IMAGE PROCESSING: INTERPOLATION

Gregory C. Peng

June 9, 2004

1 Introduction

The aim of computer graphics these days is headed toward the third dimension; companies are focusing their assets on creating more realistic 3-d graphics, applying physics and real life observations to their projects. However, with all attention on the third dimension, many forget about the aspects of the second dimension that have yet to be touched upon or mastered.

It is my view the computer entertainment industry has become lazy in the use of two-dimensional graphics. Capcom, and many other leading software companies, consistently come out with software with noticeable pixelation. Obviously the aesthetic quality of a program is very important in order for it to appeal to the masses.

However, with the application of a simple graphical filter, the resolution and quality of the images that appear on the screen are greatly increased. In this case, one can possibly even port software from 1980s and have them appear in a marketable, somewhat presentable fashion. Thus, time and money can be saved since graphics do not need to be completely redrawn.

Additionally, image interpolation can be especially useful when one needs to reduce file size or increase the amount of image data sent at one time. With the image size reduced, the image quality does not have to be significantly reduced as well.

This tech-lab project bravely ventures into the fields of image processing and computer graphics. It is my aim to explore the boundaries of 2D graphics and animation and research the application of image filters and processing.

2 Background

Interpolation is way through which images are enlarged. There are many different types of interpolation methods, each resulting in a different ?look? to the final picture. Thus, it is best if the quality, or visible distinction for each pixel, is retained throughout the enlargement process.

Thus, one cannot simply have a number of pixels directly represent a single original pixel; this is not sufficient for commercial use. Conspicuous blocks of single color will be visible, and depending on size of enlargement, the original image will be unrecognizable.

Older methods of linear interpolation somewhat addressed this problem. By finding a mean pixel value between neighboring pixels, one was able to produce an effect of blurred edges and smoothed details. Bilinear re-sampling uses the values from the four surrounding pixels, and new pixel values are calculated by weighting the averages of the four closest pixels based on distance. The new pixel value is determined by calculating a weighted average of the four closest pixels (2x2 array) based on distance. However, bilinear interpolation seems to work better for image reduction rather than image enlargement.

Linear interpolation methods can only go so far, and it has been found that non-linear methods are superior. Some non-linear interpolation methods include Bi-Cubic, Soft Directional, and non-linear interpolation through extended permutation filters. Bi-cubic interpolation uses the nearest sixteen pixels (4x4 array) based on distance, which produces a much better effect than linear interpolation.

Most high-end image manipulation and viewing programs today have some sort of interpolation process used when resizing an image. Such include ACDSee, Adobe Photoshop, IfranView, and even Internet Explorer. They implement many commercial interpolation methods. For instance Lanczos Interpolation, standard Kneson, and pxl SmartScale.

3 Blitters

Typically blitters are, quite simply, used to display graphics. However, it is not uncommon for many blitters to implement special video modes that manipulate the original screen output.

The Super2XSai blitter is an existing interpolation engine use primarily in the PC emulation of arcade machines and video game consoles. It increases the size of the original video output by twice-fold, and applies a unique interpolation filter onto each frame.

There are many other interpolation video blitters for emulation use, such as SuperEagle, Zoom 2X Software, and various scanlined video blitters. However, the effects that these blitters produce are very stylized and significantly change the look of the original video.

4 Developmental Resources

For this project, the main coding languages used were C and C++. Additionally, SDL (Simple DirectMedia Layer) and OpenGL have been used to display the interpolated images and graphical statistical representation of pixel data. The SDL graphical library was learned specifically for this project.

5 Image Filetype

Although there have been forays into researching bitmap header and RAW image filetypes, as of now the official image filetype of this project is the PPM (Portable PixMap), due to its ease of use.

A PPM can be read quite with little difficulty. After a very simple file header with select information such as P identifier, resolution, and maximum pixel value, all the pixel data is listed. Being able to view a PPM directly in a text editor, since all of the data is encoded in ASCII, allows for easy debugging of code and exact results.

