Some comman used VisionX command:

unsigned char A[N][M];

short B[N][M];

int i,j;

for ( i=0, i< N, i++ ) {

for ( j=0, j< M, j++ ) {

A[i,j] = B[i,j];

}

}

1) Vfstruct (Aim); /\* declare the image structures \*/

2) Vfstruct (Bim);

3) int y,x;

4) /\* ... read in image Bim and declare the structure for Aim.. \*/

5) /\* in this process the size, type, and channels of the \*/

6) /\* images are dynamically specified. \*/

7) for ( y=Bim.ylo, y<=Bim.yhi, y++ ) {

8) for ( x=Bim.xlo, x <=Bim.xhi, x++ ) {

9) Aim.u[y][x] = Bim.s[y][x];

10) }

11) }

Vfread(&im, IVAL); /\* read file and initialize input structure \*/

Vfembed(&tm, &im,1,1,1,1); /\* image structure with border \*/

The first statement reads an input image "im" from the input file. The second statement declares the structure "tm" and copies the image data from "im". Vfembed also adds a border of elements (set to 0) around the edge of the original image (as specified by the last four arguments of "1"). That is, "tm" has two addition rows and two additional columns with respect to "im".

The main program will compute a function from the padded image "tm" and store it in "im" and then write out the original image data structure "im" (with modified contents).

The Template Program vtemp.c

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* vtemp Compute local max operation on a single byte image \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include "VisXV4.h" /\* VisionX structure include file \*/

#include "Vutil.h" /\* VisionX utility header files \*/

VXparam\_t par[] = /\* command line structure \*/

{ /\* prefix, value, description \*/

{ "if=", 0, " input file vtemp: local max filter "},

{ "of=", 0, " output file "},

{ 0, 0, 0} /\* list termination \*/

};

#define IVAL par[0].val

#define OVAL par[1].val

main(argc, argv)

int argc;

char \*argv[];

{

Vfstruct (im); /\* i/o image structure \*/

Vfstruct (tm); /\* temp image structure \*/

int y,x; /\* index counters \*/

VXparse(&argc, &argv, par); /\* parse the command line \*/

Vfread(&im, IVAL); /\* read image file \*/

Vfembed(&tm, &im, 1,1,1,1); /\* image structure with border \*/

if ( im.type != VX\_PBYTE ) { /\* check image format \*/

fprintf(stderr, "vtemp: no byte image data in input file\n");

exit(-1);

}

for (y = im.ylo ; y <= im.yhi ; y++) { /\* compute the function \*/

for (x = im.xlo; x <= im.xhi; x++) { /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

im.u[y][x] = MAX(tm.u[y][x], /\* You only need to \*/

MAX(tm.u[y][x+1], /\* change this section \*/

MAX(tm.u[y+1][x], /\* for your program \*/

MAX(tm.u[y][x-1], /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

tm.u[y-1][x]))));

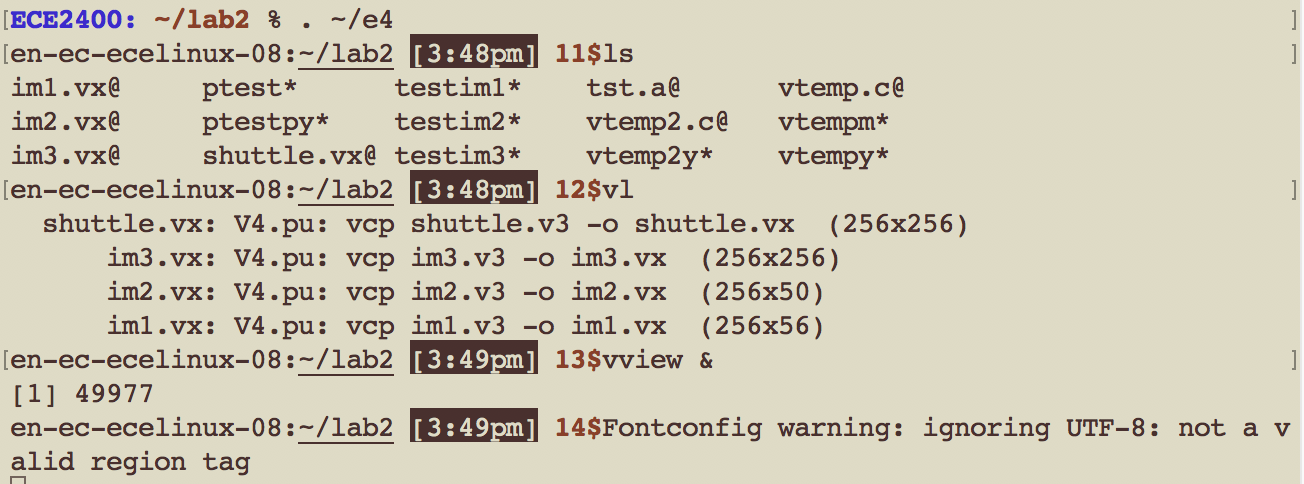
}

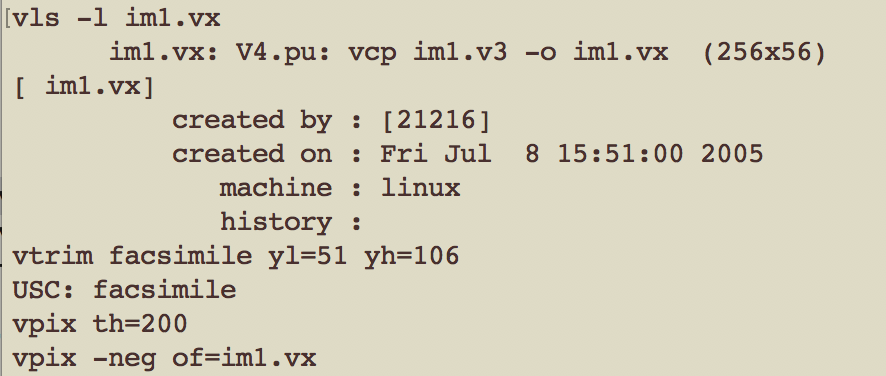
}

Vfwrite(&im, OVAL); /\* write image file \*/

exit(0);

}





After this command, visionX windows dropped out

vdview im1.vx

1. *In vview double click on the file names to view the images. To see the image properties use "Tools→ Image Statistics"*

|  |  |
| --- | --- |
| **Command** | **Action** |
|  |  |
| vls [-l] [files] | provide information on VisionX files |
| vl [-l] | list VisionX files in the current directory (in the order that they were created |
| vdview | display a VisionX image file |

*To access the user manual information for any visionx command go to "Help→ vman VX commands", scroll or search for the command name and double click on it.*

