diffuser illuminated as the test.

Syntax

```
XYZ=luv2xyz(Luv,XYZR)
```

`Luv`	For N colors, Nx3 matrix, containing, in columns, the lightness *L**, and the chromaticity coordinates *u** and *v**.
`XYZR`	Tristimulus values of the reference stimulus. If the reference stimulus is the same for all the test stimuli, this is a 1x3 matrix. If the reference is different for each test stimulus `XYZR` is a Nx3 matrix.
`XYZ`	Tristimulus values of the test stimuli. For N colors, this is a Nx3 matrix.

Related Functions

`xyz2luv, luv2perc, perc2luv`

Required Functions

newbasis

Purpose Represent a set of tristimulus vectors in a new basis.

Description `newbasis` applies the matrix *Mpp'* (that relates the systems *P* and *P'*) to the input vectors (expressed in the system *P*).

For each vector input, *Tp*, we will have an output, *Tp'*, given by:

$$T_{P'} = M_{PP'} T_P$$

The matrix *Mpp'* can be computed with `chngmtx.`

Syntax

```
TT=newbasis(T,M);
```

`M` The change-of-basis matrix that relates the systems P and P' (3*3 matrix)

`T` Input set of tristimulus vectors (color-like variable, N*3 matrix)

`TT` Output set of tristimulus vectors (color-like variable, N*3 matrix)

Related Functions `chngmtx, newconst, defsys, defsysm`

Required Functions

newconst

Purpose Compute the constants defining a new color basis.

Description The constants that define a color basis are:

- The color matching functions, *$T_i(l)$*.

- Trichromatic units, *$Y_w(P_i)$*.

- Change-of-basis matrix, *M_{sx}*, that relates the system of primaries, *P_i*, to the standard system CIE XYZ.

- (for *COLORLAB* user convenience) the chromaticities of the monitor in the system: the *COLORLAB* matrix `tm`.

Syntax `[T_l2,Yw2,M2x,tm2]=newconst(M12,T_l1,Yw1,M1x,tm1)`

`M12`	3*3 Matrix that relates the system 1 to the system 2.
`T_l1`	Color matching functions in the system 1 (Spectral-like variable, N*4 matrix).
`Yw1`	Trichromatic units of the system 1 (1*3 vector).
`M1x`	3*3 Change-of-basis matrix that relates the system 1 to the CIEXYZ system.
`tm1`	Chromaticities of the monitor in the system 1 (calibration data, 7*m matrix).
`T_l2`	Color matching functions in the system 2 (Spectral-like variable, N*4 matrix).
`Yw2`	Trichromatic units of the system 2 (1*3 vector).

`M2x` 3*3 change-of-basis matrix that relates the system 2 to the CIEXYZ system.

`tm2` Chromaticities of the monitor in the system 2 (calibration data, 7*m matrix).

Related Functions `chngmtx, newbasis, defsys, defsysm`

Required Functions `coor2tri, newbasis`

pal2true

Purpose Convert the image+palette representation to true color representation.

Description Digital images are arrays of *M*N* pixels. Each pixel may have a different color. Therefore, digital color images may be described in two ways:

* True color image.

A true color image consists of 3 *M*N* matrices indicating the tristimulus value (or any other color component) of the image in each pixel (spatial location).

This is the straightforward description of the color samples obtained from a natural image. This is referred to as true color image because you may have a lot of different colors in the image: the number of different colors is only limited by the size of the image (*M*N*) and the available resolution of the guns, $(2^b)^3$, (b=8 in standard VGAs).

* Indexed image and color palette

This description consists of one *M*N* matrix (the indexed image) and one *C*3* matrix (the color palette). The rows of the color palette contain the tristimulus values (or any other color components) of the *C* colors in the scene. Each number of the indexed matrix is an integer that indicates the color of the palette (the row of the palette) that corresponds to that pixel (spatial location).

In principle in the image+palette representation you may have the same number of colors than in a true color image but the image+palette representation was originally intended to work with a very restricted set of colors to save data (see below). This is why, when more powerful systems were available and it began to be possible to work with 3 *M*N* matrices people called that representation 'true color' because it hadn’t the *C*-colors limitation.

The image+palette representation may be convenient due to two reasons:

* (Important) It is useful to work separately with the color content. For instance to desaturate the image or equalize the luminances you only have to work on the palette (spatial position doesn't matter).

* (Historical) It may be more efficient than true color representation. If the number of different colors in the image is small ($C < \frac{2}{3} M * N$), this representation requires less data than the true color representation.

  This had a lot of importance in the past when VGAs had limited graphic memory and couldn't support true color. In that case the limitation of *C* was the (trivial) way to save memory. *C*=256 (only 256 different colors at a time) was a typical limitation in old VGAs.

`pal2true` converts the image+palette representation to true color representation

Syntax

```
im=pal2true(ind_im,map,wtbar);
im=pal2true(ind_im,map);
```

INPUT variables

`ind_im` Indexed image (M*N matrix).

`map` Palette or colormap (color-like variable: C*3 matrix).
The input colormap (C*3 matrix) may use any 3D color representation (not only tristimulus values in any space, but also chromatic coordinates and luminance, *MATLAB* digital values or standard- 8 bit digital values).

`wtbar` If `wtbar==1`, a waitbar is displayed to monitor de progress of the operation.
If `wtbar==0`, no waitbar is shown.
By default, `wtbar=1;`

OUTPUT variables

`im` True color image (M*N*3 matrix)

Related Functions

`true2pal`

Required Functions

`waitbar (image processing toolbox)`

perc2atd

Purpose Compute *ATD* from perceptual descriptors with an *ATD* model.

Description `perc2atd` computes the output of the last stage of an ATD model, including the non-linearities, from their Brightness (*B*), Hue (*H*) and Saturation (*S*).

-Jameson and Hurvich model

$$|T| = HSB$$
$$|D| = 1 - H$$
$$|A| = B - |T| - |D|$$

- In the rest of linear first-stage models:

$$A = \frac{B}{\sqrt{1+S^2}}$$
$$D = \frac{BS}{\sqrt{1+S^2}} sin(H)$$
$$T = \frac{BS}{\sqrt{1+S^2}} \cos(H)$$

- The perceptual stage of ATD95 has not an analytical inverse. D_2 and T_2 are written as a function of A_2:

$$D_2 = A_2 S sin(H)$$
$$T_2 = A_2 S\, cos(H)$$

From these values, the first-opponent stage descriptors, A_1, T_1 and D_1 are computed analytically. The value of A_2 minimizing

$\left| B - \sqrt{A_1{}^2 + T_1{}^2 + D_1{}^2} \right|$ is determined by the simplex method as implemented in `fmins`.

Syntax

`ATD=perc2atd(BHS,model)`

`BHS`	For N colors, Nx3 matrix containing in the first column the Brightness (*B*), in the second the hue angle (0≤*H*≤2*π) and in the third the saturation (*S*) of each input stimulus.
`model`	Number identifying the model (1-13). See `xyz2atd` for details.
`ATDE`	*ATD* descriptors of the last stage of the model.

