VIRTUAL CLINICAL TRIALS

VCT Examples:

Technical report

by

BARCO NV

April 1, 2015

SHORT CONTENTS

CHAPTER 1

INTRODUCTION

The examples of how to use the Virtual Clinical Trials Simulation platform are described in this technical report. The simulation is an extension of the platform MEVIC (MEdical Virtual Imaging Chain)[Mar 08]. The role of the VCT platform was described in [MBC 13].

Each chapter describes one simulation.

CHAPTER 2

SIMULATION: APPLYING 1D LUTS TO RGB INPUT

Contents

I Introduction

This simulation aims to load an RGB image or sequence as input and then apply 1D LUTs one per channel (R,G,B).

The output is saved in raw and in txt files.

II Input

1 Input image

The input image is a 24 bit RGB image saved in a raw file (64x64pixels stored as 24bit RGB).

To open it with ImageJ by instance, use ”import raw file” with the following options:

- Image type: 24-bit RGB
- Width: 64 pixels
- Height: 64 pixels
- Offset to first image: 0 bytes
- Number of images: 1
- Gap between images: 0
- White is not zero
- Little-endian byte order
- Not open all files in folder
- Not use virtual stack

The input image is a grayscale gradient is shown in the figure 2.1.

2 Input LUTs

Three input 1D LUTs are given to the simulation. They are simply text files with 256 lines and 2 columns. The first column corresponds to the Digital Driving Level and the second level is the LUT value normalized between 0 and 1 like:

```
0 0
1 0.003921569
2 0.007843137
3 0.011764706
4 0.015686275
5 0.019607843
6 0.023529412
...
...
248 0.97254902
```

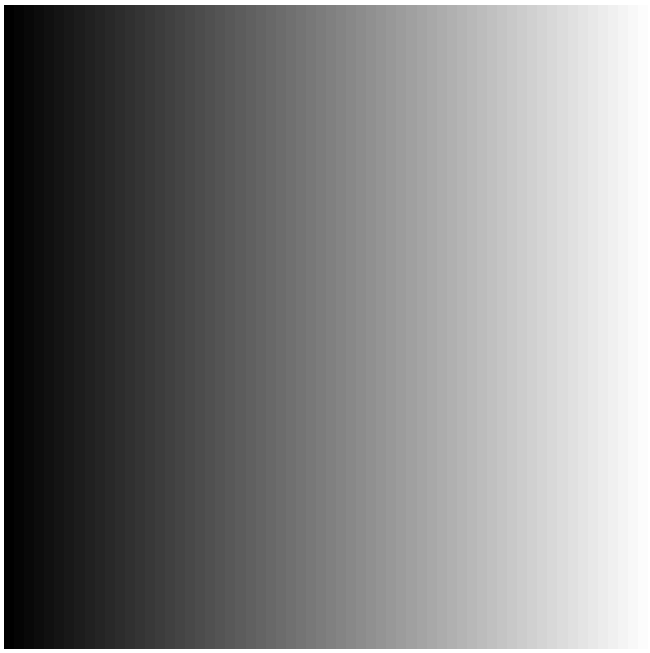

Figure 2.1: Input grayscale gradient image.

```
249  0.976470588
250  0.980392157
251  0.984313725
252  0.988235294
253  0.992156863
254  0.996078431
255  1
```

There are three 1D LUTs:

- Red: ”lut_red.txt
- Green: ”lut_green.txt”
- Blue: ”lut_blue.txt”

III Command

For running the simulation, the following command is needed:

```
"..\vct\VCT.exe" SuperXML ".\xml\test_Apply3x1DLutModule.xml" "."
```

The first argument is the path to the VCT executable. The second argument is simply the key word ”SuperXML” and the third one is the path to the xml file. Any additional parameters are used within the xml file, in this case, it corresponds to an input path.

IV xml

The input xml file is:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SuperXML>
  <LIST_OF_ITERATIONS name = "pipeline" value="1">
  </LIST_OF_ITERATIONS>

  <LIST_OF_PARAMETERS>
    <LocalParameter name = "REF">
      <SequenceRawGeneratorModule name="directory" value = "#0\input\"/>
      <Apply3x1DLutModule name="filenamePathLutRed" value = "#0\input\lut_red.
          txt"/>
      <Apply3x1DLutModule name="filenamePathLutGreen" value = "#0\input\
          lut_green.txt"/>
      <Apply3x1DLutModule name="filenamePathLutBlue" value = "#0\input\lut_blue.
          txt"/>
    </LocalParameter>
  </LIST_OF_PARAMETERS>

  <TEMPLATE_MEVIC_SIMULATION>
    <REF>
      <SequenceRawGeneratorModule>
        <directory dataType="char" value = "$"/>
        <number_of_slices dataType="int" value = "1"/>
        <width dataType = "int" value = "64"/>
        <height dataType = "int" value = "64"/>
        <nbBitsRange dataType = "int" value = "8"/>
        <_0bigEndian_1littleEndian dataType = "int" value = "0"/>
        <nbBitsOutput dataType = "int" value = "8"/>
        <nbBitsPrecision dataType = "int" value = "24"/>
        <_1WhiteIs0_0Otherwise dataType = "int" value = "0"/>
        <frame_repeat dataType = "int" value = "1"/>
        <_1RGB_0GRAY dataType = "int" value = "1"/>
      </SequenceRawGeneratorModule>
      <Apply3x1DLutModule>
        <filenamePathLutRed dataType="string" value="$"/>
        <filenamePathLutGreen dataType="string" value="$"/>
        <filenamePathLutBlue dataType="string" value="$"/>
        <nbBits dataType="unsigned int" value="8"/>
      </Apply3x1DLutModule>
      <SaveFrameRAWModule>
        <Filename dataType="char" value="#0\output\RGBLUT"/>
        <ComponentToWrite dataType="int" value="1"/>
        <FrameToWrite dataType="int" value="-1"/>
        <ChannelToWrite dataType="int" value="-1"/>
      </SaveFrameRAWModule>
      <SaveFrameTXTModule>
        <Filename dataType="char" value="#0\output\RGBIN"/>
        <ComponentToWrite dataType="int" value="0"/>
        <FrameToWrite dataType="int" value="-1"/>
        <ChannelToWrite dataType="int" value="-1"/>
      </SaveFrameTXTModule>
      <SaveFrameTXTModule>
        <Filename dataType="char" value="#0\output\RGBLUT"/>
        <ComponentToWrite dataType="int" value="1"/>
        <FrameToWrite dataType="int" value="-1"/>
        <ChannelToWrite dataType="int" value="-1"/>
      </SaveFrameTXTModule>
    </REF>
  </TEMPLATE_MEVIC_SIMULATION>
</SuperXML>
```

For the module ”SequenceRawGeneratorModule”, the character sequence ”$\#0$” is replaced by the additional input argument given at the execution of the program.

V Output

The output is a an image or a sequence or RGB frames.

VI Module: SequenceRawGeneratorModule

1 Parameter: ”directory”

This module is used for loading an image or a sequence of images. The first parameter of this module corresponds to a path of a directory. If the directory has more than one file with the extension ”raw”, the module will open directly the files with the raw extension depending on the following parameter: ”number_of_slices”.

2 Parameter: ”number_of_slices”

If this parameter is equal to ”1” then only one image is opened.

3 Parameter: image-sequence

The parameters from ”” to ”” correspond to the image or the sequence.

4 Parameter: ”frame_repeat”

In order to generate a sequence of frames for the display simulation, this parameter can be used. If this parameter is equal to 1, one slice from the input sequence corresponds to 1 frame in the simulation. In order to repeat each slice this parameter can be used with a value superior to 1. This frame repeat parameter can be in float like described in [Mar 12] in order to simulate a browsing speed that is not an integer.

Let F_{browse} show the slice browsing speed, $F_{refresh}$ show the frame refresh rate (a display property in Hz), and FR show frame repeat. F_{browse} = $\frac{F_{refresh}}{F}$. For example, at $F_{refresh}$ of 50 frame per second (fps), if each slice is fed twice to the display at consecutive refreshes ($FR = 2$), the apparent slice browsing speed is $\frac{50}{2} = 25$ slice per second (sps). In other words, $FR = \frac{F_{refresh}}{F_{browse}} = \frac{50}{25} = 2$. Hence, the browsing speeds that can be simulated with integer FRs are very limited. By allowing a fractional frame repeat, one can have arbitrary browsing speeds as follows. As an example, F_{browse} of 40 sps can be achieved if we make 5 frames out of every 4 slices. In this case $FR = \frac{F_{refresh}}{F_{browse}} = \frac{50}{40} = \frac{5}{4}$. To that end, we use an error accumulation method to find out which slices should be repeated: starting from the beginning of the stack (the residue is initially set to zero), each slice is copied floor(FR+residue) times, generating that many frames, and the residue is updated to $FR + residue - floor(FR + residue)$. This way, when the residue goes above one, an extra frame with a copy of the current slice is inserted. To have a slice browsing speed of 40 sps, on a 41-slice stack (comprised of slices 1, 2,

..., 41), when $F_{refresh}$ is 50 fps, the following slices are written to the frame buffer: 1 2 3 4 4 5 6 7 8 8 9 10 11 12 12 13 14 15 16 16 17 18 19 20 20 21 22 23 24 24 25 26 27 28 28 29 30 31 32 32 33 34 35 36 36 37 38 39 40 40 41. In this example, slice n is copied twice if $mod(n, 4) = 0$, and all other slices are copied only once.

VII Module: "Apply3x1DLutModule"

1 Parameter: "filenamepathLutRed"

Path and filename of the input text file of the 1D Red Lut.

2 Parameter: "filenamepathLutGreen"

Path and filename of the input text file of the 1D Green Lut.

3 Parameter: "filenamepathLutBlue"

Path and filename of the input text file of the 1D Blue Lut.

VIII Module: SaveFrameRAWModule

This module saves a component in a binary file.

1 Parameter: "Filename"

This parameter is the path and filename of the output file without the extension ".raw".

2 Parameter: "ComponentToWrite"

This parameter selections the component to save. The module "SequenceRawGeneratorModule" creates a component with the index "0". The module "Apply3x1DLutModule" creates a component with the index "1". The module "SaveFrameRawModule" does not create any component.

Therefore if this parameter has the value "1", it will save the component of the module "Apply3x1DLutModule"

3 Parameter: "FrameToWrite"

This parameter selections the frame to save. If the value is "-1" then all frames will be saved.

4 Parameter: "ChannelToWrite"

This parameter selections the channel to save, in case of RGB by instance, three channels can be saved. If the value is "-1" then all channels will be saved.

The filename is given as input and a filename extension will automatically generated with the extension ”.raw”.

An initialization txt file is automatically generated with the parameter to give for manipulating the raw file or open with ImageJ by instance:

```
file name: ./output/RGBLUT
width: 64
height: 64
type imageJ: 32-bit Unsigned
```

The filename uses the filename given as input parameter and adds the suffix ”_description_raw” before the extension ”.raw”.

IX Module: SaveFrameTXTModule

This module saves a component in a txt file.

1 Parameter: ”Filename”

This parameter is the path and filename of the output file without the extension ”.txt”. The extension is automatically generated.

2 Parameter: ”ComponentToWrite”

This parameter selections the component to save. The module ”SequenceRawGeneratorModule” creates a component with the index ”0”. The module ”Apply3x1DLutModule” creates a component with the index ”1”. The module ”SaveFrameTxtModule” does not create any component.

Therefore if this parameter has the value ”1”, it will save the component of the module ”Apply3x1DLutModule”.

3 Parameter: ”FrameToWrite”

This parameter selections the frame to save. If the value is ”-1” then all frames will be saved.

4 Parameter: ”ChannelToWrite”

This parameter selections the channel to save, in case of RGB by instance, three channels can be saved. If the value is ”-1” then all channels will be saved.

The filename is given as input and an extension will be automatically generated with the extension ”.txt”.

The generated file can be simply edited or imported as ”txt image” with ImageJ.

CHAPTER 3

sRGB DISPLAY SIMULATION

Contents

This simulation loads a RGB image or sequence and then simulate a pure sRGB display by converting the RGB values to XYZ triplets.

I Introduction

This simulation aims to load an RGB image or sequence as input and then apply 1D LUTs one per channel (R,G,B).

The output is saved in raw and in txt files.

II Input

1 Input image

The input image is a 24 bit RGB image saved in a raw file (64x64pixels stored as 24bit RGB).

To open it with ImageJ by instance, use ”import raw file” with the following options:

- Image type: 24-bit RGB
- Width: 64 pixels
- Height: 64 pixels
- Offset to first image: 0 bytes
- Number of images: 1
- Gap between images: 0
- White is not zero
- Little-endian byte order
- Not open all files in folder
- Not use virtual stack

The input image is a grayscale gradient is shown in the figure 3.1.

III Command

For running the simulation, the following command is needed:

```
"..\vct\VCT.exe" SuperXML ".\xml\test_SRgbDisplayModule.xml" "."
```

The first argument is the path to the VCT executable. The second argument is simply the key word ”SuperXML” and the third one is the path to the xml file. Any additional parameters are used within the xml file, in this case, it corresponds to an input path.

IV xml

The input xml file is:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SuperXML>
  <LIST_OF_ITERATIONS name = "pipeline" value="1">
  </LIST_OF_ITERATIONS>
```

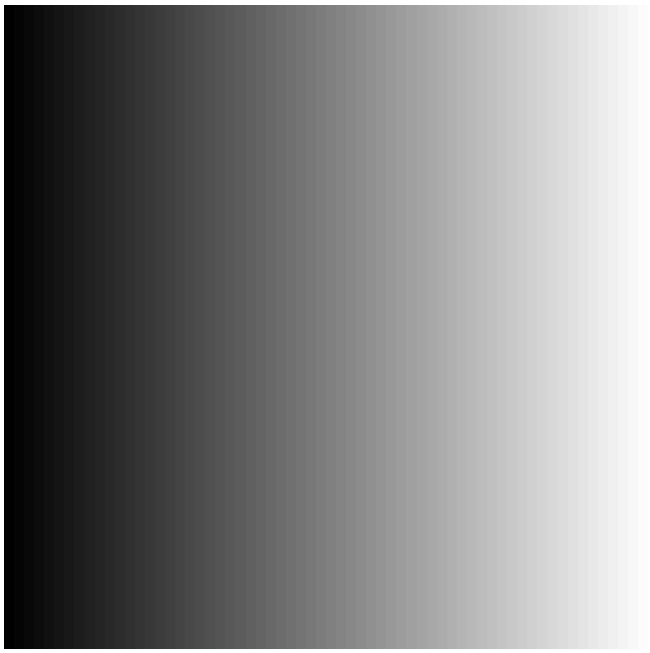

Figure 3.1: Input grayscale gradient image.

```
<LIST_OF_PARAMETERS>
  <LocalParameter name = "REF">
    <SequenceRawGeneratorModule name="directory" value = "#0\input\"/>
  </LocalParameter>
</LIST_OF_PARAMETERS>

<TEMPLATE_MEVIC_SIMULATION>
  <REF>
    <SequenceRawGeneratorModule>
      <directory dataType="char" value = "$"/>
      <number_of_slices dataType="int" value = "1"/>
      <width dataType = "int" value = "64"/>
      <height dataType = "int" value = "64"/>
      <nbBitsRange dataType = "int" value = "8"/>
      <_0bigEndian_1littleEndian dataType = "int" value = "0"/>
      <nbBitsOutput dataType = "int" value = "8"/>
      <nbBitsPrecision dataType = "int" value = "24"/>
      <_1WhiteIs0_0Otherwise dataType = "int" value = "0"/>
      <frame_repeat dataType = "int" value = "1"/>
      <_1RGB_0GRAY dataType = "int" value = "1"/>
    </SequenceRawGeneratorModule>
    <SRgbDisplayModule>
      <gamma dataType="float" value="2.2"/>
      <lumMax dataType="float" value="250.0"/>
      <contrast dataType="unsigned int" value="750"/>
      <illum dataType="string" value="D50"/>
    </SRgbDisplayModule>
    <SaveFrameRAWModule>
      <Filename dataType="char" value="#0\output\sRGB"/>
      <ComponentToWrite dataType="int" value="0"/>
      <FrameToWrite dataType="int" value="-1"/>
      <ChannelToWrite dataType="int" value="-1"/>
    </SaveFrameRAWModule>
    <SaveFrameRAWModule>
      <Filename dataType="char" value="#0\output\sRGB2XYZD50"/>
```

```
        <ComponentToWrite dataType="int" value="1"/>
        <FrameToWrite dataType="int" value="-1"/>
        <ChannelToWrite dataType="int" value="-1"/>
      </SaveFrameRAWModule>
    </REF>
  </TEMPLATE_MEVIC_SIMULATION>
</SuperXML>
```

For the module "SequenceRawGeneratorModule", the character sequence "#0" is replaced by the additional input argument given at the execution of the program.

V Output

The output is a an image or a sequence or XYZ frames.

VI Module: SequenceRawGeneratorModule

1 Parameter: "directory"

This module is used for loading an image or a sequence of images. The first parameter of this module corresponds to a path of a directory. If the directory has more than one file with the extension "raw", the module will open directly the files with the raw extension depending on the following parameter: "number_of_slices".

2 Parameter: "number_of_slices"

If this parameter is equal to "1" then only one image is opened.

3 Parameter: image-sequence

The parameters from "" to "" correspond to the image or the sequence.

4 Parameter: "frame_repeat"

In order to generate a sequence of frames for the display simulation, this parameter can be used. If this parameter is equal to 1, one slice from the input sequence corresponds to 1 frame in the simulation. In order to repeat each slice this parameter can be used with a value superior to 1. This frame repeat parameter can be in float like described in [Mar 12] in order to simulate a browsing speed that is not an integer.

Let F_{browse} show the slice browsing speed, $F_{refresh}$ show the frame refresh rate (a display property in Hz), and FR show frame repeat. $F_{browse} = \frac{F_{refresh}}{F}$. For example, at $F_{refresh}$ of 50 frame per second (fps), if each slice is fed twice to the display at consecutive refreshes ($FR = 2$), the apparent slice browsing speed is $\frac{50}{2} = 25$ slice per second (sps). In other words, $FR = \frac{F_{refresh}}{F_{browse}} = \frac{50}{25} = 2$. Hence, the browsing speeds that can be simulated with integer FRs are very limited. By allowing a fractional frame repeat, one can have arbitrary browsing speeds as follows. As an example, F_{browse} of 40 sps can

be achieved if we make 5 frames out of every 4 slices. In this case $FR = \frac{F_{refresh}}{F_{browse}} = \frac{50}{40} = \frac{5}{4}$. To that end, we use an error accumulation method to find out which slices should be repeated: starting from the beginning of the stack (the residue is initially set to zero), each slice is copied floor(FR+residue) times, generating that many frames, and the residue is updated to $FR + residue - floor(FR + residue)$. This way, when the residue goes above one, an extra frame with a copy of the current slice is inserted. To have a slice browsing speed of 40 sps, on a 41-slice stack (comprised of slices 1, 2, ..., 41), when $F_{refresh}$ is 50 fps, the following slices are written to the frame buffer: 1 2 3 4 4 5 6 7 8 8 9 10 11 12 12 13 14 15 16 16 17 18 19 20 20 21 22 23 24 24 25 26 27 28 28 29 30 31 32 32 33 34 35 36 36 37 38 39 40 40 41. In this example, slice n is copied twice if $mod(n, 4) = 0$, and all other slices are copied only once.

VII Module: SRgbDisplayModule

This module converts RGB input to XYZ values that correspond to a perfect $sRGB$ display.

1 Parameter: ”gamma”

By default the gamma value is ”$\gamma = 2.2$”.

2 Parameter: ”lumMax”

This parameter ”lumMax” corresponds to the maximum luminance in cd/m^2 given by the white point. The default value is $lumMax = 250cd/m^2$.

3 Parameter: ”contrast”

This parameter ”contrast” corresponds to the contrast of the display defined as:

$$C = \frac{lumMax}{lumMin} \tag{3.1}$$

The contrast has values superior to ”1”. The default value is $C = 750$.

4 Parameter: ”illum”

The parameter ”illum” is the illuminant by default ”D65”. I can be either ”D50” or ”D65”.

Based on the different parameters, the conversion is done according to:

$R, G, B \in [0, 1]$

$$if((R, G, B) > 0.04045)$$
$$(R, G, B) = (\frac{((R, G, B) + 0.055)}{1.055})^{\gamma};$$
$$else$$

$$(R,G,B) = \frac{(R,G,B)}{12.92};$$

In case of "D65":

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} 0.4124564 & 0.3575761 & 0.1804375 \\ 0.2126729 & 0.7151522 & 0.0721750 \\ 0.0193339 & 0.1191920 & 0.9503041 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{3.2}$$

The white point (X_0,Y_0,Z_0):

$$\begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \end{bmatrix} = \begin{bmatrix} 0.4124564 & 0.3575761 & 0.1804375 \\ 0.2126729 & 0.7151522 & 0.0721750 \\ 0.0193339 & 0.1191920 & 0.9503041 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \tag{3.3}$$

In case of "D50":

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} 0.4360747 & 0.3850649 & 0.1430804 \\ 0.2225045 & 0.7168786 & 0.0606169 \\ 0.0139322 & 0.0971045 & 0.7141733 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{3.4}$$

The white point:

$$\begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \end{bmatrix} = \begin{bmatrix} 0.4360747 & 0.3850649 & 0.1430804 \\ 0.2225045 & 0.7168786 & 0.0606169 \\ 0.0139322 & 0.0971045 & 0.7141733 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \tag{3.5}$$

$$lumMin = \frac{lumMax}{C}$$

The triplet (X,Y,Z) is:

$$X = lumMin.X_0 + X'.(lumMax - lumMin).X_0$$

$$Y = lumMin + Y'.(lumMax - lumMin)$$

$$Z = lumMin.Z_0 + Z'.(lumMax - lumMin).Z_0$$

VIII Module: SaveFrameRAWModule

This module saves a component in a binary file.

1 Parameter: "Filename"

This parameter is the path and filename of the output file without the extension ".raw".

2 Parameter: ”ComponentToWrite”

This parameter selections the component to save. The module ”SequenceRawGeneratorModule” creates a component with the index ”0”. The module ”SRgbDisplayModule” creates a component with the index ”1”. The module ”SaveFrameRawModule” does not create any component.

Therefore if this parameter has the value ”1”, it will save the component of the module ”SRgbDisplayModule”.

3 Parameter: ”FrameToWrite”

This parameter selections the frame to save. If the value is ”-1” then all frames will be saved.

4 Parameter: ”ChannelToWrite”

This parameter selections the channel to save, in case of RGB by instance, three channels can be saved. If the value is ”-1” then all channels will be saved.

The filename is given as input and a filename extension will automatically generated with the extension ”.raw”.

An initialization txt file is automatically generated with the parameter to give for manipulating the raw file or open with ImageJ by instance:

SequenceRawGeneratorModule:

```
file name: ./output/sRGB
width: 64
height: 64
type imageJ: 32-bit Unsigned
```

SRgbDisplayModule:

```
file name: ./output/sRGB2XYZD50
width: 64
height: 64
type imageJ: 32-bit Real
```

The filename uses the filename given as input parameter and adds the suffix ”_description_raw” before the extension ”.raw”.

CHAPTER 4

RGB TO XYZ DISPLAY SIMULATION USING A MASKING MODEL

Contents

The simulation described in this chapter corresponds to the conversion of RGB to XYZ tripplets using the modified masking model firstly introduced by Tamura in [Tam 02].

I Introduction

The basic principle of the masking model is to construct the color not only with Red, Green and Blue components, but also with secondary colors and gray. So the three channels (R, G and B) are simultaneously used during the measurements (given as output X, Y and Z values in cd/m^2). Therefore channel interaction is measured and present in the measurements.

Therefore the 7 primary, secondary and gray curves are needed input for this simulation.

II Input

1 Input image

The input image is a 24 bit RGB image saved in a raw file (64x64pixels stored as 24bit RGB).

To open it with ImageJ by instance, use ”import raw file” with the following options:

- Image type: 24-bit RGB
- Width: 64 pixels
- Height: 64 pixels
- Offset to first image: 0 bytes
- Number of images: 1
- Gap between images: 0
- White is not zero
- Little-endian byte order
- Not open all files in folder
- Not use virtual stack

The input image is a pink flat-field image and is shown in the figure 4.1.

