P-1196

GENERATING IMAGE PROCESSING ALGORITHMS FOR DSPs and FPGAs

PROJECT REPORT

Submitted in partial fulfillment of the Requirement
for the award of the degree of the

**Bachelor of Computer Science
and Engineering
Of
Bharathiyar University, Coimbatore.**

Submitted by

**Vasanth Prabhu.S
0027KO209**

Under the Guidance of

**Mrs. M. S. HEMA B.E.,
Lecturer**

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

**KUMARAGURU COLLEGE OF TECHNOLOGY
COIMBATORE – 641 006.**

MARCH 2004

CERTIFICATE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY
(Affiliated to Bharathiyar University, Coimbatore)

CERTIFICATE

This is to certify that the project entitled

GENERATING IMAGE PROCESSING ALGORITHMS FOR DSPs and FPGAs

is done by

Vasanth Prabhu. S
0027K0209

and submitted in partial fulfillment of the Requirement
for the award of the degree of the

Bachelor of Computer Science and Engineering
Of
Bharathiyar University, Coimbatore.

[illegible] [illegible]

Professor & Head of the department **Guide**
(Dr. S. THANGASAMY) **(Mrs. M. S. HEMA)**

Certified that the candidate was examined by us in the project work
Viva voce examination held on 23-03-2004

[illegible] 23/3/'04 [illegible] 23/3

Internal Examiner **External Examiner**

Soliton
True engineering

CERTIFICATE OF PROJECT COMPLETION

This is to certify that ***Mr. S.Vasanth Prabhu,*** a student of Kumaraguru College of Technology, Coimbatore, doing his final year, Bachelor of Engineering in Computer Science Engineering has successfully completed his final year project in our Company from November 01st, 2003 to March 01st, 2004.

During this period he worked on the design and implementation of ***Image processing algorithms for DSP and FPGA***. He did the following activities:

- Studied various image processing algorithms, which was selected for use with DSP and FPGA chips.
- Studied various image representation formats and designed a good data structure.

He is very sound in programming and mathematical concepts and applied it well in design and development of the library. He is a good hard worker and has good communication skills.

We thank him for his good work and wish Mr. Vasanth Prabhu all the very best for his future.

Date: March 01st, 2004

Anand Prasad Chinnaswamy

[illegible]

Director & CTO

DECLARATION

DECLARATION

I,

S. Vasanth Prabhu   0027K0209

declare that the project entitled "Generating Image Processing Algorithms for DSPs and FPGAs' is done by me and to the best of my knowledge a similar work has not been submitted to the Bharathiyar University or any other institution, for fulfillment of the requirement of the course study.

This report is submitted on the partial fulfillment of the requirements for all awards of degree of Bachelor of Computer Science and Engineering of Bharathiyar University.

Place: COIMBATORE

Date: 16-03-2004

S. Vasanth Prabhu

S. Vasanth Prabhu

Countersigned: [illegible]

GUIDE: Mrs. M. S. HEMA, B. E.

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

The exhilaration achieved upon the successful completion of any task should be definitely shared with the people behind the venture. This project is an amalgam of study and experience of many people without whose help this project would not have taken shape.

At the onset, I take this opportunity to thank the management of my college for having provided us excellent facilities to work with. I express my deep gratitude to our Principal Dr.K.K.Padmanabhan B.Sc (Engg.), M.Tech., for ushering us in the path of triumph.

I am always thankful to our beloved Professor and the Head of the Department, Prof.S.Thangasamy B.E. (HONS)., whose consistent support and enthusiastic involvement helped us a great deal.

I am greatly indebted to my beloved guide Mrs.M.S.Hema B.E., Lecturer, Department of Computer Science and Engineering for her excellent guidance and timely support during the course of this project. As a token of my esteem and gratitude, I honor her for her assistance towards this cause.

I also thank my project coordinator Ms.D.Chandrakala M.E., and my beloved class advisor Mrs.M.S.Hema, B.E., for their invaluable assistance.

I also feel elated in manifesting my deep sense of gratitude to all the staff and lab technicians in the Department of Computer Science and Engineering.

I feel proud to pay my respectful thanks to my Parents for their enthusiasm and encouragement and also I thank my friends who have associated themselves to bring out this project successfully.