Related Functions

`atd2perc, xyz2atd, atd1atd2.`

Required Functions

`errl`

perc2lab

Purpose Compute lightness and chromaticity coordinates in CIELAB space from perceptual descriptors.

Description Lightness (*L**) and the chromaticity coordinates *a** and *b** of CIELAB space are obtained from the perceptual descriptors of that space, lightness, chroma (*C**) and hue angle (*h**) in radians:

$$a^* = C * cos(h^*)$$
$$b^* = C * sin(h^*)$$

Syntax

```
Lab=perc2lab(LhC)
```

`LhC` For N stimuli, Nx3 matrix. The first column contains the lightness *L**, the second the hue angle ($0 \le h^* \le 2*\pi$) and the third the Chroma, *C**.

`LAB` Lightness and chromaticity coordinates of the stimuli in CIELAB. For N stimuli, this is a Nx3 matrix.

Related Functions `lab2perc, xyz2lab, lab2xyz`

Required Functions

perc2luv

Purpose

Compute lightness and chromaticity coordinates in CIELUV space from perceptual descriptors.

Description

Lightness (L^*) and the chromaticity coordinates u^* and v^* of CIELUV space are obtained from the perceptual descriptors of that space, lightness, chroma (C^*) and hue angle (h^*):

$$u^* = C * \cos(h^*)$$
$$v^* = C * \sin(h^*)$$

Syntax

`Luv=perc2luv(LhC)`

`LhC` For N stimuli, Nx3 matrix. The first column contains the lightness L^*, the second the hue angle ($0 \le h^* \le 2*\pi$) and the third the Chroma, C^*.

`Luv` Lightness and chromaticity coordinates of the stimuli in CIELUV. For N stimuli, this is a Nx3 matrix.

Related Functions

`luv2perc, xyz2luv, luv2xyz`

Required Functions

replocus

Purpose Represent the spectral colors in the current chromatic diagram.

Description `replocus` may also plot two convenient 'all positive' triangles:

- The limits of colors with positive tristimulus values in the current basis.
- The limits of the color gamut reproducible in the current monitor (given by the chromaticities of the guns with maximum saturation).

`replocus` is the basis for the higher-level functions `colordgm` and `colord_c`. `replocus` is not intended to be used independently but to be called from these other functions.

`replocus` allows you to choose many parameters of the plot: the color and width of the lines and the font size in the axis and labels.

Syntax

```
replocus(T_l,tm,extra_limits,showtriang,linecolors,
linewidths,linestyles,fontsizes)
```

`T_l` Color Matching Functions in the current basis (N*4 spectral-like matrix).

`tm` Chromaticities of the monitor. 7*N matrix with the calibration data. (see `calibrat` or `loadmon`).

`extra_limits` [min_t1 max_t1 min_t2 max_t2]
If you want to plot colors outside the locus of real colors you may want to extend the limits of the plot (otherwise computed from the locus). In this case you need to include these extra limits by hand.

`showtriang` Parameter to enable/disable the plot of the 'all positive' triangles
If `showtriang`=0

`replocus` doesn’t plot any triangle at all

If `showtriang=|1`

It ONLY plots the triangle of the colors with all positive tristimulus values.

If `showtriang=2`

It ONLY plots the triangle of generable colors.

If `showtriang=3`

It plots BOTH triangles.

`linecolors` 5*3 matrix containing the colors of the following lines
- Axis of the bounding box
- Locus of spectral colors
- Axis of the plot
- Triangle of primaries of the system
- Triangle of 'primaries' of the monitor

`linewidths` 1*5 vector containing the widths of the previous lines (in the order given above).

`linestyles` 5*2 matrix containing the strings that define the style of the lines (in the order given above) according to the convention used in `plot.`

`fontsizes` 1*2 vector containing the sizes of the numbers in the axis and the labels of the axes respectively.

If you dot want to waste your time thinking on aesthetics, here you have an example you can use to begin with (cut, paste and explore!):

```
showtriang=3;
linecolors=[0 0 0;0 0 1;0 0 0.5;0 0.5 0;0.5 0 0];
linewidths=[0.5 0.5 0.5 0.5 0.5];
linestyles=['- ';'- ';'- ';'- ';': '];
fontsizes=[10 12];

replocus(T_l,tm,showtriang,linecolors,linewidths,
linestyles,fontsizes);
```

This means that it will plot the locus and both triangles, the bounding box will be black, the locus will be blue, the axis will be dark blue, the triangle of primaries will be dark green and the monitor triangle will be dark red. All the lines will be solid but the monitor line and all of them will be 0.5 width. The numbers of the axis will be size 10 and the labels size 12.

Related Functions `colordgm, colorspc`

Required Functions `mini, maxi, niv2coor, ganadora`

rlab2xyz

Purpose Compute *XYZ* from perceptual descriptors in RLAB space.

Description `rlab2xyz` computes the tristimulus values from the lightness, *L*, chroma, C and hue, *H*, in degrees, in the RLAB space.

The transform needs information about the reference white and about the observation conditions. The absolute luminance of the reference value and its tristimulus *XYZ* values with the luminance normalized to 100 are required. The observation conditions are described by two parameters, *D* and *M0*. *D* distinguishes between hard-copy and soft-copy scenes. *M0* describes the lightness of the surround.

Syntax `XYZ=rlab2xyz(LCH,XYZW,YW,D,MO)`

`LCH` [*L C H*(º)] of the colors. For N colors, this is a Nx3 matrix.

`XYZW` Relative tristimulus values of the reference white.

`YW` Luminance (cd/m^2) of the reference white (1x1).

`D` .....1 hard-copy,
.....0 soft-copy,
.....For other situations, give intermediate values between these two.

`MO` observation conditions
...1/2.3 for average surround
...1/2.9 for dim surround,
...1/3.5 for dark surround.

`XYZ` Relative tristimulus values of the samples.
For N colors, this is a Nx3 matrix.

Related Functions `xyz2rlab.`

Required Functions

savecol

Purpose Save a set of colors in *COLORLAB* format.

Description The meaning of a given color characterization depends on its nature -for instance, *T* versus (*t*,*Y*)- and on the color space where the color was defined -for instance CIERGB versus NTSCRGB-.

The *COLORLAB* format for color storage includes the necessary additional information to ensure an appropriate retrieval from any other tristimulus space and required representation.

A *COLORLAB* color file consists of three variables:

- `T`, a N*3 color-like matrix that specifies the N colors: .
- `Msx`, a 3*3 change-of-basis matrix that specifies the color space where `T` is defined. It relates the system with the CIEXYZ system.
- `characteriz`, a parameter that specifies the representation used in `T`.