2 Input Measurements

The 7 primary, secondary and gray curves are given as input in csv files. The first column corresponds to the digital driving level and the following columns are the X, Y and Z corresponding values. There are seven csv files, hereby a sample of the red curve:

```
0,0.231550150928,0.27,0.392476936462,
1,0.232805131007,0.27,0.392869413398,
2,0.235034943077,0.27,0.393262282812,
...
...
252,112.932819448,53.19,2.35753007203,
253,114.100589483,53.74,2.38192031594,
254,115.310823883,54.31,2.4071974778,
255,116.538441874,54.88,2.41509104764,
```

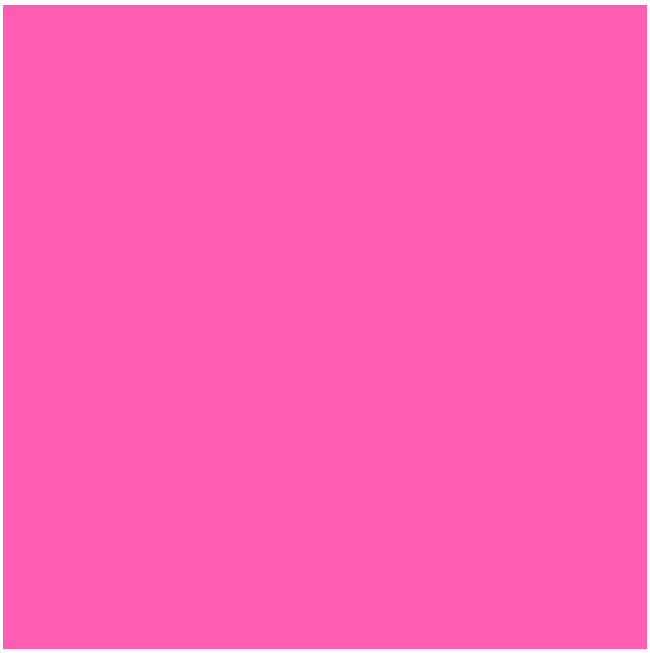

Figure 4.1: Input pink flat-field image.

There are seven curves:

- Red: ”red.csv
- Green: ”green.csv”
- Blue: ”blue.csv”
- Cyan: ”cyan.csv”
- Magenta: ”magenta.csv”
- Yellow: ”yellow.csv”
- Gray: ”gray.csv”

III Command

For running the simulation, the following command is needed:

```
"..\vct\VCT.exe" SuperXML ".\xml\test_Rgb2XYZDisplayModule.xml" "."
```

The first argument is the path to the VCT executable. The second argument is simply the key word ”SuperXML” and the third one is the path to the xml file. Any additional parameters are used within the xml file, in this case, it corresponds to an input path.

IV xml

The input xml file is:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SuperXML>
  <LIST_OF_ITERATIONS name = "pipeline" value="1">
  </LIST_OF_ITERATIONS>

  <LIST_OF_PARAMETERS>
    <LocalParameter name = "REF">
      <SequenceRawGeneratorModule name="directory" value = "#0\input"/>
    </LocalParameter>
  </LIST_OF_PARAMETERS>

  <TEMPLATE_MEVIC_SIMULATION>
    <REF>
      <SequenceRawGeneratorModule>
        <directory dataType="char" value = "$"/>
        <number_of_slices dataType="int" value = "1"/>
        <width dataType = "int" value = "64"/>
        <height dataType = "int" value = "64"/>
        <nbBitsRange dataType = "int" value = "8"/>
        <_0bigEndian_1littleEndian dataType = "int" value = "0"/>
        <nbBitsOutput dataType = "int" value = "8"/>
        <nbBitsPrecision dataType = "int" value = "24"/>
        <_1WhiteIs0_0Otherwise dataType = "int" value = "0"/>
        <frame_repeat dataType = "int" value = "1"/>
        <_1RGB_0GRAY dataType = "int" value = "1"/>
      </SequenceRawGeneratorModule>
      <DisplayModule>
        <_1Color_0Monochrome dataType="int" value="1"/>
        <filenameNativeCurveGray dataType="char" value = "#0\input\gray.csv"/>
        <filenameNativeCurveRed dataType="char" value = "#0\input\red.csv"/>
        <filenameNativeCurveGreen dataType="char" value = "#0\input\green.csv"/>
        <filenameNativeCurveBlue dataType="char" value = "#0\input\blue.csv"/>
        <filenameNativeCurveCyan dataType="char" value = "#0\input\cyan.csv"/>
        <filenameNativeCurveMagenta dataType="char" value = "#0\input\magenta.
            csv"/>
        <filenameNativeCurveYellow dataType="char" value = "#0\input\yellow.csv"
            />
        <nbBits dataType="int" value="8"/>
        <frequency dataType="int" value="50"/>
      </DisplayModule>
      <Rgb2XYZDisplayModule>
      </Rgb2XYZDisplayModule>
      <SaveFrameTXTModule>
        <Filename dataType="char" value="#0\outputTest\init_txt"/>
        <ComponentToWrite dataType="int" value="0"/>
        <FrameToWrite dataType="int" value="-1"/>
        <ChannelToWrite dataType="int" value="-1"/>
      </SaveFrameTXTModule>
      <SaveFrameRAWModule>
        <Filename dataType="char" value="#0\outputTest\init_raw"/>
        <ComponentToWrite dataType="int" value="0"/>
        <FrameToWrite dataType="int" value="-1"/>
        <ChannelToWrite dataType="int" value="-1"/>
      </SaveFrameRAWModule>
      <SaveFrameTXTModule>
        <Filename dataType="char" value="#0\outputTest\txt"/>
        <ComponentToWrite dataType="int" value="1"/>
        <FrameToWrite dataType="int" value="-1"/>
        <ChannelToWrite dataType="int" value="-1"/>
```