## Compiling and Testing a (VisionX) program

Compile the template program vtemp

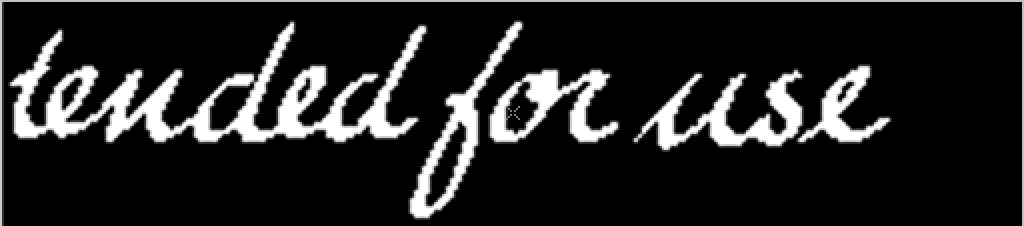
vcc vtemp.c -o vtemp

Test it with image data

vtemp im1.vx of=im1.mx

vview

then click on “im1.vx” to see the original image, as below:



Click on “im1.mx” to see our test result, as below:



Explore the use of scripts for program development. First review the script testim1

less testim1



Then execute it to create the image testim1.vx

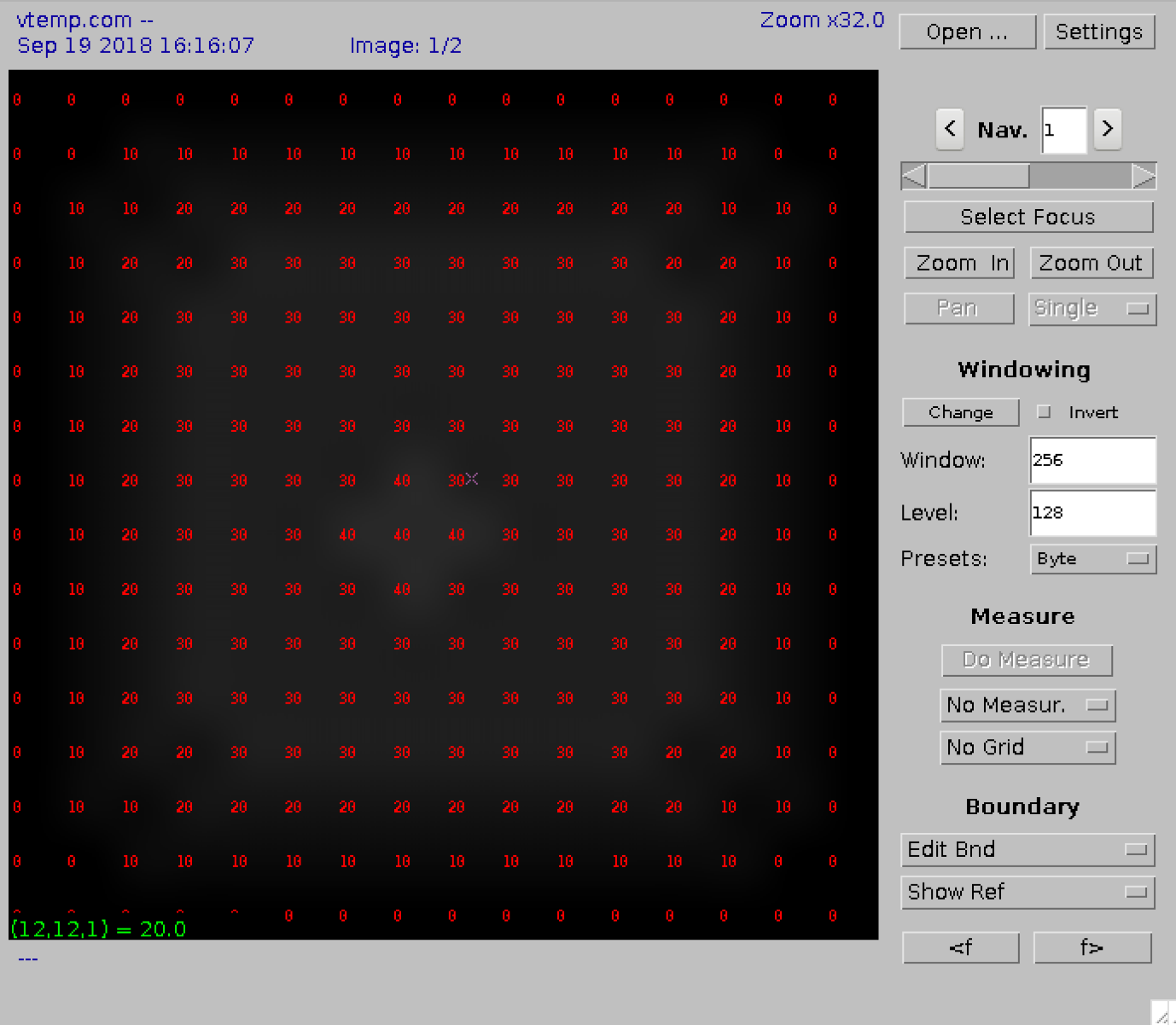
testim1

Review the script "ptest" which is designed to: (a) compile a program, (b) execute that program on a specified image, (c) create a composite image that is suitable for visual review of output vs input and (d) display that image for your review.

less ptest

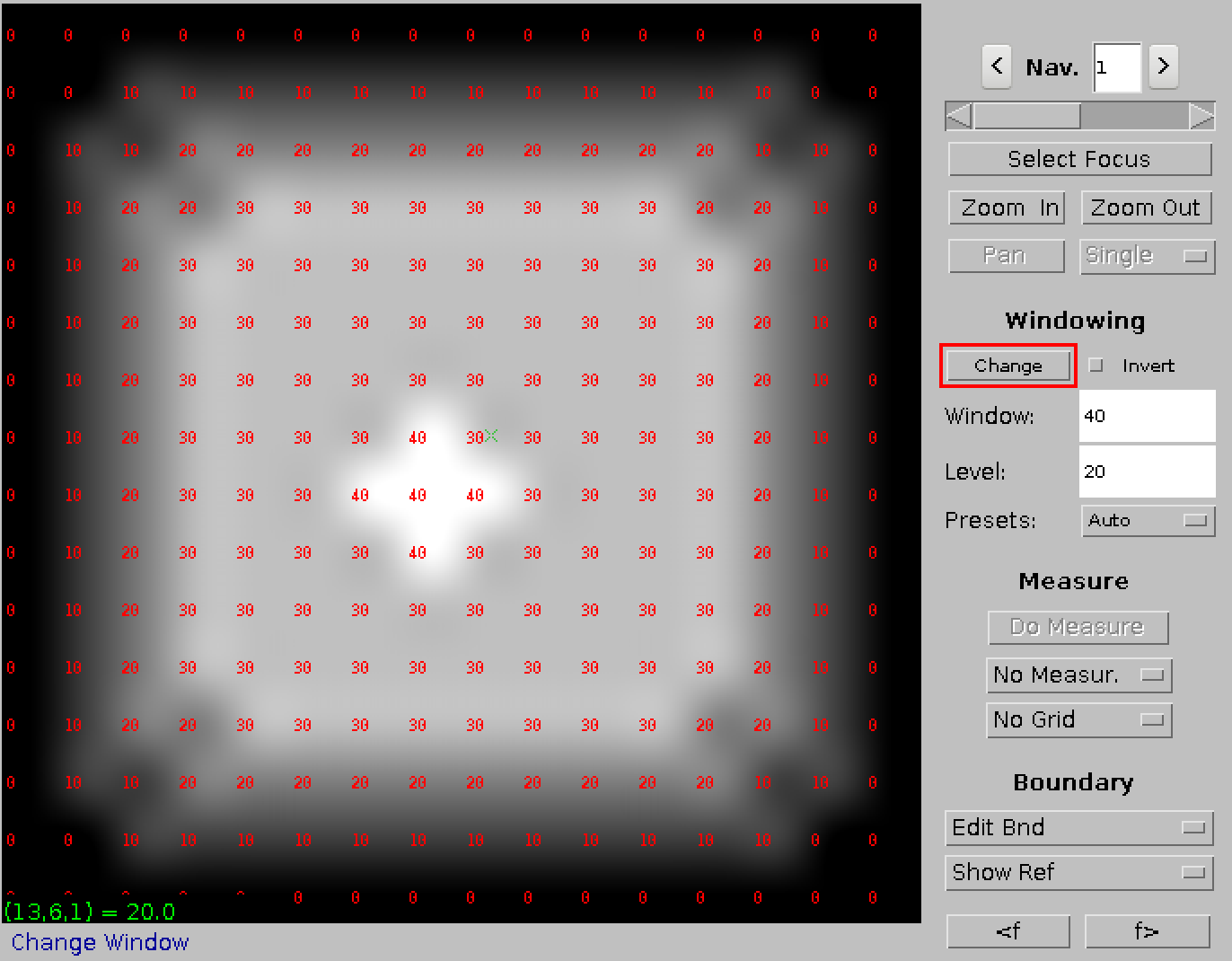
Run the script on your program with the test image.