A typical PPM goes like this:

P3 (identifier) 200 300 (width height) 255 (max pixel value)

0 0 121 1 2 4 242 0 1 etc. etc. etc. (image data)

However, as time passes, the Bitmap has become crucial to this project as well, especially when graphical libraries such as SDL are involved. Typical Bitmap header types are as follows:

```
Bytes_Per_Raster = (X2 - X1 + 1) * 3;

If ((Bytes_Per_Raster % 4) == 0)
```

```
  Raster_Pad = 0;

Else Raster_Pad = 4 - (Bytes_Per_Raster % 4);

Bytes_Per_Raster = Bytes_Per_Raster + Raster_Pad;


Long int bfSize = 0; //(* Size of File, Computed below *)

int bfReserved1 = 0; //(* Always Zero *)

int bfReserved2 = 0; //(* Always Zero *)

int bfOffbits = SizeOf(Header); //(* Pointer to Image Start *)

int biSize = 40; //(* Bytes in bi Section *)

int biWidth = X2 - X1 + 1; //(* Width of Image *)

int biHeight = Y2 - Y1 + 1; //(* Height of Image *)

int biPlanes = 1; //(* One Plane (all colors packed)*)

int biBitCount = 24; //(* Bits per Pixel *)

int biCompression = 0; //(* No Compression *)

int biSizeImage = Bytes_Per_Raster * biHeight; //(* Size of Image *)

int biXPelsPerMeter = 0; //(* Always Zero *)

int biYPelsPerMeter = 0; //(* Always Zero *)

int biClrUsed = 0; //(* No Palette in 24 bit mode *)

int biClrImportant = 0; //(* No Palette in 24 bit mode *)

int bfSize = SizeOf(Header) + biSizeImage;
```

Since BITMAP is more or less the STANDARD, it was crucial to be able to use it in this project.

6 Nearest Neighbor

Probably the most basic form of interpolation. As the actual pixels are proportionally copied to their new locations, their position in relation to one another remains the same. Since the image is enlarged, filler pixels must be placed in between the actual pixels.

With the most basic nearest neighbor interpolation, just copy the exact same pixel values over to the filler pixel closest to the pixel. Since my images have been initialized with the pixel at 0,0 being the same pixel in the original and enlarged image, I choose the pixel to the right or the pixel below, dependant on where the filler pixel is placed.

```
DIAGRAM


AB

CD


For 2X Turns into


AABB

AABB

CCDD

CCDD
```

7 Linear Interpolation

A better algorithm than nearest neighbor that takes into account the gradual transition of pixel color values. By finding the means between two pixel values, the filler pixel is better suited for overall image enhancement. In other words, it just looks plain better.

```
Once again we return to the trusty diagram


AB

CD


With filler it turns into


AFBF

FFFF

CFDF

FFFF


Thus for every F, calculate the mean of the surrounding pixels.

Eventually, you will be able to calculate the mean for every F,

even those that were originally surrounded by all F.


AFB

FFF

CFD   <--For example, eventually, the middle F should
```

`calculate out to being (A+B+C+D)/4.`

8 Bi-Cubic Interpolation

As noted before bi-cubic interpolation utilizes a 4x4 array weighted by distance

Interpolation kernel (cubic weighting function) is defined by:

$h(t)_1^3 = 1 - 2|t|^2 + |t|^3$, if —t—¡1

$4 - 8|t| + 5|t|^2 - |t|^3$, if $1 <= |t| < 2$

$0, otherwise$

$h_3(x, y) = h_3^1(x)h_3^1(y)$

9 Edge Enhancement

After linear interpolations, edges are blurred. To remedy this, spline interpolation is used. After interpolation, the edges are more visible so edge enhancement is much more sucessful and visible.

Edge detection works by taking a weighted sum of pixels around a single pixel to determine its new value. For instance, we apply the following array to each pixel in multiplication

```
-1 -1 -1

-1  8 -1

-1 -1 -1
```

10 Graphical and Stastical Analysis

In order to analyze the results of interpolation in a more quantitative method, as compared to the qualitative focus that has been the backbone of the project for most of the year, a graphical and statistical research program was developed in OpenGL.

For each channel of color (RED, GREEN, BLUE), pixel data is arranged and tallied into a separate data document. The occurence for each pixel value is printed into the document, with each number following correspoding to an ascending pixel value. ropix.txt, gopix.txt, and bopix.txt contain the pixel data of the original image file, while ripix.txt, gipix.txt, and bipix.txt contain the pixel data of the interpolated image file.

Then, each data document is loaded into a graphical analysis program. With this program, the mean, median, and mode pixel values are calculated, and a standard bar graph, scaled dependant on the most frequent pixel value, is displayed using OpenGL.