SYNOPSIS

SYNOPSIS

My project, **Generating Image processing algorithms for DSPs and FPGAs,** aims in the implementation of the entire standard image processing functions in a middle level language(C). The reason for implementing them in middle-level language is that 'all the image processing softwares provide high-level interaction with the user but when going for the interface with the hardware part, they abruptly fail. This is because interfacing with hardware needs these functions to be implemented using middle-level languages and not by using built in software add ons.

The list of image processing functions implemented are:

1. Threshold
2. Edge Detection
3. Finding holes
4. Filling holes
5. Image inversion
6. Operators
7. Obtaining image information
8. Reading and writing images
9. Color to grayscale conversion
10. Extracting color planes
11. Image copying

Some of these functions are implemented for both color and grayscale images. In case of color images there are 24 bits per pixel whereas in case of grayscale images there are only 8 bits per pixel. Also the header portions of the color and grayscale images vary.

CONTENTS

CONTENTS

1. Introduction **1**

1.1 Existing system and limitations

1.2 Proposed system and advantages

2. Company Profile **3**

3. Software requirement and analysis **5**

3.1 Product definition

3.2 Project plan

4. Software requirement specifications **7**

4.1 Purpose

4.2 Scope

4.3 Product overview and summary

4.4 Development and operating environment

4.5 External interfaces and data flow

4.6 Functional specifications

4.7 Exceptional handling

4.8 Optimization and future enhancements

5. Proposed approach to the system **12**

5.1 Problem definition

5.2 Existing system

5.3 Need for proposed system

6. System design **14**

6.1 Input design

6.2 Output design

7. System testing **18**

7.1 Testing objectives

7.2 Testing principles

7.3 Levels of testing

8. Future enhancements **22**

9. Conclusion **24**

10. Bibliography **26**

11. Annexure **28**

11.1 Source code

11.2 Output screens

INTRODUCTION

1. INTRODUCTION

My project deals with the implementation of image processing functions in a middle-level language.

1.1 Existing system and limitations:

Though a number of image processing softwares are available in the market for the processing the images, there are a lot of disadvantages in going for them. They are:

1. First of all, they can be used only with a specific other software. No software is available to satisfy the needs of an individual user.

2. Next, in order to implement simple functions on the image, the user have to buy the entire software, which is costly and consist of unnecessary functions.

3. Finally, all the image-processing softwares provide high-level interaction with the user but when going for the interface with the hardware part, they abruptly fail. This is because interfacing with hardware needs these functions to be implemented using middle-level languages and not by using built in software add ons.

1.2 Proposed system and advantages:

The ultimate solution for the above problems is to code the entire image processing functions using middle-level languages as individual modules and to use them as and when needed. This will provide the following advantages.

1. The user need not buy softwares to implement the functions. He can just download the modules and use them, which is very much cheaper.

2. They have good hardware compatibility since they are coded using middle-level languages.

2. Any type of software can be developed using these modules by just integrating them together. Even a standalone user could build his/her own software if needed.

COMPANY PROFILE

2. COMPANY PROFILE

SOLITON AUTOMATION PVT. INDIA LTD.
(True Engineering)

Bangalore

Soliton Automation Pvt. India Ltd., is one of the leading companies in the following fields in the whole of India.

1. Virtual Instrumentation
2. Machine Vision
3. Embedded Systems
4. Image Processing

Soliton is steering ahead in the field of embedded control design with its team of dedicated, intelligent and positive thinking to global customers. The company deals with projects based on embedded systems and implement them using a team of experts. They not only deal with the projects but also with the training of students in the projects which ever they like to specialize.

The company has the state of art development tools including FPGA kits, Incircuit debugger and emulators. They provide cost effective and time bound solution to customers in the field of engineering.

Engineering Service:

With the state-of-art embedded development facility at Chennai and Bangalore they offer a diverse range of embedded services. They have the experience and technical competence in design prototyping, engineering and transferring technology in the field of virtual instrumentation and machine vision.

SOFTWARE REQUIREMENT ANALYSIS

3. SOFTWARE REQUIREMENT ANALYSIS

System study is an activity that encompasses most of the tasks that we have collectively called computer system engineering. System study is conducted with the following objectives.

- Identify the needs.
- Evaluate the system concept for feasibility.
- Perform economic and technical analysis.
- Allocate function to hardware, software, people and other system elements.
- Create a system definition that forms the foundation for all subsequent engineering works.

