#include <stdio.h>

#include <stdlib.h>

#include <opencv/cv.h>

#include <opencv/highgui.h>

#define ALTOBLOQUE 32

#define ANCHOBLOQUE 32

void copiarBloque(int x0,int y0, IplImage\* imagenOrigen,int x1,int y1, IplImage\* imagenDestino){

for (int fila = 0; fila < ALTOBLOQUE; fila++){

uchar \*pImg1 = imagenOrigen->imageData + (fila+y0)\*imagenOrigen->widthStep + x0\*imagenOrigen->nChannels;

uchar \*pImg2 = imagenDestino->imageData + (fila+y1)\*imagenDestino->widthStep + x1\*imagenDestino->nChannels;

for(int col = 0; col < ANCHOBLOQUE; col++){

\*pImg2++ = \*pImg1++;

\*pImg2++ = \*pImg1++;

\*pImg2++ = \*pImg1++;

}

}

}

int compararBloques(int x0,int y0, IplImage\* imagenOrigen,int x1,int y1, IplImage\* imagenDestino){

int dif=0;

for (int fila = 0; fila < ALTOBLOQUE; fila++){

uchar \*pImg1 = imagenOrigen->imageData + ((fila+y0)\*imagenOrigen->widthStep) + (x0\*imagenOrigen->nChannels);

uchar \*pImg2 = imagenDestino->imageData + ((fila+y1)\*imagenDestino->widthStep) + (x1\*imagenDestino->nChannels);

for(int col = 0; col < ANCHOBLOQUE; col++){

dif = dif + abs(\*pImg2++ - \*pImg1++);

dif = dif + abs(\*pImg2++ - \*pImg1++);

dif = dif + abs(\*pImg2++ - \*pImg1++);

}

}

return dif;

}

void buscaBloque(int x0, int y0, IplImage\* imagenOrigen, IplImage\* imagenDestino, int\* x\_parecido, int\* y\_parecido){

int minor\_dif = 3\*255\*ANCHOBLOQUE\*ALTOBLOQUE;

int dif;

int b\_x = imagenOrigen->width/ANCHOBLOQUE;

int b\_y = imagenOrigen->height/ALTOBLOQUE;

for (int i = 0; i < b\_x; i++){

for(int j = 0; j < b\_y; j++){

dif = compararBloques( x0\*ANCHOBLOQUE, y0\*ALTOBLOQUE, imagenOrigen, i\*ANCHOBLOQUE, j\*ALTOBLOQUE, imagenDestino);

if(dif < minor\_dif){

\*x\_parecido = i;

\*y\_parecido = j;

minor\_dif = dif;

}

}

}

}

int main(int argc, char\*\* argv) {

if (argc != 3) {

printf("Usage: %s image\_file\_name\n", argv[0]);

return EXIT\_FAILURE;

}

//CV\_LOAD\_IMAGE\_COLOR = 1 forces the resultant IplImage to be colour.

//CV\_LOAD\_IMAGE\_GRAYSCALE = 0 forces a greyscale IplImage.

//CV\_LOAD\_IMAGE\_UNCHANGED = -1

IplImage\* Img1 = cvLoadImage(argv[1], CV\_LOAD\_IMAGE\_COLOR);

IplImage\* Img2 = cvLoadImage(argv[2], CV\_LOAD\_IMAGE\_COLOR);

// Always check if the program can find a file

if (!Img1) {

printf("Error: fichero %s no leido\n", argv[1]);

return EXIT\_FAILURE;

}

if (!Img2) {

printf("Error: fichero %s no leido\n", argv[2]);

return EXIT\_FAILURE;

}

// a visualization window is created with title 'image'

cvNamedWindow(argv[1], CV\_WINDOW\_AUTOSIZE);

cvNamedWindow(argv[2], CV\_WINDOW\_AUTOSIZE);

// img is shown in 'image' window

cvShowImage(argv[1], Img1);

cvShowImage(argv[2], Img2);

cvWaitKey(0);

int b\_x = Img1->width/ANCHOBLOQUE;

int b\_y = Img1->height/ALTOBLOQUE;

int x\_parecido, y\_parecido;

for (int y0 = 0; y0 < b\_y; y0++){

for(int x0 = 0; x0 < b\_x; x0++){

buscaBloque(x0, y0, Img2, Img1, &x\_parecido, &y\_parecido);

copiarBloque( x\_parecido\*ANCHOBLOQUE, y\_parecido\*ALTOBLOQUE, Img1, x0\*ANCHOBLOQUE, y0\*ALTOBLOQUE, Img2);

cvShowImage(argv[1], Img1);

cvShowImage(argv[2], Img2);

cvWaitKey(1);

}

}

cvWaitKey(0);

// memory release for img before exiting the application

cvReleaseImage(&Img1);

cvReleaseImage(&Img2);

// Self-explanatory

cvDestroyWindow(argv[1]);

cvDestroyWindow(argv[2]);

return EXIT\_SUCCESS;

}