From this information, one can find correlations by comparing the original image graph to the interpolated image graph. Depending the similarity (or dissimilarity) of the two graphs, mean, median, and mode, one can see how closely the interpolated image reflects the original one.

11 Results and Analysis: Qualitative

Running a 209 by 96 pixel image through the interpolation function yields a 418 by 192 pixel image.

As you can see plainly, the image quality increases from nearest neighbor (2) to bilinear interpolation (3).

Figure 1: Original image

Figure 2: Nearest Neighbor interpolated image

Figure 3: Bilinear interpolated image

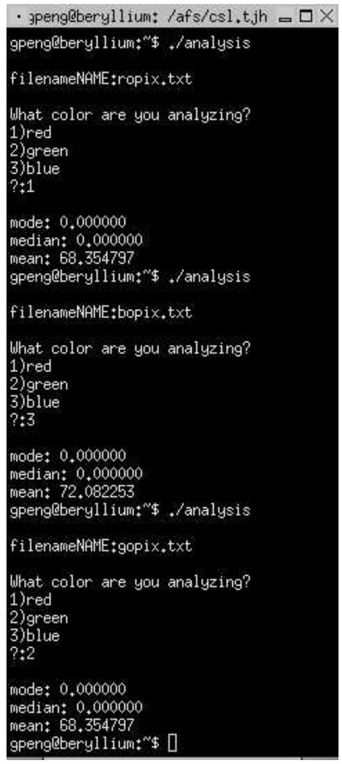

Figure 4: graphical analysis program terminal run

12 Results and Analysis: Quantitative

After running the statistical graphical analysis on the original image, such results are yielded.

In comparison, the following results were received from a statistical and graphical analysis on the bilinear interpolated image.

As you can see, the occurrence of middle values (9 10 11) that would normally not be present is documented. It is through these middle values that the interpolation is created. Moreover, even with the creation of middle values, the ratios of frequency per pixel value has not changed drastically, a good indication of successful interpolation. Moreover, the mean, median, and mode are nearly, if not exactly, the same as well. (4 8)

Of course, due to the overwhelmingly dark nature of the photo, the 0 pixel value is