ptest testim1.vx vtemp

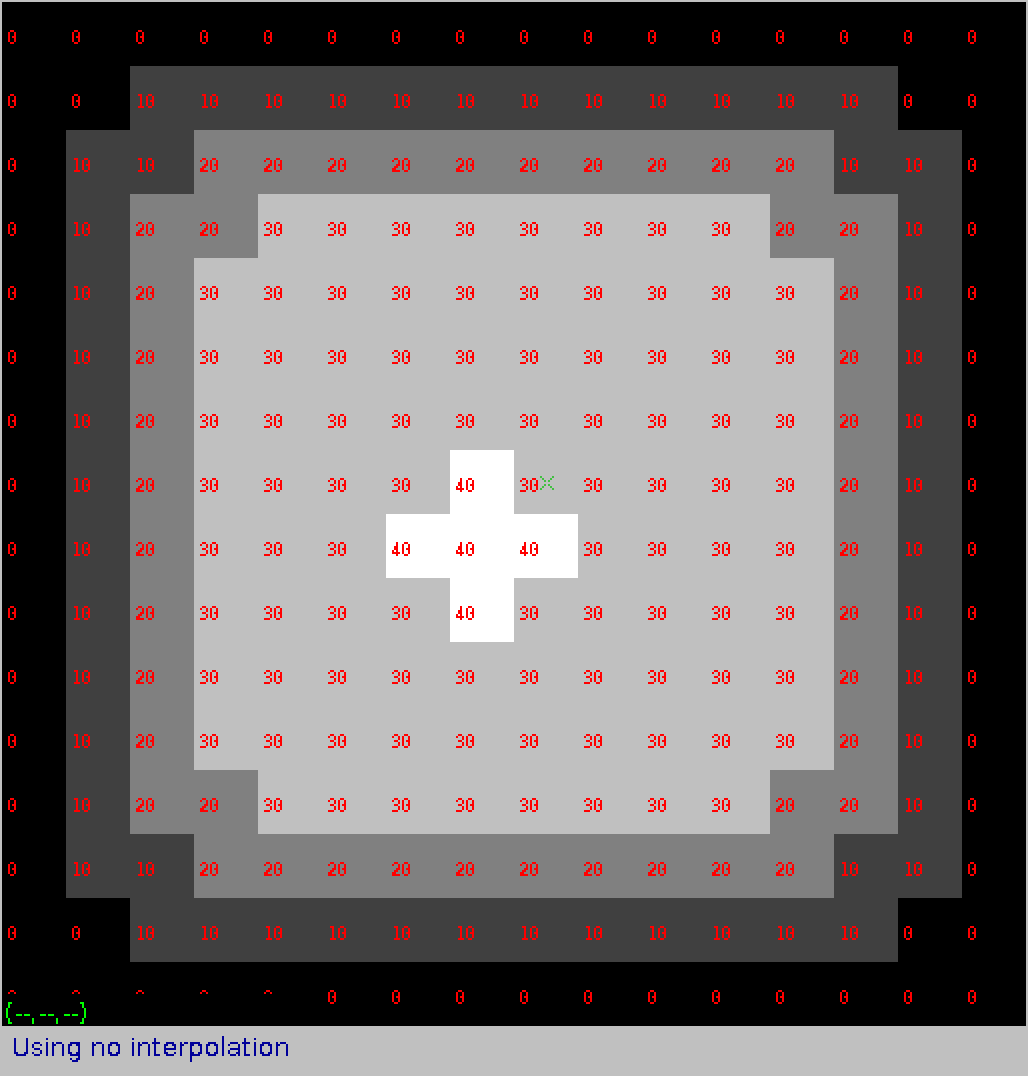


In the image that is displayed (a) set the window "Presets:" to "Auto",(b) in "settings" select "No Interpolation" and (c) compare the input and output images by rotating the mouse wheel.

