

OCR

Generated by Doxygen 1.8.12



# Contents

<b>1</b>	<b>Optical Character Recognition</b>	<b>1</b>
<b>2</b>	<b>Class Index</b>	<b>3</b>
2.1	Class List . . . . .	3
<b>3</b>	<b>File Index</b>	<b>5</b>
3.1	File List . . . . .	5
<b>4</b>	<b>Class Documentation</b>	<b>7</b>
4.1	s_bitmap Struct Reference . . . . .	7
4.2	s_bitmapFileHeader Struct Reference . . . . .	7
4.3	s_bitmapInfoHeader Struct Reference . . . . .	8
4.4	s_color Struct Reference . . . . .	8
4.5	s_element Struct Reference . . . . .	8
4.5.1	Detailed Description . . . . .	8
4.5.2	Member Data Documentation . . . . .	9
4.5.2.1	next . . . . .	9
4.5.2.2	obj . . . . .	9
4.6	s_histogram Struct Reference . . . . .	9
4.6.1	Detailed Description . . . . .	9
4.6.2	Member Data Documentation . . . . .	9
4.6.2.1	dc . . . . .	9
4.6.2.2	deltaX . . . . .	10
4.6.2.3	deltaY . . . . .	10
4.6.2.4	tc . . . . .	10

4.6.2.5	x	10
4.6.2.6	y	10
4.7	s_network Struct Reference	10
4.7.1	Detailed Description	11
4.7.2	Member Data Documentation	11
4.7.2.1	delta	11
4.7.2.2	layers	11
4.7.2.3	nlayer	11
4.7.2.4	out	11
4.7.2.5	threshold	11
4.7.2.6	weight	11
4.8	s_queue Struct Reference	12
4.8.1	Detailed Description	12
4.8.2	Member Data Documentation	12
4.8.2.1	first	12
4.8.2.2	last	12
4.8.2.3	length	12
4.9	Zone Struct Reference	12
4.9.1	Detailed Description	13
<b>5</b>	<b>File Documentation</b>	<b>15</b>
5.1	bitmap.c File Reference	15
5.1.1	Detailed Description	16
5.1.2	Function Documentation	16
5.1.2.1	autoContrast()	16
5.1.2.2	binarize()	16
5.1.2.3	draw()	17
5.1.2.4	freeBitmap()	17
5.1.2.5	loadBmp()	17
5.1.2.6	newBitmap()	17
5.1.2.7	newColor()	18

5.1.2.8	resize()	18
5.1.2.9	rotate()	18
5.1.2.10	saveBmp()	18
5.2	bitmap.h File Reference	19
5.2.1	Detailed Description	19
5.2.2	Function Documentation	20
5.2.2.1	autoContrast()	20
5.2.2.2	binarize()	20
5.2.2.3	draw()	20
5.2.2.4	freeBitmap()	20
5.2.2.5	loadBmp()	21
5.2.2.6	newBitmap()	21
5.2.2.7	newColor()	21
5.2.2.8	resize()	21
5.2.2.9	rotate()	22
5.2.2.10	saveBmp()	22
5.3	detection.c File Reference	22
5.3.1	Detailed Description	23
5.3.2	Function Documentation	24
5.3.2.1	binerizeCopy()	24
5.3.2.2	checkClass()	25
5.3.2.3	colorRectangle()	25
5.3.2.4	cutBmp()	25
5.3.2.5	detectText()	26
5.3.2.6	heightTravel()	26
5.3.2.7	histoToImage()	26
5.3.2.8	letterAverage()	27
5.3.2.9	makeHistogram()	27
5.3.2.10	merge()	27
5.3.2.11	putColumnMarker()	28

5.3.2.12	putLineMarker()	28
5.3.2.13	rlsa()	28
5.3.2.14	segmentation()	29
5.3.2.15	textToHisto()	29
5.3.2.16	widthTravel()	30
5.4	detection.h File Reference	30
5.4.1	Detailed Description	30
5.4.2	Function Documentation	30
5.4.2.1	detectText()	30
5.4.2.2	segmentation()	31
5.5	graphical.c File Reference	31
5.5.1	Detailed Description	32
5.5.2	Function Documentation	32
5.5.2.1	a2i()	32
5.5.2.2	cbOpen()	33
5.5.2.3	fileChoose()	33
5.5.2.4	leaveDialog()	33
5.5.2.5	process()	33
5.5.2.6	rotation()	34
5.5.2.7	saveFile()	34
5.5.2.8	start()	34
5.6	graphical.h File Reference	35
5.6.1	Detailed Description	35
5.6.2	Function Documentation	36
5.6.2.1	a2i()	36
5.6.2.2	cbOpen()	37
5.6.2.3	fileChoose()	37
5.6.2.4	leaveDialog()	37
5.6.2.5	process()	37
5.6.2.6	rotation()	38

5.6.2.7	saveFile()	38
5.6.2.8	start()	38
5.7	learning.c File Reference	39
5.7.1	Detailed Description	39
5.7.2	Function Documentation	39
5.7.2.1	createResults()	39
5.7.2.2	createSamples()	40
5.7.2.3	freeSamples()	40
5.7.2.4	learn()	40
5.7.2.5	learning()	41
5.7.2.6	randomize()	41
5.7.2.7	shuffleSample()	41
5.8	main.c File Reference	42
5.8.1	Detailed Description	42
5.8.2	Function Documentation	42
5.8.2.1	main()	42
5.9	network.c File Reference	43
5.9.1	Detailed Description	43
5.9.2	Function Documentation	43
5.9.2.1	freeNetwork()	43
5.9.2.2	generateNetwork()	44
5.9.2.3	loadNetwork()	44
5.9.2.4	newNetwork()	44
5.9.2.5	saveNetwork()	44
5.10	network.h File Reference	45
5.10.1	Detailed Description	45
5.10.2	Function Documentation	45
5.10.2.1	freeNetwork()	45
5.10.2.2	generateNetwork()	46
5.10.2.3	loadNetwork()	46
5.10.2.4	newNetwork()	46
5.10.2.5	saveNetwork()	46





# Chapter 1

## Optical Character Recognition

This software can recognize text in images.