3.1 Product Definition:

The processing of images using the individual modules developed using a middle-level language provides a greater flexibility. If these functions are uploaded in the Internet, they can be downloaded and used as and when necessary. This makes the entire process simple. Also the codes implemented were time efficient that speedy processing is possible. They also provide a greater flexibility when interfaced with the hardware part.

3.2 Project plan:

The project mainly aims at developing time efficient codes for the entire set of basic standard image processing routines in a middle-level language. The middle-level language is turbo C, which is easier both in coding and also in debugging. Also the codes are designed such that they can be easily interfaced with the hardware side, that is, with DSPs and FPGAs.

SOFTWARE REQUIREMENT SPECIFICATION

4. SOFTWARE REQUIREMENT SPECIFICATION

4.1 Purpose:

The primary purpose of the Software Requirements Specification (SRS) is to document the previously agreed to functionality, external interfaces, attributes, and the performance of the **IMAGE PROCESSING ALGORITHMS.** This specification is the primary document upon which all of the subsequent design source code, and test plan will be based.

4.2 Scope:

The scope of this document, Software Requirement Specification (SRS) is to describe the requirements definition effort. The SRS documentation for Generating Image Processing Algorithms for DSPs and FPGAs describes the functions, external interfaces, attributes, and performance issues specified in the product definition.

4.3 Product overview and summary:

My product provides a reliable, robust and efficient means of processing of images. The project basically involves pixel manipulation of image file and writing them over to another image file. The system first reads the input image, separates the header and data portion, then performs the manipulation on the data portion, stores the output into another image file.

4.4 Development and Operating Environment:

The development environment gives the minimum hardware and software requirements.

Hardware Specification:

Processor – Pentium III

RAM – 256 MB

Hard Disk – 10 GB

Floppy Drive – 1.44 MB

Monitor – 14" Monitor

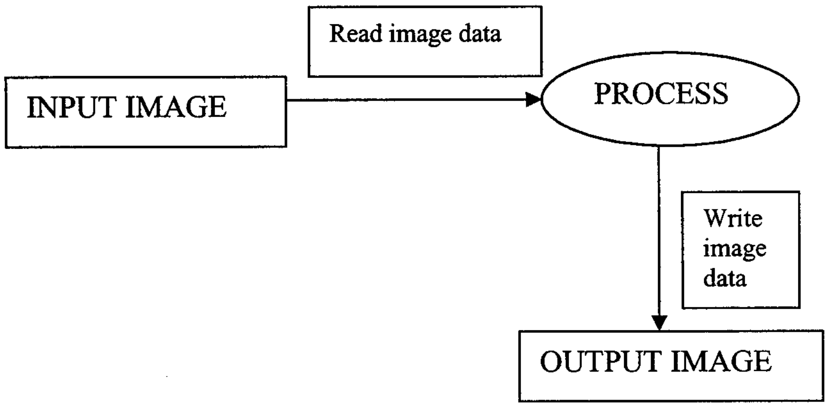
Software Specification:

Operating System – Windows 98

Platform – Turbo C

4.5 External interfaces and Dataflow:

The external interface is nothing but the input image file and output image file are mentioned as command line arguments in the DOS screen.



4.6 Functional specifications:

The description of the modules is as follows:

1. **Threshold** - If the intensities of pixels are greater than some user defined threshold level, they are kept at a maximum else they are kept 0. This makes bright points brighter and dark points darker.
2. **Edge Detection** - This function was implemented by using the concept that

the difference between adjacent pixel and intensities do not go over some maximum threshold level. This can be implemented for both color and grayscale images.

3. **Finding Holes**- The holes are those areas where the intensities of the pixels are less than 100. So each pixel value is obtained, checked for a hole and highlighted by making it a lower value of brightness.
4. **Filling Holes -** This function was implemented by choosing the pixels having intensities less than 150 and changing them to some maximum intensity.
5. **Image Inversion -** Here all the bytes are read separately. Each value will vary from 0 to 255. Subtracting the current value from 255 can do inversion. Image inversion can be applied for both color and grayscale images.
6. **Operators** - This function is the addition of some fixed color to the image. The color is obtained as a constant from the user and added to all pixel values.
7. **Obtaining Image Information -** Actually each image has header part, palette and data portion. Header part contains 15 fields describing the image. A structure can be used to read these fields and print them.
8. **Reading and Writing Images -** The header part occupies 54 bytes. The palette portion occupies 1024 bytes. The data portion occupies either 640*480 bytes or 640*480*3 bytes depending on either the image is grayscale or color image. The image file can be written by finding out whether the image is grayscale image or color image from the header part.
9. **Color to Grayscale Conversion -** A color image can be converted to grayscale image by just avoiding some header fields and palette information, and by just converting 3 bytes of R, G and B to a single byte by using some combination formula.
10. **Extracting color planes** - The intensities of each color (red, green or blue) can be obtained from a color image onto a grayscale image whose pixels have a value between 0 and 255. In case of color image, each pixel consists of red, green and blue color each of 8 bits or 1 byte. So if a red color plane is to be extracted alone the first byte of each pixel is read into a grayscale image.
11. **Copying Image -** This can be done just by copying the pixels from one image

