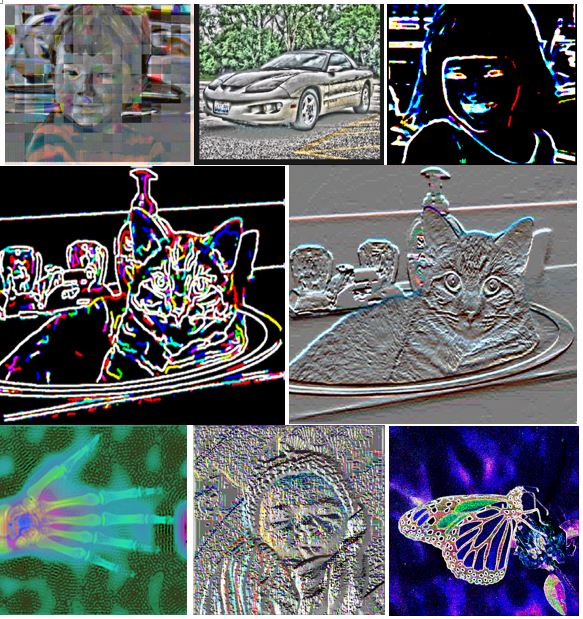
SOLUTIONS MANUAL

Version 3.1

2018

*DIGITAL IMAGE PROCESSING & ANALYSIS: Applications with Matlab and CVIPtools*

*Edition 3*



SCOTT E UMBAUGH

Southern Illinois University Edwardsville

**TABLE OF CONTENTS**

[Solutions for Chapter 1: Introduction to Computer Imaging 3](#_Toc269924756)

[Solutions for Chapter 2: Computer Imaging Systems 4](#_Toc269924757)

[Supplementary Exercises: 9](#_Toc269924758)

[Solutions for Chapter 3: Introduction to Digital Image Analysis 11](#_Toc269924759)

[Supplementary Exercises 16](#_Toc269924760)

[Solutions for Chapter 4: Segmentation and Edge/Line Detection 20](#_Toc269924761)

[SupplementaryExercises 27](#_Toc269924762)

[Solutions for Chapter 5: Discrete Transforms 31](#_Toc269924763)

[Supplementary Exercises 39](#_Toc269924764)

[Solutions for Chapter 6: Feature Analysis and Pattern Classification 43](#_Toc269924765)

[Supplementary Exercises 49](#_Toc269924766)

[Solutions for Chapter 7: Digital Image Processing and Visual Perception 62](#_Toc269924767)

[Supplementary Exercises 68](#_Toc269924768)

[Solutions for Chapter 8: Image Enhancement 70](#_Toc269924769)

[Supplementary Exercises 81](#_Toc269924770)

[Solutions for Chapter 9: Image Restoration 84](#_Toc269924771)

[Supplementary Exercises 93](#_Toc269924772)

[Solutions for Chapter 10: Image Compression 96](#_Toc269924773)

[Supplementary Exercises 108](#_Toc269924774)

[CVIPlab with Sample Programs and Setup Functions 112](#_Toc269924775)

[CVIPlab Header File 131](#_Toc269924776)

[Threshold Function 132](#_Toc269924777)

[Binary Features Function 134](#_Toc269924778)

[Labeling Function 136](#_Toc269924779)

[Euler Number Function 140](#_Toc269924780)

[Complement Function 142](#_Toc269924781)

[Logic Functions 144](#_Toc269924782)

[Subtraction Function 146](#_Toc269924783)

[Addition Function 148](#_Toc269924784)

[Rotate Function 150](#_Toc269924785)

[Convolution Mask (Filter) Function 153](#_Toc269924786)

[Histogram Stretch/Shrink Function 162](#_Toc269924787)

[Histogram Slide Function 164](#_Toc269924788)

[Robert’s Edge Detector Function 167](#_Toc269924789)

[Sobel Edge Detector Function 169](#_Toc269924790)

[Template Matching Function 172](#_Toc269924791)

[CVIPlab with Sample CVIPtools Libraries Function Calls 176](#_Toc269924792)

## Solutions for Chapter 1: Introduction to Computer Imaging

1. *Digital image processing* is also referred to as *computer imaging* and can be defined as the acquisition and processing of visual information by computer. It can be divided into application areas of computer vision and human vision; where in computer vision applications the end user is a computer and in human vision applications the end user is a human. Image analysis ties these two primary application areas together, and can be defined as the examination of image data to solve a computer imaging problem. A computer vision system can be thought of as a deployed image analysis system. To develop human vision applications requires extensive use of image analysis and its methods.

