Sergio Coronado
16.484 Assignment #
Created :2/2/15
Due: 2/91016

Objective:

The purpose of this assignment was to learn how to read in an image and perform so basic operations on the image.

Background:

This assignment was based as a way of introducing us to the most basic operations necessary in the field of computer vision. In this assignment we were required to read in a n image from a file and output an modified image. For part 1 of the assignment we were required to mask out the first half of the image. For the second part we were required to reduce the size of the image by using only the the even pixels in the image.

Algorithm Used:

The image was read in as from the file using fread() and the output was written to a file using fwrite(). A header was created on the output file so that the image could ber opened using xv without th needed of manipulation. Once the image had been read it was processed in part 1 by masking off the left half of the image, this was accomplished by writing a block of 0's to the fist x/2 pixels of the row, and the writing from the image file for the second half. The second part was accomplished by writing every other pixel of the read in image to the output of the image.

Results:

The program was successful in reading and modifying the Image as required. For part A an image was successfully loaded and masked at the left half as shown in figures 1 and 2.

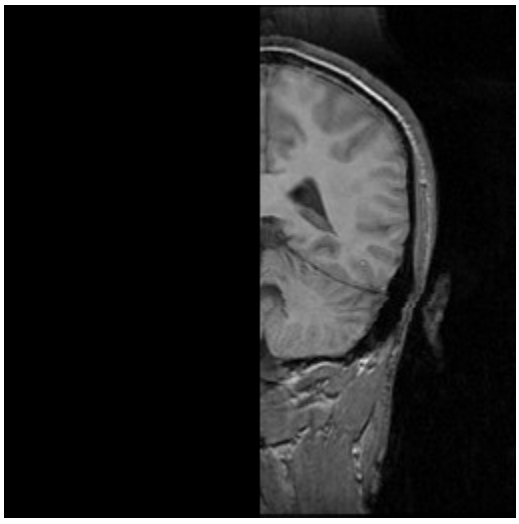

Figure 1. Masked Mri Image

Figure 2. Masked Porsche Image

In part B the images where reduced by wiring to the output file only every other pixel. What resulted was an image that is smaller in size as shown in figures 3 and 4. The resultant images have a jagged quality to them, although the object can be perfectly recognized. As intended both programs performed the manipulations required by the assignment however in order to improve the quality of the image processed in part b it would be better to write every-other pixel as the average of the pixels around it, this could potentially result in a smother image.

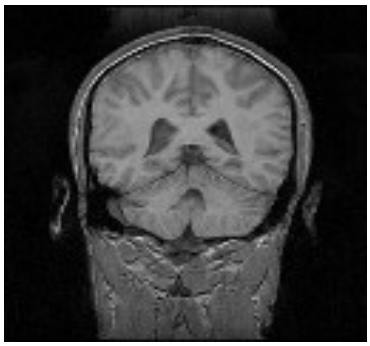

Figure 3. Reduced MRI

Figure 4. Reduced Porsche

Conclusion and Observations

The exercise was successful in demonstrating images can be read and manipulated in program. As mentioned in the results the down-sampling of the image resulted a jagged quality in the image it might make a smoother image if for every other pixel the weighted average of the other intensities around was written into the output.

Readme

////////////////////// Sergio Coronado Assignment 1 README /////////////////

////BUILD

make sure theat both homework1a.c and homework1b.c have the variable CMD writte
to be : /usr/X11R6/bin/xv when runnign on the anaconda machines

$ make

///RUN

bin/HW1a <input-file> <output-file> <xSize> <ySize>
bin/HW1a <input-file> <output-file> <xSize> <ySize>

