# OCR

Generated by Doxygen 1.8.12

# Contents

# Chapter 1

# Optical Character Recognition

This software car 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:

# Chapter 3

# File Index

## 3.1 File List

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

# 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 nblayer

```
unsigned s_network::nblayer
```

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 fisrt 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**

| | |
|---|---|
| *img* | the bitmap to contrast |

#### 5.1.2.2 binarize()

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

put the image in black and white

**Parameters**

| | |
|---|---|
| *img* | the image which will be binarize |

#### 5.1.2.3 draw()

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

Draw an image in console.

**Parameters**

| | |
|---|---|
| *img* | the image which will be draw |

#### 5.1.2.4 freeBitmap()

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

free the bitmap

**Parameters**

| | |
|---|---|
| *img* | the bitmap to free |

#### 5.1.2.5 loadBmp()

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

Load a bmp file with a path in argument.

**Parameters**

| | |
|---|---|
| *path* | 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**

| | |
|---|---|
| *width* | is the with of the new bitmap |
| *height* | is the height of the new bitmap |
| *content* | 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**

| | |
|---|---|
| *r* | red component of the new color |
| *g* | green component of the new color |
| *b* | 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**

| | |
|---|---|
| *img* | the bitmap to rotate |
| *angle* | the angle of the rotation, in degree |

#### 5.1.2.10 saveBmp()

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

save the bmp with the path name

**Parameters**

| | |
|---|---|
| *img* | the picture to save |
| *path* | 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**

| | |
|---|---|
| *img* | the bitmap to contrast |

#### 5.2.2.2 binarize()

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

put the image in black and white

**Parameters**

| | |
|---|---|
| *img* | the image which will be binarize |

#### 5.2.2.3 draw()

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

Draw an image in console.

**Parameters**

| | |
|---|---|
| *img* | the image which will be draw |

#### 5.2.2.4 freeBitmap()

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

free the bitmap

**Parameters**

| | |
|---|---|
| *img* | the bitmap to free |

#### 5.2.2.5 loadBmp()

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

Load a bmp file with a path in argument.

**Parameters**

| | |
|---|---|
| *path* | 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**

| | |
|---|---|
| *width* | is the with of the new bitmap |
| *height* | is the height of the new bitmap |
| *content* | 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**

| | |
|---|---|
| *r* | red component of the new color |
| *g* | green component of the new color |
| *b* | 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**

| | |
|---|---|
| *img* | the bitmap to rotate |
| *angle* | the angle of the rotation, in degree |

#### 5.2.2.10 saveBmp()

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

save the bmp with the path name

**Parameters**

| | |
|---|---|
| *img* | the picture to save |
| *path* | 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**

| | |
|---|---|
| *src* | 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**

| | |
|---|---|
| *histo* | is the hsitogram of a block |
| *hm* | 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**

| | |
|---|---|
| *img* | the full image |
| *min* | x min |
| *max* | y min |
| *width* | the width of the rectangle |
| *height* | 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**

| | |
|---|---|
| *img* | the full image |
| *min* | x min |
| *max* | y min |
| *width* | the width of the cuted Image |
| *height* | 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**

| | |
|---|---|
| *src* | is the original image |
| *nbCharacter* | is the number of character |
| *nbLetter* | is the number of letter |

#### 5.3.2.6 heightTravel()

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

create a new bitmap with height offset

**Parameters**

| | |
|---|---|
| *src* | 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**

| | |
|---|---|
| *final* | is the image that we modify |

**Parameters**

| | |
|---|---|
| *src* | is the original image |
| *histoQueue* | is the queue with contain all histograms |
| *hm* | 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**

| | |
|---|---|
| *columnMarker* | array where there are marker for each pixel |
| *width* | 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**

| | |
|---|---|
| *bmp* | is the text block |
| *original* | is copy binarize of the original image |
| *x* | higth left corner of the block on the copy in abscisse |
| *y* | higth left corner of the block on the copy in ordonate |
| *q* | 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**

| | |
|---|---|
| *src1* | the first bitmap |
| *src2* | 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**

| | |
|---|---|
| *img* | one line of the full image |
| *min* | x min |
| *max* | x max |
| *array* | where marker is put in function of the img |
| *pos* | the x min pos in the original image |
| *width* | 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**

| | |
|---|---|
| *img* | the full image |
| *array* | where marker is put in function of the img |
| *pos* | the x min pos in the original image |
| *width* | 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**

| | |
|---|---|
| *src* | is the original image |
| *imgQueue* | is the queue where we stock images |
| *posQueue* | 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**

| | |
|---|---|
| *img* | the full image |
| *nbCharacter* | the number of character in the image |
| *nbLetter* | the number of letter int the image |
| *q* | is the queue where is stocked all letter |
| *cutedImage* | is the the image that we use |
| *pos* | 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**

| | |
|---|---|
| *histoQueue* | is the queue with all histogram of a image |
| *src* | the cuted image |
| *original* | is the original image |
| *hm* | is the height average of text block |
| *pos* | is the x min pos |