2. In general, a computer vision system has an imaging device, such as a camera, and a computer running analysis software to perform a desired task. Such as: A system to inspect parts on an assembly line. A system to aid in the diagnosis of cancer via MRI images. A system to automatically navigate a vehicle across Martian terrain. A system to inspect welds in an automotive assembly factory.

3. The image analysis process requires the use of tools such as image segmentation, image transforms, feature extraction and pattern classification. *Image segmentation* is often one of the first steps in finding higher level objects from the raw image data. *Feature extraction* is the process of acquiring higher level image information, such as shape or color information, and may require the use of *image transforms* to find spatial frequency information. *Pattern classification* is the act of taking this higher level information and identifying objects within the image.

4. The major topics within the field of image processing include image restoration, image enhancement, and image compression. IP applications: restore old, degraded photographs; restore satellite images distorted by mechanical jitter on a spacecraft; sharpen an image to bring out details; compressing images in a way so they still look good; for special effect sin movies, etc

5. These questions will need to be considered: How much compression do we need? What quality of image do we need? How will we measure that quality? What visual information is important for this application? Do we need to be able to recreate the image exactly, or will an approximation do?

6. Image restoration requires knowledge of the degradation process and uses a model to reverse the distortion. Image enhancement takes advantage of the human visual system’s response and creates an image looks better, so does not model the “distortion”.

**Sample CVIPlab Solutions to Programming Exercises.**

### CVIPlab with Sample Programs and Setup Functions

/\*=========================================================================

\*

\* Computer Vision and Image Processing Lab - Dr. Scott Umbaugh SIUE

\*

\* =========================================================================

\*

\* File Name: CVIPlab.c

\* Description: This file contains the skeleton program CVIPlab

\* with added Setup functions for programming

\* exercises from Chapters 3,4,6 and 8

\* Initial Coding Date: June 26, 2004

\* Last update Date: April 2, 2009

\* Portability: Standard (ANSI) C

\* Credit(s): Scott Umbaugh,Zhen Li, Kun Luo, Dejun Zhang

\* Southern Illinois University Edwardsville

\* Name: Husain Kagalwalla

\*

\*\* Copyright (c) 1995, 1996, 2004, 2010 SIUE - Scott Umbaugh

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

/\*

\*\* include header files

\*/

#include "CVIPtoolkit.h"

#include "CVIPconvert.h"

#include "CVIPdef.h"

#include "CVIPimage.h"

#include "CVIPlab.h"

#define CASE\_MAX 20

/\* Put the command here, as VIDEO\_APP, to run your image acquisition

application program \*/

#define VIDEO\_APP "PC-VCR.exe"

/\*

\*\* function declarations

\*/

Image \*threshold\_Setup(Image \*inputImage);

Image \*binaryFeatures\_Setup(Image \*inputImage);

Image \*labeling\_Setup(Image \*inputImage);

Image \*Eulernumber\_Setup(Image \*inputImage);

Image \*complement\_Setup(Image \*inputImage);

Image \*logic\_Setup(Image \*inputImage1, Image \*inputImage2);

Image \*subtraction\_Setup(Image \*inputImage1, Image \*inputImage2);

Image \*addition\_Setup(Image \*inputImage1, Image \*inputImage2);

Image \*zoom\_Setup(Image \*cvipImage );

Image \*rotate\_Setup(Image \*inputImage);

Image \*mask\_Setup(Image \*inputImage);

Image \*histogram\_Setup(Image \*inputImage);

Image \*Histogram\_lab2(Image \*inputImage);

Image \*Robert\_Setup(Image \*inputImage);

Image \*Sobel\_Setup(Image \*inputImage);

static int Template\_Match\_Setup(Image\*\* image, Image\*\* template, float\* threshold, int \*method);

Image \*input();

/\*

\*\* start main function

\*/

void main\_cviplab(){

IMAGE\_FORMAT format; /\* the input image format \*/

Image \*cvipImage, \*cvipImage1,\*cvipImage2;

/\* pointer to the CVIP Image structure \*/

Image \*templateImage; /\* template Image for matching \*/

float matchThreshold; /\* % threshold of posible maximum error\*/

/\* A match will occur when the error \*/

/\* measure is less than this threshold \*/

int errorMeasure; /\* indicates error measure method \*/

/\* can be EUCLIDEAN\_DISTANCE or \*/

/\* CITY\_BLOCK\_DISTANCE \*/

char \*outputfile; /\* output file name \*/

int choice;