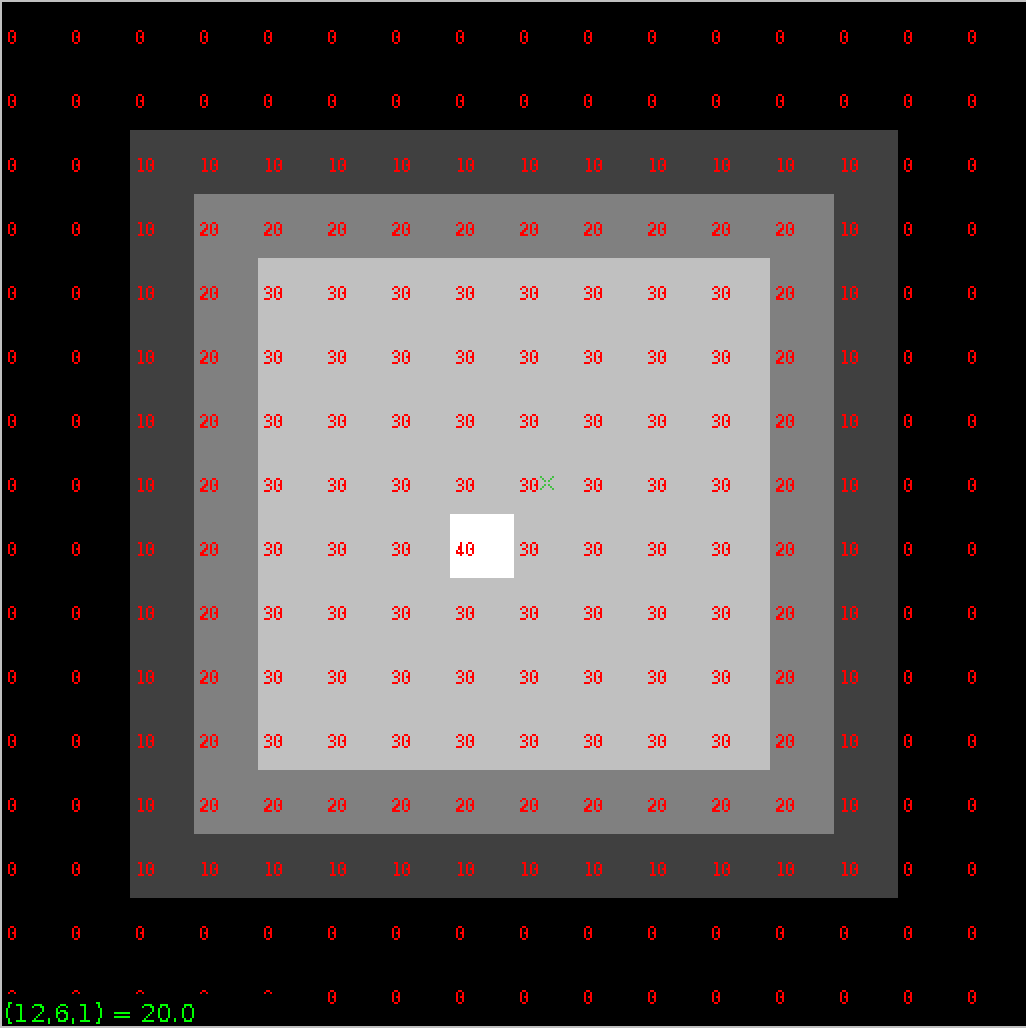
(a)



(b)



(c)



**Using Scripts:** Note, the scripts testim1, testim2, testim3, and ptest are editable text files. You may make copies of them and modify them to match your needs. If you copy a script file you may need to set the execution permision in the file desctiption so that the system will recognize it as a file that can be executed. The command to do this is "chmod +x". For example

cp testim1 testim1cp

cp testim2 testim2cp

cp testim3 testim3cp

chmod +x testim1

chmod +x testim2

chmod +x testim3

set the execute permission for the three files

Prepare images for printing or for inclusion in your lab report

vpix -neg im1.vx | vcapt c="input image im1.vx" of=p1.vx

vpix -neg im1.mx | vcapt c="vtemp processed image" of=p2.vx

This command inverts the image so that you would use less "ink" if you wanted to print it.

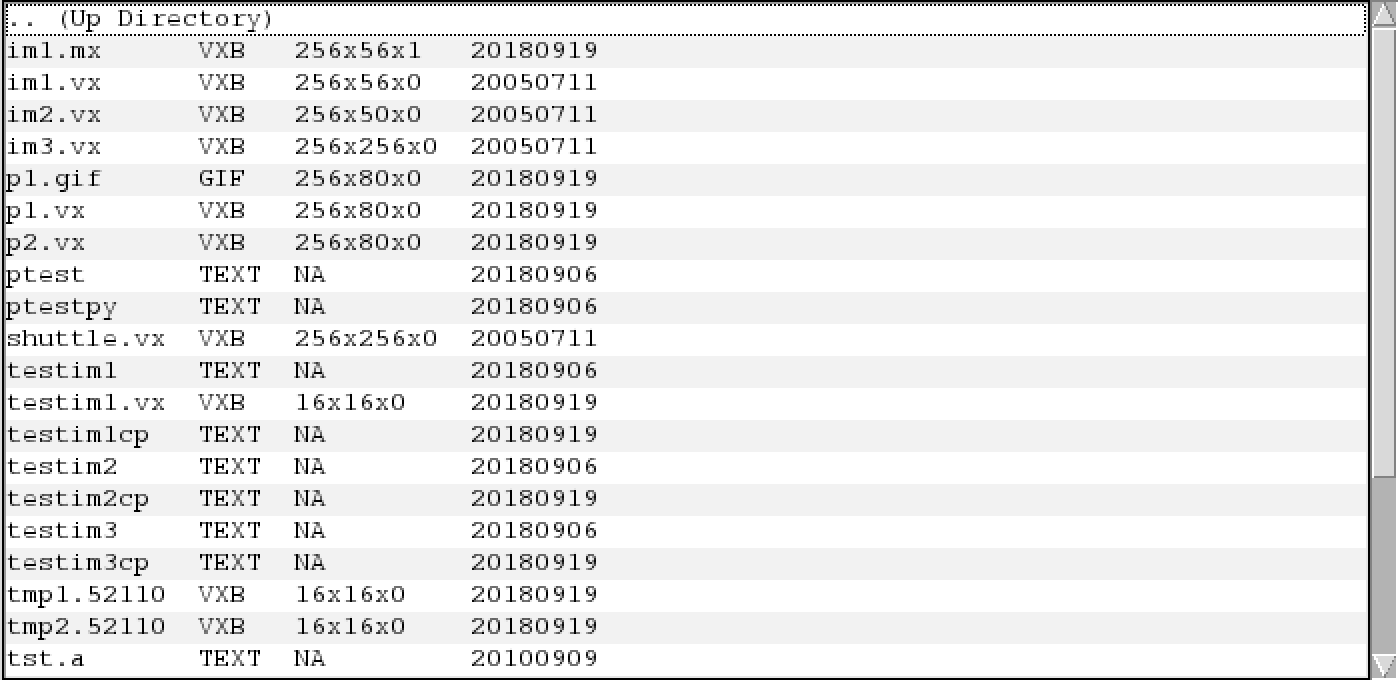
View my results, as below:





Convert VisionX images into png, jpeg or gif files (for importing into LibreOffice Write and Draw programs for your report)

*these commands will create two new image files "p1.gif" and "p2.gif" If you click on the Refresh button these files will become visible in vview. They may be displayed with either the vdview or "Tab view" viewers.*



1. n general you should use the gif format for gray images and png or jpeg for color images. To generate png or jpeg images replace gif with png or jpeg in the above commands.