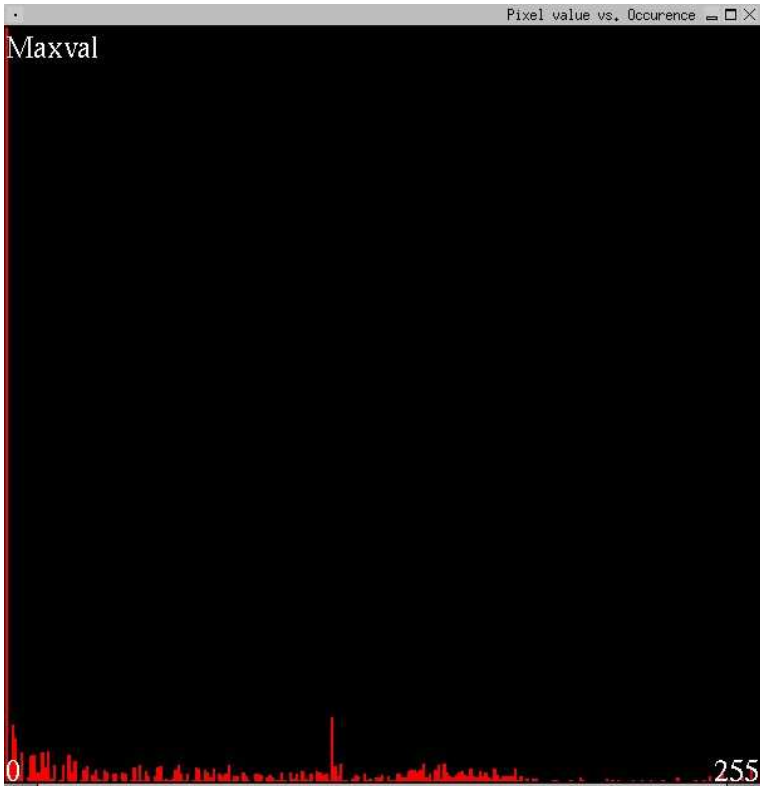

Figure 5: RED ORIGINAL PIXEL VALUES

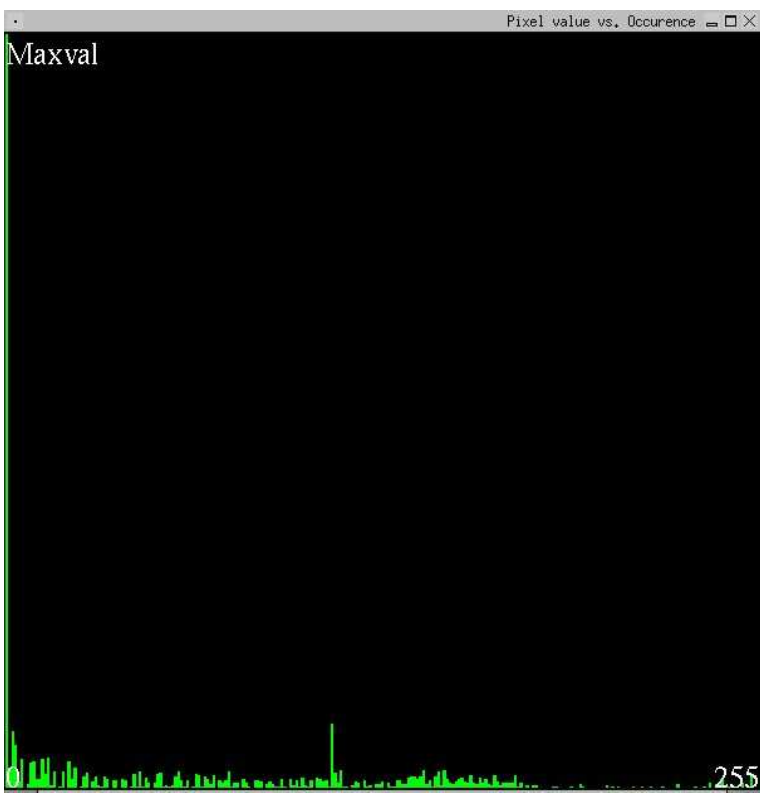

Figure 6: GREEN ORIGINAL PIXEL VALUES

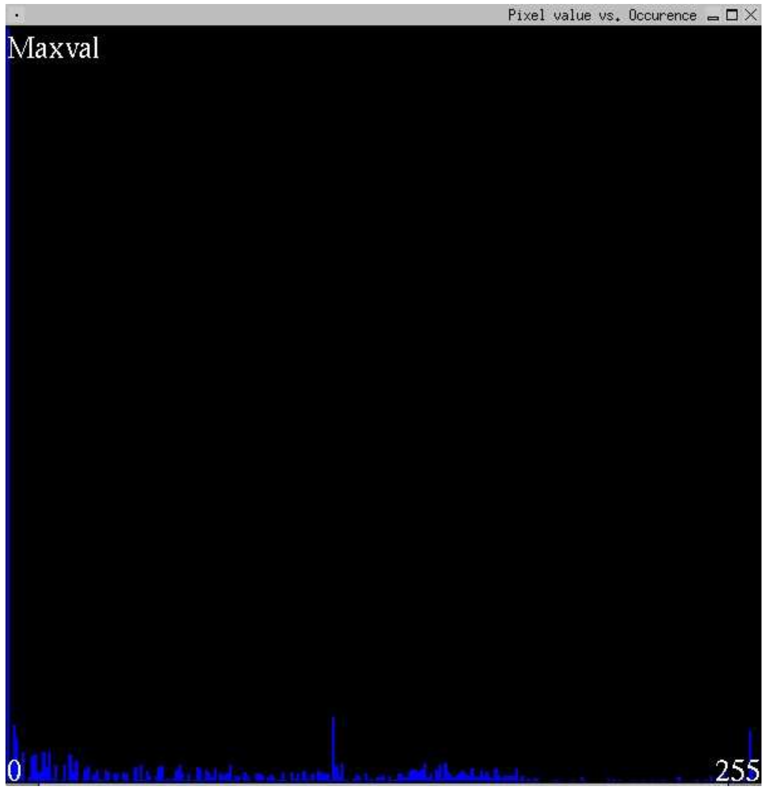

Figure 7: BLUE ORIGINAL PIXEL VALUES

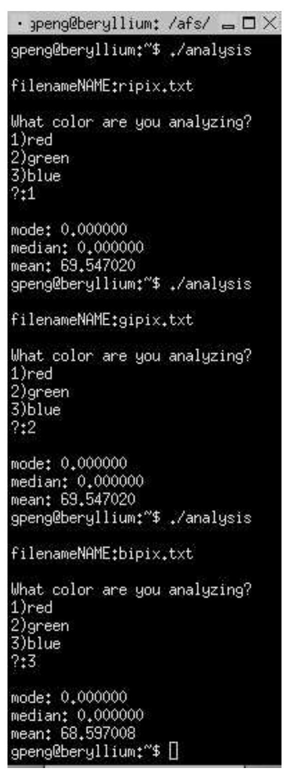

Figure 8: graphical analysis program terminal run 2

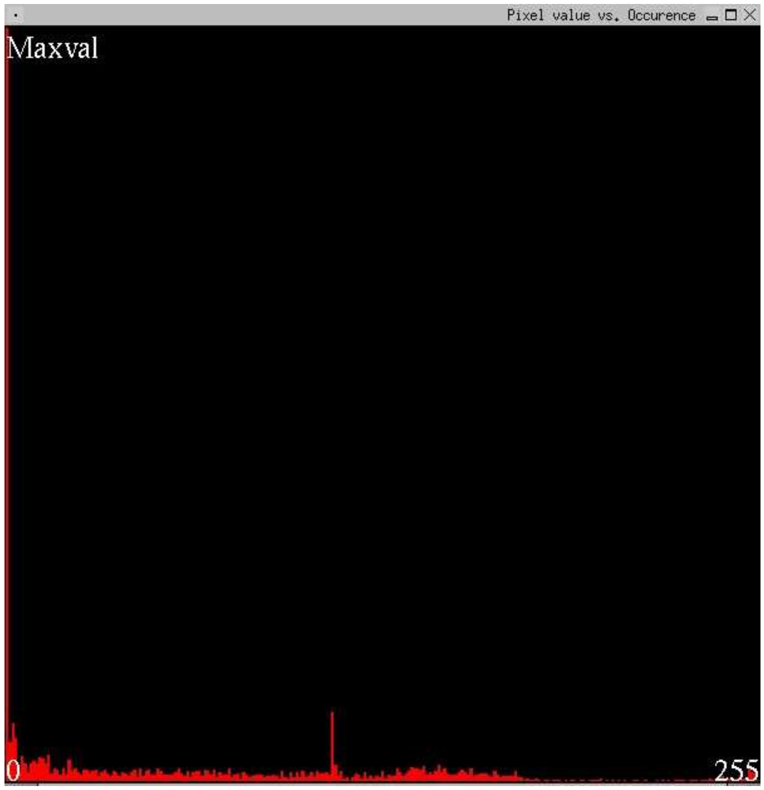

Figure 9: RED INTERPOLATED PIXEL VALUES

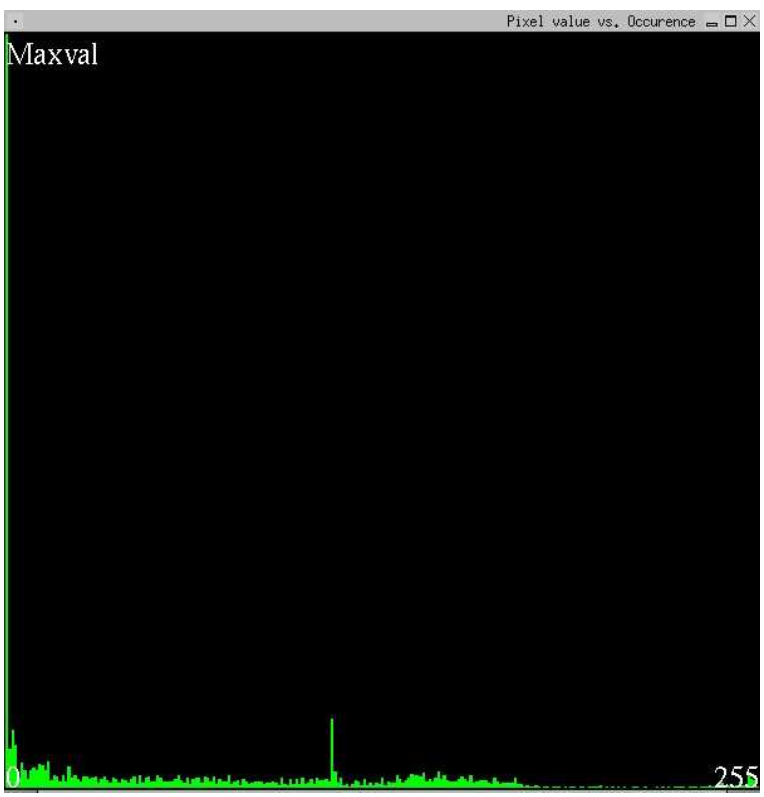

Figure 10: GREEN INTERPOLATED PIXEL VALUES

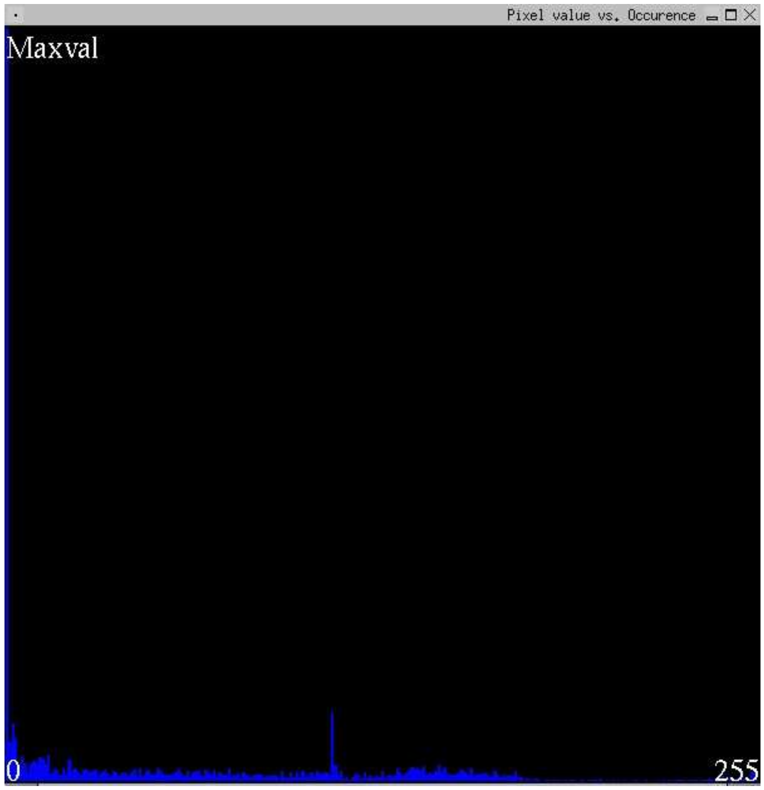