file onto another. The header part of the image and the palette information should be written before the data is to be written.

4.7 Exception handling:

The package can handle only bitmap. If any other type of images are used a warning message will be displayed. The product can handle both the color and grayscale images.

4.8 Product Optimization:

The product is about to be implemented on 16-bit environment, if the same can be implemented on 32-bit environment the image processing will considerably get optimized.

PROPOSED APPROACH TO THE SYSTEM

5. PROPOSED APPROACH TO THE SYSTEM

5.1 Problem Definition:

The project can be used for a variety of applications such as finding out holes in rice in rice mills, finding out the alignment of a pencil lead in the center of a pencil for stability and so on.

5.2 Existing System:

In case of existing system, many types of software are available to implement these image-processing functions but they are not suitable for the above-mentioned applications. This is because interfacing with DSP chips in real time applications need these functions to be implemented in middle-level language like C.

5.3 Need for proposed system:

If the image processing functions are implemented in a middle-level language like C, then, there will be no problem in interfacing with hardware parts. The executables can easily be burnt inside the chips and are highly suited for real time applications. Also each module can individually be shipped into the Internet, which when needed by any user, can be downloaded quickly and used.

SYSTEM DESIGN

6. SYSTEM DESIGN

The system design phase is an interesting phase. It provides the way the information is to be fed and how the output is to be obtained. The design goes through logical and physical stages of development. Logical stage involves preparing the input and output preparation of the system. The physical stage details the hardware and the system implementations.

6.1 Input Design:

The input to the project is a bitmap image, either color or grayscale. The input image is passed as a command line argument. Now the header part of the input image is read. From the header part we are able to get the following information:

1. Type of bitmap
2. Size of file
3. Offset of data from this position
4. Size of rest of the header
5. Width of bitmap
6. Height of bitmap
7. Planes of bitmap
8. Number of bits per pixel
9. Types of compression
10. Size of image
11. Number of pixel per meter (X)
12. Number of pixel per meter (Y)
13. Number of colors used
14. Number of colors considered important

With the above information it is possible to process the entire image. The header portion of any bitmap image occupies 54 bytes. The remaining data portion occupies either 640 * 480 bytes or 640 * 480 * 3 bytes depending on whether the image is grayscale or color image. In case of grayscale image, the number of bits per pixel is 8, whereas in case of color image it is 24 with R, G and B components.

6.2 Output Design:

The output of this project is also an image. The image data portion gets modified according to the function used.

In case of threshold, all the pixel values greater than the threshold value are made 255 and those below the threshold value are made 0.

In case of edge detection, the difference between the adjacent pixels is calculated. If the difference exceeds a considerable limit, then it means that there is a transition from a higher to lower or a lower to higher intensity. So those pixel transitions must be an edge. Those pixels alone are made blacker.

In case of finding holes, all those pixels whose intensities are less than 50 are made to 0, since, pixels with intensities less than 50 will be black in color and hence they are holes.

In case of filling holes, all those pixels whose intensities are less than 50 are made to 255, since pixels with intensities less than 50 will be holes and 255 will be white color needed.

In case of image inversion, all the pixel values are subtracted from a value of 255. This makes the image invert.

Operators are nothing but addition, subtraction, multiplication and division operations can be performed on an image. A constant is obtained from the user and it is manipulated with all the pixels according to function needed.

Obtaining image information is nothing but reading the entire datum specified above from the image by using structures.

In case of Reading and Writing images, the header portion is first read from the input file and then the data portion is written into the output text file. The number of bytes written depends on whether the image is color or grayscale.