CVIP\_BOOLEAN done = CVIP\_NO; /\* end of menu loop variable \*/

print\_CVIP("\n\n\n\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

print\_CVIP("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ");

print\_CVIP("\n\*\t\t Computer Vision and Image Processing Lab\t \*");

print\_CVIP("\n\*\t\t\t <Husain A. Kagalwalla> \t\t \*");

print\_CVIP("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

print\_CVIP("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n\n");

while(!done) {

print\_CVIP("\t\t0.\tExit \n\n");

print\_CVIP("\t\t1.\tGrab and Snap an Image \n\n");

print\_CVIP("\t\t2.\tThreshold Operation \n\n");

print\_CVIP("\t\t3.\tbinaryFeatures \n\n");

print\_CVIP("\t\t4.\tlabeling \n\n");

print\_CVIP("\t\t5.\tEulernumber \n\n");

print\_CVIP("\t\t6.\tcomplement \n\n");

print\_CVIP("\t\t7.\tlogic \n\n");

print\_CVIP("\t\t8.\tsubtraction \n\n");

print\_CVIP("\t\t9.\taddition \n\n");

print\_CVIP("\t\t10.\tzoom \n\n");

print\_CVIP("\t\t11.\trotate \n\n");

print\_CVIP("\t\t12.\tmask \n\n");

print\_CVIP("\t\t13.\thistogram \n\n");

print\_CVIP("\t\t14.\tSelect Unsharp Masking \n\n");

print\_CVIP("\t\t15.\tRoberts Edge Detection \n\n");

print\_CVIP("\t\t16.\tSobels Edge Detection \n\n");

print\_CVIP("\t\t17.\tTemplate Matching \n\n");

print\_CVIP("\n\nCVIPlab>>");

/\*

\*\* obtain an integer between 0 and CASE\_MAX from the user

\*/

choice = getInt\_CVIP(10, 0, CASE\_MAX);

switch(choice) {

case 0:

done=CVIP\_YES;

break;

case 1:

if ( ShellExecute(NULL,"Open",VIDEO\_APP,NULL,NULL, SW\_SHOW )<= 32)

print\_CVIP("Error while running Video Program");

break;

case 2:

/\*Get the input image \*/

cvipImage = input();

if(cvipImage == NULL) {

error\_CVIP("main", "could not read input image");

break;

}

/\* calls the threshold function \*/

cvipImage = threshold\_Setup(cvipImage);

if (!cvipImage) {

perror\_CVIP("main", "threshold fails");

break;

}

print\_CVIP("\n\t\tEnter the Output File Name: ");

outputfile = getString\_CVIP();

/\*

\*\* display the resultant image

\*/

view\_Image(cvipImage,outputfile);

/\*

\*\* saves the resulting image data out to disk in <outputfile>

\*/

format = getFileFormat\_Image(cvipImage);

write\_Image(cvipImage, outputfile, CVIP\_NO, CVIP\_NO, format, 1);

/\*

\*\*IMPORTANT: free the dynamic allocated memory when it is not

\*\* needed

\*/

free(outputfile);

break;

case 3:

/\*Get the input image \*/

cvipImage = input();

if(cvipImage == NULL) {

error\_CVIP("main", "could not read input image");

break;

}

/\* calls the binary Features \*/

binaryFeatures\_Setup(cvipImage);

break;

case 4:

/\*Get the input image \*/

cvipImage = input();

if(cvipImage == NULL) {

error\_CVIP("main", "could not read input image");

break;

}

/\* calls the labeling function \*/

cvipImage = labeling\_Setup(cvipImage);

if (!cvipImage) {

error\_CVIP("main", "labeling fails");

break;

}

print\_CVIP("\n\t\tEnter the Output File Name: ");

outputfile = getString\_CVIP();

/\*

\*\* display the resultant image

\*/

view\_Image(cvipImage,outputfile);

/\*

\*\* saves the resulting image data out to disk in <outputfile>

\*/

format = getFileFormat\_Image(cvipImage);

write\_Image(cvipImage, outputfile, CVIP\_NO, CVIP\_NO, format, 1);

/\*

\*\*IMPORTANT: free the dynamic allocated memory when it is not

\*\* needed

\*/

free(outputfile);

break;

case 5:

/\*Get the input image \*/

cvipImage = input();

if(cvipImage == NULL) {

error\_CVIP("main", "could not read input image");

break;

}

/\* calls the Eulernumber setup function \*/

Eulernumber\_Setup(cvipImage);

break;

case 6:

/\*Get the input image \*/

cvipImage = input();

if(cvipImage == NULL) {

error\_CVIP("main", "could not read input image");

break;

}

/\* calls the complement function \*/

cvipImage = complement\_Setup(cvipImage);

if (!cvipImage) {

perror\_CVIP("main", "complement fails");

break;

}

print\_CVIP("\n\t\tEnter the Output File Name: ");

outputfile = getString\_CVIP();

/\*

\*\* display the resultant image

\*/

view\_Image(cvipImage,outputfile);

/\*

\*\* saves the resulting image data out to disk in <outputfile>

\*/

format = getFileFormat\_Image(cvipImage);

write\_Image(cvipImage, outputfile, CVIP\_NO, CVIP\_NO, format, 1);

/\*

\*\*IMPORTANT: free the dynamic allocated memory when it is not

\*\* needed

\*/

free(outputfile);

break;

case 7:

/\*Get the input image \*/

cvipImage1 = input();

if(cvipImage1 == NULL) {

error\_CVIP("main", "could not read input image");

break;

}

cvipImage2 = input();

if(cvipImage2 == NULL) {

error\_CVIP("main", "could not read input image");

break;

}

/\* calls the logic function \*/

cvipImage = logic\_Setup(cvipImage1, cvipImage2);

if (!cvipImage) {

perror\_CVIP("main", "logic fails");

break;

}

print\_CVIP("\n\t\tEnter the Output File Name: ");

outputfile = getString\_CVIP();

/\*

\*\* display the resultant image

\*/

view\_Image(cvipImage,outputfile);

/\*

\*\* saves the resulting image data out to disk in <outputfile>

\*/

format = getFileFormat\_Image(cvipImage);

write\_Image(cvipImage, outputfile, CVIP\_NO, CVIP\_NO, format, 1);

/\*

\*\*IMPORTANT: free the dynamic allocated memory when it is not

\*\* needed

\*/

free(outputfile);

break;

case 8:

/\*Get the input image \*/

cvipImage1 = input();

if(cvipImage1 == NULL) {

error\_CVIP("main", "could not read input image");

break;

}

cvipImage2 = input();

if(cvipImage2 == NULL) {

error\_CVIP("main", "could not read input image");

break;

}

/\* calls the subtraction function \*/

cvipImage = subtraction\_Setup(cvipImage1, cvipImage2);

if (!cvipImage) {

perror\_CVIP("main", "subtraction fails");

break;

}

print\_CVIP("\n\t\tEnter the Output File Name: ");

outputfile = getString\_CVIP();

/\*

\*\* display the resultant image

\*/

view\_Image(cvipImage,outputfile);

/\*

\*\* saves the resulting image data out to disk in <outputfile>

\*/

format = getFileFormat\_Image(cvipImage);

write\_Image(cvipImage, outputfile, CVIP\_NO, CVIP\_NO, format, 1);

/\*

\*\*IMPORTANT: free the dynamic allocated memory when it is not

\*\* needed

\*/

free(outputfile);

break;

case 9:

/\*Get the input image \*/

cvipImage1 = input();

if(cvipImage1 == NULL) {

error\_CVIP("main", "could not read input image");

break;

}

cvipImage2 = input();

if(cvipImage2 == NULL) {

error\_CVIP("main", "could not read input image");

break;

}

/\* calls the addition function \*/

cvipImage = addition\_Setup(cvipImage1, cvipImage2);

if (!cvipImage) {

perror\_CVIP("main", "addition fails");

break;

}

print\_CVIP("\n\t\tEnter the Output File Name: ");

outputfile = getString\_CVIP();

/\*

\*\* display the resultant image

\*/

view\_Image(cvipImage,outputfile);

/\*

\*\* saves the resulting image data out to disk in <outputfile>

\*/

format = getFileFormat\_Image(cvipImage);

write\_Image(cvipImage, outputfile, CVIP\_NO, CVIP\_NO, format, 1);

/\*

\*\*IMPORTANT: free the dynamic allocated memory when it is not

\*\* needed

\*/

free(outputfile);

break;

case 10:

/\*Get the input image \*/

cvipImage = input();

if(cvipImage == NULL) {

error\_CVIP("main", "could not read input image");

break;

}

/\* calls the zoom function \*/

cvipImage = zoom\_Setup(cvipImage);

if (!cvipImage) {

perror\_CVIP("main", "zoom fails");

break;

}

print\_CVIP("\n\t\tEnter the Output File Name: ");

outputfile = getString\_CVIP();