Syntax `savecol(T,Msx,characteriz,'file')`

`T`	Specification of the N colors: the variable, `T`, N*3 color-like matrix.
`Msx`	Specification of the color system: the variable, `Msx`, 3*3 change-of-basis matrix that relates the system with the CIEXYZ. (This matrix is given in the definition of the system. See `defsys)`
`characteriz`	Specification of the representation: the variable, `characteriz`, describes the nature of the representation used in `T`. `characteriz` = 1.....Tristimulus vectors `characteriz` = 2.....Chromatic coordinates and luminance `characteriz` = 3.....Dominant wavelength, excitation

purity and luminance
`characteriz` = 4.....Dominant wavelength, colorimetric purity and luminance

`file` String with the path to the desired `*.mat` file

Related Functions `defcolor, loadcol, defcroma`

Required Functions

saveillu

Purpose Save illuminants.

Description Illuminants are spectral-like variables defined with `defillu` or loaded with `loadillu`.

`saveillu` normalizes the spectral radiance of the illuminant to give a relative distribution in the range [0 1].

`loadillu` undoes this normalization to obtain the required luminance or radiance.

Syntax

```
saveillu(illum,['path']);
```

`illum`	Illuminant
`['path']`	String containing the path to the file. For example:

```
'c:/MATLAB/toolbox/COLORLAB/colordat/illumin/,...
illum_CIE_A.mat'
```

Related Functions `loadillu, loadillum, defillu`

Required Functions

savemon

Purpose Save monitor calibration data to a (manually) given file.

Description The *'COLORLAB format for CRT calibration data storage'* is just a name convention for the variables involved in the CRT calibration. Such a convention is convenient for future automatic transform of the data to the required color system when using `loadmon`

COLORLAB assumes additivity and independence between the guns of the CRT monitor. *COLORLAB* also assumes an exponential relation (gamma relation) between the luminance of the gun, *i*, and the corresponding digital value, n_i (i=1,2,3). *COLORLAB* makes no assumption on the chromaticities of the guns, which may depend on the digital value in any complex way.

With these assumptions, the calibration data include:

- Chromaticities of the gun, *i*, as a function of the digital step, n_i.
- Parameters (a_i and g_i) of the gamma relation between the luminance of the gun, *i*, and the digital value, n_i:

$$Y_i = a_i n_i^{g_i}$$

Syntax `savemon(Msx,tm,a,g,'path')`

`Msx` 3*3 change-of-basis matrix relating the system S (where the data are described) to the CIEXYZ system.

`tm` 7*N matrix that contains the chromaticities of the R, G and B guns for N calibration points of the digital value (see `calibrat` for details).
The chromaticities are given in the system at hand even though they are stored in the CIEXYZ system (this is why `Msx` is needed).

`a` 1*3 vector including the constants a_i of the gamma relation for each gun.

`g` 1*3 vector including the constants g_i of the gamma relation for each gun.

`'path'` String with the path to the file containing the data.
For example:
`'c:\`*`MATLAB`*`\toolbox\`*`COLORLAB`*`\colordat\...`
`monitor\monito.mat'`

Related Functions `loadmon, loadmonm, calibrat, tri2val, val2tri`

Required Functions `coor2tri, newbasis`

savereﬂ

Purpose Save reflectances/transmittances.

Description Reflectances (and transmittances) are spectral-like variables defined with `defrefl` or loaded with `loadrefl`.

Syntax `saverefl(refl,['path']);`

`refl`	Reflectance (or Transmittance).
`['path']`	String containing the path to the file. For example: `'c:/`*MATLAB*`/toolbox/`*COLORLAB*`/colordat/,... reflec/yellow.mat'`

Related Functions `loadrefl, loadrefm, defrefl`

Required Functions

savesysm

Purpose Save the data of a color basis to a (manually) given file.

Description The data that defines a color basis are:

1. Color matching functions
2. Trichromatic units
3. The matrix that relates this representation to the CIE XYZ representation

Syntax

```
savesysm(T_l,Yw,Mbx,'path');
```

`T_l`	Color matching functions (spectral-like, N*4 variable)
`Yw`	Trichromatic units (1*3 vector)
`Mbx`	The change-to-CIEXYZ matrix (3*3 matrix)
`'path'`	String with the path to a file with color basis data. For example, `'c:/`*`MATLAB`*`/toolbox/`*`COLORLAB`*`/colordat/systems/ciergb.mat'`

Related Functions `defsys, defsysm, loadsys, loadsysm, startcol, chngmtx, newconst, newbasis`

Required Functions

spec2tri

Purpose Compute tristimulus vectors from spectral data.

Description `spec2tri` computes the N tristimulus vectors (color-like variable) from N chromatic stimuli (spectral-like variable).

The N chromatic stimuli can be described in three different ways (options):

1. With N spectral radiances [in Wstr^{-1}m^{-2}].

2. With N reflectances/transmittances and some spectral illuminant in absolute units [in Wstr^{-1}m^{-2}].

3. With N reflectances/transmittances, some relative illuminant and the desired luminance for this illuminant.

Option 1 requires the color matching functions, `T_l`, and the palette of radiances `R`.
Option 2 requires the color matching functions, `T_l`, a palette of reflectances `R`, and some illuminant `S`.
Option 3 requires the color matching functions, `T_l`, a palette of reflectances `R`, the illuminant `S`, the desired luminance `Y`, and the trichromatic units, `Yw`.

`spec2tri` returns a color-like variable, `T`, and a spectral-like variable, `RR`, with the tristimulus values and the radiances of the input stimuli. The rows of `T` correspond to the columns of `RR` and `R`.

Syntax

```
[T,RR]=spec2tri(option,T_l,D_lambda,R,S,Y,Yw);
```

`option` 1,2,3 determines the interpretation of `R`, `S`, `Y`, `Yw`. (see above).

`T_l` Color Matching Functions.

`D_lambda`	Wavelength step used in the spectrum interpolation for tristimulus vector computation. Note that (FORTUNATELY!) the sampling of the input color matching functions, reflectances and illuminant do not have to be the same. (You don’t have to care about that: `spec2tri` resamples for you).
`R`	Radiances (option 1) or reflectances/transmittances (option 2,3)
`S`	Illuminant (options 2,3). You may introduce 0 in option 1.
`Y`	Required luminance (option 3). You may introduce 0 in options 1,2.
`Yw`	Trichromatic units (option 3). You may introduce 0 in options 1,2.

Related Functions

`tri2spec`

Required Functions

startcol

Purpose Load the data required for *COLORLAB* to work.

Description `startcol` opens two dialog boxes to load the data required for *COLORLAB* to work.

In order to start working with *COLORLAB* you need to define a reference color system an you need the CRT calibration data to display color images. (See `loadsys`, `loadmon` and `calibrat`)

The names given to these variables are:

`T_l` Color matching functions (spectral-like, N*4 variable)

`Yw` Trichromatic units (1*3 vector)

`Msx` 3*3 change-of-basis matrix that relates the system at hand to the CIEXYZ system.

`tm` 7*N matrix that contains the chromaticities of the R, G and B guns for N calibration points of the digital value (see `calibrat` for details).
The chromaticities are given in the system at hand even though they are stored in the CIEXYZ system (this is why `Msx` is needed).