- Utilization :

- Graphical :

- `./main`

- Terminal :

- `./main 2 [img path]`

- `* Generating a new 'network.save' : './main 0'`

- `* Learning : './main 1 [learning file]'`



## Chapter 2

# Class Index

### 2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">s_bitmap</a>	.....	7
<a href="#">s_bitmapFileHeader</a>	.....	7
<a href="#">s_bitmapInfoHeader</a>	.....	8
<a href="#">s_color</a>	.....	8
<a href="#">s_element</a>		
Element of an queue	.....	8
<a href="#">s_histogram</a>		
All information on a text block is in	.....	9
<a href="#">s_network</a>		
The structure of a neural network	.....	10
<a href="#">s_queue</a>		
Queue	.....	12
<a href="#">Zone</a>		
Struct that contains all the info of the graphical interface	.....	12



## Chapter 3

# File Index

### 3.1 File List

Here is a list of all documented files with brief descriptions:

<a href="#">bitmap.c</a>	Structs and fonctions on bitmap . . . . .	15
<a href="#">bitmap.h</a>	Header of <a href="#">bitmap.c</a> . . . . .	19
<a href="#">detection.c</a>	Detect text and only it in a text . . . . .	22
<a href="#">detection.h</a>	The header of <a href="#">detection.c</a> . . . . .	30
<a href="#">graphical.c</a>	Create the graphical interface and call the function . . . . .	31
<a href="#">graphical.h</a>	Create the graphical interface and call the function . . . . .	35
<a href="#">learning.c</a>	All functions for learn and use a neural network . . . . .	39
<a href="#">learning.h</a>	. . . . .	??
<a href="#">main.c</a>	Dynamic storage . . . . .	42
<a href="#">network.c</a>	Create, detroy, load and save a neural network . . . . .	43
<a href="#">network.h</a>	The header of <a href="#">network.c</a> . . . . .	45
<a href="#">ocr.h</a>	. . . . .	??
<a href="#">queue.h</a>	. . . . .	??



## Chapter 4

# Class Documentation

### 4.1 s\_bitmap Struct Reference

#### Public Attributes

- unsigned **width**
- unsigned **height**
- [color](#) \* **content**

The documentation for this struct was generated from the following file:

- [bitmap.h](#)

### 4.2 s\_bitmapFileHeader Struct Reference

#### Public Attributes

- uint16\_t **bfType**
- uint32\_t **bfSize**
- uint16\_t **bfReserved1**
- uint16\_t **bfReserved2**
- uint32\_t **bfOffBytes**

The documentation for this struct was generated from the following file:

- [bitmap.c](#)

## 4.3 s\_bitmapInfoHeader Struct Reference

### Public Attributes

- uint32\_t **biSize**
- uint32\_t **biWidth**
- uint32\_t **biHeight**
- uint16\_t **biPlanes**
- uint16\_t **biBitCount**
- uint32\_t **biCompression**
- uint32\_t **biSizeImage**
- uint32\_t **biXPelsPerMeter**
- uint32\_t **biYPelsPerMeter**
- uint32\_t **biClrUsed**
- uint32\_t **biClrImportant**

The documentation for this struct was generated from the following file:

- [bitmap.c](#)

## 4.4 s\_color Struct Reference

### Public Attributes

- unsigned char **b**
- unsigned char **g**
- unsigned char **r**

The documentation for this struct was generated from the following file:

- [bitmap.h](#)

## 4.5 s\_element Struct Reference

is an element of an queue

```
#include <queue.h>
```

### Public Attributes

- void \* [obj](#)
- struct [s\\_element](#) \* [next](#)

### 4.5.1 Detailed Description

is an element of an queue



## 4.5.2 Member Data Documentation

### 4.5.2.1 next

```
struct s_element* s_element::next
```

is the next element of the queue

### 4.5.2.2 obj

```
void* s_element::obj
```

obj which is stock in the element

The documentation for this struct was generated from the following file:

- queue.h

## 4.6 s\_histogram Struct Reference

All information on a text block is in.

### Public Attributes

- unsigned [x](#)
- unsigned [y](#)
- unsigned [deltaX](#)
- unsigned [deltaY](#)
- unsigned [dc](#)
- unsigned [tc](#)

### 4.6.1 Detailed Description

All information on a text block is in.

## 4.6.2 Member Data Documentation

### 4.6.2.1 dc

```
unsigned s_histogram::dc
```

number of black pixel in the block

#### 4.6.2.2 deltaX

`unsigned s_histogram::deltaX`

width of the block

#### 4.6.2.3 deltaY

`unsigned s_histogram::deltaY`

height of the block

#### 4.6.2.4 tc

`unsigned s_histogram::tc`

number of dif between the original and the rlsa block

#### 4.6.2.5 x

`unsigned s_histogram::x`

min x pos of the block

#### 4.6.2.6 y

`unsigned s_histogram::y`

min y pos of the block

The documentation for this struct was generated from the following file:

- [detection.c](#)

## 4.7 s\_network Struct Reference

The structure of a neural network.

```
#include <network.h>
```

### Public Attributes

- unsigned [nblayer](#)
- unsigned \* [layers](#)
- float \*\* [threshold](#)
- float \*\* [out](#)
- float \*\* [delta](#)
- float \*\*\* [weight](#)

### 4.7.1 Detailed Description

The structure of a neural network.

### 4.7.2 Member Data Documentation

#### 4.7.2.1 delta

```
float** s_network::delta
```

The deltas of each neurone

#### 4.7.2.2 layers

```
unsigned* s_network::layers
```

The number of neurone for each layer

#### 4.7.2.3 nlayer

```
unsigned s_network::nlayer
```

The number of layer

#### 4.7.2.4 out

```
float** s_network::out
```

The activation value of each neurone

#### 4.7.2.5 threshold

```
float** s_network::threshold
```

The threshold or bias of the neural network

#### 4.7.2.6 weight

```
float*** s_network::weight
```

The weight of each neurone

The documentation for this struct was generated from the following file:

- [network.h](#)

## 4.8 s\_queue Struct Reference

is the queue

```
#include <queue.h>
```

### Public Attributes

- int [length](#)
- [element](#) \* [first](#)
- [element](#) \* [last](#)

### 4.8.1 Detailed Description

is the queue

### 4.8.2 Member Data Documentation

#### 4.8.2.1 first

```
element* s_queue::first
```

the first element of the queue

#### 4.8.2.2 last

```
element* s_queue::last
```

the last element of the queue

#### 4.8.2.3 length

```
int s_queue::length
```

the length of the queue

The documentation for this struct was generated from the following file:

- [queue.h](#)

## 4.9 Zone Struct Reference

Struct that contains all the info of the graphical interface.

```
#include <graphical.h>
```

## Public Attributes

- GtkWidget \* **image**
- GtkWidget \* **text**
- GtkWidget \* **pWindow**
- char \* **path**

### 4.9.1 Detailed Description

Struct that contains all the info of the graphical interface.