/\*

\*\* display the resultant image

\*/

view\_Image(cvipImage,outputfile);

/\*

\*\* saves the resulting image data out to disk in <outputfile>

\*/

format = getFileFormat\_Image(cvipImage);

write\_Image(cvipImage, outputfile, CVIP\_NO, CVIP\_NO, format, 1);

/\*

\*\*IMPORTANT: free the dynamic allocated memory when it is not

\*\* needed

\*/

free(outputfile);

break;

case 11:

/\*Get the input image \*/

cvipImage = input();

if(cvipImage == NULL) {

error\_CVIP("main", "could not read input image");

break;

}

/\* calls the rotate function \*/

cvipImage = rotate\_Setup(cvipImage);

if (!cvipImage) {

perror\_CVIP("main", "rotate fails");

break;

}

print\_CVIP("\n\t\tEnter the Output File Name: ");

outputfile = getString\_CVIP();

/\*

\*\* display the resultant image

\*/

view\_Image(cvipImage,outputfile);

/\*

\*\* saves the resulting image data out to disk in <outputfile>

\*/

format = getFileFormat\_Image(cvipImage);

write\_Image(cvipImage, outputfile, CVIP\_NO, CVIP\_NO, format, 1);

/\*

\*\*IMPORTANT: free the dynamic allocated memory when it is not

\*\* needed

\*/

free(outputfile);

break;

case 12:

/\*Get the input image \*/

cvipImage = input();

if(cvipImage == NULL) {

error\_CVIP("main", "could not read input image");

break;

}

/\* calls the mask function \*/

cvipImage = mask\_Setup(cvipImage);

if (!cvipImage) {

perror\_CVIP("main", "mask fails");

break;

}

print\_CVIP("\n\t\tEnter the Output File Name: ");

outputfile = getString\_CVIP();

/\*

\*\* display the resultant image

\*/

view\_Image(cvipImage,outputfile);

/\*

\*\* saves the resulting image data out to disk in <outputfile>

\*/

format = getFileFormat\_Image(cvipImage);

write\_Image(cvipImage, outputfile, CVIP\_NO, CVIP\_NO, format, 1);

/\*

\*\*IMPORTANT: free the dynamic allocated memory when it is not

\*\* needed

\*/

free(outputfile);

break;

case 13:

/\*Get the input image \*/

cvipImage = input();

if(cvipImage == NULL) {

error\_CVIP("main", "could not read input image");

break;

}

/\* calls the histogram function \*/

cvipImage = histogram\_Setup(cvipImage);

if (!cvipImage) {

perror\_CVIP("main", "histogram fails");

break;

}

print\_CVIP("\n\t\tEnter the Output File Name: ");

outputfile = getString\_CVIP();

/\*

\*\* display the resultant image

\*/

view\_Image(cvipImage,outputfile);

/\*

\*\* saves the resulting image data out to disk in <outputfile>

\*/

format = getFileFormat\_Image(cvipImage);

write\_Image(cvipImage, outputfile, CVIP\_NO, CVIP\_NO, format, 1);

/\*

\*\*IMPORTANT: free the dynamic allocated memory when it is not

\*\* needed

\*/

free(outputfile);

break;

case 14:

/\*Get the input image \*/

cvipImage = input();

if(cvipImage == NULL) {

error\_CVIP("main", "could not read input image");

break;

}

/\* calls the histogram function \*/

cvipImage = Histogram\_lab2(cvipImage);

if (!cvipImage) {

perror\_CVIP("main", "histogram fails");

break;

}

print\_CVIP("\n\t\tEnter the Output File Name: ");

outputfile = getString\_CVIP();

/\*

\*\* display the resultant image

\*/

view\_Image(cvipImage,outputfile);

/\*

\*\* saves the resulting image data out to disk in <outputfile>

\*/

format = getFileFormat\_Image(cvipImage);

write\_Image(cvipImage, outputfile, CVIP\_NO, CVIP\_NO, format, 1);

/\*

\*\*IMPORTANT: free the dynamic allocated memory when it is not

\*\* needed

\*/

free(outputfile);

break;

case 15:

/\*Get the input image \*/

cvipImage = input();

if(cvipImage == NULL) {

error\_CVIP("main", "could not read input image");

break;

}

/\* calls the Robert's function \*/

cvipImage = Robert\_Setup(cvipImage);

if (!cvipImage) {

perror\_CVIP("main", "Robert fails");

break;

}

print\_CVIP("\n\t\tEnter the Output File Name: ");

outputfile = getString\_CVIP();