Figure 11: BLUE INTERPOLATED PIXEL VALUES

the most predominant in all the color channels.

For more insight into this matter, the data documents from which the graphs were derived are below:

gopix.txt

19393 528 1447 1102 161 734 0 84 631 677 222 232 735 343 779 70 406 0 43 408 9 693 211 510 0 41 287 392 0 167 277 53 47 277 229 0 190 177 1 216 189 9 33 362 27 416 44 258 132 13 127 370 382 20 78 45 73 298 404 181 35 187 13 30 347 313 16 299 218 93 308 5 193 123 180 429 38 121 136 1 214 120 29 7 189 208 162 12 102 152 96 47 32 228 20 3 226 75 185 21 259 58 46 187 2 265 226 170 177 29 1655 395 116 444 47 42 7 55 175 140 0 209 41 72 8 89 80 158 8 139 17 31 187 187 111 97 257 300 305 226 289

453 112 185 20 301 432 62 445 223 126 82 162 231 258 146 121 322 143 105 263 114 230 99 50 323 156 61 115 137 119 117 324 19 127 0 57 10 71 57 0 0 14 14 0 40 16 0 0 2 1 20 0 1 82 17 0 0 14 0 14 15 13 60 0 23 0 30 15 0 15 0 39 0 0 33 0 0 16 8 0 1 27 0 1 0 0 81 0 0 0 2 0 42 1 0 42 0 120 4 100 134 221 16 30 42 19 41 13 120 3 47 302 2 8 4141