The documentation for this struct was generated from the following files:

- [graphical.c](#)
- [graphical.h](#)



## Chapter 5

# File Documentation

### 5.1 bitmap.c File Reference

Structs and fonctions on bitmap.

```
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <err.h>
#include <math.h>
#include "bitmap.h"
```

#### Classes

- struct [s\\_bitmapFileHeader](#)
- struct [s\\_bitmapInfoHeader](#)

#### Typedefs

- typedef struct [s\\_bitmapFileHeader](#) **bitmapFileHeader**
- typedef struct [s\\_bitmapInfoHeader](#) **bitmapInfoHeader**

#### Functions

- [color newColor](#) (unsigned char r, unsigned char g, unsigned char b)  
*Create a new color with RGB as argument.*
- [bitmap \\* newBitmap](#) (unsigned width, unsigned height, [color](#) \*content)  
*Create a new bitmap with width, height and and color array of it.*
- void [freeBitmap](#) ([bitmap](#) \*img)  
*free the bitmap*
- void [draw](#) ([bitmap](#) \*img)  
*Draw an image in console.*
- void [binarize](#) ([bitmap](#) \*img)  
*put the image in black and white*

- void `resize` (`bitmap` \*img)  
*resize an image in 16x16 pixels*
- `bitmap` \* `loadBmp` (char \*path)  
*Load a bmp file with a path in argument.*
- void `saveBmp` (char \*path, `bitmap` \*bmp)  
*save the bmp with the path name*
- void `autoContrast` (`bitmap` \*img)  
*equalize the histogram of the bitmap to raise the contrast*
- void `rotate` (`bitmap` \*img, double angle)  
*rotate the bitmap by angle degree*

### 5.1.1 Detailed Description

Structs and fonctions on bitmap.

#### Author

astain\_d and issarn\_t

#### Date

10/04/2016

### 5.1.2 Function Documentation

#### 5.1.2.1 `autoContrast()`

```
void autoContrast (
    bitmap * )
```

equalize the histogram of the bitmap to raise the contrast

#### Parameters

<i>img</i>	the bitmap to contrast
------------	------------------------

#### 5.1.2.2 `binarize()`

```
void binarize (
    bitmap * img )
```

put the image in black and white

#### Parameters

<i>img</i>	the image which will be binarize
------------	----------------------------------



### 5.1.2.3 draw()

```
void draw (
    bitmap * img )
```

Draw an image in console.

#### Parameters

<i>img</i>	the image which will be draw
------------	------------------------------

### 5.1.2.4 freeBitmap()

```
void freeBitmap (
    bitmap * img )
```

free the bitmap

#### Parameters

<i>img</i>	the bitmap to free
------------	--------------------

### 5.1.2.5 loadBmp()

```
bitmap* loadBmp (
    char * path )
```

Load a bmp file with a path in argument.

#### Parameters

<i>path</i>	is image path
-------------	---------------

### 5.1.2.6 newBitmap()

```
bitmap* newBitmap (
    unsigned width,
    unsigned height,
    color * content )
```

Create a new bitmap with width, height and and color array of it.

#### Parameters

<i>width</i>	is the with of the new bitmap
<i>height</i>	is the height of the new bitmap
<i>content</i>	is the full image

#### 5.1.2.7 newColor()

```
color newColor (
    unsigned char r,
    unsigned char g,
    unsigned char b )
```

Create a new color with RGB as argument.

##### Parameters

<i>r</i>	red component of the new color
<i>g</i>	green component of the new color
<i>b</i>	blue component of the new color

#### 5.1.2.8 resize()

```
void resize (
    bitmap * img )
```

resize an image in 16x16 pixels

img the image which will be resize

#### 5.1.2.9 rotate()

```
void rotate (
    bitmap * ,
    double )
```

rotate the bitmap by angle degree

##### Parameters

<i>img</i>	the bitmap to rotate
<i>angle</i>	the angle of the rotation, in degree

#### 5.1.2.10 saveBmp()

```
void saveBmp (
    char * ,
    bitmap * )
```

save the bmp with the path name

##### Parameters

<i>img</i>	the picture to save
<i>path</i>	the name for the saved picture

## 5.2 bitmap.h File Reference

header of [bitmap.c](#)

```
#include <stdio.h>
#include <stdlib.h>
```

### Classes

- struct [s\\_color](#)
- struct [s\\_bitmap](#)

### Typedefs

- typedef struct [s\\_color](#) **color**
- typedef struct [s\\_bitmap](#) **bitmap**
- typedef struct [s\\_bitmapFileHeader](#) **bitmapFileHeader**
- typedef struct [s\\_bitmapInfoHeader](#) **bitmapInfoHeader**

### Functions

- [color](#) **newColor** (unsigned char, unsigned char, unsigned char)  
*Create a new color with RGB as argument.*
- [bitmap \\*](#) **newBitmap** (unsigned, unsigned, [color \\*](#))  
*Create a new bitmap with width, height and and color array of it.*
- void **freeBitmap** ([bitmap \\*](#))  
*free the bitmap*
- void **draw** ([bitmap \\*](#))  
*Draw an image in console.*
- void **binarize** ([bitmap \\*](#))  
*put the image in black and white*
- void **resize** ([bitmap \\*](#))  
*resize an image in 16x16 pixels*
- [bitmap \\*](#) **loadBmp** (char \*)  
*Load a bmp file with a path in argument.*
- void **saveBmp** (char \*, [bitmap \\*](#))  
*save the bmp with the path name*
- void **autoContrast** ([bitmap \\*](#))  
*equalize the histogram of the bitmap to raise the contrast*
- void **rotate** ([bitmap \\*](#), double)  
*rotate the bitmap by angle degree*

### 5.2.1 Detailed Description

header of [bitmap.c](#)

#### Author

astain\_d and issarn\_t

#### Date

10/01/2016

## 5.2.2 Function Documentation

### 5.2.2.1 autoContrast()

```
void autoContrast (
    bitmap * )
```

equalize the histogram of the bitmap to raise the contrast

#### Parameters

<i>img</i>	the bitmap to contrast
------------	------------------------

### 5.2.2.2 binarize()

```
void binarize (
    bitmap * img )
```

put the image in black and white

#### Parameters

<i>img</i>	the image which will be binarize
------------	----------------------------------

### 5.2.2.3 draw()

```
void draw (
    bitmap * img )
```

Draw an image in console.

#### Parameters

<i>img</i>	the image which will be draw
------------	------------------------------

### 5.2.2.4 freeBitmap()

```
void freeBitmap (
    bitmap * img )
```

free the bitmap

#### Parameters

<i>img</i>	the bitmap to free
------------	--------------------

### 5.2.2.5 loadBmp()

```
bitmap* loadBmp (
    char * path )
```

Load a bmp file with a path in argument.

#### Parameters

<i>path</i>	is image path
-------------	---------------

### 5.2.2.6 newBitmap()

```
bitmap* newBitmap (
    unsigned width,
    unsigned height,
    color * content )
```

Create a new bitmap with width, height and and color array of it.