/\*

\*\* display the resultant image

\*/

view\_Image(cvipImage,outputfile);

/\*

\*\* saves the resulting image data out to disk in <outputfile>

\*/

format = getFileFormat\_Image(cvipImage);

write\_Image(cvipImage, outputfile, CVIP\_NO, CVIP\_NO, format, 1);

/\*

\*\*IMPORTANT: free the dynamic allocated memory when it is not

\*\* needed

\*/

free(outputfile);

break;

case 16:

/\*Get the input image \*/

cvipImage = input();

if(cvipImage == NULL) {

error\_CVIP("main", "could not read input image");

break;

}

/\* calls the Sobel's function \*/

cvipImage = Sobel\_Setup(cvipImage);

if (!cvipImage) {

perror\_CVIP("main", "Sobel fails");

break;

}

print\_CVIP("\n\t\tEnter the Output File Name: ");

outputfile = getString\_CVIP();

/\*

\*\* display the resultant image

\*/

view\_Image(cvipImage,outputfile);

/\*

\*\* saves the resulting image data out to disk in <outputfile>

\*/

format = getFileFormat\_Image(cvipImage);

write\_Image(cvipImage, outputfile, CVIP\_NO, CVIP\_NO, format, 1);

/\*

\*\*IMPORTANT: free the dynamic allocated memory when it is not

\*\* needed

\*/

free(outputfile);

break;

case 17: /\* Template Matching \*/

if (0 == (Template\_Match\_Setup(

&cvipImage, &templateImage,

&matchThreshold, &errorMeasure))) {

template\_match\_lab(cvipImage, templateImage,

matchThreshold, errorMeasure);

}

break;

default:

print\_CVIP("Sorry ! You Entered a wrong choice ");

break;

}

}

}

/\*

\*\* end of the function main

\*/

/\*

\*\* The following function reads in the image file specified by the user,

\*\* stores the data and other image info. in a CVIPtools Image structure,

\*\* and displays the image.

\*/

Image \*input(){

char \*inputfile;

Image \*cvipImage;

/\*

\*\* get the name of the file and stores it in the string 'inputfile '

\*/

print\_CVIP("\n\t\tEnter the Input File Name: ");

inputfile = getString\_CVIP();

/\*

\*\* creates the CVIPtools Image structure from the input file

\*/

cvipImage = read\_Image(inputfile, 1);

if(cvipImage == NULL) {

error\_CVIP("init\_Image", "could not read image file");

free(inputfile);

return NULL;

}

/\*

\*\* display the source image

\*/

view\_Image(cvipImage,inputfile);

/\*

\*\*IMPORTANT: free the dynamic allocated memory when it is not needed

\*/

free(inputfile);

return cvipImage;

}

/\*

\*\* The following setup function asks the threshold value from the user. After

\*\* it gets the threshold value, it will call the threshold\_Image() function.

\*/

Image \*threshold\_Setup(Image \*inputImage){

unsigned int threshval; /\* Threshold value \*/

/\*

\*\* Gets a value between between 0 and 255 for threshsold

\*/

print\_CVIP("\n\t\tEnter the threshold value: ");

threshval = getInt\_CVIP(10, 0, 255);

return threshold\_lab(inputImage, threshval);

}

Image \*binaryFeatures\_Setup(Image \*inputImage){

return binaryFeatures\_lab(inputImage);

}

Image \*labeling\_Setup(Image \*inputImage){

return labeling\_lab(inputImage);

}

Image \*Eulernumber\_Setup(Image \*inputImage){

return Eulernumber\_lab(inputImage);

}

Image \*complement\_Setup(Image \*inputImage){

unsigned int pixel;

print\_CVIP("\n\t\tEnter the Pixel value: ");

pixel = getInt\_CVIP(10, 0, 255);

return complement\_lab(inputImage, pixel);

}

Image \*logic\_Setup(Image \*inputImage1, Image \*inputImage2){

unsigned int operator;

print\_CVIP("\n\t\t What do you want to do 1. and /2. or /3. xor ?");

operator=getInt\_CVIP(10,0,255);

return logic\_lab(inputImage1, inputImage2, operator);

}

Image \*subtraction\_Setup(Image \*inputImage1, Image \*inputImage2){

return subtraction\_lab(inputImage1, inputImage2);

}

Image \*addition\_Setup(Image \*inputImage1, Image \*inputImage2){

return addition\_lab(inputImage1, inputImage2);

}

/\*\* The following setup function calls the zoom function.