```
        </SaveFrameTXTModule>
        <SaveFrameRAWModule>
          <Filename dataType="char" value="#0\outputTest\raw"/>
          <ComponentToWrite dataType="int" value="1"/>
          <FrameToWrite dataType="int" value="-1"/>
          <ChannelToWrite dataType="int" value="-1"/>
        </SaveFrameRAWModule>
      </REF>
    </TEMPLATE_MEVIC_SIMULATION>
</SuperXML>
```

For the module ”SequenceRawGeneratorModule”, the character sequence ”#0” is replaced by the additional input argument given at the execution of the program.

V Output

The output is a an image or a sequence or XYZ frames.

VI Module: SequenceRawGeneratorModule

1 Parameter: ”directory”

This module is used for loading an image or a sequence of images. The first parameter of this module corresponds to a path of a directory. If the directory has more than one file with the extension ”raw”, the module will open directly the files with the raw extension depending on the following parameter: ”number_of_slices”.

2 Parameter: ”number_of_slices”

If this parameter is equal to ”1” then only one image is opened.

3 Parameter: image-sequence

The parameters from ”” to ”” correspond to the image or the sequence.

4 Parameter: ”frame_repeat”

In order to generate a sequence of frames for the display simulation, this parameter can be used. If this parameter is equal to 1, one slice from the input sequence corresponds to 1 frame in the simulation. In order to repeat each slice this parameter can be used with a value superior to 1. This frame repeat parameter can be in float like described in [Mar 12] in order to simulate a browsing speed that is not an integer.

Let F_{browse} show the slice browsing speed, $F_{refresh}$ show the frame refresh rate (a display property in Hz), and FR show frame repeat. $F_{browse} = \frac{F_{refresh}}{F}$. For example, at $F_{refresh}$ of 50 frame per second (fps), if each slice is fed twice to the display at consecutive refreshes ($FR = 2$), the apparent slice

browsing speed is $\frac{50}{2} = 25$ slice per second (sps). In other words, $FR = \frac{F_{refresh}}{F_{browse}} = \frac{50}{25} = 2$. Hence, the browsing speeds that can be simulated with integer FRs are very limited. By allowing a fractional frame repeat, one can have arbitrary browsing speeds as follows. As an example, F_{browse} of $40\ sps$ can be achieved if we make 5 frames out of every 4 slices. In this case $FR = \frac{F_{refresh}}{F_{browse}} = \frac{50}{40} = \frac{5}{4}$. To that end, we use an error accumulation method to find out which slices should be repeated: starting from the beginning of the stack (the residue is initially set to zero), each slice is copied floor(FR+residue) times, generating that many frames, and the residue is updated to $FR + residue - floor(FR + residue)$. This way, when the residue goes above one, an extra frame with a copy of the current slice is inserted. To have a slice browsing speed of $40\ sps$, on a 41-slice stack (comprised of slices 1, 2, ..., 41), when $F_{refresh}$ is $50\ fps$, the following slices are written to the frame buffer: 1 2 3 4 4 5 6 7 8 8 9 10 11 12 12 13 14 15 16 16 17 18 19 20 20 21 22 23 24 24 25 26 27 28 28 29 30 31 32 32 33 34 35 36 36 37 38 39 40 40 41. In this example, slice n is copied twice if $mod(n, 4) = 0$, and all other slices are copied only once.

VII Module: DisplayModule and Rgb2XYZDisplayModule

The combination of the modules DisplayModule and Rgb2XYZDisplayModule convert RGB input to a perfect XYZ values using an accurate model named modified masking model [Tam 02].

1 Module: DisplayModule

This module does not create any new component. It is a module that creates utilities to be shared with other modules like Rgb2XYZDisplayModule.

a Parameter: _1Color_0Monochrome

If the input image is gray, so only a gray curve is needed and this parameter should be equal to "1" otherwise it must be equal to "0".

b Parameters: input curves

- filenameNativeCurveGray: path to file with gray measurements
- filenameNativeCurveRed: path to file with red measurements
- filenameNativeCurveGreen: path to file with green measurements
- filenameNativeCurveBlue: path to file with blue measurements
- filenameNativeCurveCyan: path to file with cyan measurements
- filenameNativeCurveMagenta: path to file with magenta measurements
- filenameNativeCurveYellow: path to file with yellow measurements

c Parameter: nbBits

This parameter is the input number of bits of the display simulation.

d Parameter: frequency

This parameter is the frequency in Hz of the display panel.

2 Module: Rgb2XYZDisplayModule

This module does not need any input parameters and use input parameters given to the module DisplayModule, convert RGB or Gray values to XYZ values in cd/m^2.

The remaining part of this chapter has been written with Paul Geniet. The values of the digital input vary between 0 and $d := 2^n - 1$.

a The Masking Model : Generic Principle

Let d_R, d_G and d_B be the values of the digital inputs and let $I(d_R, d_G, d_B)$ be the associated vector $\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$. Let I_R, I_G, I_B, I_C, I_Y, I_M and I_K be the color native curves :

$$\forall ddl \in [0, d], \begin{cases} \text{Red} & I_R(ddl) = I(ddl, 0, 0) \\ \text{Green} & I_G(ddl) = I(0, ddl, 0) \\ \text{Blue} & I_B(ddl) = I(0, 0, ddl) \\ \text{Yellow} & I_Y(ddl) = I(ddl, ddl, 0) \\ \text{Magenta} & I_M(ddl) = I(ddl, 0, ddl) \\ \text{cyan} & I_C(ddl) = I(0, ddl, ddl) \\ \text{Gray} & I_K(ddl) = I(ddl, ddl, ddl) \end{cases} .$$

The digital inputs d_R, d_G et d_B are associated to a construction of a color using red, green and blue. The color can also be constructed using a gray, a secondary and a primary color.

First, an associated formal tool to this construction is defined and named "the masking basis". Then, the principle of PSK Factorization standing for "Primary-Secondary-Kay" is defined, this is a particular masking basis that is used to construct the color in the masking Model.

The basics of the masking basis are defined as follow.

Definition 1 (*Masking basis*)

A masking basis is a color triplet that is composed of a gray color component, a secondary color component and a primary color component.

In the masking model, the colors are constructed by using the masking basis. That is the PSK Factorization of a color defined as follow.

Definition 2 (*PSK factorization*)

The PSK Factorization of a color is a masking basis and is composed of:

- *P: The primary color (with the highest digital input)*
- *S: The obtained secondary color by the mixing of two primary colors with higher digital inputs than the third one*
- *K: The gray color*

The figure 4.2 summarizes the construction of the PSK factorization.

Remark : Each color admits one PSK factorization.

This PSK factorization is unique if (and only if) the three digital input are all different.

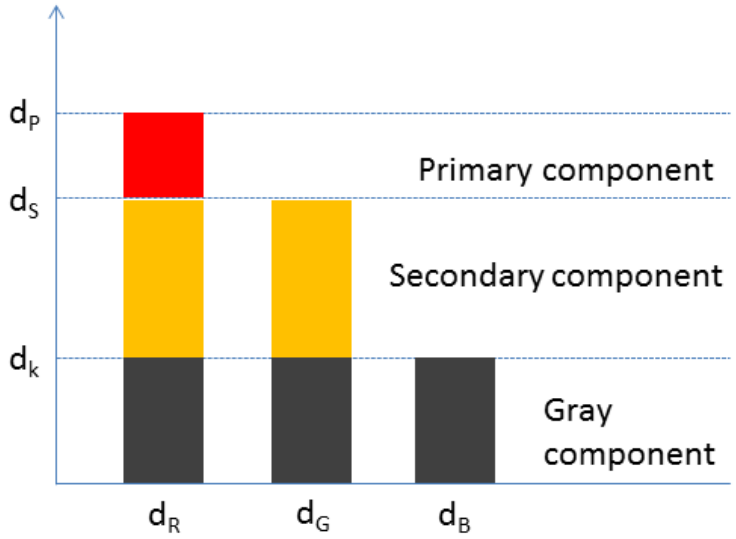

Figure 4.2: Construction of a PSK Factorization.

We use this factorization to construct one color triplet. It consists in blending the colors present in this factorization. From this point of view the quantity which must be blended has to be known. For this reason some fictive digital input are defined: d_K, d_S and d_P, as follow.

Definition 3 (*Fictive digital inputs*)

The fictive digital inputs are sorted as follow : $d_K \leqslant d_S \leqslant d_P$.
The fictive digital inputs concord with the digital inputs : $\{d_K, d_S, d_P\} = \{d_R, d_G, d_B\}$.

Remark : K stands for the gray (like the letter K in the word black). S and P respectively stand for a secondary and a primary color (from the PSK Factorization).

Example : Let C be the obtained color with the following digital input: $d_R = 128$, $d_G = 25$ and $d_B = 44$.

With $d_G \leqslant d_B \leqslant d_R$, the fictive digital input is defined by: $d_K = d_G = 25$, $d_S = d_B = 44$ and $d_P = d_R = 128$.

Therefore, the secondary color for the PSK factorization is obtained by blending red and blue. So the secondary is magenta, and the primary color of the PSK factorization is red.

The PSK factorization is therefore {Gray, Magenta, Red}.

We construct the color by mixing the colors from the PSK factorization. The XYZ value of the color is calculated with the addition of three terms: one per color from the masking basis.

The construction is summarized in the following formula (formula 1).

Formula 1 (*Color constructions*)

$$\begin{array}{ll} \textit{Conventional construction} & I(d_R,d_G,d_B) = I_R(d_R) + I_G(d_G) + I_B(d_B) \\ \textit{Masking construction} & I(d_R,d_G,d_B) = I_K(d_K) + [I_S(d_S) - I_S(d_K)] + [I_P(d_P) - I_P(d_S)] \end{array}$$

By introducing I_S and I_K channel interactions can be considered. The masking construction is therefore more accurate than the conventional construction which does not take into account the channel interaction. $I(d_R,d_G,d_B)$ for the masking construction is approximated by $J(d_R,d_G,d_B)$ defined as follow:

Formula 2 (*Color approximation*)

$$J(d_R,d_G,d_B) = J_K(d_K) + [J_S(d_S) - J_S(d_K)] + [J_P(d_P) - J_P(d_S)]$$

b The masking model in practice

b.1 Data processing A Principal Component Analysis (PCA) is applied on the measured XYZ values of each channel (R,G,B,C,Y,M,K). Therefore a single vector, $I_{PCA,i}$ stands for each channel $(I_R,I_G,I_B,I_C,I_Y,I_M,I_K)$). So for each channel, the XYZ triplet values can be processed like scalar values. It is summarized in the following definition.

Definition 4 (*Approximated color curves*)

Let i be a color (R, G, B, C, Y, M, C, K).
$\forall ddl \in [0,d]$, $I_i(ddl)$ is approximated by $J_i(ddl)$ defined as follow.

$$J_i(ddl) = c_i(ddl)\, I_{PCA,i} + I(0,0,0).$$

where J, I_{PCA} and I correspond to vectors with size (3,1), where c_i is a scalar function.

The color construction defined in the formula 1 can be therefore written as a matrix by using a masking transfer matrix form as follow:

Definition 5 (*Masking transfer matrix*)

Let $\mathscr{B} = (P,S,K)$ be a masking basis. The masking transfer matrix associated to $\mathscr{B}$ is the matrix $M_{\mathscr{B}} := (I_{PCA,P} \mid I_{PCA,S} \mid I_{PCA,K})$

The development of the defined masking color construction in formula 1 is:

Formula 3 (*Development of the color construction, step 1*)

$$J(d_R,d_G,d_B) = J_K(d_K) + [J_S(d_S) - J_S(d_K)] + [J_P(d_P) - J_P(d_S)]$$

where $J_{K,S,P}$ is given by definition 4.

By injecting J_i from definition 4 in formula 3, the following formula is obtained (formula 4):

Formula 4 (*Development of the color construction, step 2*)

$$\begin{array}{rl} J(d_R,d_G,d_B) = & c_K(d_K)\, I_{PCA,K} + I(0,0,0) + \cdots \\ & [(c_S(d_S)\, I_{PCA,S} + I(0,0,0)) - (c_S(d_K)\, I_{PCA,S} + I(0,0,0))] + \cdots \\ & [(c_P(d_P)\, I_{PCA,P} + I(0,0,0)) - (c_P(d_S)\, I_{PCA,P} + I(0,0,0))] \end{array}$$

Formula 5 (*Development of the color construction, step 3*)

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}(d_R,d_G,d_B) = c_K(d_K) \begin{pmatrix} X_{PCA,K} \\ Y_{PCA,K} \\ Z_{PCA,K} \end{pmatrix} + I(0,0,0) + \cdots$$

$$\left[\left(c_S(d_S) \begin{pmatrix} X_{PCA,S} \\ Y_{PCA,S} \\ Z_{PCA,S} \end{pmatrix} + I(0,0,0) \right) - \left(c_S(d_K) \begin{pmatrix} X_{PCA,S} \\ Y_{PCA,S} \\ Z_{PCA,S} \end{pmatrix} + I(0,0,0) \right) \right] + \cdots$$

$$\left[\left(c_P\left(d_P\right)\begin{pmatrix}X_{PCA,P}\\Y_{PCA,P}\\Z_{PCA,P}\end{pmatrix}+I\left(0,0,0\right)\right)-\left(c_P\left(d_S\right)\begin{pmatrix}X_{PCA,P}\\Y_{PCA,P}\\Z_{PCA,P}\end{pmatrix}+I\left(0,0,0\right)\right)\right]$$

Formula 6 (*The matrix $M_{\mathscr{B}}$, step 4*)

$$M_{\mathscr{B}}=\begin{pmatrix}X_{PCA,P} & X_{PCA,S} & X_{PCA,K}\\Y_{PCA,P} & Y_{PCA,S} & Y_{PCA,K}\\Z_{PCA,P} & Z_{PCA,S} & Z_{PCA,K}\end{pmatrix}$$

Formula 7 (*Development of the color construction, step 5*)

$$I\left(d_R,d_G,d_B\right)=\begin{pmatrix}X_{PCA,P} & X_{PCA,S} & X_{PCA,K}\\Y_{PCA,P} & Y_{PCA,S} & Y_{PCA,K}\\Z_{PCA,P} & Z_{PCA,S} & Z_{PCA,K}\end{pmatrix}\begin{pmatrix}c_p\left(d_p\right)-c_p\left(d_s\right)\\c_s\left(d_s\right)-c_s\left(d_K\right)\\c_K\left(d_K\right)\end{pmatrix}+I\left(0,0,0\right)$$

In formula 6 and 7, the matrix $M_{\mathscr{B}}$ corresponds to the new primary, secondary and gray components resulting from the principal component analysis. The 3 column vectors are unitary vectors. The coefficient's vector corresponds then to a norm and therefore to color intensity in the new basis.

Proposal 1 (*Matrix writing of the color construction*)

Let $\mathscr{B}$ be the masking basis associated to the color construction.

$$I\left(d_R,d_G,d_B\right)=M_{\mathscr{B}}\begin{pmatrix}c_p\left(d_p\right)-c_p\left(d_s\right)\\c_s\left(d_s\right)-c_s\left(d_K\right)\\c_K\left(d_K\right)\end{pmatrix}+I\left(0,0,0\right).$$

The three axes of the principal component analysis form an orthonormal basis. Therefore the scalar functions c_i can be calculated as follow.

Proposal 2 (*c_i values*)

Let i be a color (R, G, B, C, Y, M, Gr) and let c_i be the scalar function defined in definition 4.
$\forall ddl \in [0,d]$, $c_i\left(ddl\right)=\left[J_i\left(ddl\right)-I\left(0,0,0\right)\right]\cdot I_{PCA,i}$.

Proof : It is the calculation of an orthonormal factorization. ■

Color intensities must be expressed in the new basis, it is a norm and therefore it does correspond to the norm of the projected original color vector to the new axis.

So, by using the last proposal, the $c_i\left(ddl\right)$ are calculated thanks to the measured data (cf curve on the figure 4.3).

The measured input values are limited. We only have 2^n input values per primary, secondary and gray curve. We need floating point digital input values. Therefore, simple linear interpolation (simple spline case) is used for calculating the c_i values for any arbitrary digital input.

To use this theory and to calculate the associated digital input to any XYZ arbitrary values, reciprocal functions of the c_i must be found. The following theorem proves the existence of these reciprocal functions.

Theorem 1 (*Bijectivity of the c_i*)

Let i be a color (R, G, B, C, Y, M, Gr) and let c_i be the scalar function defined in definition 4.
c_i is a bijective function.

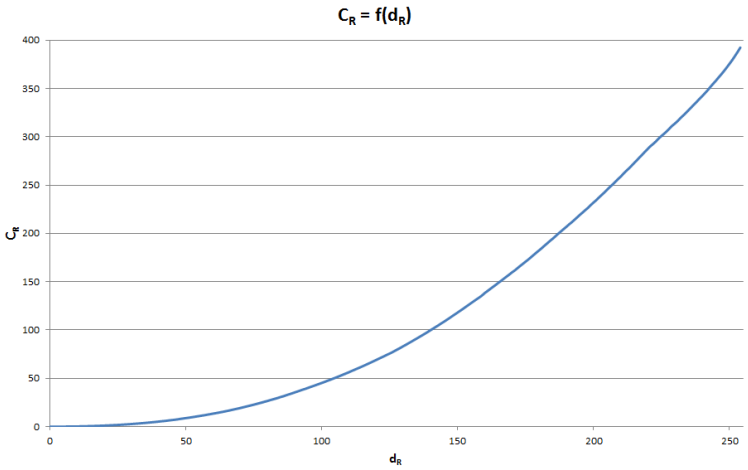

Figure 4.3: Coefficient of the first principal component for red.

Proof : X, Y and Z are increasing continuous functions of the digital inputs. So the c_i functions are also increasing and continuous. So, using the bijection theorem, the c_i functions are bijective. ■

In the two next subsections, the processing of the measures is supposed to have been made. So, the 7 I_{PCAi} vectors and the c_i functions (as their reverse functions) are supposed to be known.

b.2 Phases for the conversion from RGB to XYZ In this subsection, the conversion phases from any arbitrary digital input triplet values (in float) to XYZ triplet values are described.

Let d_R, d_G and d_B be the three arbitrary digital inputs.

First of all, we calculate the fictive digital inputs, d_P, d_S and d_K as given in definition 3.

Secondly we calculate the PSK factorization (called $\mathscr{B}$) of the color as given in definition 2.

We calculate thirdly the masking transfer matrix, $M_{\mathscr{B}}$, and the associated coefficients $c_i\left(d_i\right)$.

Finally the resulting triplet $J\left(d_R, d_G, d_B\right)$ or (X,Y,Z) is calculated.

c Modified Masking Model

The generic principle of the modified masking model does not differ too much compared to the masking model [Tam 02].

c.1 Conversion from RGB to XYZ The realized approximation given in definition 4 is not effective since the deviation from linearity is very large.

The common method described in [Tam 02] consists of including the second axis of the principal component analysis. So the approximation of the color of the formula 1 is transformed by the following definition:

Definition 6 (*Approximated color curves in the Modified Masking Model*)

Let i be a color (R, G, B, C, Y, M, C, Gr).
$\forall ddl \in [0, d]$, $I_i\left(ddl\right)$ is approximate by $J_i\left(ddl\right)$ defined as follow.

$$J_i\left(ddl\right) = I\left(0,0,0\right) + c_{i,1}\left(ddl\right) I_{PCA,i,1} + \mathbf{c_{i,2}}\left(\mathbf{ddl}\right) \mathbf{I_{PCA,i,2}}$$

Using the same process than in the masking model, the value of the functions $c_{i,j}$ can be calculated and the value of $I\left(d_R, d_G, d_B\right)$ can be approximated by $I\left(d_R, d_G, d_B\right)$ calculated like in the masking model.

It is also possible to use this model to calculate the XYZ value associated to the digital input.

The matrix $M_{\mathscr{B}}$ for the modified masking model is:

Formula 8 (*The matrix $M_{\mathscr{B}}$ for the modified masking model*)

$$M_{\mathscr{B}} = \begin{pmatrix} X_{PCA,P,1} & X_{PCA,S,1} & X_{PCA,K,1} & X_{PCA,P,2} & X_{PCA,S,2} & X_{PCA,K,2} \\ Y_{PCA,P,1} & Y_{PCA,S,1} & Y_{PCA,K,1} & Y_{PCA,P,2} & Y_{PCA,S,2} & Y_{PCA,K,2} \\ Z_{PCA,P,1} & Z_{PCA,S,1} & Z_{PCA,K,1} & Z_{PCA,P,2} & Z_{PCA,S,2} & Z_{PCA,K,2} \end{pmatrix}$$

The color construction for the modified masking model is:

Formula 9 (*Color construction*)

$$I\left(d_R, d_G, d_B\right) =$$

$$\begin{pmatrix} X_{PCA,P,1} & X_{PCA,S,1} & X_{PCA,K,1} & X_{PCA,P,2} & X_{PCA,S,2} & X_{PCA,K,2} \\ Y_{PCA,P,1} & Y_{PCA,S,1} & Y_{PCA,K,1} & Y_{PCA,P,2} & Y_{PCA,S,2} & Y_{PCA,K,2} \\ Z_{PCA,P,1} & Z_{PCA,S,1} & Z_{PCA,K,1} & Z_{PCA,P,2} & Z_{PCA,S,2} & Z_{PCA,K,2} \end{pmatrix} \begin{pmatrix} c_{P,1}\left(d_P\right) - c_{P,1}\left(d_S\right) \\ c_{P,1}\left(d_S\right) - c_{S,1}\left(d_K\right) \\ c_{K,1}\left(d_K\right) \\ c_{P,2}\left(d_P\right) - c_{P,2}\left(d_S\right) \\ c_{S,2}\left(d_S\right) - c_{S,2}\left(d_K\right) \\ c_{K,2}\left(d_K\right) \end{pmatrix}$$

$$+ I\left(0,0,0\right)$$

VIII Module: SaveFrameTXTModule

This module saves a component in a txt file.

1 Parameter: ”Filename”

This parameter is the path and filename of the output file without the extension ”.txt”. The extension is automatically generated.

2 Parameter: ”ComponentToWrite”

This parameter selections the component to save. The module ”SequenceRawGeneratorModule” creates a component with the index ”0”. The module ”Rgb2XYZDisplayModule” creates a component with the index ”1”. The module ”SaveFrameTxtModule” does not create any component.

Therefore if this parameter has the value ”1”, it will save the component of the module ”Apply3x1DLutModule”

3 Parameter: ”FrameToWrite”

This parameter selections the frame to save. If the value is ”-1” then all frames will be saved.

4 Parameter: ”ChannelToWrite”

This parameter selections the channel to save, in case of RGB by instance, three channels can be saved. If the value is ”-1” then all channels will be saved.

The filename is given as input and an extension will be automatically generated with the extension ”.txt”.

The generated file can be simply edited or imported as ”txt image” with ImageJ.

IX Module: SaveFrameRAWModule

This module saves a component in a binary file.

1 Parameter: ”Filename”

This parameter is the path and filename of the output file without the extension ”.raw”.

2 Parameter: ”ComponentToWrite”

This parameter selections the component to save. The module ”SequenceRawGeneratorModule” creates a component with the index ”0”. The module ”Rgb2XYZDisplayModule” creates a component with the index ”1”. The module ”SaveFrameRawModule” does not create any component.

Therefore if this parameter has the value ”1”, it will save the component of the module ”SRgbDisplayModule”.

3 Parameter: ”FrameToWrite”

This parameter selections the frame to save. If the value is ”-1” then all frames will be saved.

4 Parameter: ”ChannelToWrite”

This parameter selections the channel to save, in case of RGB by instance, three channels can be saved. If the value is ”-1” then all channels will be saved.

The filename is given as input and a filename extension will automatically generated with the extension ”.raw”.

An initialization txt file is automatically generated with the parameter to give for manipulating the raw file or open with ImageJ by instance:

SequenceRawGeneratorModule:

```
file name: ./output/sRGB
width: 64
height: 64
type imageJ: 32-bit Unsigned
```

Rgb2XYZDisplayModule:

```
file name: ./output/sRGB2XYZD50
width: 64
height: 64
type imageJ: 32-bit Real
```

The filename uses the filename given as input parameter and adds the suffix ”_description_raw” before the extension ”.raw”.

CHAPTER 5

ΔE_{2000} DISPLAY SIMULATION

Contents

This simulation compares two colored XYZ images or sequences and computes the ΔE_{2000}.

I Introduction

The simulation loads two previously generated simulations.

1 1^{st} input

This xml is used for generating the outcome of a monitor on which a gradient is displayed (cf Figure 5.1). Except the last module the other modules have been described in the previous chapters.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SuperXML>
  <LIST_OF_ITERATIONS name = "pipeline" value="1">
  </LIST_OF_ITERATIONS>

  <LIST_OF_PARAMETERS>
    <LocalParameter name = "REF">
      <SequenceRawGeneratorModule name="directory" value = "#0\input\"/>
      <Apply3x1DLutModule name="filenamePathLutRed" value = "#0\input\lut_red.
          txt"/>
```

```
      <Apply3x1DLutModule name="filenamePathLutGreen" value = "#0\input\
          lut_green.txt"/>
      <Apply3x1DLutModule name="filenamePathLutBlue" value = "#0\input\lut_blue.
          txt"/>
    </LocalParameter>
  </LIST_OF_PARAMETERS>

  <TEMPLATE_MEVIC_SIMULATION>
    <REF>
      <SequenceRawGeneratorModule>
        <directory dataType="char" value = "$"/>
        <number_of_slices dataType="int" value = "1"/>
        <width dataType = "int" value = "64"/>
        <height dataType = "int" value = "64"/>
        <nbBitsRange dataType = "int" value = "8"/>
        <_0bigEndian_1littleEndian dataType = "int" value = "0"/>
        <nbBitsOutput dataType = "int" value = "8"/>
        <nbBitsPrecision dataType = "int" value = "24"/>
        <_1WhiteIs0_0Otherwise dataType = "int" value = "0"/>
        <frame_repeat dataType = "int" value = "1"/>
        <_1RGB_0GRAY dataType = "int" value = "1"/>
      </SequenceRawGeneratorModule>
      <Apply3x1DLutModule>
        <filenamePathLutRed dataType="string" value="$"/>
        <filenamePathLutGreen dataType="string" value="$"/>
        <filenamePathLutBlue dataType="string" value="$"/>
        <nbBits dataType="unsigned int" value="8"/>
      </Apply3x1DLutModule>
      <SRgbDisplayModule>
        <gamma dataType="float" value="2.2"/>
        <lumMax dataType="float" value="250.0"/>
        <contrast dataType="unsigned int" value="750"/>
        <illum dataType="string" value="D50"/>
      </SRgbDisplayModule>
      <SaveFrameTXTModule>
        <Filename dataType="char" value="#0\output\RGBINREF"/>
        <ComponentToWrite dataType="int" value="0"/>
        <FrameToWrite dataType="int" value="-1"/>
        <ChannelToWrite dataType="int" value="-1"/>
      </SaveFrameTXTModule>
      <SaveFrameTXTModule>
        <Filename dataType="char" value="#0\output\XYZOUTREF"/>
        <ComponentToWrite dataType="int" value="2"/>
        <FrameToWrite dataType="int" value="-1"/>
        <ChannelToWrite dataType="int" value="-1"/>
      </SaveFrameTXTModule>
      <WriterModule>
        <Filename dataType="char" value="#0\output\XYZOUTREF.bin"/>
        <compression dataType="bool" value="1"/>
        <savelastcomponent dataType="bool" value="1"/>
      </WriterModule>
    </REF>
  </TEMPLATE_MEVIC_SIMULATION>
</SuperXML>
```

The input image is a grayscale gradient and is shown in the figure 2.1.

This sub simulation generates a bin file that is the input of to the ΔE_{2000} simulation.

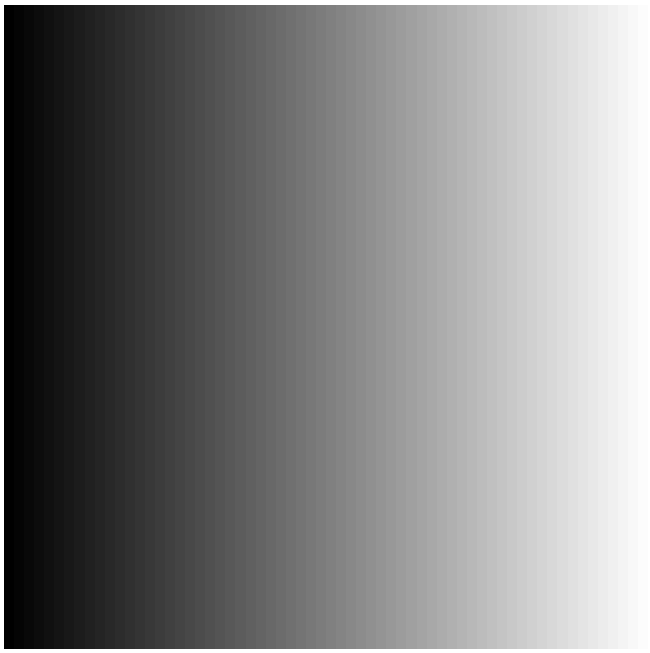

Figure 5.1: Input grayscale gradient image.

a WriterModule

This module is used for saving the simulation in a binary file called container file and its extension is ".bin".

a.1 Filename Input filename and path of the output file.

a.2 compression This parameter should be equal to "0" or "1". "1" means that the data are compressed.

a.3 savelastcomponent This parameter should be equal to "0" or "1". "1" means that only the last component of the simulation is saved.

2 2^{nd} input

This xml is used for generated the outcome of a monitor on which a noisy gradient is display (cf Figure **??**). Except the last module the other modules have been described in the previous chapters.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SuperXML>
  <LIST_OF_ITERATIONS name = "pipeline" value="1">
  </LIST_OF_ITERATIONS>

  <LIST_OF_PARAMETERS>
    <LocalParameter name = "REF">
      <SequenceRawGeneratorModule name="directory" value = "#0\input\"/>
      <Apply3x1DLutModule name="filenamePathLutRed" value = "#0\input\lut_red.
          txt"/>
```

```
      <Apply3x1DLutModule name="filenamePathLutGreen" value = "#0\input\
          lut_green.txt"/>
      <Apply3x1DLutModule name="filenamePathLutBlue" value = "#0\input\lut_blue.
          txt"/>
    </LocalParameter>
  </LIST_OF_PARAMETERS>