gipix.txt

68934 3535 5286 3879 967 2322 1660 828 1560 1847 1611 2226 2013 1612 2396 672 1174 1080 585 1158 576 1921 898 1135 751 518 1044 1031 783 939 1050 410 824 1010 736 473 942 830 496 836 730 405 829 1035 290 1108 408 907 1002 541 597 1154 950 514 519 588 515 775 1143 697 434 890 341 745 965 949 456 1041 881 523 1010 304 773 562 573 1193 343 559 661 238 755 580 393 443 619 618 544 295 456 507 609 456 274 859 354 276 798 334 840 364 873 393 502 771 355 926 733 659 748 549 6270 1494 496 897 218 328 136 317 635 536 179 858 233 476 278 412 394 864 225 599 400 401 1096 792 459 714 914 1230 1156 1157 982 1323 577 838 565 917 1464 833 1170 718 563 599 548 833 989 782 576 1143 459 614 658 669 718 412 305 907 541 301 457 422 486 483 880 324 348 165 193 98 156 186 122 47 60 66 52 110 55 99 28 55 40 149 56 45 118 112 50 78 162 74 93 189 47 101 30 105 41 90 38 113 52 47 58 97 26 51 68 34 37 45 87 29 55 26 139 31 77 120 66 106 31 110 54 135 41 79 84 57 261 61 172 424 454 70 170 199 125 241 558 257 188 1406 773 695 64 13438

ropix.txt

19393 528 1447 1102 161 734 0 84 631 677 222 232 735 343 779 70 406 0 43 408 9 693 211 510 0 41 287 392 0 167 277 53 47 277 229 0 190 177 1 216 189 9 33 362 27 416 44 258 132 13 127 370 382 20 78 45 73 298 404 181 35 187 13 30 347 313 16 299 218

93 308 5 193 123 180 429 38 121 136 1 214 120 29 7 189 208 162 12 102 152 96 47 32 228 20 3 226 75 185 21 259 58 46 187 2 265 226 170 177 29 1655 395 116 444 47 42 7 55 175 140 0 209 41 72 8 89 80 158 8 139 17 31 187 187 111 97 257 300 305 226 289 453 112 185 20 301 432 62 445 223 126 82 162 231 258 146 121 322 143 105 263 114 230 99 50 323 156 61 115 137 119 117 324 19 127 0 57 10 71 57 0 0 14 14 0 40 16 0 0 2 1 20 0 1 82 17 0 0 14 0 14 15 13 60 0 23 0 30 15 0 15 0 39 0 0 33 0 0 16 8 0 1 27 0 1 0 0 81 0 0 0 2 0 42 1 0 42 0 120 4 100 134 221 16 30 42 19 41 13 120 3 47 302 2 8 4141

ripix.txt

68934 3535 5286 3879 967 2322 1660 828 1560 1847 1611 2226 2013 1612 2396 672 1174 1080 585 1158 576 1921 898 1135 751 518 1044 1031 783 939 1050 410 824 1010 736 473 942 830 496 836 730 405 829 1035 290 1108 408 907 1002 541 597 1154 950 514 519 588 515 775 1143 697 434 890 341 745 965 949 456 1041 881 523 1010 304 773 562 573 1193 343 559 661 238 755 580 393 443 619 618 544 295 456 507 609 456 274 859 354 276 798 334 840 364 873 393 502 771 355 926 733 659 748 549 6270 1494 496 897 218 328 136 317 635 536 179 858 233 476 278 412 394 864 225 599 400 401 1096 792 459 714 914 1230 1156 1157 982 1323 577 838 565 917 1464 833 1170 718 563 599 548 833 989 782 576 1143 459 614 658 669 718 412 305 907 541 301 457 422 486 483 880 324 348 165 193 98 156 186 122 47 60 66 52 110 55 99 28 55 40 149 56 45 118 112 50 78 162 74 93 189 47 101 30 105 41 90 38 113 52 47 58 97 26 51 68 34 37 45 87 29 55 26 139 31 77 120 66 106 31 110 54 135 41 79 84 57 261 61 172 424 454 70 170 199 125 241 558 257 188 1406 773 695 64 13438