`a` 1*3 vector including the constants a_i of the gamma relation for each gun.

`g` 1*3 vector including the constants g_i of the gamma relation for each gun.

Syntax `startcol;`

Related Functions

`defsysm, defsys, loadsys, loadsysm, savesysm, chngmtx, newconst, newbasis`

Required Functions

`loadsys, loadmon, coor2tri, newbasis`

svf2xyz

Purpose Transform SVF descriptors into XYZ.

Description `svf2xyz` computes the tristimulus values, in CIE-1931 space of the stimuli whose descriptors in the SVF space are *VF_1F_2* when the background is *XYZW*.

Syntax

```
XYZ=svf2xyz(VF1F2,XYZW)
```

`VF1F2`	[*V* (value) *F_1 F_2* (chromaticity coordinates)]
`XYZW`	Tristimulus values of the refence white. This variable may contain a single white or a white for each stimulus.
`XYZ`	Tristimulus values of the samples. For N colors, this is a Nx3 matrix.

Related Functions `xyz2svf`

Required Functions `acop, mixi, lator`

td2lum

Purpose Compute luminance from retinal illuminance.

Description This is the inverse function of `lum2td`. The retinal illuminance, *I*, in trolands, for an object of luminance *Y*, in cd/m^2, is defined as:

$$I = Y \cdot A(Y)$$

where *A(Y)* is the luminance-dependent pupil area, in mm^2.

Two different options for computing this area are available:

3. The pupil is assumed to be circular and Crawford's formula is used to compute the pupil diameter, *d*:

$$d = 5 - 3tanh(0.4\,log(Y))$$

With this option, the luminance is computed numerically. The simplex method, as implemented in fmins, is used to determine the value Y minimizing the expression:

$$\left| I - \frac{\pi}{4} Y \cdot (5 - 3tanh(0.4\,log\,Y))^2 \right|$$

4. Guth's formula for the pupil area:

$$A(Y) = 18Y^{-0.2}$$

With this formula, the luminance can be analytically computed:

$$Y = \sqrt[0.8]{\left(\frac{I}{18}\right)}$$

Syntax

```
Y=td2lum(I,form)
Y=td2lum(I)
```

`I` Retinal illuminance (trolands)

`form` Optional parameter determining the expression for computing *I*.
If `form`=1, the pupil diameter is computed with Crawford's formula
If `form`=2, Guth's formula is used. This is the default value.

`Y` Stimulus luminance (cd/m^2)

Related Functions

`lum2td, xyztd2l, nguth`

Required Functions

tri2coor

Purpose Compute chromaticity coordinates and luminance.

Description `tri2coor` obtains chromatic coordinates and luminance from tristimulus values. To compute the luminance, the trichromatic units of the system are required.

Syntax `t=tri2coor(T,Yw);`

`T`	Input tristimulus vectors (color-like variable, colors in rows).
`Yw`	Trichromatic units [$Y_w(P_1)$ $Y_w(P_2)$ $Y_w(P_3)$]
`t`	Output chromatic coordinates and luminance (color-like variable, [t_1 t_2 Y]).

Related Functions `coor2tri, coor2lp, lp2coor`

Required Functions

tri2spec

Purpose Assign reflectance(s) to tristimulus vector(s).

Description Sometimes you may want to look for a stimulus (a reflectance) that gives rise to a particular color under a given illuminant.

This is not a well defined transform because all metameric stimuli give rise to the same color. Although this is a non invertible problem, a particular or approximate solution may be useful for stimulus design.

For each tristimulus vector of the input, `tri2spec` associates a reflectance from the set of 'target reflectances'. `tri2spec` selects the reflectance that gives rise to the closest chromaticity and scales the reflectance to fit the luminance of the input color. (Incorrect) Euclidean distance is used to compute the chromaticity differences.

A relative illuminant has to be introduced with an additional luminance (in cd/m^2).

CAUTION: Note that for low-energy illuminants it may be necessary to apply high factors on the reflectances to obtain the required luminance. This may lead to reflectances > 1 for some wavelengths (this is why not all chromaticities are generable from physical reflectances). It may be necessary to apply physical constraints (reflectances < 1) to the output of this routine.

Syntax

```
RT=tri2spec(T,R,S,Y,T_l,Yw);
```

`T` Input tristimulus vectors (color-like variable)

`R` Target reflectances to choose from (spectral-like variable). Hint: The file munsell.mat contains a set of reflectances covering a big region of the chromaticity space, so it may be useful in this routine.

`S` Illuminant (spectral-like variable).

`Y` Luminance of the illuminant.

`T_l` Color matching functions.

`Yw` Trichromatic units.

`RT` Output reflectances (spectral-like variable)

Related Functions

`spec2tri`

Required Functions

`spec2tri.m, tri2coor.m, ganadora.m`

tri2val

Purpose Compute digital values, *n*, from tristimulus vectors, *T*.

Description Computers do not use an appropriate (colorimetrically meaningful) color description, but 3D arrays of parameters $n=[n_1\ n_2\ n_3]$ that control the voltages of each gun of the CRT. In most standard image formats these n_i values (digital values) are given using 8 bit integers (uint8 variables ranging from 0 to 255). This is a device-dependent characterization because the color obtained from a given array, *n*, depends on the particular color reproduction device.

In *MATLAB* colormaps the digital values that describe the colors are real numbers with values in [0,1] (ranging from minimum luminance to maximum luminance of the gun).

`tri2val` computes this (meaningless) device-dependent color characterization, *n*, from the (colorimetrically meaningful) characterization using tristimulus vectors, *T*.

In this way you will be able to generate *MATLAB* colormaps (and to represent color images or save them in standard formats) from the corresponding tristimulus characterization.

To do so, the CRT calibration (see `calibrat`) has to be taken into account.

The limitations of the color reproduction device (the CRT) imply certain restrictions on the gamut of colors that can be represented using the representation with digital values. Remember that n_i is restricted to be in the range [0,1], so luminance is limited and certain (highly saturated) chromaticities are not available.

Remember also that the VGA has a limited (8 bit) resolution, so very close n_i give rise to the same output luminance. This implies that only a discrete set (grid) of colors inside a limited gamut is available. As a result, not every real color, *T*, can be represented by *n*. In general a certain (quantization or gamut limitation) error is make when using the digital value representation. Therefore, the nearest available color, T_n, corresponding to the array, n, will be different from the desired color, T.

`tri2val` computes the array, n, that minimizes the error $|T - T_n|$ in the tristimulus space you are working. `tri2val` looks for the best array, n, taking into account the resolution (in bits/channel) of the VGA at hand.

Syntax

```
[n,saturat,Tn]=tri2val(T,Yw,tm,a,g,res,wtbar);
[n,saturat,Tn]=tri2val(T,Yw,tm,a,g,res);
```

INPUT variables

`T` Input color-like variable (N*3 variable) with N colors (tristimulus vectors).

`Yw` Trichromatic units

`tm` 7*M matrix that contains the chromaticities of the R, G and B guns for M calibration points of the digital value (see `calibrat`).