In case of color to grayscale conversion, the 3 bytes per pixel in the color image are read and converted into a single byte by using the standard conversion formula

GrayValue = (int)(0.299*redValue + 0.587*greenValue + 0.114*blueValue);

In case of extracting color planes, only the required byte of the 3 bytes(R, G or B) from each pixel is written into the output image.

The image copying function is performed by blindly reading all the bytes of the input image (irrespective of whether the image is grayscale or color) and writing into a new image file.

Since there is a wide variation in headers between the color and grayscale images, in case of writing a grayscale image the header portion alone is copied from a standard grayscale image.

Some of the above operations are performed for both color and grayscale images. Such functions are listed below:

1. Threshold
2. Edge detection
3. Finding holes
4. Filling holes
5. Image inversion
6. Operators
7. Image copying

In case of processing of data portion of color and grayscale images, the only difference between them is that the color image processing involves 3 times overhead than grayscale image. This is due to difference in number of bytes.

If the images other than bitmap image is specified, the project will display an error message. The header of bitmap differs from other image types.

Using command line arguments runs most of the modules. This is because, in case of real time applications the input and output image files, and the functions are called dynamically at run time. So the format of running the project is specifying the program name, the input image name and the output image name.

SYSTEM TESTING

7. SYSTEM TESTING

Testing is an activity to verify that a correct system is being built and is performed with the intent of finding faults in the system. Testing is an activity, however not restricted to being performed after the development phase is complete. But this is to be carried out in parallel with all stages of system development, starting with requirement specification. Testing results once gathered and evaluated, provide a qualitative indication of software quality and reliability and serve as a basis for design modification if required.

System Testing is a process of checking whether the development system is working according to the original objectives and requirements. The system should be tested experimentally with test data so as to ensure that the system works according to the required specification. When the system is found working, test it with actual data and check performance.

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. The increasing visibility of software as a system element and the attendant "cost" associated with a software failure are the motivation forces for a well planned, thorough testing.

6.1 Testing Objectives:

The testing objectives are summarized in the following three steps. Testing is the process of executing a program with the intent of finding an error. A good test case is one that has high probability of finding an error. A successful test is one that uncovers as –yet-undiscovered errors.

6.2 Testing Principles:

All tests should be traceable to customer requirements. Tests should be planned long before testing begins, that is, the test planning can begin as soon as the requirements model is complete. Testing should begin "in the small" and progress towards resting "in large". The focus of testing will shift progressively from programs to individual modules and finally to the entire project. Exhaustive testing is not possible. To be more effective, testing should be one, which has highest probability of finding errors.

The following are the attributes of good tests:

- A good test has a high probability of finding an error.
- A good test is not redundant.
- A good test should be "best of breed"
- A good test should be neither too simple nor too complex.

6.3 Levels of Testing:

The details of the software functionality tests are given below. The testing procedure that has been used is as follow:

- Unit Testing
- Integration Testing
- Validation Testing
- Output Testing

Unit Testing:

Unit testing is carried out to verify and uncover errors within the boundary of the smallest unit or a module. In this testing step, each module was found to be working satisfactory as per the expected output of the module. Unit testing exercise specific paths in the modules control structure to ensure complete coverage and maximum error detection. The project has 21 modules. These modules are developed separately and verified whether they function properly.

Integration Testing:

Integration Testing address the issues associated with the dual problems of verification and program construction. After the software has been integrated a set of higher-order tests are conducted. The main objective in this testing process is to take unit-tested modules and build a program structure that has been dictated by design.

Validation Testing:

At the end of integration testing, software is completely assembled as a package, interfacing errors have been uncovered and correction testing begins.

Validation Test Criteria:

Software testing and validation is achieved through series of black box tests that demonstrate conformity with the requirements are achieved, documentation is correct and other requirement are met. Here the 21 modules which are checked in the integration testing are assembled and tested for any correction.

Output Testing:

Output testing is series of different test whose primary purpose is to fully exercise the computer based system. Although each test has a different purpose, all the work should be verified so that all system elements have properly integrated and perform allocated functions.

Output testing is the stage of implantation, which is aimed at ensuring that the system works accurately and efficiently before live operation commences. The input screens, output documents were checked and required modification made to suite the program specification. Then using rest data prepared, the whole system was tested and found to be a successful one.