\*\*/

Image \*zoom\_Setup(Image \*cvipImage ){

int r,c,w,h,q ;

float f ;

r=c=w=h= -1;

/\* The following statement will print the statement in the

\*\* selection menu

\*/

print\_CVIP("\n\tEnter the quad no. : ");

print\_CVIP("\n\tEnter 1 for upper right ");

print\_CVIP("\n\tEnter 2 for upper left ");

print\_CVIP("\n\tEnter 3 for lower left ");

print\_CVIP("\n\tEnter 4 for lower right ");

print\_CVIP("\n\tEnter 5 for full image ");

print\_CVIP("\n\tEnter 6 for part of Image you want : ");

q = getInt\_CVIP(10,1,6);

print\_CVIP("\n\tEnter the factor value: ");

f = getFloat\_CVIP(1,10);

/\* If we select the part of the image option then we have to

\*\* enter the row, column numbers and width and height values

\*\* we want to zoom.

\*/

if (q==6)

{

print\_CVIP("\n\tEnter the column value: ");

c = getInt\_CVIP(10,0,255);

print\_CVIP("\n\tEnter the row value: ");

r = getInt\_CVIP(10,0,255);

print\_CVIP("\n\tEnter the width value: ");

w = getInt\_CVIP(10,0,255);

print\_CVIP("\n\tEnter the height value: ");

h = getInt\_CVIP(10,0,255);

}

/\* The following function will return the zoomed image

\*/

return zoom(cvipImage,q,c,r,w,h,f ) ;

}

/\*

\*\* The following setup function asks the image from the user. After

\*\* it gets the image it asks the value for degree of rotation.

\*/

Image \*rotate\_Setup(Image \*inputImage){

float degree;

print\_CVIP("\n\t\tEnter the degree of rotation: ");

degree = getFloat\_CVIP(0, 360);

return rotate\_lab(inputImage, degree);

}

Image \*mask\_Setup(Image \*inputImage){

int choice;

print\_CVIP("\n\t\t 1. Low Pass Filter");

print\_CVIP("\n\t\t 2. High Pass Filter");

print\_CVIP("\n\t\t 3. User input for a 3\*3 mask");

print\_CVIP("\n\n\t\t Enter your choice: ");

choice = getInt\_CVIP(10,0,3);

return mask1\_lab(inputImage, choice);

}

/\*

\*\* The following setup function is a histogram setup function

\*/

Image \*histogram\_Setup(Image \*cvipImage){

int choice, max, min, offset ;

print\_CVIP("\n\t\t 1. Stretch/Shrink ");

print\_CVIP("\n\t\t 2. Slide");

print\_CVIP("\n\n\t\t Enter your choice: ");

choice = getInt\_CVIP(10,0,2);

if (choice == 1)

{

print\_CVIP("\n\t\t input max value ");

max = getInt\_CVIP(10, 0,255);

print\_CVIP("\n\t\t input min value ");

min = getInt\_CVIP(10, 0,255);

return stretch\_shrink\_lab(cvipImage, max, min);

}

if (choice == 2)

{

print\_CVIP("\n\t\t input offset value ");

offset = getInt\_CVIP(10, -255, 255);

return slide\_lab(cvipImage, offset);

}

}

Image \*Histogram\_lab2(Image \*inputImage){

char \*inputfile,\*outputfile;

Image \*copyimage, \*copyimage1;

float low\_clip, high\_clip;

int min, max;

IMAGE\_FORMAT format;

copyimage = duplicate\_Image(inputImage);

if(copyimage == NULL)

{

error\_CVIP("init\_Image", "could not read image file");

free(inputfile);

return NULL;

}

copyimage1= mask1\_lab(copyimage,1);

print\_CVIP("\n\t\t Enter the Output File Name: ");

outputfile = getString\_CVIP();

view\_Image(copyimage1, outputfile); /\*display the resultant image\*/

format = getFileFormat\_Image(copyimage1);

write\_Image(copyimage1, outputfile, CVIP\_YES, CVIP\_NO, format, 1); /\* save the image\*/

free(outputfile);

/\*Performing Histogram Shrink\*/

print\_CVIP("\n\n\t\t Enter the Minimum Value (between 0 to 255): ");

min = getInt\_CVIP(10,0,255);

print\_CVIP("\n\n\t\t Enter the Maximum Value (between 0 to 255): ");

max = getInt\_CVIP(10,0,255);

copyimage1 = stretch\_shrink\_lab(copyimage1, max, min);

print\_CVIP("\n\t\t Enter the Output File Name: ");

outputfile = getString\_CVIP();

view\_Image(copyimage1, outputfile); /\*display the resultant image\*/

format = getFileFormat\_Image(copyimage1);

write\_Image(copyimage1, outputfile, CVIP\_YES, CVIP\_NO, format, 1); /\* save the image\*/

free(outputfile);