#### Parameters

<i>width</i>	is the with of the new bitmap
<i>height</i>	is the height of the new bitmap
<i>content</i>	is the full image

### 5.2.2.7 newColor()

```
color newColor (
    unsigned char r,
    unsigned char g,
    unsigned char b )
```

Create a new color with RGB as argument.

#### Parameters

<i>r</i>	red component of the new color
<i>g</i>	green component of the new color
<i>b</i>	blue component of the new color

### 5.2.2.8 resize()

```
void resize (
    bitmap * img )
```

resize an image in 16x16 pixels

img the image which will be resize

### 5.2.2.9 rotate()

```
void rotate (
    bitmap * ,
    double )
```

rotate the bitmap by angle degree

#### Parameters

<i>img</i>	the bitmap to rotate
<i>angle</i>	the angle of the rotation, in degree

### 5.2.2.10 saveBmp()

```
void saveBmp (
    char * ,
    bitmap * )
```

save the bmp with the path name

#### Parameters

<i>img</i>	the picture to save
<i>path</i>	the name for the saved picture

## 5.3 detection.c File Reference

Detect text and only it in a text.

```
#include <err.h>
#include "bitmap.h"
#include "queue.h"
#include "detection.h"
```

### Classes

- struct [s\\_histogram](#)  
*All information on a text block is in.*

### Typedefs

- typedef struct [s\\_histogram](#) **histogram**

## Functions

- **bitmap \* binerizeCopy** (**bitmap** \*src)  
*binerize the copy of an image*
- void **putLineMarker** (**bitmap** \*img, char \*array, int pos, unsigned width)  
*Put a marker for each line with a letter.*
- void **putColumnMarker** (**bitmap** \*img, unsigned min, unsigned max, char \*array, int pos, unsigned width)  
*Put a marker for each column with a letter.*
- **bitmap \* cutBmp** (**bitmap** \*img, unsigned x, unsigned y, unsigned width, unsigned height)  
*Put a marker for each column with a letter.*
- float **letterAverage** (char \*columnMarker, unsigned width)  
*find the length average of letters for one line*
- void **colorRectangle** (**bitmap** \*src, int x, int y, unsigned width, unsigned height)  
*frame a rectangle*
- void **segmentation** (**bitmap** \*img, size\_t \*nbCharacter, size\_t \*nbLetter, **queue** \*q, **bitmap** \*cutedImage, int pos)  
*Create a queue with all letter in a bitmap.*
- **bitmap \* widthTravel** (**bitmap** \*src)  
*create a new bitmap with width offset*
- **bitmap \* heightTravel** (**bitmap** \*src)  
*create a new bitmap with height offset*
- **bitmap \* merge** (**bitmap** \*src1, **bitmap** \*src2)  
*Fusion two bitmap in one other.*
- char **checkClass** (**histogram** \*histo, float \*hm)  
*check is a block is a block of text*
- void **columnCut** (**bitmap** \*src, **queue** \*imgQueue, **queue** \*posQueue)
- void **makeHistogram** (**bitmap** \*bmp, **bitmap** \*original, unsigned x, unsigned y, **queue** \*q)  
*make the histogram of one block and add it in a queue*
- void **textToHisto** (**queue** \*histoQueue, **bitmap** \*src, **bitmap** \*original, float \*hm, unsigned pos)  
*Return an image without black line et draw.*
- void **histoToImage** (**bitmap** \*final, **bitmap** \*src, **queue** \*histoQueue, float \*hm)  
*make image from a queue of histogram*
- **bitmap \* rlsa** (**bitmap** \*src, **queue** \*imgQueue, **queue** \*posQueue)  
*return a image without line and draw*
- **queue \* detectText** (**bitmap** \*src, size\_t \*nbCharacter, size\_t \*nbLetter)  
*return a queue which contain all letter*

### 5.3.1 Detailed Description

Detect text and only it in a text.

#### Author

issarn\_t

#### Date

09/17/2016

## 5.3.2 Function Documentation

### 5.3.2.1 binerizeCopy()

```
bitmap* binerizeCopy (  
    bitmap * src )
```

binerize the copy of an image



## Parameters

<i>src</i>	is the image to copy
------------	----------------------

## 5.3.2.2 checkClass()

```
char checkClass (
    histogram * histo,
    float * hm )
```

check is a block is a block of text

## Parameters

<i>histo</i>	is the histogram of a block
<i>hm</i>	is the average height of blocks

## 5.3.2.3 colorRectangle()

```
void colorRectangle (
    bitmap * src,
    int x,
    int y,
    unsigned width,
    unsigned height )
```

frame a rectangle

## Parameters

<i>img</i>	the full image
<i>min</i>	x min
<i>max</i>	y min
<i>width</i>	the width of the rectangle
<i>height</i>	the height of the rectangle

## 5.3.2.4 cutBmp()

```
bitmap* cutBmp (
    bitmap * img,
    unsigned x,
    unsigned y,
    unsigned width,
    unsigned height )
```

Put a marker for each column with a letter.

**Parameters**

<i>img</i>	the full image
<i>min</i>	x min
<i>max</i>	y min
<i>width</i>	the width of the cuted Image
<i>height</i>	the height of the image

**5.3.2.5 detectText()**

```
queue* detectText (
    bitmap * src,
    size_t * nbCharacter,
    size_t * nbLetter )
```

return a queue which contain all letter

**Parameters**

<i>src</i>	is the original image
<i>nbCharacter</i>	is the number of character
<i>nbLetter</i>	is the number of letter

**5.3.2.6 heightTravel()**

```
bitmap* heightTravel (
    bitmap * src )
```

create a new bitmap with height offset

**Parameters**

<i>src</i>	the orginal bitmap
------------	--------------------

**5.3.2.7 histoToImage()**

```
void histoToImage (
    bitmap * final,
    bitmap * src,
    queue * histoQueue,
    float * hm )
```

make image from a queue of histogram

**Parameters**

<i>final</i>	is the image that we modify
--------------	-----------------------------

## Parameters

<i>src</i>	is the original image
<i>histoQueue</i>	is the queue with contain all histograms
<i>hm</i>	is the average height of block in the image

## 5.3.2.8 letterAverage()

```
float letterAverage (
    char * columnMarker,
    unsigned width )
```

find the length average of letters for one line

## Parameters

<i>columnMarker</i>	array where there are marker for each pixel
<i>width</i>	is the width of the columnMarker

## 5.3.2.9 makeHistogram()

```
void makeHistogram (
    bitmap * bmp,
    bitmap * original,
    unsigned x,
    unsigned y,
    queue * q )
```

make the histogram of one block and add it in a queue

## Parameters

<i>bmp</i>	is the text block
<i>original</i>	is copy binarize of the original image
<i>x</i>	high left corner of the block on the copy in abscisse
<i>y</i>	high left corner of the block on the copy in ordonate
<i>q</i>	is the queue where we add the hsitogram

## 5.3.2.10 merge()

```
bitmap* merge (
    bitmap * src1,
    bitmap * src2 )
```

Fusion two bitmap in one other.