bopix.txt

19393 528 1447 1102 161 734 0 84 631 677 222 232 735 343 779 70 406 0 43 408 9 693 211 510 0 41 287 392 0 167 277 53 47 277 229 0 190 177 1 216 189 9 33 362 27 416 44 258 132 13 127 370 382 20 78 45 73 298 404 181 35 187 13 30 347 313 16 299 218 93 308 5 193 123 180 429 38 121 136 1 214 120 29 7 189 208 162 12 102 152 96 47 32 228 20 3 226 75 185 21 259 58 46 187 2 265 226 170 177 29 1655 395 116 444 47 42 7 55 175 140 0 209 41 72 8 89 80 158 8 139 17 31 187 187 111 97 257 300 305 226 289 453 112 185 20 301 432 62 445 223 126 82 162 231 258 146 121 322 143 105 263 114 230 99 50 323 156 61 115 137 119 117 324 19 127 0 57 10 71 57 0 0 14 14 0 40 16 0 0 2 1 20 0 1 82 17 0 0 14 0 14 15 13 60 0 23 0 30 15 0 15 0 39 0 0 33 0 0 16 8 0 1 27 0 1 0 0 81 0 0 0 2 0 42 1 0 42 0 120 4 100 134 221 16 30 42 19 41 13 120 3 1301 302 2 8 4141

bipix.txt

68934 3535 5286 3879 967 2322 1660 828 1560 1847 1611 2226 2013 1612 2396 672 1174 1080 585 1158 576 1921 898 1135 751 518 1044 1031 783 939 1050 410 824 1010 736 473 942 830 496 836 730 405 829 1035 290 1108 408 907 1002 541 597 1154 950 514 519 588 515 775 1143 697 434 890 341 745 965 949 456 1041 881 523 1010 304 773 562 573 1193 343 559 661 238 755 580 393 443 619 618 544 295 456 507 609 456 274 859 354 276 798 334 840 364 873 393 502 771 355 926 733 659 748 549 6270 1494 496 897 218 328 136 317 635 536 179 858 233 476 278 412 394 864 225 599 400 401 1096 792 459 714 914 1230 1156 1157 982 1323 577 838 565 917 1464 833 1170 718 563 599 548 833 989 782 576 1143 459 614 658 669 718 412 305 907 541 301 457 422 486 483 880 324 348 165 193 98 156 186 122 47 60 66 52 110 55 99 28 55 40 149 56 45 118 112 50 78 162 74 93 189 47 101 30 105 41 90 38 113 52 47 58 97 26 51 68 34 37 45 87 29 55 26 139 31 77 120 66 106 31 110

54 135 41 79 84 57 261 61 172 424 454 70 170 199 125 241 558 257 188 152 773 695 64

13438

13 Conclusion

Image processing has become an integral part of our everyday lives. And one of the first steps to higher level image processing is interpolation. Cancer detection via computers relies heavily on image processing methods, such as edge detection, but the first step must be interpolation.

But it is not to be misconstrued that image interpolation is a mere stepping stone for further action. Nearly every digitally edited image has gone through some sort of interpolation, and the same applies to the moving image as well. Interpolation makes it possible for image file-sizes to be reduced, without having to drastically reduce the quality as well.

14 References

EFFICIENT IMAGE MAGNIFICATION BY BICUBIC SPLINE INTERPOLATION: http://members.bellatlantic.net/ vze2vrva/design.html

4.2.2.3 Bi-cubic interpolation: http://ct.radiology.uiowa.edu/ jiangm/courses/dip/html/node68.html

Interpolation: http://www.nut-n-but.net/CCPCUG/TechBrief

Appendix A: Bicubic Interpolation: http://www.npac.syr.edu/projects/nasa/MILOJE/final/node36.html

55:248 Advanced Image Processing:

http://www.engineering.uiowa.edu/ gec/248$_s00_students/blake_carlson/hw2/$

A Brief Tutorial On Interpolation for Image Scaling:

http://www.cs.wisc.edu/graphics/Courses/cs-638-1999/image$_scaling.htm$

Geometric Transformation of Digital Images Interpolation and Image Rotation:

http://micro.magnet.fsu.edu/primer/java/digitalimaging/processing/geometricaltransformation

http://www.magic-software.com/Source/Interpolation/WmlIntpBicubic2.cpp

Bicubic Interpolation for Image Scaling:

http://astronomy.swin.edu.au/ pbourke/colour/bicubic/

24 Bit .BMP Files: http://www.cs.umass.edu/ verts/cs32/save$_24b.html$

Quick Image Stretching Technique:

http://www.codeguru.com/bitmap/quick$_stretch.html$

[SDL] Locked Surface during Blit?:

http://www.libsdl.org/pipermail/sdl/2002-January/041153.html

Bitmap objects: http://games.linux.sk/docs/allegro/alleg008.html

www.tjhsst.edu/ gpeng/techlab/berkICIP2000.pdf