//Cleanup

$ make clean

Code

```
 1 ////////////////////// homework1a.c ////////////////////////////////////////
 2 /*
 3 By:    Sergio Coronado
 4         16.484 Computer Vision
 5         Assignemnt #1
 6         Part 1
 7
 8 PURPOSE:
 9   Program reads in an image specfied by user and outputs a file specified by user
10    that has reduced to a quarter of the original size by keeping even pixels and
11    discarding odd ones.
12
13 USAGE:
14
15    HW1a <input-file> <output-file> <xSize> <ySize>
16
17 */
18
19 ////////////////////// Includes ////////////////////////////////////////////
20
21 #include <unistd.h>
22 #include <stdio.h>
23 #include <stdlib.h>
24 #include <string.h>
25 #include "CursorCntl.h"
26 #include <sys/wait.h>
27
28 ////////////////////// Constant Definitions ////////////////////////////////
29
30 #define SINGLE_BYTE 1
31 #define HEADER_SIZE 20
32 #define NUM_ARGS 3
33 #define CMD_LENGTH 2
34
35 /* Command String for launching XV
36    Note:
37         For Runnign on School Lab this must be : /usr/X11R6/bin/xv
38 */
39
40 //static char *CMD = "/usr/bin/xv";
41 static char *CMD = "/usr/X11R6/bin/xv";
42
43 ////////////////////// Function Prototypes /////////////////////////////////
44
45
46 unsigned char ** ReadImage(
47   FILE * openedFile, unsigned xSize, unsigned ySize, unsigned * numRows);
```

```
48
49 ////////////////////// Main ////////////////////////////////////////////////
50
51 int main (int argc, char **  argv)
52 {
53  ////////////////////// Variables ////////////////////////////////////////
54
55  unsigned char ** img;           //Matrix Holding Image Values
56  unsigned xSize;                 //NUmber of horizontal pixels
57  unsigned ySize;                 //NUmber of vertical pixels
58  unsigned numRows;               //rows Read in
59  unsigned char * zeroBuff;       //Buffer used to mask image
60  pid_t xvFork;                   //pid for xv launched
61  char headerBuffer[HEADER_SIZE]; //array contianign header params for the image
62  char * xvArgs[NUM_ARGS];        //Arguments for XV
63  int status;                     ///Status variable
64
65
66  FILE * openedFile;              //File to be read In
67  FILE * outputFile;              //file to be written to
68
69  int i, j;      //counter variables
70
71  /* Check correct n umber of arguments */
72
73  if (argc < 5)
74  {
75   printError("Invalid Parameters\n");
76   exit (-1);
77  }
78
79  /* Openede image File */
80
81  openedFile = fopen(argv[1], "r");
82
83  xSize = atoi(argv[3]);
84
85  ySize = atoi(argv[4]);
86
87  if (openedFile == NULL)
88  {
89   printError("Error Could Not Open file\n");
90   exit (-1);
91  }
92
93  img = (unsigned char **) malloc( sizeof(unsigned char *) * ySize);
94
95  zeroBuff = (unsigned char *) calloc (xSize/2, sizeof(unsigned char ));
96
```

```
97   printOK("Reading Image\n");
98
99   /* Read Image */
100
101   img = ReadImage(openedFile, xSize, ySize, &numRows);
102
103   /* Close input file and open output file */
104   /* Designed in this order in case user wishes to over-write image */
105
106   fclose(openedFile);
107
108   outputFile = fopen(argv[2], "w");
109
110   printOK("Building Image\n");
111
112   sprintf(headerBuffer, "P5\n%d %d\n255\n", xSize, ySize);
113
114   fwrite(headerBuffer, sizeof(unsigned char), strlen(headerBuffer), outputFile);
115
116   for ( i = 0; i < numRows; i++)
117   {
118     /* Write the first half of the row  as blank then write the second half
119        from the read-in image */
120
121     fwrite( zeroBuff, sizeof(unsigned char), xSize/2, outputFile);
122     for (j = xSize/2 ; j < xSize; j ++)
123     {
124       fwrite(&img[i][j], sizeof(unsigned char), SINGLE_BYTE, outputFile);
125     }
126   }
127
128   fclose(outputFile);
129
130
131   printOK("Image Built Starting XV\n");
132
133   xvArgs[0] = CMD;