`a` 1*3 vector including the constants a_i of the gamma relation for each gun (see `calibrat`)

`g` 1*3 vector including the constants g_i of the gamma relation for each gun (see `calibrat`)

`res` Resolution (in bits/channel) of the VGA. (8 bits in the standard VGAs)

`wtbar` If `wtbar==1`, a waitbar is displayed to monitor de progress of the operation.
If `wtbar==0`, no waitbar is shown.
By default, `wtbar=1;`

OUTPUT variables

`n` Input color-like variable (N*3) with N colors (digital values).

`saturat`	Colors with high luminance may lie outside the gamut of available colors. If this is the case, `tri2val` reduces their luminance until they do not saturate any digital value. The variable `saturat` is a N*1 variable that indicates the colors that have been reduced in luminance because of this saturation. Zero in this variable indicates that the luminance of the corresponding color hasn't been modified. Conversely, a value of one indicates that the luminance of the corresponding color has been modified.
`Tn`	Color-like N*3 variable with the tristimulus values corresponding to the colors in the colormap `n`. The colors in `Tn` are the nearest colors to `T` available using the discrete device-dependent representation.

Related Functions `val2tri, loadmon, loadmonm, savemon, calibrat`

Required Functions `t2n.m, miniecol.m, n2t.m, tri2lum.m, errcol.m, ganadora.m, lum2niv.m, niv2coor.m`

true2pal

Purpose Generate the indexed image and the palette from the true color image.

Description Digital images are arrays of *M*N* pixels. Each pixel may have a different color. Therefore, digital color images may be described in two ways:

* True color image.

A true color image consists of 3 *M*N* matrices indicating the tristimulus value (or any other color component) of the image in each pixel (spatial location).

This is the straightforward description of the color samples obtained from a natural image. This is referred to as true color image because you may have a lot of different colors in the image: the number of different colors is only limited by the size of the image (*M*N*) and the available resolution of the guns, $(2^b)^3$, (b=8 in standard VGAs).

* Indexed image and color palette

This description consists of one *M*N* matrix (the indexed image) and one *C*3* matrix (the color palette). The rows of the color palette contain the tristimulus values (or any other color components) of the *C* colors in the scene. Each number of the indexed matrix is an integer that indicates the color of the palette (the row of the palette) that corresponds to that pixel (spatial location).

In principle in the image+palette representation you may have the same number of colors than in a true color image but the image+palette representation was originally intended to work with a very restricted set of colors to save data (see below). This is why, when more powerful systems were available and it began to be possible to work with 3 *M*N* matrices people called that representation 'true color' because it hadn’t the *C*-colors limitation.

The image+palette representation may be convenient due to two reasons:

* (Important) It is useful to work separately with the color content. For instance to desaturate the image or equalize the luminances you only have to work on the palette (spatial position doesn't matter).

* (Historical) It may be more efficient than true color representation. If the number of different colors in the image is small ($C < \frac{2}{3} M * N$), this representation requires less data than the true color representation.

  This had a lot of importance in the past when VGAs had limited graphic memory and couldn't support true color. In that case the limitation of *C* was the (trivial) way to save memory. *C*=256 (only 256 different colors at a time) was a typical limitation in old VGAs.

`true2pal` computes the image+palette representation from the true color image.

`true2pal` normalizes the colors in the scene to put them in a unit volume cube and then it applies `rgb2ind` to select the palette.

`rgb2ind` converts a true color image to indexed image with limited size palette using one of three different methods: uniform quantization, minimum variance quantization, colormap approximation. Also it may use direct translation: i.e. it may generate an indexed image with a non-restricted palette (containing all the colors of the true color image).

Syntax

```
[X,map]=true2pal(im3,p);
```

OUTPUT variables

`X`	Indexed image (M*N matrix).
`map`	Palette (color-like variable: C*3 matrix).

INPUT variables

`im3` True color image (M*N*3 matrix)

The input true color image (M*N*3 matrix) may use any 3D color representation (not only tristimulus values in any space, but also chromatic coordinates and luminance, *MATLAB* digital values or (standard) 8 bit digital values).

`p` Parameter to control the palette selection algorithm in `rgb2ind`.

We have four different options:

(1) Uniform quantization of each color axis (scalar quantization)
`[X,MAP]=true2pal(im3,TOL)` converts the true color image, `im3`, to an indexed image X using uniform quantization. `MAP` contains at most `c=(floor(1/TOL)+1)^3` colors. `TOL` must be between 0.0 and 1.0.

(2) Minimum Variance Quantization (vector quantization)
`[X,MAP] = true2pal(im3,C)` converts the true color image, `im3`, to an indexed image `X` using minimum variance quantization. `MAP` contains at most C colors.

(3) Colormap approximation
`X=true2pal(im3,MAP)` converts the true color image, `im3`, to an indexed image `X` with colormap (palette) `MAP` by matching colors in RGB with the nearest color in the colormap `MAP.`

(4) Direct translation
`[X,MAP] = true2pal(im3)` converts the true color image, `im3`, to an indexed image `X` with colormap `MAP` using direct translation. The resulting colormap may be very long, as it has one entry for each pixel in RGB.

Related Functions

See `imread` and `imwrite` to learn on how to read and write images in standard formats

See `image` (*MATLAB*), `imshow` (*MATLAB*) and `dispim` (*COLORLAB*) to learn how to display an image.

WARNING: Note that *MATLAB* 5.3 versions of image and imshow do not support colormaps (palettes) with more than 256 colors. In order to show images with bigger palettes use `dispim` (*COLORLAB*) or the true color format (M*N*3 matrix).

Required Functions

`rgb2ind (image processing toolbox), cmunique (image processing toolbox)`

val2tri

Purpose Compute tristimulus vectors from *MATLAB* digital values.

Description `val2tri` computes *T* taking into account the CRT calibration data (see `calibrat` and `tri2val`).

Syntax

```
T=val2tri(n,Yw,tm,a,g);
```

INPUT variables

`n` Input color-like variable (N*3 variable) with N colors (digital values).

`Yw` Trichromatic units

`tm` 7*M matrix that contains the chromaticities of the R, G and B guns for M calibration points of the digital value (see `calibrat`).

`a` 1*3 vector including the constants a_i of the gamma relation for each gun (see `calibrat`)

`g` 1*3 vector including the constants g_i of the gamma relation for each gun (see `calibrat`)

OUTPUT variables

`T` Output color-like variable (N*3 variable) with N colors (tristimulus vectors).

Related Functions `tri2val, loadmon, loadmonm, savemon, calibrat`

Required Functions `niv2coor, ganadora`

xyz2atd

Purpose Transform XYZ tristimulus values into ATD values.

Description `xyz2atd` transforms the XYZ tristimulus values of a set of colors into the first and, when possible, second stage responses of the achromatic and chromatic mechanisms of an ATD model, in the specified observation conditions.

COLORLAB includes a set of classic models: cone spaces (models 1-3 and 12), first-stage linear models (4-7), opponent modulation model (13) and non-linear two-opponent stages models (8-11).