FUTURE ENHANCEMENTS

8. FUTURE ENHANCEMENTS

Image processing is a rapidly developing subject in the engineering field. It finds a variety of applications from finding holes in falling rice to checking the alignment of pencil lead in the middle of the pencil.

This project is applicable to all the real time applications. It also satisfies the time constraint of most of the applications.

All the modules in this project are done without the usage of arrays. This is because if the images are tried to store in an array they will result in stack overflow error. The number of bytes for a color image without header portion is 921600 whereas in case of grayscale image it is 307200. Such large number of bytes cannot be stored in ordinary arrays.

So in order to achieve the storing of images in arrays, the arrays should be split into many parts and linked to one another. This may increase the time consumption of the modules but it is worth when it comes to processing since the stored image can be easily processed.

Also the current working environment of the project is 16-bit. In future this can be made to work in 32-bit environments with only slight modifications.

CONCLUSION

9. CONCLUSION

The complete design and development of the system **"GENERATING IMAGE PROCESSING ALGORITHMS FOR DSPs AND FPGAs"** is presented in this dissertation. The system has user-friendly features. It is possible for any user to use this system.

The programming techniques used in the design of the system provide a scope for further expansion and implementation of any changes, which may occur in the future. The system has been tested by connecting with many systems and they provide satisfactory performance.

This system is developed with the specifications and abiding by the existing rules and regulations of the company.

Since the requirements of any organization and their standards are changing day to day the system has been designed in such a way that its scope and boundaries could be expanded in future with little modification. As a further enhancement this system can be integrated with any other system.

This system has been developed using turbo-C. The main aim behind the development of this system is to provide a solution for the costly image processing softwares.

BIBLIOGRAPHY

10. Bibliography

Nick Efford, "**Image processing in c"**, Pearson Education, March 2000.

Web Sites Visited:

www.search.cpan.org - 20-DEC-2003

www.w3schools.com - 15-JAN-2003

www.devspan.com - 29-JAN-2003

ANNEXURE

11. ANNEXURE

11.1 Source Code:

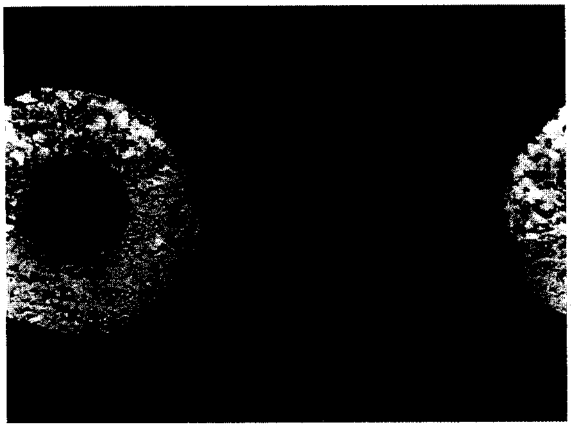
Grayscale Threshold:

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

/*-------STRUCTURES---------*/
typedef struct {int rows; int cols; unsigned char* data;} sImage;

/*-------PROTOTYPES---------*/
long getImageInfo(FILE*, long, int);
void copyImageInfo(FILE* inputFile, FILE* outputFile);
void copyColorTable(FILE* inputFile, FILE* outputFile, int nColors);


int main(int argc, char* argv[])
{
 FILE              *bmpInput, *bmpOutput;
 sImage            originalImage;
 unsigned char          someChar;
 unsigned char*    pChar;
 int               nColors;  /* BMP number of colors */
 long              fileSize; /* BMP file size */
 int               vectorSize; /* BMP vector size */
 int               r, c;     /* r = rows, c = cols */


 /* initialize pointer */
 someChar = '0';
 pChar = &someChar;

 if(argc < 2)
 {
  printf("Usage: %s bmpInput.bmp\n", argv[0]);
  exit(0);
 }
 printf("Reading filename %s\n", argv[1]);

 /*--------READ INPUT FILE------------*/
 bmpInput = fopen(argv[1], "rb");
 fseek(bmpInput, 0L, SEEK_END);
/*--------DECLARE OUTPUT FILE--------*/
bmpOutput = fopen(argv[2], "wb");

/*--------GET BMP DATA---------------*/
originalImage.cols = (int)getImageInfo(bmpInput, 18, 4);
originalImage.rows = (int)getImageInfo(bmpInput, 22, 4);
fileSize = getImageInfo(bmpInput, 2, 4);
nColors = getImageInfo(bmpInput, 46, 4);
vectorSize = fileSize - (14 + 40 + 4*nColors);