  <TEMPLATE_MEVIC_SIMULATION>
    <REF>
      <SequenceRawGeneratorModule>
        <directory dataType="char" value = "$"/>
        <number_of_slices dataType="int" value = "1"/>
        <width dataType = "int" value = "64"/>
        <height dataType = "int" value = "64"/>
        <nbBitsRange dataType = "int" value = "8"/>
        <_0bigEndian_1littleEndian dataType = "int" value = "0"/>
        <nbBitsOutput dataType = "int" value = "8"/>
        <nbBitsPrecision dataType = "int" value = "24"/>
        <_1WhiteIs0_0Otherwise dataType = "int" value = "0"/>
        <frame_repeat dataType = "int" value = "1"/>
        <_1RGB_0GRAY dataType = "int" value = "1"/>
      </SequenceRawGeneratorModule>
      <Apply3x1DLutModule>
        <filenamePathLutRed dataType="string" value="$"/>
        <filenamePathLutGreen dataType="string" value="$"/>
        <filenamePathLutBlue dataType="string" value="$"/>
        <nbBits dataType="unsigned int" value="8"/>
      </Apply3x1DLutModule>
      <SRgbDisplayModule>
        <gamma dataType="float" value="2.2"/>
        <lumMax dataType="float" value="250.0"/>
        <contrast dataType="unsigned int" value="750"/>
        <illum dataType="string" value="D50"/>
      </SRgbDisplayModule>
      <SaveFrameTXTModule>
        <Filename dataType="char" value="#0\output\RGBINTEST"/>
        <ComponentToWrite dataType="int" value="0"/>
        <FrameToWrite dataType="int" value="-1"/>
        <ChannelToWrite dataType="int" value="-1"/>
      </SaveFrameTXTModule>
      <SaveFrameTXTModule>
        <Filename dataType="char" value="#0\output\XYZOUTTEST"/>
        <ComponentToWrite dataType="int" value="2"/>
        <FrameToWrite dataType="int" value="-1"/>
        <ChannelToWrite dataType="int" value="-1"/>
      </SaveFrameTXTModule>
      <WriterModule>
        <Filename dataType="char" value="#0\output\XYZOUTTEST.bin"/>
        <compression dataType="bool" value="1"/>
        <savelastcomponent dataType="bool" value="1"/>
      </WriterModule>
    </REF>
  </TEMPLATE_MEVIC_SIMULATION>
</SuperXML>
```

The input image is a grayscale gradient and is shown in the figure 5.2.

This sub simulation generates a bin file with the simulation to give to the deltaE2000 simulation.

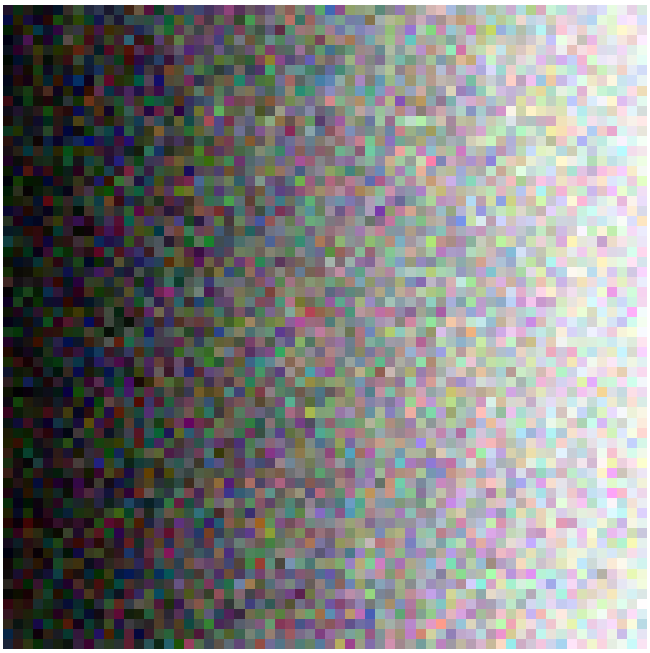

Figure 5.2: Input grayscale gradient noisy image.

a WriterModule

This module is used for saving the simulation in a binary file called container file and its extension is ".bin".

a.1 Filename Input filename and path of the output file.

a.2 compression This parameter should be equal to "0" or "1". "1" means that the data are compressed.

a.3 savelastcomponent This parameter should be equal to "0" or "1". "1" means that only the last component of the simulation is saved.

II Command

```
"..\vct\VCT.exe" SuperXML ".\xml\test_DeltaE2000Module.xml" "."
```

The first argument is the path to the VCT executable. The second argument is simply the key word "SuperXML" and the third one is the path to the xml file. Any additional parameters are used within the xml file, in this case, it corresponds to an input path.

III xml

The input xml file is:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SuperXML>
```

```
  <LIST_OF_ITERATIONS name = "pipeline" value="1">
  </LIST_OF_ITERATIONS>
  <LIST_OF_PARAMETERS>
    <LocalParameter name = "REF">
    </LocalParameter>
  </LIST_OF_PARAMETERS>
  <TEMPLATE_MEVIC_SIMULATION>
    <REF>
      <ReaderModule name="ReaderModule">
        <Filename dataType="string" value="#0\input_ref\output\XYZOUTREF.bin" />
        <Compression dataType="bool" value="1" />
      </ReaderModule>
      <ReaderModule name="ReaderModule 2">
        <Filename dataType="string" value="#0\input_test\output\XYZOUTTEST.bin"
            />
        <Compression dataType="bool" value="1" />
      </ReaderModule>
      <DeltaE2000Module name="DeltaE2000Module">
      </DeltaE2000Module>
      <SaveFrameTXTModule>
        <Filename dataType="char" value="#0\output\dE2000"/>
        <ComponentToWrite dataType="int" value="0"/>
        <FrameToWrite dataType="int" value="-1"/>
        <ChannelToWrite dataType="int" value="-1"/>
      </SaveFrameTXTModule>
    </REF>
  </TEMPLATE_MEVIC_SIMULATION>
</SuperXML>
```

IV Output

The output is a an image or a sequence or frames containing the metric value ΔE_{2000}, pixel per pixel comparison between the first and the second input (cf Figure **??**). The minimum ΔE_{2000} value is 0.5 and the maximum value is 36.51.

V Module: ReaderModule

1 Parameter: "Filename"

Input filename and path of the input ".bin" file.

.4 compression This parameter should be equal to "0" or "1". "1" means that the data are compressed.

VI Module: DeltaE2000Module

This module does not have any input parameter, it computes the ΔE_{2000} as defined by the CIE from two input containers.

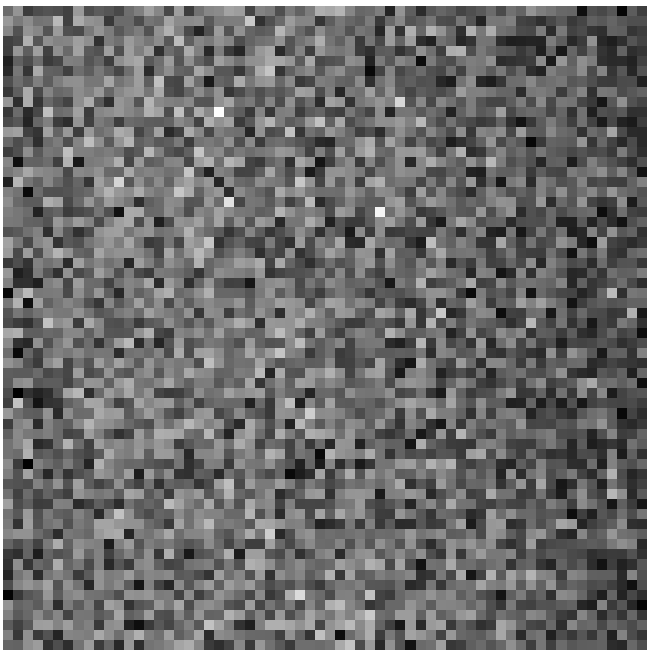

Figure 5.3: Visualization of the map of delta E 2000 differences in grayscale between image from Figure 5.1 and Figure 5.2. The minimum ΔE_{2000} value is 0.5 and the maximum value is 36.51. The perception threshold is 1.

CHAPTER 6

CLINICAL STUDY SIMULATION

Contents

The goal of this simulation is to run a virtual clinical trial with breast images that were previously used in the study described in [Mar 12]. The breast images were generated using the Bakic's anthropomorphic breast phantom developed at the University of Pennsylvania.

In this virtual clinical trials 8 bits breast images with and without signals are used as input. Half of the images do contain a signal in the center of the images and half do not contain signals.

These images are then used for simulating the displayed images represented in the XYZ color space. The simulated is a grayscale mammographic display with a contrast of $1200 : 1$ with a maximum luminance of $600cd/m^2$. The full chain is in 8 bits.

Then the second component Y is used by a single slice CHO with Laguerre-Gauss channels [Pla 09, Mye 87].

Finally a Multi-reader-Multi-Cases study is carried out by training and testing multiple single slice CHO observers using the technique described by Gallad in [Gal 06].

I Introduction

There are three steps for this simulation:

- Simulate displayed images
- Simulate one observer
- Run statistical analysis

II Simulate displayed images or ”00_displaySimulation”

1 Input image

The input image is a 8 bit image saved in a raw file (64x64pixels stored as 16 bits unsigned) (cf Figure 6.1). Even stored in 16 bits unsigned, the pixel values are all between ”0” and ”1”.

To open it with ImageJ by instance, use ”import raw file” with the following options:

- Image type: 16 bit unsigned RGB
- Width: 64 pixels
- Height: 64 pixels
- Offset to first image: 0 bytes
- Number of images: 1
- Gap between images: 0
- White is not zero
- Little-endian byte order
- Not open all files in folder
- Not use virtual stack

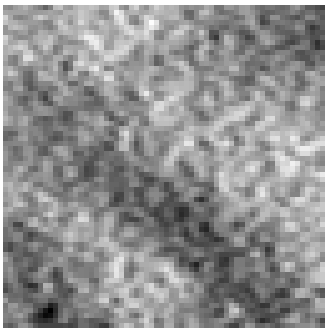

Figure 6.1: Breast image.

2 XML file

This xml is used for generating the outcome of a monitor on which breast images are displayed.The modules except "DisplayLutModule" "ConversionDDL2CDModule" have been described in the previous chapters.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SuperXML>
  <LIST_OF_ITERATIONS name = "pipeline" value="1">
  </LIST_OF_ITERATIONS>

  <LIST_OF_PARAMETERS>
    <LocalParameter name = "REF">
      <SequenceRawGeneratorModule name="directory" value = "#0"/>
      <WriterModule name="Filename" value="#2"/>
    </LocalParameter>
  </LIST_OF_PARAMETERS>

  <TEMPLATE_MEVIC_SIMULATION>
    <REF>
      <SequenceRawGeneratorModule>
        <directory dataType="char" value = "$"/>
        <number_of_slices dataType="int" value = "1"/>
        <width dataType = "int" value = "64"/>
        <height dataType = "int" value = "64"/>
        <nbBitsRange dataType = "int" value = "8"/>
        <_0bigEndian_1littleEndian dataType = "int" value = "1"/>
        <nbBitsOutput dataType = "int" value = "8"/>
        <nbBitsPrecision dataType = "int" value = "16"/>
        <_1WhiteIs0_0Otherwise dataType = "int" value = "0"/>
        <frame_repeat dataType = "int" value = "1"/>
        <_1RGB_0GRAY dataType = "int" value = "0"/>
      </SequenceRawGeneratorModule>
       <VideoCardModule name="VideoCardModule">
        <number_of_bits_out dataType="int" description="LUT" value="8" />
      </VideoCardModule>
      <DisplayModule>
        <_1Color_0Monochrome dataType="int" value="0"/>
        <filenameNativeCurveGray dataType="char" value = ".\input\gray.csv"/>
        <nbBits dataType="int" value="8"/>
        <frequency dataType="int" value="50"/>
      </DisplayModule>
      <DisplayLutModule>
      </DisplayLutModule>
      <ConversionDDL2CDModule name ="ConversionDDL2CDModule">
        <_1ddl2cd_0cd2ddl dataType="char" value="1"/>
      </ConversionDDL2CDModule>
      <SaveFrameTXTModule>
        <Filename dataType="char" value="#1\GRAYIN"/>
        <ComponentToWrite dataType="int" value="0"/>
        <FrameToWrite dataType="int" value="-1"/>
        <ChannelToWrite dataType="int" value="-1"/>
      </SaveFrameTXTModule>
      <SaveFrameTXTModule>
        <Filename dataType="char" value="#1\VIDEOCARD"/>
        <ComponentToWrite dataType="int" value="1"/>
        <FrameToWrite dataType="int" value="-1"/>
        <ChannelToWrite dataType="int" value="-1"/>
      </SaveFrameTXTModule>
```

```
        <SaveFrameTXTModule>
          <Filename dataType="char" value="#1\GRAYGSDF"/>
          <ComponentToWrite dataType="int" value="2"/>
          <FrameToWrite dataType="int" value="-1"/>
          <ChannelToWrite dataType="int" value="-1"/>
        </SaveFrameTXTModule>
        <SaveFrameTXTModule>
          <Filename dataType="char" value="#1\XYZ"/>
          <ComponentToWrite dataType="int" value="3"/>
          <FrameToWrite dataType="int" value="-1"/>
          <ChannelToWrite dataType="int" value="-1"/>
        </SaveFrameTXTModule>
        <WriterModule>
          <Filename dataType="char" value="$"/>
          <compression dataType="bool" value="1"/>
          <savelastcomponent dataType="bool" value="1"/>
        </WriterModule>
      </REF>
    </TEMPLATE_MEVIC_SIMULATION>
</SuperXML>
```

3 Command

run_simulation.bat

In this command the VCT.exe is executed several times for generating the necessary input for the virtual observer displayed images with and without signals. The images with signals are named ”diseased” and the ones without signals are named ”healthy”.

Part of the file ”run_simulation.bat”:

```
@echo off
"../../vct/VCT.exe" SuperXML "./xml/test_DisplayLutModule.xml" "./input/diseased
    /train/"0000 "./output/diseased/train/"0000 "./output/diseased/train/0000.bin
    "
...
...
...
"../../vct/VCT.exe" SuperXML "./xml/test_DisplayLutModule.xml" "./input/healthy/
    test/"0049 "./output/healthy/test/"0049 "./output/healthy/test/0049.bin"
```

Then the data are generated and organized correctly for running the single slice CHO.

4 Module: DisplayLutModule

This module does not take any input parameter. It applies a 1D LUT to the RGB digital driving levels for calibrating the display according to the Grayscale Standard Display Function as described in [DIC 98]. Based on the native curve given as input to the module DisplayModule and based on the minimum and maximum luminance values, the 1D LUT can be generated and applied.

After this step, the RGB levels are transfomed in such way that the perceived differences between two consecutive grayscale levels are constant over the full range. The perception of grayscale levels has been linearized.

5 Module: ConversionDDL2CDModule

This module converts the RGB into XYZ values using the input native curve.

The outcome is 2D maps with X, Y and Z values.

The simulation of the display generates .txt and .raw that can be easily opened with ImageJ by instance.

The simulation generates also

III Simulate one single slice observer: ”01_ssCHOSimulation”

The files generated by the display simulation are used by this module for running a clinical study. An observer is trained with 100 images and then tested with 100 other images.

1 XML file

This xml is used for generating the outcome of a monitor on which breast images are displayed.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SuperXML>
  <LIST_OF_ITERATIONS name = "pipeline" value="1">
  </LIST_OF_ITERATIONS>

  <LIST_OF_PARAMETERS>
  </LIST_OF_PARAMETERS>

  <TEMPLATE_MEVIC_SIMULATION>
    <REF>
      <SingleSliceCHOModule>
        <reader_id dataType = "unsigned long" value = "1"/>
        <image_height dataType = "unsigned long" value = "64"/>
        <image_width dataType = "unsigned long" value = "64"/>
        <image_depth dataType = "unsigned long" value = "1"/>
        <one_image_file_per_frame dataType = "bool" value = "false"/>
        <target_slice_id dataType = "unsigned long" value = "1"/>
        <num_image_pairs_training dataType = "unsigned long" value = "50"/>
        <num_image_pairs_testing dataType = "unsigned long" value = "50"/>
        <channels_number dataType = "unsigned long" value = "15"/>
        <channels_type dataType = "char" value = "LG"/>
        <channels_lg_spread dataType = "float" value = "22"/>
        <save_channelized_images dataType = "bool" value = "false"/>
        <auto_create_outputfolders dataType = "bool" value = "true"/>
        <dirpath_in_training_healthy dataType = "char" value = "#1\00
            _displaySimulation\output\healthy\train"/>
        <dirpath_in_training_diseased dataType = "char" value = "#1\00
            _displaySimulation\output\diseased\train"/>
        <dirpath_in_testing_healthy dataType = "char" value = "#1\00
            _displaySimulation\output\healthy\test"/>
        <dirpath_in_testing_diseased dataType = "char"  value = "#1\00
            _displaySimulation\output\diseased\test"/>
        <dirpath_out_channelized_training_healthy dataType = "char" value = "#0"
            />
        <dirpath_out_channelized_training_diseased dataType = "char" value = "#0
            "/>
```

```
        <dirpath_out_channelized_testing_healthy dataType = "char" value = "#0"/
            >
        <dirpath_out_channelized_testing_diseased dataType = "char" value = "#0"
            />
        <filepath_out_training_tpf_fpf dataType = "char" value = "#0\outputTest\
            training_tpf_fpf.txt"/>
        <filepath_out_training_auc_snr dataType = "char" value = "#0\outputTest\
            training_auc_snr.txt"/>
        <filepath_out_testing_tpf_fpf dataType = "char" value = "#0\outputTest\
            testing_tpf_fpf.txt"/>
        <filepath_out_testing_auc_snr dataType = "char" value = "#0\outputTest\
            testing_auc_snr.txt"/>
      </SingleSliceCHOModule>
    </REF>
  </TEMPLATE_MEVIC_SIMULATION>
</SuperXML>
```

2 Command

run_simulation.bat

or

"..\..\vct\VCT.exe" **SuperXML** ".\xml\test_ssCHO.xml" "." "../"

In this command the VCT.exe is executed for training and testing an observer.

3 output

Four files are generated:

- training_tpf_fpf.txt: True Positive Fraction in function of False Positive Fraction, training phase
- training_auc_snr.txt: AUC (Area Under the Curve) and SNR (Signal Noise Ratio) SNR, training phase
- testing_tpf_fpf.txt: True Positive Fraction in function of False Positive Fraction, testing phase
- testing_auc_snr.txt: AUC (Area Under the Curve) and SNR (Signal Noise Ratio) SNR, testing phase

4 SingleSliceCHOModule

a Parameter: "reader_id"

This is the reader identification number.

b Parameter: "image_height"

This is the image width in pixels.

c Parameter: "image_width"

This is the image height in pixels.

d Parameter: "image_depth"

This is the image depths in frame number.

e Parameter: "one_image_file_per_frame"

This is for indicating if the frames are stored within one file or split. It must be equal to false for the single slice CHO.

f Parameter: "target_slice_id"

This is the target slide or frame for training the observer. It must be equal to 1 for the single slice CHO.

g Parameter: "num_image_pairs_training"

This is the number of pairs of images for the training phase.

h Parameter: "num_image_pairs_testing"

This is the number of pairs of images for the testing phase.

i Parameter: "channels_number"

This is the number of LG channels.

j Parameter: "channels_type"

This is the type of channels but only Laguerre Gauss channels are supported.

k Parameter: "channels_lg_spread"

This is a parameter intrinsic to LG channels, it the is the spread in pixels.

l Parameter: "save_channelized_images"

This is for saving or not the channels.

m Parameter: "auto_create_outputfolders"

This for automatically or not automatically creating the subfolders.

n Parameter: ”dirpath_in_training_healthy”

This is the path of the training healthy images.

o Parameter: ”dirpath_in_training_diseased”

This is the path of the training diseased images.

p Parameter: ”dirpath_in_testing_healthy”

This is the path of the testing healthy images.

q Parameter: ”dirpath_in_testing_diseased”

This is the path of the testing diseased images.

r Parameter: ”dirpath_out_channelized_training_healthy”

This is the output path of the training healthy results.

s Parameter: ”dirpath_out_channelized_training_diseased”

This is the output path of the training diseased results.

t Parameter: ”dirpath_out_channelized_testing_healthy”

This is the output path of the testing healthy results.

u Parameter: ”dirpath_out_channelized_testing_diseased”

This is the output path of the testing diseased results.

v Parameter: ”filepath_out_training_tpf_fpf”

This is the output path and filename for the training TPF vs FPF results for the training phase.

w Parameter: ”filepath_out_training_auc_snr”

This is the output path and filename for the AUC and SNR results for the training phase.

x Parameter: ”filepath_out_testing_tpf_fpf”

This is the output path and filename for the training TPF vs FPF results for the testing phase.

y Parameter: ”filepath_out_testing_auc_snr”

This is the output path and filename for the AUC and SNR results for the testing phase.

IV Statistical analysis: ”02_MRMCSimulation”

The files generated by the display simulation are used by this module for running a clinical study. 5 observers are trained and tested on the same dataset