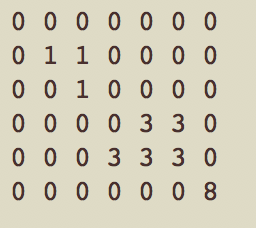
|  |  |
| --- | --- |
| **Command** | **Action** |
|  |  |
| vcc | compile a c program with VisionX include files and library |
| vpix | perform point operations on an image |
| vcapt | add a caption to the bottom of an image |
| vxport | change image format from VisionX format |

## Small Image Manipulation

you develop is first tested on very small images that you can check the result. the following VisionX commands are used to create and display small test images:

Examine the ASCII "image" tst.a

less tst.a



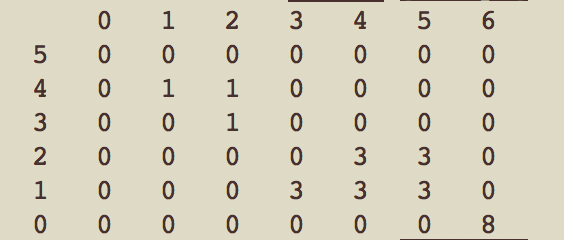
*The file "test.a" is also shown in vview as a TEXT file. By double clicking on this name you can view its contents. You can also edit the file in vview and save the changes by clicking on the "Save" button.*

Generate a VisionX image

vrawtovx -t tst.a of=tst.vx

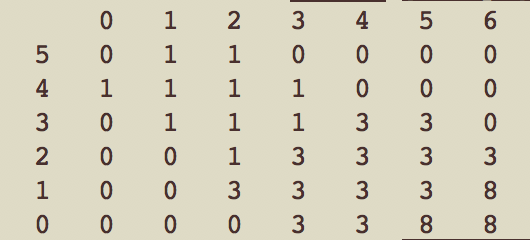
Display the VisionX image

vppr tst.vx



Test the program vtemp

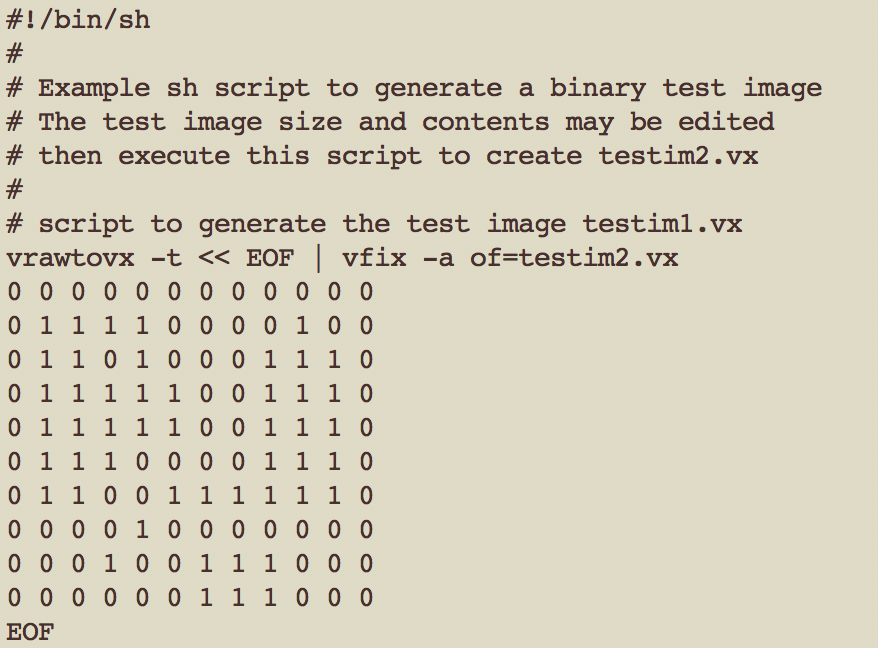
vtemp tst.vx | vppr



1. Use scripts as an alternative approach. Create a testimge by executing script testim2 and then use ptest to test vtemp with the image imtest2.vx.

review the script testim2

less testim2



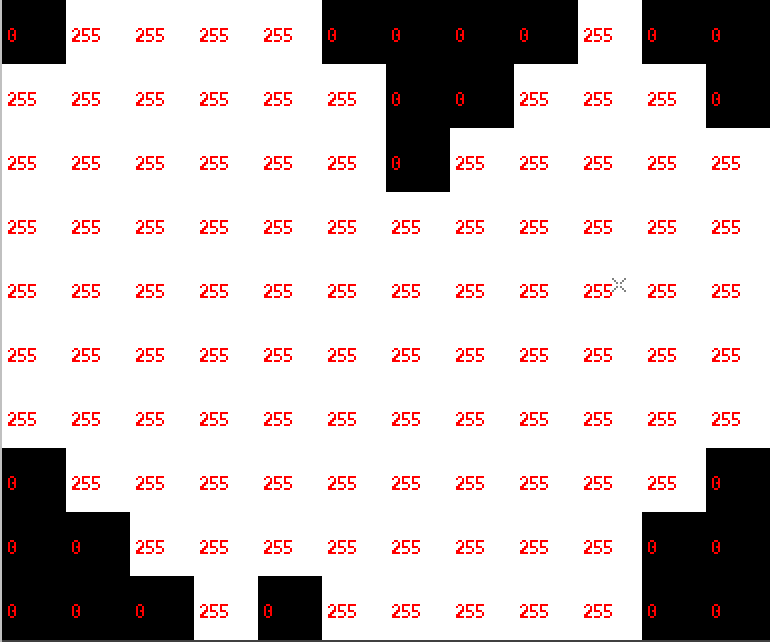
execute it to create the image testim2.vx