**Parameters**

<i>src1</i>	the first bitmap
<i>src2</i>	the second bitmap

**5.3.2.11 putColumnMarker()**

```
void putColumnMarker (
    bitmap * img,
    unsigned min,
    unsigned max,
    char * array,
    int pos,
    unsigned width )
```

Put a marker for each column with a letter.

**Parameters**

<i>img</i>	one line of the full image
<i>min</i>	x min
<i>max</i>	x max
<i>array</i>	where marker is put in function of the img
<i>pos</i>	the x min pos in the original image
<i>width</i>	the width of the cuted Image

**5.3.2.12 putLineMarker()**

```
void putLineMarker (
    bitmap * img,
    char * array,
    int pos,
    unsigned width )
```

Put a marker for each line with a letter.

**Parameters**

<i>img</i>	the full image
<i>array</i>	where marker is put in function of the img
<i>pos</i>	the x min pos in the original image
<i>width</i>	the width of the cuted Image

**5.3.2.13 rlsa()**

```
bitmap* rlsa (
    bitmap * src,
```

```
queue * imgQueue,  
queue * posQueue )
```

return a image without line and draw

#### Parameters

<i>src</i>	is the original image
<i>imgQueue</i>	is the queue where we stock images
<i>posQueue</i>	is the queue where we stock x min position of each images

#### 5.3.2.14 segmentation()

```
void segmentation (  
    bitmap * img,  
    size_t * nbCharacter,  
    size_t * nbLetter,  
    queue * q,  
    bitmap * cutedImage,  
    int pos )
```

Create a queue with all letter in a bitmap.

#### Parameters

<i>img</i>	the full image
<i>nbCharacter</i>	the number of character in the image
<i>nbLetter</i>	the number of letter int the image
<i>q</i>	is the queue where is stocked all letter
<i>cutedImage</i>	is the the image that we use
<i>pos</i>	is the x min

#### 5.3.2.15 textToHisto()

```
void textToHisto (  
    queue * histoQueue,  
    bitmap * src,  
    bitmap * original,  
    float * hm,  
    unsigned pos )
```

Return an image without black line et draw.

#### Parameters

<i>histoQueue</i>	is the queue with all histogram of a image
<i>src</i>	the cuted image
<i>original</i>	is the original image
<i>hm</i>	is the height average of text block
<i>pos</i>	is the x min pos

### 5.3.2.16 widthTravel()

```
bitmap* widthTravel (
    bitmap * src )
```

create a new bitmap with width offset

#### Parameters

<code>src</code>	the original bitmap
------------------	---------------------

## 5.4 detection.h File Reference

The header of [detection.c](#).

```
#include <stdio.h>
#include <stdlib.h>
#include "bitmap.h"
#include "queue.h"
```

### Functions

- [queue](#) \* [detectText](#) ([bitmap](#) \*, [size\\_t](#) \*, [size\\_t](#) \*)  
*return a queue which contain all letter*
- void [segmentation](#) ([bitmap](#) \*, [size\\_t](#) \*, [size\\_t](#) \*, [queue](#) \*, [bitmap](#) \*, int)  
*Create a queue with all letter in a bitmap.*

### 5.4.1 Detailed Description

The header of [detection.c](#).

#### Author

issarn\_t

#### Date

09/17/2016

### 5.4.2 Function Documentation

#### 5.4.2.1 detectText()

```
queue* detectText (
    bitmap * src,
    size_t * nbCharacter,
    size_t * nbLetter )
```

return a queue which contain all letter

## Parameters

<i>src</i>	is the original image
<i>nbCharacter</i>	is the number of character
<i>nbLetter</i>	is the number of letter

## 5.4.2.2 segmentation()

```
void segmentation (
    bitmap * img,
    size_t * nbCharacter,
    size_t * nbLetter,
    queue * q,
    bitmap * cutedImage,
    int pos )
```

Create a queue with all letter in a bitmap.

## Parameters

<i>img</i>	the full image
<i>nbCharacter</i>	the number of character in the image
<i>nbLetter</i>	the number of letter int the image
<i>q</i>	is the queue where is stocked all letter
<i>cutedImage</i>	is the the image that we use
<i>pos</i>	is the x min

## 5.5 graphical.c File Reference

Create the graphical interface and call the function.

```
#include <gdk-pixbuf/gdk-pixbuf.h>
#include <stdio.h>
#include <stdlib.h>
#include <err.h>
#include <time.h>
#include <unistd.h>
#include <gtk/gtk.h>
#include "ocr.h"
#include "bitmap.h"
#include "detection.h"
#include "queue.h"
#include "network.h"
#include "learning.h"
```

## Classes

- struct [Zone](#)

*Struct that contains all the info of the graphical interface.*

## Functions

- int [a2i](#) (const char \*s)  
*Convert a constant string to an integer.*
- void [rotation](#) (GtkWidget \*window, gpointer data)  
*Launch the rotation on the current image.*
- void [process](#) (GtkWidget \*window, gpointer data)  
*Process the optical recognition of character algorithm on the chosen picture.*
- void [saveFile](#) (GtkWidget \*window, gpointer data)  
*Save the file at the location chosen by the user.*
- void [fileChoose](#) (GtkWidget \*widget, gpointer data)  
*Return the path of a file chosen by the user.*
- void [cbOpen](#) (GtkWidget \*widget, gpointer data)  
*Open the file chosen by the user. Must be a image type file.*
- void [leaveDialog](#) (GtkWidget \*widget, gpointer data)  
*Create a dialog that make the user confirm if he really wants to quit the program.*
- int [start](#) (int argc, char \*\*argv)  
*Create the parent window, the menus, the zones, ...*

### 5.5.1 Detailed Description

Create the graphical interface and call the function.

#### Author

decret\_t

#### Date

October 19th 2016

Generate the graphical user interface (GUI) and link all the buttons with the functions to launch.

### 5.5.2 Function Documentation

#### 5.5.2.1 a2i()

```
int a2i (
    const char * s )
```

Convert a constant string to an integer.

#### Parameters

s	The constant string to convert
---	--------------------------------



### 5.5.2.2 cbOpen()

```
void cbOpen (
    GtkWidget * widget,
    gpointer data )
```

Open the file chosen by the user. Must be a image type file.