134   xvArgs[1] = argv[2];
135   xvArgs[2] = NULL;
136
137   xvFork = fork();
138
139   if (xvFork == 0)
140   {
141     status = execv(CMD, xvArgs);
142     if (status < 0)
143     {
144       printError("Could Not Initiate XV\n");
145     }
```

```
146
147   }
148   else
149   {
150     waitpid(xvFork, 0, 0);
151   }
152
153 return 0;
154
155 }
156
157
158 ////////////////////////// ReadImage() ////////////////////////////////////////]
159 /*
160 PURPOSE
161 The read image function reads an image from a file of size specified in the
162 parameters
163 INPUT
164   FILE * openedFile : pointer to the file being opened must be opened priot to
165                       call
166   unsigned xSize  `  :  number of columns in image to be read
167   unsigned ySize    : number of rows in image to be read
168   unsigned char *** img   : Pointer to be set to the
169
170 OUTPUT
171
172   unsigned char ** img   : read-in matrix of pixel densities
173
174 */
175
176 unsigned char ** ReadImage(
177  FILE * openedFile, unsigned xSize, unsigned ySize, unsigned * numRows)
178 {
179   unsigned char ** img;      //pointer to hold the address of the image matrix
180   unsigned bytesRead;        // variable counting the bytes read per line
181   unsigned i, j;             // COUNTER VARIABLES
182
183   *(numRows) = 0;
184
185   img = (unsigned char **) malloc( sizeof(unsigned char *) * ySize);
186
187   for ( i = 0; i < ySize; i ++)
188   {
189     img[i] = (unsigned char *) malloc (sizeof(unsigned char) * xSize);
190     bytesRead = fread(img[i], sizeof(unsigned char), xSize , openedFile );
191     if (bytesRead < xSize)
192     {
193       if (feof(openedFile))
194       {
```

```
195         printWarning("Reached Unexpected EOF Attemptin padding to size\n");
196           for (j = (bytesRead -1); j < xSize; j++)
197           {
198             img[i][j] = 0;
199           }
200       }
201       else
202       {
203         printError("Error Reading File");
204       }
205
206     }
207     *(numRows) += 1;
208
209   }
210   return img;
211 }
```

```
 1 ////////////////////////// homework1b.c ////////////////////////////////////////
 2 /*
 3 By:    Sergio Coronado
 4         16.484 Computer Vision
 5         Assignemnt #1
 6         Part 2
 7
 8 PURPOSE:
 9   Program reads in an image specfied by user and outputs a file specified by user
10    that has reduced to a quarter of the original size by keeping even pixels and
11    discarding odd ones.
12
13 USAGE:
14
15    HW1b <input-file> <output-file> <xSize> <ySize>
16
17 */
18
19 ////////////////////////// Includes ////////////////////////////////////////////
20
21
22 #include <unistd.h>
23 #include <stdio.h>
24 #include <stdlib.h>
25 #include <string.h>
26 #include "CursorCntl.h"
27 #include <sys/wait.h>
28
29 ////////////////////////// Constant Definitions ////////////////////////////////
30
31 #define SINGLE_BYTE 1
```

```
32 #define HEADER_SIZE 20
33 #define NUM_ARGS 3
34 #define CMD_LENGTH 2
35
36 /* Command String for launching XV
37    Note:
38        For Runnign on School Lab this must be : /usr/X11R6/bin/xv
39 */
40
41 //static char *CMD = "/usr/bin/xv";
42 static char *CMD = "/usr/X11R6/bin/xv";
43
44 ////////////////////////// Function Prototypes ///////////////////////////////////
45
46
47 unsigned char ** ReadImage(
48  FILE * openedFile, unsigned xSize, unsigned ySize, unsigned * numRows);
49
50 ////////////////////////// Main //////////////////////////////////////////////////
51
52
53 int main (int argc, char **  argv)
54 {
55
56   ////////////////////// Variables ////////////////////////////////////////
57
58
59   unsigned char ** img;           //Matrix Holding Image Values
60   unsigned xSize;                 //NUmber of horizontal pixels
61   unsigned ySize;                 //NUmber of vertical pixels
62   unsigned numRows;                //rows Read in