In all the cone-spaces considered, LMS is computed as a linear transform of XYZ. With these models, use CON2XYZ.

The first stage models need at most the information about the background or the adaptation matrix.

Vos and Walraven's fundamentals
  Y=L+M+S
  Cone space. Use only with XYZ2CON and CON2XYZ
  XYZ must be computed with the CIE-1931 observer modified by Judd (1951).

Wyszecki and Stiles's fundamentals.
  L(475.5)+M(475.5)=16S(475.5)
  Cone space. Use only with XYZ2CON and CON2XYZ
  XYZ must be computed with the CIE-1931 observer modified by Judd (1951).

MacLeod-Boynton's fundamentals.
  L(400)+M(400)=S(400)
  Cone space. Use only with XYZ2CON and CON2XYZ
  XYZ must be computed the CIE-1931 observer modified by Vos (1978).

Jameson and Hurvich (1957).

One opponent linear stage with adaptation matrix.
XYZ must be computed with the CIE-1931 colorimetric observer.

Ingling and Tsou (1977).

One opponent linear stage, with different adaptation matrixes for threshold and suprathreshold conditions.

Boynton (1986)

L(498)+M(498)=S(498)
One opponent linear stage, without adaptation.

Guth (1980)

L, M and S normalized to one.
One opponent linear stage with adaptation matrix.

Guth (1990).

XYZ is normalized so that Y=0.0015 equals 100 Td
Smith and Pokorny fundamentals, normalized to one
L:M:S 0.66:1:0.55
Cone gain-control:

$$LMS_{gc} = LMS\left(0.01 + 0.99\frac{0.05}{LMS(A) + 0.05}\right)$$

where LMS is the linear cone-response to the test stimulus, LMS(A) the cone-responses to the adapting stimulus. Both the test and the background XYZB contribute to the adapting stimulus in the proportion indicated by weight:

weight=[Test contribution, Background contribution]

These values sum to one. In the different updates of the model, they may take any value.
Two opponent non-linear stages. The adaptation mechanism is already included in the model. The same happens with models 9-11.

Guth (1993): Guth90 modified.

XYZ are transformed to trolands.
Smith and Pokorny fundamentals, normalized to one
L:M:S   0.66:1:0.45
Gain-control

$$LMS' = LMS \frac{sigma}{sigma + LMSA}$$

`sigma`=400 but to compute MacAdam's ellipses
`sigma`=220.
By default, `sigma`=400.

Guth (1994):Guth93 modified.
XYZ are transformed to trolands.
Smith and Pokorny fundamentals, normalized to one
L:M:S   0.66:1:3
Gain-control

$$LMS' = LMS \frac{sigma}{sigma + LMSA}$$

`sigma`=400 but to compute MacAdam's ellipses
`sigma`=220.
By default, `sigma`=400.

Guth (1995):Guth94 modified.
XYZ are transformed to trolands.
Smith and Pokorny fundamentals, normalized to one
L:M:S   0.66:1:0.43
Gain-control

$$LMS' = LMS \frac{sigma}{sigma + LMSA}$$

`sigma`=300 but to compute MacAdam's ellipses
`sigma`=220.
By default, `sigma`=300.

Stockman and Sharpe's fundamentals (2000)

Derrington, Krauskopf y Lennie (DKL)
Opponent modulation space. LMS must be increments on a background with tristimulus values XYZB.

The only condition on LMS is that they must be the Smith-Pokorny fundamentals, with Y=L+M; the scaling conditions on the S-cones is inmaterial. In this function, Boynton's scaling is used.

Syntax

`[ATD1,ATD2]=xyz2atd(XYZ,model,XYZB,adap,weight,mode,sig)`

`XYZ` Tristimulus values of the test stimuli. The color observer may be the CIE-1931 standard observer, the observer modified by Judd or by Judd and Vos, depending on the model.

`model` Integer (4-11, 13) describing the model.
`models`1:3 and 12 do not have opponent stages.
`model`=4 Jameson and Hurvich (1957)
`model`=5 Ingling and Tsou (1977)
`model`=6 Boynton (1986)
`model`=7 Guth (1980)
`model`=8 Guth (1990)
`model`=9 Guth (1993)
`model`=10 Guth (1994)
`model`=11 Guth (1995)
`model`=13 DKL

`XYZB` Background stimulus (3x1)

`adap` In models 4 and 7, `adap` is a 3x3 matrix scaling the ATD responses. In model 5, is a scalar, and in model 13 a 1x3 vector with the white tristimulus values.

`model`=4	At threshold, `adap=eye(3)`. This is the default value.
`model`=5	At threshold `adap`=1. Otherwise, `adap`=2. By default, `adap`=2;
`model`=7	At threshold, `adap=eye(3)`. This is the default value.
`model`=13	`adap=XYZB`;

`adap` 3x3 or 3x1 matrix, depending on the model, including adaptation effects or adaptation conditions.

`weight` Only with models 8-11. Contributions of test stimuli and the background to the adaptation stimulus.

`mode` If ATD are increments on XYZB, mode=1. Otherwise, mode=2. Use only with models 8-11.

`sig` Scalar parameter controlling the non-linearity at the cone stage. Use only with models 8-11.

`ATD1, ATD2` Responses of the achromatic (A), red-green (T) and blue-yellow (D) mechanism to the test stimuli at the first and second opponent stages,respectively.
For N stimuli, these are Nx3 matrixes.

First stage models need at most the information about the background or the adaptation matrix, so the function may be used as

`ATD=xyz2atd(XYZ,model,[],adap)` (models 4, 5 and 7)

`ATD=xyz2atd(XYZ,model)` (model 6)

`ATD=xyz2atd(XYZ,model,XYZB)` (model 13).

If a model does not use a given parameter, write []. To use the default values of a parameter, use [] if any parameter with a user-defined value comes behind, or just leave them out if they are the last inputs of the function.

For instance,

```
[ATD1,ATD2]=xyz2atd(XYZ,11,XYZB,[],weights,mode)
```

uses the default value of `sig` in Guth95 (300), and indicates that adap is not used in this model. `[ATD1,ATD2]=xyz2atd(XYZ,11)` would set all the parameters to their default values.

Related Functions	`atd2xyz, atd2perc, perc2atd, atdf2con, con2xyz, incomcop, ingancon, xyzn2xyz`
Required Functions	`xyz2con, con2atd, gancon, compcop, xyz2xyzn`

xyz2con

Purpose Compute cone responses from XYZ.

Description `xyz2con` transforms the tristimulus values XYZT into the cone-space defined by a set of fundamentals or a color-vision model.

This transform needs information about the model or set of fundamentals used (`model`), the background (`XYZB`), and the non-linear mechanisms of the model (`weights`, `mode` and `sigma`).

FUNDAMENTALS AND MODELS

`model`=1 Vos and Walraven's fundamentals (Y=L+M+S)
XYZ are computed with the CIE-1931 observer modifiedby Judd (1951).