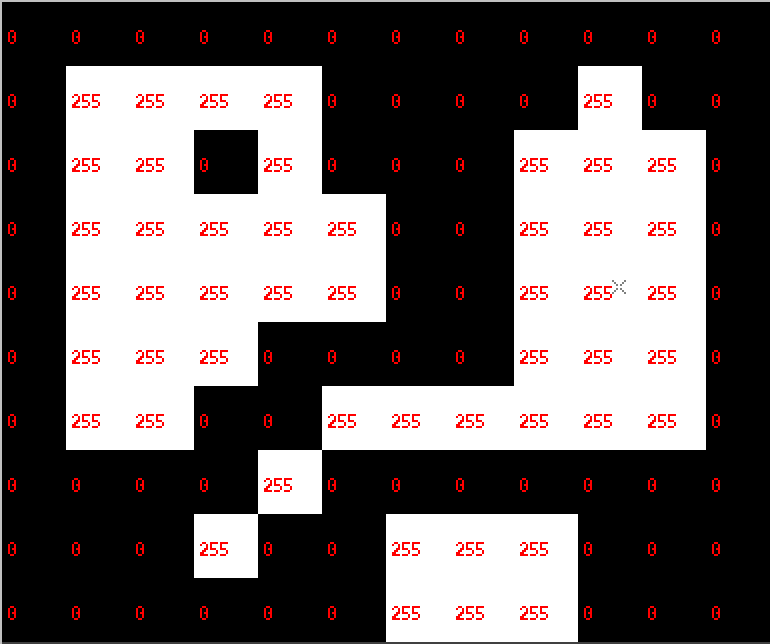
testim2

so we got

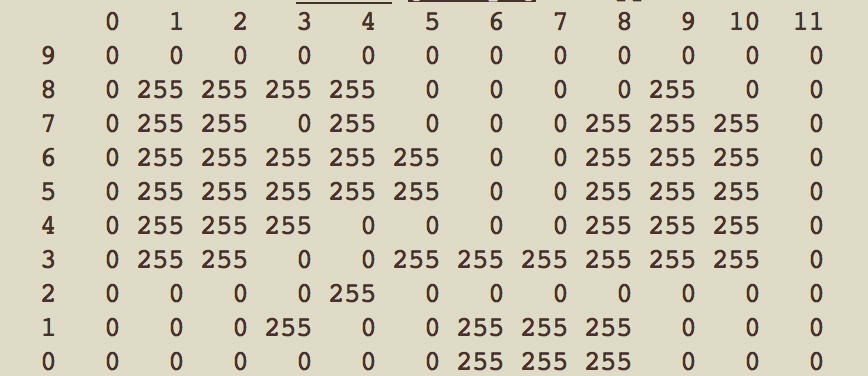
testim2.vx

ptest testim2.vx vtemp

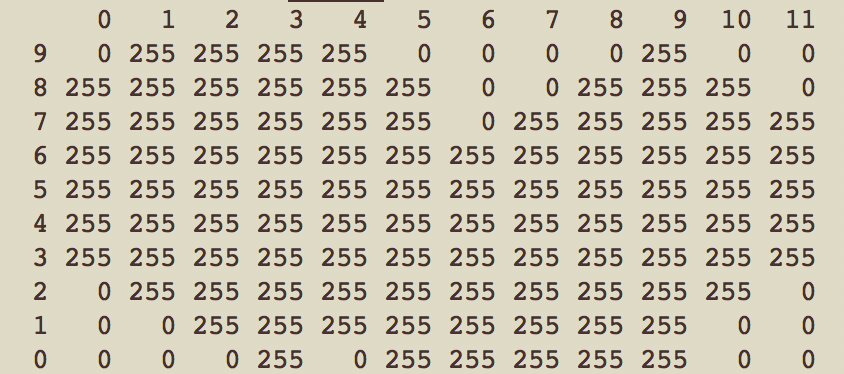




vvpr testim2.vx



vtemp testim2.vx | vppr



Make a new test image of your own design to test vtemp. You can do this with either the direct method or by using scripts. While the direct method may be simpler, the script method is superior for efficient program development but requires some understanding of the script language. It is suggested that you attempt both.

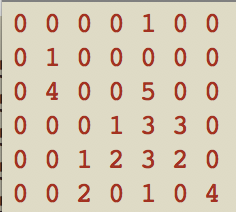
I Modify my own copy of the ASCII image using the vim text editor

cp tst.a mine.a

vim mine.a

and I revise my mine.a

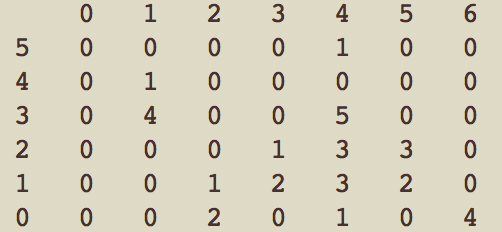
as below:

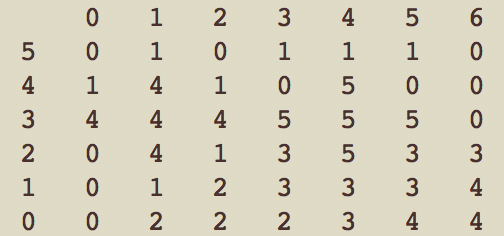
Test the program vtemp with the new image