#### Parameters

<i>widget</i>	The parent window
<i>data</i>	The pointer to the created zone struct

### 5.5.2.3 fileChoose()

```
void fileChoose (
    GtkWidget * widget,
    gpointer data )
```

Return the path of a file chosen by the user.

#### Parameters

<i>widget</i>	The parent window
<i>data</i>	The pointer to the created zone struct

### 5.5.2.4 leaveDialog()

```
void leaveDialog (
    GtkWidget * widget,
    gpointer data )
```

Create a dialog that make the user confirm if he really wants to quit the program.

#### Parameters

<i>widget</i>	The parent window
<i>data</i>	The pointer to the created zone struct

### 5.5.2.5 process()

```
void process (
    GtkWidget * window,
    gpointer data )
```

Process the optical recognition of character algorithm on the chosen picture.

**Parameters**

<i>window</i>	The parent window
<i>data</i>	The pointer to the created zone struct

**5.5.2.6 rotation()**

```
void rotation (
    GtkWidget * window,
    gpointer data )
```

Launch the rotation on the current image.

**Parameters**

<i>window</i>	The current parent window
<i>data</i>	The pointer to the <a href="#">Zone</a>

**5.5.2.7 saveFile()**

```
void saveFile (
    GtkWidget * window,
    gpointer data )
```

Save the file at the location chosen by the user.

**Parameters**

<i>window</i>	The parent window
<i>data</i>	The pointer to the created zone struct

**5.5.2.8 start()**

```
int start (
    int argc,
    char ** argv )
```

Create the parent window, the menus, the zones, ...

**Parameters**

<i>argc</i>	The argc parameter of the main fuction
<i>argv</i>	The argv parameter of the main fuction

## 5.6 graphical.h File Reference

Create the graphical interface and call the function.

```
#include "ocr.h"
#include <gtk/gtk.h>
#include "bitmap.h"
#include "detection.h"
#include "queue.h"
#include "network.h"
```

### Classes

- struct [Zone](#)  
*Struct that contains all the info of the graphical interface.*

### Functions

- double [a2i](#) (const char)  
*Convert a constant string to a double.*
- void [rotation](#) (GtkWidget \*, gpointer)  
*Launch the rotation on the current image.*
- void [process](#) (GtkWidget \*, gpointer)  
*Process the optical recognition of character algorithm on the chosen picture.*
- void [saveFile](#) (GtkWidget \*, gpointer)  
*Save the file at the location chosen by the user.*
- void [fileChoose](#) (GtkWidget \*, gpointer)  
*Return the path of a file chosen by the user.*
- void [cbOpen](#) (GtkWidget \*, gpointer)  
*Open the file chosen by the user. Must be a image type file.*
- void [leaveDialog](#) (GtkWidget \*, gpointer)  
*Create a dialog that make the user confirm if he really wants to quit the program.*
- int [start](#) (int, char \*\*)  
*Create the parent window, the menus, the zones, ...*

#### 5.6.1 Detailed Description

Create the graphical interface and call the function.

##### Author

decret\_t

##### Date

October 19th 2016

Generate the graphical user interface (GUI) and link all the buttons with the functions to launch.

## 5.6.2 Function Documentation

### 5.6.2.1 a2i()

```
double a2i (  
    const char  )
```

Convert a constant string to a double.

**Parameters**

<i>s</i>	The constant string to convert
----------	--------------------------------

**5.6.2.2   cbOpen()**

```
void cbOpen (
    GtkWidget * widget,
    gpointer data )
```

Open the file chosen by the user. Must be a image type file.

**Parameters**

<i>widget</i>	The parent window
<i>data</i>	The pointer to the created zone struct

**5.6.2.3   fileChoose()**

```
void fileChoose (
    GtkWidget * widget,
    gpointer data )
```

Return the path of a file chosen by the user.

**Parameters**

<i>widget</i>	The parent window
<i>data</i>	The pointer to the created zone struct

**5.6.2.4   leaveDialog()**

```
void leaveDialog (
    GtkWidget * widget,
    gpointer data )
```

Create a dialog that make the user confirm if he really wants to quit the program.

**Parameters**

<i>widget</i>	The parent window
<i>data</i>	The pointer to the created zone struct

**5.6.2.5   process()**

```
void process (
```

```
GtkWidget * window,
gpointer data )
```

Process the optical recognition of character algorithm on the chosen picture.

#### Parameters

<i>window</i>	The parent window
<i>data</i>	The pointer to the created zone struct

#### 5.6.2.6 rotation()

```
void rotation (
    GtkWidget * window,
    gpointer data )
```

Launch the rotation on the current image.

#### Parameters

<i>window</i>	The current parent window
<i>data</i>	The pointer to the <a href="#">Zone</a>

#### 5.6.2.7 saveFile()

```
void saveFile (
    GtkWidget * window,
    gpointer data )
```

Save the file at the location chosen by the user.

#### Parameters

<i>window</i>	The parent window
<i>data</i>	The pointer to the created zone struct

#### 5.6.2.8 start()

```
int start (
    int argc,
    char ** argv )
```

Create the parent window, the menus, the zones, ...

#### Parameters

<i>argc</i>	The argc parameter of the main fuction
<i>argv</i>	The argv parameter of the main fuction

## 5.7 learning.c File Reference

All functions for learn and use a neural network.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>
#include <err.h>
#include "bitmap.h"
#include "detection.h"
#include "network.h"
#include "ocr.h"
#include "queue.h"
#include "learning.h"
```

### Functions

- void [shuffleSample](#) (float \*\*samples, float \*\*results, unsigned length)  
*suffle the samples and results table*
- void [randomize](#) ([network](#) \*n, float r)  
*randomize the weight and thresholds of a network*
- float [learn](#) ([network](#) \*n, float \*\*samples, float \*\*results, unsigned nbSample, float speed, size\_t sizeBatch)  
*entrain a neural network with a batch of samples*
- void [learning](#) (char \*learnFiles[], size\_t nbFile)  
*learn to the network "network.save"*
- int [createSamples](#) ([queue](#) \*text, float \*\*samples)  
*fill the samples with a segmented image and return the size*
- float \*\* [createResults](#) (char text[], int nbSample)  
*Create the results of samples.*
- void [freeSamples](#) (float \*\*samples, int nbSample)  
*free all components of a sample*

### 5.7.1 Detailed Description

All functions for learn and use a neural network.

The header of [learning.c](#).

Author

amsall\_f

Date

09/17/2016

### 5.7.2 Function Documentation

#### 5.7.2.1 createResults()