`model`=2 Wyszecki and Stiles's fundamentals
(L(475.5)+M(475.5)=16S(475.5))
XYZ are computed with the CIE-1931 observer modified by Judd (1951).

`model`=3 MacLeod-Boynton's fundamentals (L(400)+M(400)=S(400))
XYZ are computed with the CIE-1931 observer modified by Vos (1978).

`model`=4 Jameson and Hurvich (1957)
XYZ are described with the CIE-1931 observer.

`model`=5 Ingling and Tsou (1977)

`model`=6 Boynton (1986)
L(498)+M(498)=S(498)

`model`=7 Guth (1980)
L, M and S normalized to one

`model`=8 Guth (1990)

XYZ is normalized so that Y=0.0015 equals 100 Td
Smith and Pokorny fundamentals, normalized to one
L:M:S 0.66:1:0.55
Cone gain-control:
XYZB represents the background or the adapting stimulus. Both the test and the background may contribute to the adapting stimulus. This is controlled by `weight`:

`weight`=[Test contribution, Background contribution]

These values sum to one. In the different updates of the model, they may take any value.
The meaning of XYZB, mode and weight in models 8-11 is the same.
Defaults: XYZB=[0 0 0], mode=2, weight=[1 0]. These values apply also to models 9-11.

`model`=9 Guth (1993)

XYZ are transformed to trolands.
Smith and Pokorny fundamentals, normalized to one
L:M:S 0.66:1:0.45
Gain-control

$$LMS' = LMS \frac{sigma}{sigma + LMSA}$$

`sigma`=400 but to compute MacAdam's ellipses
`sigma`=220.
By default, `sigma`=400.

`model`=10 Guth (1994):XYZ are transformed to trolands.

Smith and Pokorny fundamentals, normalized to one
L:M:S 0.66:1:3
Gain-control

$$LMS' = LMS \frac{sigma}{sigma + LMSA}$$

`sigma`=400 but to compute MacAdam's ellipses
`sigma`=220.

By default, `sigma`=400.

`model`=11 Guth (1995):XYZ are transformed to trolands.
Smith and Pokorny fundamentals, normalized to one
L:M:S 0.66:1:0.43
Gain-control

$$LMS' = LMS \frac{sigma}{sigma + LMSA}$$

`sigma`=300 but to compute MacAdam's ellipses
`sigma`=220.
By default, `sigma`=300.

`model`=12 Stockman and Sharpe's fundamentals (2000)

Except for `model`=1 and `model`=12, the Smith and Pokorny fundamentals, with different normalization conditions, are used.

Syntax

`LMS=xyz2con(XYZT,model,XYZB,weight,mode,sigma)`

`XYZ`	Input tristimulus values. The tristimulus space is either CIE-1931 or this space as modified by Judd or by Judd and Vos.
`model`	Integer (1-13) identifying the model. A brief description of each model and the necessary parameters can be seen below.
`XYZB`	Background stimulus (3x1)
`weight`	Contributions of test stimuli and the background to the adaptation stimulus. Use only with models 8-11.
`mode`	If `LMS` are increments on `XYZB`, mode=1. Otherwise, mode=2. Use only with models 8-11.
`sigma`	Scalar parameter controlling the non-linearity at the cone

stage.
Use only with models 8-11.

`LMS` Cone responses. Color format.

With `xyz2con(XYZ,model)`, the rest of the parameters are set to zero in `model`≤7 and `model`=12 and to the default parameters in the rest.

Related Functions

`xyz2xyzn, cambial, nguth, lum2td, inguth, td2lum, xyzl2td, lum2td, gancon`

Required Functions

`con2xyz, xyz2atd, atd2xyz, atd2perc, perc2atd, con2atd, atdf2con, atd1atd2`

xyz2lab

Purpose

Compute lightness and chromaticity coordinates in CIELAB space from XYZ.

Description

`xyz2lab` computes the lightness, L^*, and chromatic coordinates a^* and b^* in CIELAB of set of colors from their XYZ tristimulus values.

CIELAB is a simple appearance model providing perceptual descriptors (lightness, hue and chroma) for related colors (colors in a scene).

In this representation, information about the illumination conditions or, alternatively, about the scene, is included in a reference stimulus. Using CIELAB in the standard conditions implies that the reference stimulus is a perfect diffuser illuminated as the test.

Syntax

`Lab=xyz2lab(XYZ,XYZR)`

`XYZ`	Tristimulus values of the test stimuli. For N colors, this is a Nx3 matrix.
`XYZR`	Tristimulus values of the reference stimulus. If the reference stimulus is the same for all the test stimuli, this is a 1x3 matrix. If the reference is different for each test stimulus `XYZR` is a Nx3 matrix.
`Lab`	For N colors, Nx3 matrix, containing, in columns, the lightness L^*, and the chromaticity coordinates a^* and b^*.

Related Functions

`lab2xyz, lab2perc, perc2lab`

Required Functions

xyz2llab

Purpose

Compute descriptors in LLAB space.

Description

`xyz2llab` computes the lightness (*L*), the opponent coordinates *a* and *b*, the chroma (*C*) and the hue angle (*H*), in degrees, in the LLAB space.

The tristimulus values of the input stimulus are normalized so that *Y*=100 for the equal-energy stimulus, illuminated as the test, which is the reference stimulus. The absolute luminance values of the reference white and the background (*`YW`* and *`YB`*) are also required. The remaining observation conditions are specified by parameter *mo*.

Syntax

`LLABch=xyz2llab(XYZS,XYZW,YW,YB,MO)`

`XYZ`	Relative tristimulus values of the samples. For N colors, this is a Nx3 matrix.
`XYZW`	Relative tristimulus values of the reference white.
`YW`	Luminance (cd/m^2) of the reference white (1x1).
`YB`	Luminance factor of the background (1x1).
`MO`	Parameter describing the observation conditions: `MO`=1.....reflection samples, average surround, field>4º `MO`=2.....reflection samples, average surround, field<4º `MO`=3.....television and VDU, dim surround `MO`=4.....cut-sheet transparency in dim surround `MO`=5.....35 mm projection transparency in dark surround
`LLABCH`	[*L a b C H*(º)]. For N colors, this is a Nx3 matrix.

Related Functions	`llab2xyz`
Required Functions	

xyz2luv

Purpose

Compute lightness and chromaticity coordinates in CIELUV space from XYZ.

Description

`xyz2luv` computes the lightness, *L**, and chromatic coordinates *u** and *v** in CIELUV of set of colors from their *XYZ* tristimulus values.

CIELUV is a simple appearance model providing perceptual descriptors (lightness, hue and chroma) for related colors (colors in a scene).

In this representation, information about the illumination conditions or, alternatively, about the scene, is included in a reference stimulus. Using CIELUV in the standard conditions implies that the reference stimulus is a perfect diffuser illuminated as the test.