63   pid_t xvFork;                   //pid for xv launched
64   char headerBuffer[HEADER_SIZE]; //array contianign header params for the image
65   char * xvArgs[NUM_ARGS];        //Arguments for XV
66   int status;                     //Status variable
67
68
69   FILE * openedFile;              // file to be read in
70   FILE * outputFile;              // file to be wrtten out
71
72   int i, j;        //counter varaibles
73
74   /* Check Correct number of argumetns */
75
76   if (argc < 5)
77   {
78     printError("Invalid Parameters\n");
79     exit (-1);
80   }
```

```
81
82  /* open image file */
83
84  openedFile = fopen(argv[1], "r");
85
86  xSize = atoi(argv[3]);
87
88  ySize = atoi(argv[4]);
89
90  if (openedFile == NULL)
91  {
92    printError("Error Could Not Open image file\n");
93    exit (-1);
94  }
95
96  numRows = 0;
97
98  printOK("Reading Image\n");
99
100  /* Read Image */
101
102  img = ReadImage(  openedFile, xSize, ySize, &numRows);
103
104  /* Close input file and open output file */
105  /* Designed in this order in case user wishes to over-write image */
106
107  fclose(openedFile);
108
109  outputFile = fopen(argv[2], "w");
110
111  if (outputFile == NULL)
112  {
113    printError("Could Not Open or create ouput file\n");
114    exit (-1);
115  }
116
117  printOK("Building Image\n");
118
119  sprintf(headerBuffer, "P5\n%d %d\n255\n", xSize/2, ySize/2);
120
121  fwrite(headerBuffer, sizeof(unsigned char), strlen(headerBuffer), outputFile);
122
123  for ( i = 0; i < numRows; i += 2)
124  {
125    for (j = 0 ; j < xSize; j += 2)
126    {
127      fwrite(&img[i][j], sizeof(unsigned char), SINGLE_BYTE, outputFile);
128    }
129  }
```

```
130
131   fclose(outputFile);
132
133   printOK("Image Built Starting XV\n");
134
135   xvArgs[0] = CMD;
136   xvArgs[1] = argv[2];
137   xvArgs[2] = NULL;
138
139   xvFork = fork();
140
141   if (xvFork == 0)
142   {
143     status = execv(CMD, xvArgs);
144     if (status < 0)
145     {
146       printError("Could Not Initiate XV\n");
147     }
148
149   }
150   else
151   {
152     waitpid(xvFork, 0, 0);
153   }
154
155 return 0;
156
157 }
158
159 ////////////////////////// ReadImage() ////////////////////////////////////////]
160 /*
161 PURPOSE
162 The read image function reads an image from a file of size specified in the
163 parameters
164 INPUT
165   FILE * openedFile : pointer to the file being opened must be opened priot to
166                       call
167   unsigned xSize  `  :  number of columns in image to be read
168   unsigned ySize    : number of rows in image to be read
169   unsigned char *** img   : Pointer to be set to the
170
171 OUTPUT
172
173   unsigned char ** img   : read-in matrix of pixel densities
174
175 */
176
177 unsigned char ** ReadImage(
178  FILE * openedFile, unsigned xSize, unsigned ySize, unsigned * numRows)
```

```
179 {
180   unsigned char ** img;      //pointer to hold the address of the image matrix
181   unsigned bytesRead;        // variable counting the bytes read per line
182   unsigned i, j;             // COUNTER VARIABLES
183
184   *(numRows) = 0;
185
186   img = (unsigned char **) malloc( sizeof(unsigned char *) * ySize);
187
188   for ( i = 0; i < ySize; i ++)
189   {
190     img[i] = (unsigned char *) malloc (sizeof(unsigned char) * xSize);
191     bytesRead = fread(img[i], sizeof(unsigned char), xSize , openedFile );
192     if (bytesRead < xSize)
193     {
194       if (feof(openedFile))
195       {
196         printWarning("Reached Unexpected EOF Attemptin padding to size\n");
197           for (j = (bytesRead -1); j < xSize; j++)
198           {
199             img[i][j] = 0;
200           }
201       }
202       else
203       {
204         printError("Error Reading File");
205       }
206
207     }
208     *(numRows) += 1;
209
210   }
211   return img;
212 }
```