/\* Subtracting the images\*/

print\_CVIP("\n\n\t\tPlease perform the Subtraction\n\n");

copyimage1 = subtraction\_lab(copyimage,copyimage1);

print\_CVIP("\n\t\t Enter the Output File Name: ");

outputfile = getString\_CVIP();

view\_Image(copyimage1, outputfile); /\*display the resultant image\*/

format = getFileFormat\_Image(copyimage1);

write\_Image(copyimage1, outputfile, CVIP\_YES, CVIP\_NO, format, 1); /\* save the image\*/

free(outputfile);

/\* Histogram Stretch performance\*/

/\*print\_CVIP("\n\n\t\t Enter the Minimum Value: ");

\*\*min = getInt\_CVIP(10,0,255);

\*\*print\_CVIP("\n\n\t\t Enter the Maximum Value: ");

max = getInt\_CVIP(10,0,255); \*/

print\_CVIP("\n\n\t\t Enter the low clip value from %f to %f: ", 0.0, 1.0);

low\_clip = getFloat\_CVIP(0.0,1.0);

print\_CVIP("\n\n\t\t Enter the high clip value from %f to %f: ", 0.0, 1.0-low\_clip);

high\_clip = getFloat\_CVIP(0.0, 1.0-low\_clip);

copyimage1 = hist\_stretch(copyimage1, min, max, low\_clip, high\_clip);

return(copyimage1);

}

Image \*Robert\_Setup(Image \*inputImage){

return robert\_lab(inputImage);

}

Image \*Sobel\_Setup(Image \*inputImage){

return sobel\_lab(inputImage);

}

/\*

\*\* Function: Template\_Match\_Setup()

\*\* Description: Get input image, template image and match threshold

\*\* Parameters: Image \*\*image: get from user the image to be matched

\*\* Image \*\*template: get from user the template image

\*\* float \*threshold: get from user the % threshold

\*\* int errorMeasure: indicates error measure method

\*\* can be EUCLIDEAN\_DISTANCE or CITY\_BLOCK\_DISTANCE

\*\* Return: 0: indicates success. -1: indicates error

\*/

static int Template\_Match\_Setup (Image\*\* image, Image\*\* template,

float\* threshold, int \*method){

char \*inputfile1, \*inputfile2;

char temp[256]; /\* holds temporary print string \*/

print\_CVIP("\n");

/\*

\*\* get the name of the file and stores it in the string 'inputfile '

\*/

sprintf(temp, "%50s", "Enter the Input File Name: ");

print\_CVIP(temp);

inputfile1 = getString\_CVIP();

/\*

\*\* creates the CVIPtools Image structure from the input file

\*/

\*image = read\_Image(inputfile1, 1);

if(\*image == NULL) {

error\_CVIP("init\_Image", "could not read image file");

free(inputfile1);

return -1;

}

/\*

\*\* get the name of the file and stores it in the string 'inputfile '

\*/

sprintf(temp, "%50s", "Enter the Template File Name: ");

print\_CVIP(temp);

inputfile2 = getString\_CVIP();

/\*

\*\* creates the CVIPtools Image structure from the input file

\*/

\*template = read\_Image(inputfile2, 1);

if(\*template == NULL) {

error\_CVIP("init\_Image", "could not read template image file");

free(inputfile1);

free(inputfile2);

return -1;

}

/\*

\*\* get the threshold

\*/

sprintf(temp, "%50s", "Enter the threshold ratio (0.0 to 1.0): ");

print\_CVIP(temp);

\*threshold = getFloat\_CVIP(0.0, 1.0);

/\*

\*\* get the error measure

\*/

sprintf(temp, "%50s", "Square Root(0) or Absolute(1) method: ");

print\_CVIP(temp);

/\* getInt\_CVIP(base, low\_limit, up\_limit) \*/

\*method = (int)getInt\_CVIP(10, 0, 1);

/\*

\*\* display the source image

\*/

view\_Image(\*image, inputfile1);

/\*

\*\* display the template image

\*/

view\_Image(\*template, inputfile2);

/\* Clean Up \*/

free(inputfile1);

free(inputfile2);

return 0; /\* success \*/

}