```
float** createResults (
    char text[],
    int nbSample )
```

Create the results of samples.

**Parameters**

<i>text</i>	the expected text
<i>nbsample</i>	the length of the text
<i>nbOutput</i>	the number of possible outputs

**5.7.2.2 createSamples()**

```
int createSamples (
    queue * text,
    float ** samples )
```

fill the samples with a segmented image and return the size

**Parameters**

<i>text</i>	the image segmented
<i>samples</i>	the table of sample to fill

**5.7.2.3 freeSamples()**

```
void freeSamples (
    float ** samples,
    int nbSample )
```

free all components of a sample

**Parameters**

<i>samples</i>	the inputs or outputs of a sample
<i>nbSample</i>	the number of character in the sample

**5.7.2.4 learn()**

```
float learn (
    network * n,
    float ** samples,
    float ** results,
    unsigned nbSample,
    float speed,
    size_t sizeBatch )
```

entrain a neural network with a batch of samples

**Parameters**

<i>n</i>	the neural network
<i>samples</i>	the inputs of samples



## Parameters

<i>results</i>	the outputs of samples
<i>nbSample</i>	the number of samples

## 5.7.2.5 learning()

```
void learning (
    char * learnFiles[],
    size_t nbFile )
```

learn to the network "network.save"

## Parameters

<i>learnFiles</i>	the table of path to learn files
<i>nbFile</i>	the number of file

## 5.7.2.6 randomize()

```
void randomize (
    network * n,
    float r )
```

randomize the weight and thresholds of a network

## Parameters

<i>n</i>	the network
<i>r</i>	the ratio of the randomization

## 5.7.2.7 shuffleSample()

```
void shuffleSample (
    float ** samples,
    float ** results,
    unsigned length )
```

shuffle the samples and results table

## Parameters

<i>samples</i>	the table of samples
<i>result</i>	the table of results
<i>length</i>	the length of both table

## 5.8 main.c File Reference

dynamic storage

```
#include <stdio.h>
#include <stdlib.h>
#include <err.h>
#include <time.h>
#include "ocr.h"
#include "detection.h"
#include "learning.h"
#include "graphical.h"
```

### Functions

- `int main (int argc, char *argv[ ])`  
*the main function*

#### 5.8.1 Detailed Description

dynamic storage

Author

issarn\_t

Date

09/29/2016

#### 5.8.2 Function Documentation

##### 5.8.2.1 main()

```
int main (
    int argc,
    char * argv[ ] )
```

the main function

Parameters

<i>argc</i>	the number of params
<i>argv</i>	a table of string

## 5.9 network.c File Reference

Create, detroy, load and save a neural network.

```
#include <stdio.h>
#include <stdlib.h>
#include <err.h>
#include <math.h>
#include "network.h"
```

### Functions

- `network * newNetwork` (unsigned nlayer, unsigned \*layers)  
*create a new neural network*
- void `freeNetwork` (`network *n`)  
*free all components of a network*
- `network * loadNetwork` (char \*path)  
*load a neural network from a save*
- void `saveNetwork` (char \*path, `network *n`)  
*save weights anf thresholds of a neural network*
- void `generateNetwork` ()  
*generate a network of three layer*

### 5.9.1 Detailed Description

Create, detroy, load and save a neural network.

#### Author

amsall\_f and astain\_d

#### Date

09/17/2016

### 5.9.2 Function Documentation

#### 5.9.2.1 freeNetwork()

```
void freeNetwork (
    network * n )
```

free all components of a network

#### Parameters

<i>n</i>	the neural network
----------	--------------------

### 5.9.2.2 generateNetwork()

```
void generateNetwork ( )
```

generate a network of three layer

#### Parameters

<i>c1</i>	the number of neurone for the first layer
<i>c1</i>	the number of neurone for the second layer
<i>c1</i>	the number of neurone for the third layer

### 5.9.2.3 loadNetwork()

```
network* loadNetwork (
    char * path )
```

load a neural network from a save

#### Parameters

<i>path</i>	location of the save
-------------	----------------------

### 5.9.2.4 newNetwork()

```
network* newNetwork (
    unsigned nlayer,
    unsigned * layers )
```

create a new neural network

#### Parameters

<i>nlayer</i>	the number of layers
<i>layers</i>	a table of the number of neurone in each layer

### 5.9.2.5 saveNetwork()

```
void saveNetwork (
    char * path,
    network * n )
```

save weights and thresholds of a neural network

#### Parameters

<i>path</i>	location of the new file
<i>n</i>	the neural network to save

## 5.10 network.h File Reference

The header of [network.c](#).

```
#include <stdio.h>
#include <stdlib.h>
```

### Classes

- struct [s\\_network](#)  
*The structure of a neural network.*

### Typedefs

- typedef struct [s\\_network](#) **network**

### Functions

- [network](#) \* [newNetwork](#) (unsigned nlayer, unsigned \*layers)  
*create a new neural network*
- void [freeNetwork](#) ([network](#) \*n)  
*free all components of a network*
- [network](#) \* [loadNetwork](#) (char \*path)  
*load a neural network from a save*
- void [saveNetwork](#) (char \*path, [network](#) \*n)  
*save weights and thresholds of a neural network*
- void [generateNetwork](#) ()  
*generate a network of three layer*

#### 5.10.1 Detailed Description

The header of [network.c](#).

##### Author

amsall\_f and astain\_d

##### Date

09/17/2016

#### 5.10.2 Function Documentation

##### 5.10.2.1 freeNetwork()

```
void freeNetwork (
    network * n )
```

free all components of a network

**Parameters**

<i>n</i>	the neural network
----------	--------------------

**5.10.2.2 generateNetwork()**

```
void generateNetwork ( )
```

generate a network of three layer

**Parameters**

<i>c1</i>	the number of neurone for the first layer
<i>c1</i>	the number of neurone for the second layer
<i>c1</i>	the number of neurone for the third layer

**5.10.2.3 loadNetwork()**

```
network* loadNetwork (
    char * path )
```

load a neural network from a save

**Parameters**

<i>path</i>	location of the save
-------------	----------------------

**5.10.2.4 newNetwork()**

```
network* newNetwork (
    unsigned nlayer,
    unsigned * layers )
```

create a new neural network

**Parameters**

<i>nlayer</i>	the number of layers
<i>layers</i>	a table of the number of neurone in each layer

**5.10.2.5 saveNetwork()**