1 XML file

This xml is used for generating the outcome of a monitor on which breast images are displayed. The modules except ”MRMCModule” have been described in the previous chapters.

```
<?xml version=”1.0” encoding=”ISO-8859-1”?>
<SuperXML>
  <LIST_OF_ITERATIONS name = ”pipeline” value=”1”>
  </LIST_OF_ITERATIONS>

  <LIST_OF_PARAMETERS>
  </LIST_OF_PARAMETERS>

  <TEMPLATE_MEVIC_SIMULATION>
    <REF>
      <SingleSliceCHOModule>
        <reader_id dataType = ”unsigned long” value = ”1”/>
        <image_height dataType = ”unsigned long” value = ”64”/>
        <image_width dataType = ”unsigned long” value = ”64”/>
        <image_depth dataType = ”unsigned long” value = ”1”/>
        <one_image_file_per_frame dataType = ”bool” value = ”false”/>
        <target_slice_id dataType = ”unsigned long” value = ”1”/>
        <num_image_pairs_training dataType = ”unsigned long” value = ”16”/>
        <num_image_pairs_testing dataType = ”unsigned long” value = ”16”/>
        <channels_number dataType = ”unsigned long” value = ”15”/>
        <channels_type dataType = ”char” value = ”LG”/>
        <channels_lg_spread dataType = ”float” value = ”22”/>
        <save_channelized_images dataType = ”bool” value = ”false”/>
        <auto_create_outputfolders dataType = ”bool” value = ”true”/>
        <dirpath_in_training_healthy dataType = ”char” value = ”#0\input\healthy
            \train1”/>
        <dirpath_in_training_diseased dataType = ”char” value = ”#0\input\
            diseased\train1”/>
        <dirpath_in_testing_healthy dataType = ”char” value = ”#0\input\healthy\
            test”/>
        <dirpath_in_testing_diseased dataType = ”char”  value = ”#0\input\
            diseased\test”/>
        <dirpath_out_channelized_training_healthy dataType = ”char” value = ””/>
        <dirpath_out_channelized_training_diseased dataType = ”char” value = ””/
            >
        <dirpath_out_channelized_testing_healthy dataType = ”char” value = ””/>
        <dirpath_out_channelized_testing_diseased dataType = ”char” value = ””/>
        <filepath_out_training_tpf_fpf dataType = ”char” value = ”#0\output_temp
            \tpf_fpf\training\reader_01_training_tpf_fpf.txt”/>
        <filepath_out_training_auc_snr dataType = ”char” value = ”#0\output_temp
            \auc_snr\training\reader_01_training_auc_snr.txt”/>
```

```
  <filepath_out_testing_tpf_fpf dataType = "char" value = "#0\output_temp\
      tpf_fpf\testing\reader_01_testing_tpf_fpf.txt"/>
  <filepath_out_testing_auc_snr dataType = "char" value = "#0\output_temp\
      auc_snr\testing\reader_01_testing_auc_snr.txt"/>
  <filepath_out_reader_score_healthy_ref dataType = "char" value = "#0\
      output_temp\scores\reader_00_scores_healthy.txt"/>
  <filepath_out_reader_score_healthy_new dataType = "char" value = "#0\
      output_temp\scores\reader_01_scores_healthy.txt"/>
  <filepath_out_reader_score_diseased_ref dataType = "char" value = "#0\
      output_temp\scores\reader_00_scores_diseased.txt"/>
  <filepath_out_reader_score_diseased_new dataType = "char" value = "#0\
      output_temp\scores\reader_01_scores_diseased.txt"/>
 </SingleSliceCHOModule>
 <SingleSliceCHOModule>
  <reader_id dataType = "unsigned long" value = "2"/>
  <image_height dataType = "unsigned long" value = "64"/>
  <image_width dataType = "unsigned long" value = "64"/>
  <image_depth dataType = "unsigned long" value = "1"/>
  <one_image_file_per_frame dataType = "bool" value = "false"/>
  <target_slice_id dataType = "unsigned long" value = "1"/>
  <num_image_pairs_training dataType = "unsigned long" value = "16"/>
  <num_image_pairs_testing dataType = "unsigned long" value = "16"/>
  <channels_number dataType = "unsigned long" value = "15"/>
  <channels_type dataType = "char" value = "LG"/>
  <channels_lg_spread dataType = "float" value = "22"/>
  <save_channelized_images dataType = "bool" value = "false"/>
  <auto_create_outputfolders dataType = "bool" value = "true"/>
  <dirpath_in_training_healthy dataType = "char" value = "#0\input\healthy
      \train2"/>
  <dirpath_in_training_diseased dataType = "char" value = "#0\input\
      diseased\train2"/>
  <dirpath_in_testing_healthy dataType = "char" value = "#0\input\healthy\
      test"/>
  <dirpath_in_testing_diseased dataType = "char"  value = "#0\input\
      diseased\test"/>
  <dirpath_out_channelized_training_healthy dataType = "char" value = ""/>
  <dirpath_out_channelized_training_diseased dataType = "char" value = ""/
      >
  <dirpath_out_channelized_testing_healthy dataType = "char" value = ""/>
  <dirpath_out_channelized_testing_diseased dataType = "char" value = ""/>
  <filepath_out_training_tpf_fpf dataType = "char" value = "#0\output_temp
      \tpf_fpf\training\reader_02_training_tpf_fpf.txt"/>
  <filepath_out_training_auc_snr dataType = "char" value = "#0\output_temp
      \auc_snr\training\reader_02_training_auc_snr.txt"/>
  <filepath_out_testing_tpf_fpf dataType = "char" value = "#0\output_temp\
      tpf_fpf\testing\reader_02_testing_tpf_fpf.txt"/>
  <filepath_out_testing_auc_snr dataType = "char" value = "#0\output_temp\
      auc_snr\testing\reader_02_testing_auc_snr.txt"/>
  <filepath_out_reader_score_healthy_ref dataType = "char" value = "#0\
      output_temp\scores\reader_01_scores_healthy.txt"/>
  <filepath_out_reader_score_healthy_new dataType = "char" value = "#0\
      output_temp\scores\reader_02_scores_healthy.txt"/>
  <filepath_out_reader_score_diseased_ref dataType = "char" value = "#0\
      output_temp\scores\reader_01_scores_diseased.txt"/>
  <filepath_out_reader_score_diseased_new dataType = "char" value = "#0\
      output_temp\scores\reader_02_scores_diseased.txt"/>
 </SingleSliceCHOModule>
 <SingleSliceCHOModule>
  <reader_id dataType = "unsigned long" value = "3"/>
```

```
  <image_height dataType = "unsigned long" value = "64"/>
  <image_width dataType = "unsigned long" value = "64"/>
  <image_depth dataType = "unsigned long" value = "1"/>
  <one_image_file_per_frame dataType = "bool" value = "false"/>
  <target_slice_id dataType = "unsigned long" value = "1"/>
  <num_image_pairs_training dataType = "unsigned long" value = "16"/>
  <num_image_pairs_testing dataType = "unsigned long" value = "16"/>
  <channels_number dataType = "unsigned long" value = "15"/>
  <channels_type dataType = "char" value = "LG"/>
  <channels_lg_spread dataType = "float" value = "22"/>
  <save_channelized_images dataType = "bool" value = "false"/>
  <auto_create_outputfolders dataType = "bool" value = "true"/>
  <dirpath_in_training_healthy dataType = "char" value = "#0\input\healthy
      \train3"/>
  <dirpath_in_training_diseased dataType = "char" value = "#0\input\
      diseased\train3"/>
  <dirpath_in_testing_healthy dataType = "char" value = "#0\input\healthy\
      test"/>
  <dirpath_in_testing_diseased dataType = "char"  value = "#0\input\
      diseased\test"/>
  <dirpath_out_channelized_training_healthy dataType = "char" value = ""/>
  <dirpath_out_channelized_training_diseased dataType = "char" value = ""/
      >
  <dirpath_out_channelized_testing_healthy dataType = "char" value = ""/>
  <dirpath_out_channelized_testing_diseased dataType = "char" value = ""/>
  <filepath_out_training_tpf_fpf dataType = "char" value = "#0\output_temp
      \tpf_fpf\training\reader_03_training_tpf_fpf.txt"/>
  <filepath_out_training_auc_snr dataType = "char" value = "#0\output_temp
      \auc_snr\training\reader_03_training_auc_snr.txt"/>
  <filepath_out_testing_tpf_fpf dataType = "char" value = "#0\output_temp\
      tpf_fpf\testing\reader_03_testing_tpf_fpf.txt"/>
  <filepath_out_testing_auc_snr dataType = "char" value = "#0\output_temp\
      auc_snr\testing\reader_03_testing_auc_snr.txt"/>
  <filepath_out_reader_score_healthy_ref dataType = "char" value = "#0\
      output_temp\scores\reader_02_scores_healthy.txt"/>
  <filepath_out_reader_score_healthy_new dataType = "char" value = "#0\
      output_temp\scores\reader_03_scores_healthy.txt"/>
  <filepath_out_reader_score_diseased_ref dataType = "char" value = "#0\
      output_temp\scores\reader_02_scores_diseased.txt"/>
  <filepath_out_reader_score_diseased_new dataType = "char" value = "#0\
      output_temp\scores\reader_03_scores_diseased.txt"/>
 </SingleSliceCHOModule>
 <SingleSliceCHOModule>
  <reader_id dataType = "unsigned long" value = "4"/>
  <image_height dataType = "unsigned long" value = "64"/>
  <image_width dataType = "unsigned long" value = "64"/>
  <image_depth dataType = "unsigned long" value = "1"/>
  <one_image_file_per_frame dataType = "bool" value = "false"/>
  <target_slice_id dataType = "unsigned long" value = "1"/>
  <num_image_pairs_training dataType = "unsigned long" value = "16"/>
  <num_image_pairs_testing dataType = "unsigned long" value = "16"/>
  <channels_number dataType = "unsigned long" value = "15"/>
  <channels_type dataType = "char" value = "LG"/>
  <channels_lg_spread dataType = "float" value = "22"/>
  <save_channelized_images dataType = "bool" value = "false"/>
  <auto_create_outputfolders dataType = "bool" value = "true"/>
  <dirpath_in_training_healthy dataType = "char" value = "#0\input\healthy
      \train4"/>
```

```
<dirpath_in_training_diseased dataType = "char" value = "#0\input\
    diseased\train4"/>
<dirpath_in_testing_healthy dataType = "char" value = "#0\input\healthy\
    test"/>
<dirpath_in_testing_diseased dataType = "char"  value = "#0\input\
    diseased\test"/>
<dirpath_out_channelized_training_healthy dataType = "char" value = ""/>
<dirpath_out_channelized_training_diseased dataType = "char" value = ""/
    >
<dirpath_out_channelized_testing_healthy dataType = "char" value = ""/>
<dirpath_out_channelized_testing_diseased dataType = "char" value = ""/>
<filepath_out_training_tpf_fpf dataType = "char" value = "#0\output_temp
    \tpf_fpf\training\reader_04_training_tpf_fpf.txt"/>
<filepath_out_training_auc_snr dataType = "char" value = "#0\output_temp
    \auc_snr\training\reader_04_training_auc_snr.txt"/>
<filepath_out_testing_tpf_fpf dataType = "char" value = "#0\output_temp\
    tpf_fpf\testing\reader_04_testing_tpf_fpf.txt"/>
<filepath_out_testing_auc_snr dataType = "char" value = "#0\output_temp\
    auc_snr\testing\reader_04_testing_auc_snr.txt"/>
<filepath_out_reader_score_healthy_ref dataType = "char" value = "#0\
    output_temp\scores\reader_03_scores_healthy.txt"/>
<filepath_out_reader_score_healthy_new dataType = "char" value = "#0\
    output_temp\scores\reader_04_scores_healthy.txt"/>
<filepath_out_reader_score_diseased_ref dataType = "char" value = "#0\
    output_temp\scores\reader_03_scores_diseased.txt"/>
<filepath_out_reader_score_diseased_new dataType = "char" value = "#0\
    output_temp\scores\reader_04_scores_diseased.txt"/>
</SingleSliceCHOModule>
<SingleSliceCHOModule>
<reader_id dataType = "unsigned long" value = "5"/>
<image_height dataType = "unsigned long" value = "64"/>
<image_width dataType = "unsigned long" value = "64"/>
<image_depth dataType = "unsigned long" value = "1"/>
<one_image_file_per_frame dataType = "bool" value = "false"/>
<target_slice_id dataType = "unsigned long" value = "1"/>
<num_image_pairs_training dataType = "unsigned long" value = "16"/>
<num_image_pairs_testing dataType = "unsigned long" value = "16"/>
<channels_number dataType = "unsigned long" value = "15"/>
<channels_type dataType = "char" value = "LG"/>
<channels_lg_spread dataType = "float" value = "22"/>
<save_channelized_images dataType = "bool" value = "false"/>
<auto_create_outputfolders dataType = "bool" value = "true"/>
<dirpath_in_training_healthy dataType = "char" value = "#0\input\healthy
    \train5"/>
<dirpath_in_training_diseased dataType = "char" value = "#0\input\
    diseased\train5"/>
<dirpath_in_testing_healthy dataType = "char" value = "#0\input\healthy\
    test"/>
<dirpath_in_testing_diseased dataType = "char"  value = "#0\input\
    diseased\test"/>
<dirpath_out_channelized_training_healthy dataType = "char" value = ""/>
<dirpath_out_channelized_training_diseased dataType = "char" value = ""/
    >
<dirpath_out_channelized_testing_healthy dataType = "char" value = ""/>
<dirpath_out_channelized_testing_diseased dataType = "char" value = ""/>
<filepath_out_training_tpf_fpf dataType = "char" value = "#0\output_temp
    \tpf_fpf\training\reader_05_training_tpf_fpf.txt"/>
<filepath_out_training_auc_snr dataType = "char" value = "#0\output_temp
    \auc_snr\training\reader_05_training_auc_snr.txt"/>
```

```
        <filepath_out_testing_tpf_fpf dataType = "char" value = "#0\output_temp\
            tpf_fpf\testing\reader_05_testing_tpf_fpf.txt"/>
        <filepath_out_testing_auc_snr dataType = "char" value = "#0\output_temp\
            auc_snr\testing\reader_05_testing_auc_snr.txt"/>
        <filepath_out_reader_score_healthy_ref dataType = "char" value = "#0\
            output_temp\scores\reader_04_scores_healthy.txt"/>
        <filepath_out_reader_score_healthy_new dataType = "char" value = "#0\
            output_temp\scores\reader_05_scores_healthy.txt"/>
        <filepath_out_reader_score_diseased_ref dataType = "char" value = "#0\
            output_temp\scores\reader_04_scores_diseased.txt"/>
        <filepath_out_reader_score_diseased_new dataType = "char" value = "#0\
            output_temp\scores\reader_05_scores_diseased.txt"/>
      </SingleSliceCHOModule>
      <MRMCModule>
        <file_path_scores_diseased dataType = "char" value = "#0\output_temp\
            scores\reader_05_scores_diseased.txt"/>
        <file_path_scores_healthy dataType = "char" value = "#0\output_temp\
            scores\reader_05_scores_healthy.txt"/>
        <file_path_results dataType = "char" value = "#0\outputTest\
            shot1result_LG_multi.txt"/>
        <flag_estimate_biased dataType = "int" value = "0"/>
        <flag_verbose_required dataType = "int" value = "1"/>
      </MRMCModule>
    </REF>
  </TEMPLATE_MEVIC_SIMULATION>
</SuperXML>
```

2 Command

run_simulation.bat

In this file, the data are reorganized with batch commands for training and testing the different observers with 17 pairs of images.

ThenIn this command the VCT.exe is executed for training and testing the observers and running the MRMC analysis.

3 output

Several files are genrated the same outcome than in the previous section for each observer and then one file for the final results:

- shot1result_LG_multi.txt

4 Module: ”MRMCModule”

Modules for running the multi readers multi cases statistical analysis.

a Parameter: ”file_path_scores_diseased”

The file path with the scores for the diseased images.

b Parameter: ”file_path_scores_healthy”

The file path with the scores for the healthy images.

c Parameter: ””file_path_results

This is the file path of the final results.

LIST OF FIGURES

CONTENTS

BIBLIOGRAPHY

[DIC 98] DICOM . DICOM supplement 28: Grayscale Standard Display Function (GSDF)., 1998.

[Gal 06] GALLAS B.D. One-shot estimate of mrmc variance: Auc. Acad. Radiol., vol. 13, pp. 353–362, 2006.

[Mar 08] MARCHESSOUX C., KIMPE T. and BERT T. A virtual image chain for perceived and clinical image quality of medical display. JOURNAL OF DISPLAY TECHNOLOGY, vol. 4, pp. 356–368, 2008.

[Mar 12] MARCHESSOUX C., AVANAKI A., BAKIC P.R., KIMPE T.R.L. and MAIDMENT A.D.A. Effects of medical display luminance, contrast and temporal compensation on cho detection performanceat various browsing speeds and on digital breast tomosynthesis images. IWDM. 2012.

[MBC 13] MAIDMENT A.D., BAKIC P.R., CHUI J.H., AVANAKI A.N., MARCHESSOUX C., POKRAJAC D.D., ESPIG K.S., KIMPE T., XTHONA A., LAGO M.A. and SHANKLA V. The role of virtual clinical trials in preclinical testing of breast imaging systems. RSNA. 2013.

[Mye 87] MYERS K.J. and BARRETT H.H. Addition of a channel mechanism to the ideal-observer model. J. Opt. Soc. Am. A, vol. 4, n°12, pp. 2447–2457, 1987.

[Pla 09] PLATISA L., GOOSSENS B., VANSTEENKISTE E., BADANO A. and PHILIPS W. Channelized Hotelling Observers for Detection Tasks in Multi-Slice Images, October 2009.

[Tam 02] TAMURA N., TSUMURA N. and MIYAKE Y. Masking model for accurate colorimetric characterization of lcd. Proc. IS&T/SID 10 th Color Imaging Conference 2002. pp. 312–316, 2002.