Syntax

```
Luv=xyz2luv(XYZ,XYZR)
```

`XYZ`	Tristimulus values of the test stimuli. For N colors, this is a Nx3 matrix.
`XYZR`	Tristimulus values of the reference stimulus. If the reference stimulus is the same for all the test stimuli, this is a 1x3 matrix. If the reference is different for each test stimulus `XYZR` is a Nx3 matrix.
`Luv`	For N colors, Nx3 matrix, containing, in columns, the lightness *L**, and the chromaticity coordinates *u** and *v**.

Related Functions

`luv2xyz, luv2perc, perc2luv`

Required Functions

xyz2rlab

Purpose Compute descriptors in RLAB space.

Description `xyz2rlab` computes the coordinates *L*, *a*, *b* in RLAB space, the chroma *C* and the hue angle (in degrees) *H*.

The transform needs information about the reference white and about the observation conditions. The absolute luminance of the reference value and its tristimulus *XYZ* values with the luminance normalized to 100 are required. The observation conditions are described by two parameters, *D* and *M0*. *D* allows to distinguish between hard-copy and soft-copy. *M0* describes the lightness of the surround.

Syntax `LABCH=xyz2rlab(XYZS,XYZW,YW,D,MO)`

`XYZS` Relative tristimulus values of the samples.
For N colors, this is a Nx3 matrix.

`XYZW` Relative tristimulus values of the reference white.

`YW` Luminance (cd/m^2) of the reference white (1x1).

`D` .....1 hard-copy,
.....0 soft-copy,
.....For other situations, give intermediate values between these two.

`MO` observation conditions
...1/2.3 for average surround
...1/2.9 for dim surround,
...1/3.5 for dark surround.

`LABCH` [*L a b C H*(º)]. For N colors, this is a Nx3 matrix.

Related Functions `rlab2xyz`

Required Functions

xyz2svf

Purpose

Transform XYZ into SVF descriptors.

Description

`svf2xyz` computes VF_1F_2 in SFV space from the tristimulus values, in CIE-1931 space of the stimuli when the background is *XYZW*.

The SVF space was designed for reflectant samples on an achromatic background and the tristimulus values are normalized to the background. Hopefully , the model will work correctly with other stimuli, (such as, for instance, $Y/Y_W > 100$)

Syntax

```
VF1F2=xyz2svf(XYZ,XYZW)
```

`XYZ`	Tristimulus values of the samples. For N colors, this is a Nx3 matrix.
`XYZW`	Tristimulus values of the reference white. This variable may contain a single white or a white for each stimulus.
`VF1F2`	[V (value) F_1 F_2 (chromaticity coordinates)]

Related Functions

`svf2xyz`

Required Functions

Chapter 5

Bibliography

[Fairchild 98] Mark D. Fairchild, *Color Appearance Models*, Addison-Wesley, Reading, MA (1998).

[Capilla 02 a] Capilla P, Artigas J.M., Pujol J., Luque M.J., Malo J., Martínez- Verdú F. *Fundamentos de Colorimetría*. Serie Educació: Materials 55. Servei de Publicacions de la Universitat de Valencia (2002).

[Capilla 02 b] Capilla P, Artigas J.M., Pujol J., Luque M.J., Malo J., Martínez- Verdú F. *Tecnología del Color*. Serie Educació: Materials 57. Servei de Publicacions de la Universitat de Valencia (2002).

[Boynton 86] Boynton R M. A system of photometry and colorimetry based on cone excitations. *Color Res. and Appl.*, 11, 244-252 (1986)

[Brainard 96] Brainard D. H. Cone contrast and opponent modulation color spaces. In Kaiser and Boynton. *Human Color Vision.* Optical Society of America, Washington D. C., pp. 563-579, (1996).

[Derrington et al. 84] Derrington A. M, Krauskopf J and Lennie, P. Chromatic mechanisms in lateral geniculate nucleus of macaque. *J. Physiol.*, **357**, 241-265 (1984).

[De Valois and De Valois 93] De Valois R. L. and De Valois K. K., A multi-stage color model. *Vision Res.*, **33**, 1053-1065 (1993).

[Guth 91] Guth S L. Model for color vision and light adaptation. *J. Opt. Soc. Am. A*, **8**, 976-993 (1991). Erratum. *J. Opt. Soc. Am. A,* **9**, 344 (1992).

[Guth 93] Guth S.L. Unified model for human color perception and visual adaptation II. Proc. SPIE- The International Society for Optical Engineering, **1913**, 440-448 (1993).

[Guth 94] Guth S.L. ATD model for color vision I: background. Proc. SPIE- The International Society for Optical Engineering, **2170**, 149-152, (1994).

[Guth 94] Guth S.L. ATD model for color vision II: applications. Proc. SPIE- The International Society for Optical Engineering, **2170**, 153-162 (1994).

[Guth 95] Guth S.L. Further applications of the ATD model for color vision. Proc. SPIE- The International Society for Optical Engineering, **2414**, 12-26 (1995).

[Guth et al. 80] Guth S.L. Massof R.W. and Benzschawel T. Vector model for normal and dichromatic vision. *J. Opt. Soc. Am.* **70**, 197-212 (1980).

[Hurvich and Jameson 57] Hurvich L. M. and Jameson D., An opponent-process theory of color vision. *Psych. Rev.*, **64**, 384-404 (1957).

[Jameson and Hurvich 55] Jameson D and Hurvich L. M. Some quantitative aspects of an opponent- colors theory. I. Chromatic responses and spectral saturation. *J. Opt. Soc. Am.*, **45**, 546-552, (1955).

[Judd 51] Judd D. B., Report of US. Secretariat Committee on Colorimetry and artificial daylight. *CIE Proc.* Vol I, Part 7, p. 11. (Stockholhm, 1951). Paris, CIE Central Bureau, (1951).

[Krauskopf et al. 82] Krauskopf P, Williams D. R and Heeley D. W. Cardinal directions of color space. *Vision Res.*, **22**, 1123-1131, (1982).

[Ingling and Tsou 77] Ingling C. R. Jr. and Tsou B. H., Orthogonal combinations of three visual channels. *Vision Res.*, **17**, 1075-1082 (1977).

[Smith and Pokorny 96] Smith V C and Pokorny J (1996). The design and use of a cone chromaticity space: A tutorial. *Color Res. Appl.*, **21**, 375-383.

[Vos and Walraven 71] Vos J J and Walraven P L. On the derivation of the foveal receptor primaries. *Vision Res.*, **11**, 799-818, (1971).

[Wiesen and Hubel 66] Wiesel T. N. and Hubel D. H., Spatial and chromatic interactions in the lateral geniculate body of the rhesus monkey. *J. Neurophysiol.*, **29**, 1115-1156 (1966).

[Wyszecki and Stiles 82] Wyszecki G. and Stiles W. S., *Color Science. Concepts and methods, quantitative data and formulae.* 2nd ed., Wiley, New York, (1982).

[Zrenner and Gouras 81] Zrenner E and Gouras P. Characteristics of the blue sensitive cone mechanism in primate retinal ganglion cells. *Vision Res.***,** **21**, 1605-1609, (1981).