```
void saveNetwork (
    char * path,
    network * n )
```

save weights and thresholds of a neural network

**Parameters**

<i>path</i>	location of the new file
<i>n</i>	the neural network to save



# Index

- a2i
  - graphical.c, [32](#)
  - graphical.h, [36](#)
- autoContrast
  - bitmap.c, [16](#)
  - bitmap.h, [20](#)
- binarize
  - bitmap.c, [16](#)
  - bitmap.h, [20](#)
- binerizeCopy
  - detection.c, [24](#)
- bitmap.c, [15](#)
  - autoContrast, [16](#)
  - binarize, [16](#)
  - draw, [16](#)
  - freeBitmap, [17](#)
  - loadBmp, [17](#)
  - newBitmap, [17](#)
  - newColor, [18](#)
  - resize, [18](#)
  - rotate, [18](#)
  - saveBmp, [18](#)
- bitmap.h, [19](#)
  - autoContrast, [20](#)
  - binarize, [20](#)
  - draw, [20](#)
  - freeBitmap, [20](#)
  - loadBmp, [20](#)
  - newBitmap, [21](#)
  - newColor, [21](#)
  - resize, [21](#)
  - rotate, [21](#)
  - saveBmp, [22](#)
- cbOpen
  - graphical.c, [32](#)
  - graphical.h, [37](#)
- checkClass
  - detection.c, [25](#)
- colorRectangle
  - detection.c, [25](#)
- createResults
  - learning.c, [39](#)
- createSamples
  - learning.c, [40](#)
- cutBmp
  - detection.c, [25](#)
- dc
  - s\_histogram, [9](#)
- delta
  - s\_network, [11](#)
- deltaX
  - s\_histogram, [9](#)
- deltaY
  - s\_histogram, [10](#)
- detectText
  - detection.c, [26](#)
  - detection.h, [30](#)
- detection.c, [22](#)
  - binerizeCopy, [24](#)
  - checkClass, [25](#)
  - colorRectangle, [25](#)
  - cutBmp, [25](#)
  - detectText, [26](#)
  - heightTravel, [26](#)
  - histoToImage, [26](#)
  - letterAverage, [27](#)
  - makeHistogram, [27](#)
  - merge, [27](#)
  - putColumnMarker, [28](#)
  - putLineMarker, [28](#)
  - rlsa, [28](#)
  - segmentation, [29](#)
  - textToHisto, [29](#)
  - widthTravel, [30](#)
- detection.h, [30](#)
  - detectText, [30](#)
  - segmentation, [31](#)
- draw
  - bitmap.c, [16](#)
  - bitmap.h, [20](#)
- fileChoose
  - graphical.c, [33](#)
  - graphical.h, [37](#)
- first
  - s\_queue, [12](#)
- freeBitmap
  - bitmap.c, [17](#)
  - bitmap.h, [20](#)
- freeNetwork
  - network.c, [43](#)
  - network.h, [45](#)
- freeSamples
  - learning.c, [40](#)
- generateNetwork
  - network.c, [44](#)

- network.h, 46
- graphical.c, 31
  - a2i, 32
  - cbOpen, 32
  - fileChoose, 33
  - leaveDialog, 33
  - process, 33
  - rotation, 34
  - saveFile, 34
  - start, 34
- graphical.h, 35
  - a2i, 36
  - cbOpen, 37
  - fileChoose, 37
  - leaveDialog, 37
  - process, 37
  - rotation, 38
  - saveFile, 38
  - start, 38
- heightTravel
  - detection.c, 26
- histoTolmage
  - detection.c, 26
- last
  - s\_queue, 12
- layers
  - s\_network, 11
- learn
  - learning.c, 40
- learning
  - learning.c, 41
- learning.c, 39
  - createResults, 39
  - createSamples, 40
  - freeSamples, 40
  - learn, 40
  - learning, 41
  - randomize, 41
  - shuffleSample, 41
- leaveDialog
  - graphical.c, 33
  - graphical.h, 37
- length
  - s\_queue, 12
- letterAverage
  - detection.c, 27
- loadBmp
  - bitmap.c, 17
  - bitmap.h, 20
- loadNetwork
  - network.c, 44
  - network.h, 46
- main
  - main.c, 42
- main.c, 42
  - main, 42
- makeHistogram
  - detection.c, 27
- merge
  - detection.c, 27
- nblayer
  - s\_network, 11
- network.c, 43
  - freeNetwork, 43
  - generateNetwork, 44
  - loadNetwork, 44
  - newNetwork, 44
  - saveNetwork, 44
- network.h, 45
  - freeNetwork, 45
  - generateNetwork, 46
  - loadNetwork, 46
  - newNetwork, 46
  - saveNetwork, 46
- newBitmap
  - bitmap.c, 17
  - bitmap.h, 21
- newColor
  - bitmap.c, 18
  - bitmap.h, 21
- newNetwork
  - network.c, 44
  - network.h, 46
- next
  - s\_element, 9
- obj
  - s\_element, 9
- out
  - s\_network, 11
- process
  - graphical.c, 33
  - graphical.h, 37
- putColumnMarker
  - detection.c, 28
- putLineMarker
  - detection.c, 28
- randomize
  - learning.c, 41
- resize
  - bitmap.c, 18
  - bitmap.h, 21
- rlsa
  - detection.c, 28
- rotate
  - bitmap.c, 18
  - bitmap.h, 21
- rotation
  - graphical.c, 34
  - graphical.h, 38
- s\_bitmap, 7

s\_bitmapFileHeader, [7](#)  
s\_bitmapInfoHeader, [8](#)  
s\_color, [8](#)  
s\_element, [8](#)  
    next, [9](#)  
    obj, [9](#)  
s\_histogram, [9](#)  
    dc, [9](#)  
    deltaX, [9](#)  
    deltaY, [10](#)  
    tc, [10](#)  
    x, [10](#)  
    y, [10](#)  
s\_network, [10](#)  
    delta, [11](#)  
    layers, [11](#)  
    nblayer, [11](#)  
    out, [11](#)  
    threshold, [11](#)  
    weight, [11](#)  
s\_queue, [12](#)  
    first, [12](#)  
    last, [12](#)  
    length, [12](#)  
saveBmp  
    bitmap.c, [18](#)  
    bitmap.h, [22](#)  
saveFile  
    graphical.c, [34](#)  
    graphical.h, [38](#)  
saveNetwork  
    network.c, [44](#)  
    network.h, [46](#)  
segmentation  
    detection.c, [29](#)  
    detection.h, [31](#)  
shuffleSample  
    learning.c, [41](#)  
start  
    graphical.c, [34](#)  
    graphical.h, [38](#)  
  
tc  
    s\_histogram, [10](#)  
textToHisto  
    detection.c, [29](#)  
threshold  
    s\_network, [11](#)  
  
weight  
    s\_network, [11](#)  
widthTravel  
    detection.c, [30](#)  
  
x  
    s\_histogram, [10](#)  
  
y  
    s\_histogram, [10](#)  
Zone, [12](#)