vrawtovx -t mine.a of=mine.vx

vppr mine.vx

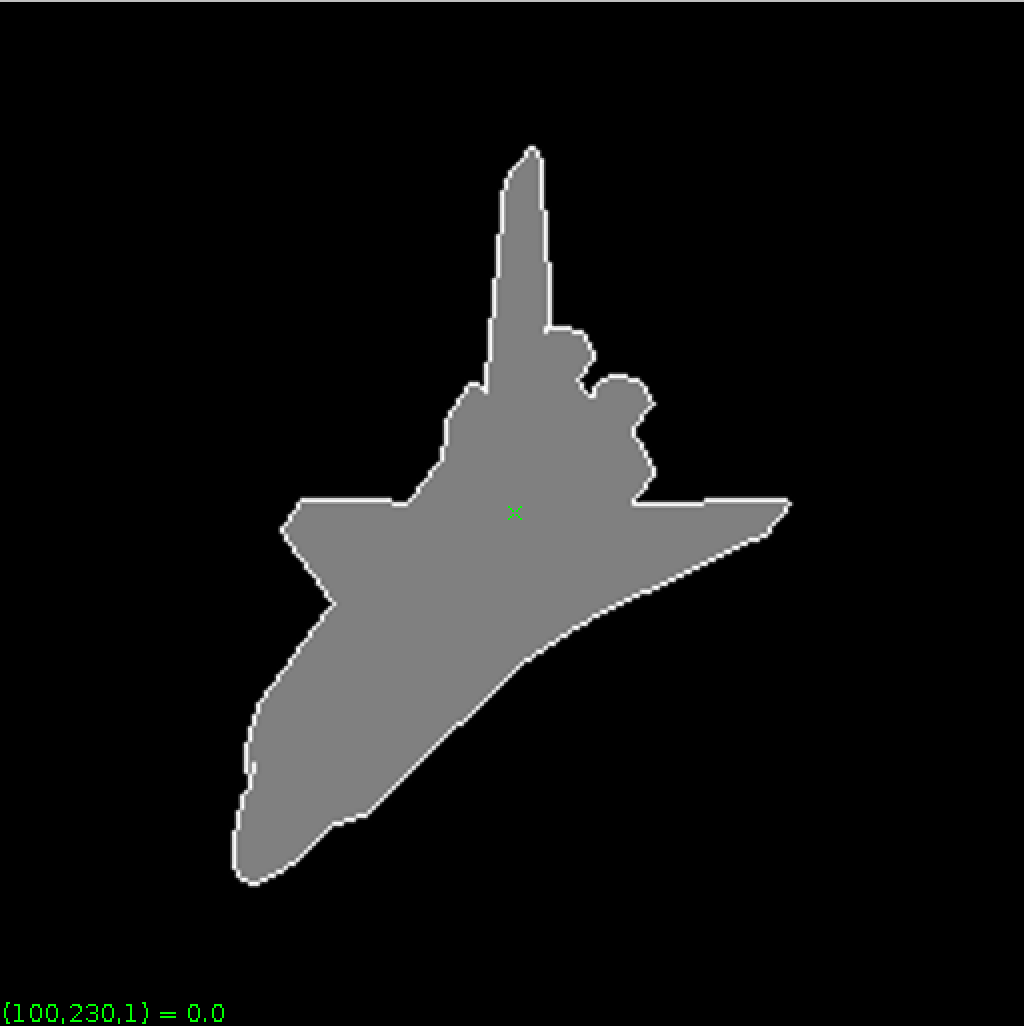


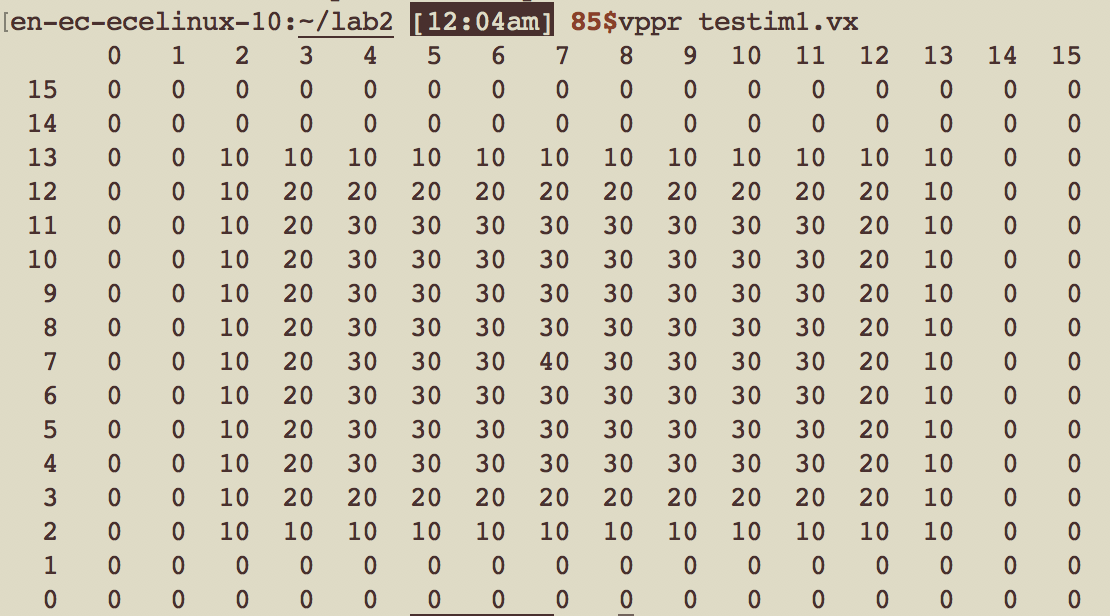
vtemp mine.vx | vppr



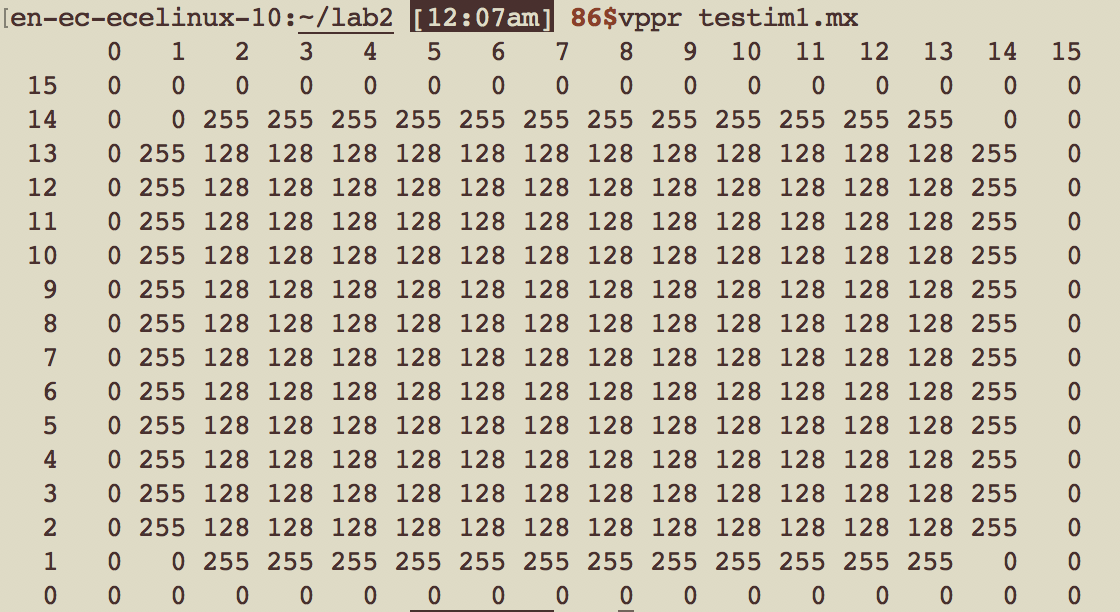
## A first program ****(bound)****

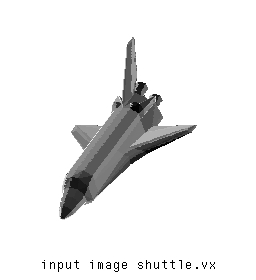
bound shuttle.vx of= shuttle.mx

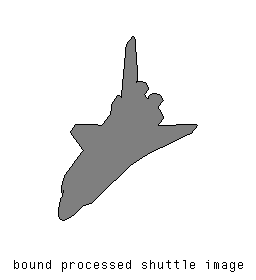






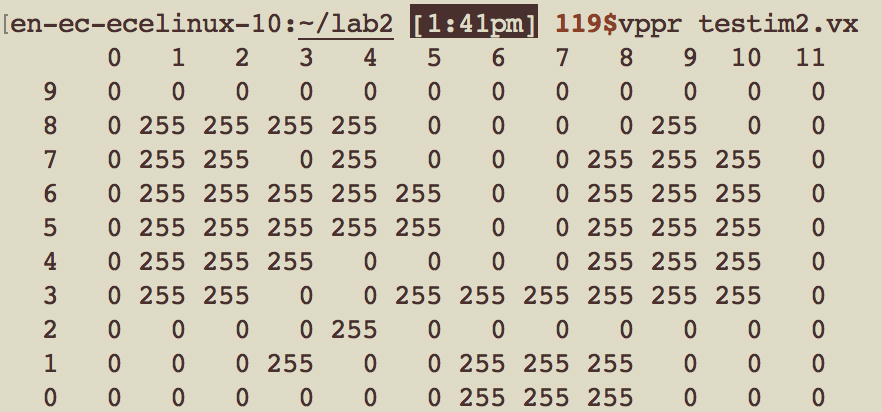




## 5. A second program ****(cclabel)****

## 

## 



en-ec-ecelinux-10:~/lab2 [1:41pm] **120$**vpix -neg testim2.vx | vcapt c="input image testim2.vx" of=cclabeltestim2.vx

en-ec-ecelinux-10:~/lab2 [1:48pm] **123$**vxport -png cclabeltestim2.vx

**** ****

en-ec-ecelinux-10:~/lab2 [1:51pm] **125$**vpix -neg test.vx | vcapt c="cclabeledtestim2.vx" of=cclabeltestim2output.vx

en-ec-ecelinux-10:~/lab2 [1:52pm] **126$**vxport -png cclabeltestim2output.vx

en-ec-ecelinux-10:~/lab2 [1:55pm] **128$**cclabel if=im1.mx of=cclabeledletters.vx

en-ec-ecelinux-10:~/lab2 [1:59pm] **132$**vpix -neg im1.vx | vcapt c="input cclabeled im1.vx" of=cclabeledim1input.vx

en-ec-ecelinux-10:~/lab2 [DING!] **133$**vxport -png cclabeledim1input.vx



en-ec-ecelinux-10:~/lab2 [1:56pm] **129$**vpix -neg cclabeledletters.vx | vcapt c="cclabeled im1.vx" of=cclabeledim1.vx

en-ec-ecelinux-10:~/lab2 [1:57pm] **131$**vxport -png cclabeledim1.vx



Appendix:

|  |
| --- |
|  |
|  | /\* vtemp Compute local max operation on a single byte image \*/ |
|  | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ |
|  |  |
|  | #include "VisXV4.h" /\* VisionX structure include file \*/ |
|  | #include "Vutil.h" /\* VisionX utility header files \*/ |
|  |  |
|  | VXparam\_t par[] = /\* command line structure \*/ |
|  | { /\* prefix, value, description \*/ |
|  | { "if=", 0, " input file vtemp: local max filter "}, |
|  | { "of=", 0, " output file "}, |
|  | { 0, 0, 0} /\* list termination \*/ |
|  | }; |
|  | #define IVAL par[0].val |
|  | #define OVAL par[1].val |
|  |  |
|  | main(argc, argv) |
|  | int argc; |
|  | char \*argv[]; |
|  | { |
|  | Vfstruct (im); /\* i/o image structure \*/ |
|  | Vfstruct (tm); /\* temp image structure \*/ |
|  | int y,x; /\* index counters \*/ |
|  | VXparse(&argc, &argv, par); /\* parse the command line \*/ |
|  |  |
|  | Vfread(&im, IVAL); /\* read image file \*/ |
|  | Vfembed(&tm, &im, 1,1,1,1); /\* image structure with border \*/ |
|  | if ( im.type != VX\_PBYTE ) { /\* check image format \*/ |
|  | fprintf(stderr, "vtemp: no byte image data in input file\n"); |
|  | exit(-1); |
|  | } |
|  | for (y = im.ylo ; y <= im.yhi ; y++) { /\* compute the function \*/ |
|  | for (x = im.xlo; x <= im.xhi; x++) { /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ |
|  | //im.u[y][x] = MAX(tm.u[y][x], /\* You only need to \*/ |
|  | // MAX(tm.u[y][x+1], /\* change this section \*/ |
|  | // MAX(tm.u[y+1][x], /\* for your program \*/ |
|  | // MAX(tm.u[y][x-1], /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ |
|  | // tm.u[y-1][x])))); |
|  |  |
|  |  |
|  | if(tm.u[y][x] == 0 && tm.u[y + 1][x] == 0 && tm.u[y][x + 1] == 0 && tm.u[y - 1][x] == 0 && tm.u[y][x - 1] == 0){ |
|  | im.u[y][x] = 0; |
|  | } |
|  |  |
|  | else if(tm.u[y][x] != 0){ |
|  | im.u[y][x] = 128; |
|  | } |
|  |  |
|  | else{ |
|  | im.u[y][x] = 255; |
|  | } |
|  | } |
|  | } |
|  |  |
|  | Vfwrite(&im, OVAL); /\* write image file \*/ |
|  | exit(0); |
|  | } |

|  |
| --- |
| /\* vtemp Compute local max operation on a single byte image \*/ |
|  | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ |
|  |  |
|  | #include "VisXV4.h" /\* VisionX structure include file \*/ |
|  | #include "Vutil.h" /\* VisionX utility header files \*/ |
|  |  |
|  | VXparam\_t par[] = /\* command line structure \*/ |
|  | { /\* prefix, value, description \*/ |
|  | { "if=", 0, " input file vtemp: local max filter "}, |
|  | { "of=", 0, " output file "}, |
|  | { 0, 0, 0} /\* list termination \*/ |
|  | }; |
|  | #define IVAL par[0].val |
|  | #define OVAL par[1].val |
|  | /\*void lmax(int, int); \*/ |
|  | /\* Blanca add the declaration of setlable function \*/ |
|  | void setlabel(int, int, int); |
|  |  |
|  | Vfstruct (im); /\* i/o image structure \*/ |
|  | Vfstruct (tm); /\* temp image structure \*/ |
|  | Vfstruct (vm); |
|  | main(argc, argv) |
|  | int argc; |
|  | char \*argv[]; |
|  | { |
|  | int y,x; /\* index counters \*/ |
|  | VXparse(&argc, &argv, par); /\* parse the command line \*/ |
|  |  |
|  | Vfread(&im, IVAL); /\* read image file \*/ |
|  | Vfembed(&tm, &im, 1,1,1,1); /\* image structure with border \*/ |
|  | Vfembed(&vm, &im, 1,1,1,1); |
|  | if ( im.type != VX\_PBYTE ) { /\* check image format \*/ |
|  | fprintf(stderr, "vtemp: no byte image data in input file\n"); |
|  | exit(-1); |
|  | } |
|  | /\* for (y = im.ylo ; y <= im.yhi ; y++) { |
|  | for (x = im.xlo ; x <= im.xhi; x++) { |
|  | lmax(x, y); |
|  | } |
|  | } |
|  | \*/ |
|  |  |
|  | // <-- Blanca's code |
|  | /\* preset all pixels in im to 0 implies unlabled \*/ |
|  | for (y = im.ylo ; y <= im.yhi ; y++){ |
|  | for (x = im.xlo ; x <= im.xhi ; x++){ |
|  | vm.u[y][x] = 0; |
|  | } |
|  | } |
|  |  |
|  | /\* when meet object and check isunlabled in im, then use lable function to lable it \*/ |
|  | int i = 20; |
|  | for (y = im.ylo ; y <= im.yhi ; y++){ |
|  | for(x = im.ylo ; x <= im.xhi ; x++){ |
|  | if(tm.u[y][x] != 0 && vm.u[y][x] == 0){ |
|  | setlabel(y, x, i); |
|  | i=i+30; |
|  |  |
|  | } |
|  | } |
|  | } |
|  |  |
|  | for (y = im.ylo ; y <= im.yhi ; y++){ |
|  | for(x = im.ylo ; x <= im.xhi ; x++){ |
|  | im.u[y][x] = vm.u[y][x]; |
|  | } |
|  | } |
|  | //Blanca's code --> |
|  |  |
|  | Vfwrite(&im, OVAL); /\* write image file \*/ |
|  | exit(0); |
|  | } |
|  |  |
|  | /\* function to compute the local maximum \*/ |
|  | /\* |
|  | \* void lmax(int x, int y) |
|  | { |
|  | im.u[y][x] = MAX(tm.u[y][x], |
|  | MAX(tm.u[y][x+1], |
|  | MAX(tm.u[y+1][x], |
|  | MAX(tm.u[y][x-1], |
|  | tm.u[y-1][x])))); |
|  | } |
|  | \*/ |
|  |  |
|  | //<-- Blanca's code |
|  | /\* setlabel(y, x, h) function here \*/ |
|  | void setlabel(int y, int x, int h){ |
|  | vm.u[y][x] = h; |
|  | if(vm.u[y][x + 1] == 0 && tm.u[y][x + 1] != 0){ |
|  | setlabel(y, x + 1, h); |
|  | } |
|  | if(vm.u[y][x - 1] == 0 && tm.u[y][x - 1] != 0){ |
|  | setlabel(y, x - 1, h); |
|  | } |
|  | if(vm.u[y - 1][x] == 0 && tm.u[y - 1][x] != 0){ |
|  | setlabel(y - 1, x, h); |
|  | } |
|  | if(vm.u[y + 1][x] == 0 && tm.u[y + 1][x] != 0){ |
|  | setlabel(y + 1, x, h); |
|  | } |
|  | } |
|  |  |
|  | //Blanca's code --> |