#### 5.3.2.16 widthTravel()

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

create a new bitmap with width offset

**Parameters**

| | |
|---|---|
| *src* | 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**

| | |
|---|---|
| *src* | is the original image |
| *nbCharacter* | is the number of character |
| *nbLetter* | 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**

| | |
|---|---|
| *img* | the full image |
| *nbCharacter* | the number of character in the image |
| *nbLetter* | the number of letter int the image |
| *q* | is the queue where is stocked all letter |
| *cutedImage* | is the the image that we use |
| *pos* | 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**

| | |
|---|---|
| *widget* | The parent window |
| *data* | 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**

| | |
|---|---|
| *widget* | The parent window |
| *data* | 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**

| | |
|---|---|
| *widget* | The parent window |
| *data* | 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**

| | |
|---|---|
| *window* | The parent window |
| *data* | 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**

| | |
|---|---|
| *window* | The current parent window |
| *data* | The pointer to the Zone |

#### 5.5.2.7 saveFile()

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

Save the file at the location chosen by the user.

**Parameters**

| | |
|---|---|
| *window* | The parent window |
| *data* | 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**

| | |
|---|---|
| *argc* | The argc parameter of the main fuction |
| *argv* | 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**

| | |
|---|---|
| *s* | 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**

| | |
|---|---|
| *widget* | The parent window |
| *data* | 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**

| | |
|---|---|
| *widget* | The parent window |
| *data* | 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**

| | |
|---|---|
| *widget* | The parent window |
| *data* | 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**

| | |
|---|---|
| *window* | The parent window |
| *data* | 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**

| | |
|---|---|
| *window* | The current parent window |
| *data* | The pointer to the Zone |

#### 5.6.2.7 saveFile()

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

Save the file at the location chosen by the user.

**Parameters**

| | |
|---|---|
| *window* | The parent window |
| *data* | 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**

| | |
|---|---|
| *argc* | The argc parameter of the main fuction |
| *argv* | 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**

| | |
|---|---|
| *text* | the expected text |
| *nbsample* | the length of the text |
| *nbOutput* | the number of possibile outputs |

#### 5.7.2.2 createSamples()

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

fill the samples with a segmented image and return the size

**Parameters**

| | |
|---|---|
| *text* | the image segmented |
| *samples* | the table of sample to fill |

#### 5.7.2.3 freeSamples()

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

free all components of a sample

**Parameters**

| | |
|---|---|
| *samples* | the inputs or outputs of a sample |
| *nbSample* | 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**

| | |
|---|---|
| *n* | the neural network |
| *samples* | the inputs of samples |

**Parameters**

| | |
|---|---|
| *results* | the outputs of samples |
| *nbSample* | the number of samples |

#### 5.7.2.5 learning()

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

learn to the network "network.save"

**Parameters**

| | |
|---|---|
| *learnFiles* | the table of path to learn files |
| *nbFile* | the number of file |

#### 5.7.2.6 randomize()

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

randomize the weight and thresholds of a network

**Parameters**

| | |
|---|---|
| *n* | the network |
| *r* | the ratio of the randomization |

#### 5.7.2.7 shuffleSample()

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

suffle the samples and results table

**Parameters**

| | |
|---|---|
| *samples* | the table of samples |
| *result* | the table of results |
| *length* | 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**

| | |
|---|---|
| *argc* | the number of params |
| *argv* | 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 nblayer, 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**

| | |
|---|---|
| *n* | the neural network |

#### 5.9.2.2 generateNetwork()

```
void generateNetwork ( )
```

generate a network of three layer

**Parameters**

| | |
|---|---|
| *c1* | the number of neurone for the first layer |
| *c1* | the number of neurone for the second layer |
| *c1* | 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**

| | |
|---|---|
| *path* | location of the save |

#### 5.9.2.4 newNetwork()

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

create a new neural network

**Parameters**

| | |
|---|---|
| *nblayer* | the number of layers |
| *layers* | a table of the number of neurone in each layer |

#### 5.9.2.5 saveNetwork()

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

save weights anf thresholds of a neural network

**Parameters**

| | |
|---|---|
| *path* | location of the new file |
| *n* | 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 nblayer, 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*

### 5.10.1 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**

| | |
|---|---|
| *n* | the neural network |

#### 5.10.2.2 generateNetwork()

```
void generateNetwork ( )
```

generate a network of three layer

**Parameters**

| | |
|---|---|
| *c1* | the number of neurone for the first layer |
| *c1* | the number of neurone for the second layer |
| *c1* | 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**

| | |
|---|---|
| *path* | location of the save |

#### 5.10.2.4 newNetwork()

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

create a new neural network

**Parameters**

| | |
|---|---|
| *nblayer* | the number of layers |
| *layers* | a table of the number of neurone in each layer |

#### 5.10.2.5 saveNetwork()

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

save weights anf thresholds of a neural network

**Parameters**

| | |
|---|---|
| *path* | location of the new file |
| *n* | the neural network to save |

# Index