/*-------PRINT DATA TO SCREEN-------------*/
printf("Width: %d\n", originalImage.cols);
printf("Height: %d\n", originalImage.rows);
printf("File size: %ld\n", fileSize);
printf("# Colors: %d\n", nColors);
printf("Vector size: %d\n", vectorSize);

copyImageInfo(bmpInput, bmpOutput);
copyColorTable(bmpInput, bmpOutput, nColors);

/*----START AT BEGINNING OF RASTER DATA-----*/
fseek(bmpInput, (54 + 4*nColors), SEEK_SET);

/*----------READ RASTER DATA----------*/
for(r=0; r<=originalImage.rows - 1; r++)
{
        for(c=0; c<=originalImage.cols - 1; c++)
        {
         /*-----read data, reflect and write to output file----*/
         fread(pChar, sizeof(char), 1, bmpInput);
                if(*pChar>128)
                        *pChar=255;
                else
                        *pChar=0;
   fwrite(pChar, sizeof(char), 1, bmpOutput);
  }
}

fclose(bmpInput);
fclose(bmpOutput);

}

/*----------GET IMAGE INFO SUBPROGRAM--------------*/
long getImageInfo(FILE* inputFile, long offset, int numberOfChars)
```

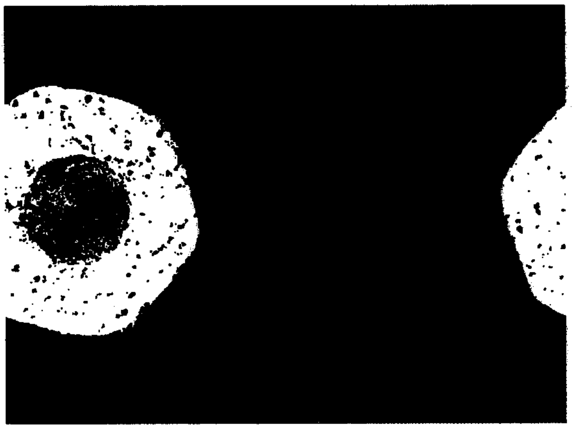
```
{
 unsigned char                    *ptrC;
 long                      value = 0L;
 unsigned char                    dummy;
 int                       i;

 dummy = '0';
 ptrC = &dummy;

 fseek(inputFile, offset, SEEK_SET);

 for(i=1; i<=numberOfChars; i++)
 {
  fread(ptrC, sizeof(char), 1, inputFile);
  /* calculate value based on adding bytes */
  value = (long)(value + (*ptrC)*(pow(256, (i-1))));
 }
 return(value);

} /* end of getImageInfo */

/*-------------COPIES HEADER AND INFO HEADER----------------*/
void copyImageInfo(FILE* inputFile, FILE* outputFile)
{
 unsigned char              *ptrC;
 unsigned char              dummy;
 int                 i;

 dummy = '0';
 ptrC = &dummy;

 fseek(inputFile, 0L, SEEK_SET);
 fseek(outputFile, 0L, SEEK_SET);

 for(i=0; i<=50; i++)
 {
  fread(ptrC, sizeof(char), 1, inputFile);
  fwrite(ptrC, sizeof(char), 1, outputFile);
 }

}

/*----------------COPIES COLOR TABLE-----------------------------*/
void copyColorTable(FILE* inputFile, FILE* outputFile, int nColors)
{
 unsigned char              *ptrC;
 unsigned char                    dummy;
 int                         i;

 dummy = '0';
 ptrC = &dummy;

 fseek(inputFile, 54L, SEEK_SET);
 fseek(outputFile, 54L, SEEK_SET);

 for(i=0; i<=(4*nColors); i++)  /* there are (4*nColors) bytesin color table */
 {
  fread(ptrC, sizeof(char), 1, inputFile);
  fwrite(ptrC, sizeof(char), 1, outputFile);
 }

}
```

11.2 Output Screens:

Original Image:



Threshold Image:



In case of thresholding, all the pixels with values greater than a fixed threshold limit are kept at a value of 255 and all the pixels with values less than that value are kept at 0. Thresholding can be applied for both grayscale and color images. In case of color images, thresholding is done for all the three bytes per pixel and for all pixels.