EEL 4851-001 Data Structures

BACKGROUND INFORMATION on Images

Prof. Sudeep Sarkar

Images are just a collection of numbers

Images are abound. We have photographic images, video images, medical CAT scan images, satellite images, and so on. One muses, can I play around with them in my computer? But, how does one capture a representation of an image in the computer? Surely one cannot use chemicals as is done for photographic films. Somehow the photographs have to be changed into collection of numbers, which are easily stored and manipulated in the computer. To enable us to accomplish this, there are various devices such as scanners or digital cameras. These devices represent an image as **an array of numbers**.

Consider for example a scanner. How does it change a photograph into an array of number? The idea is simple. Imagine overlaying a rectangular grid on a photographic image. The cells of the grid corresponds to the individual elements of the array, which we will refer to as the pixels. An image is just a collection of numbers representing the average value of the intensity (brightness) of each of these grid cells or pixels.

Different aspects of the image to array coding.

An image as defined on, say a photographic film, is a continuous function of brightness values. Each point on the developed film can be associated with a gray level value representing how bright that particular point is. To store images in a computer we have to sample and quantize (digitize) the image function. **Sampling** refers to considering the image only at a finite number of points. And **quantization** refers to the representation of the gray level value at the sampling point using finite number of bits. Each image sample is called a **pixel**. Your typical desktop image scanner does sampling and quantization for you.

One of the simpler schemes for sampling is for a **regular** grid of squares or rectangles. We can visualize the process as overlaying an uniform grid on the image and summing the image function within each grid square. Finer the grid, better the resolution of the image; coarser the grid, the more is the observed "pixelization" (see Fig. 1).

At each pixel (or at each grid square) we usually represent the *continuous* gray level value using an **discrete** range of integers, typically from 0 for black to 255 for fully white. This conversion from a continuous range of brightness to a discrete range of brightness is referred to as **quantization**.

In what format are the images stored in a computer?

A digital image is essentially a collection of pixel values. There are various formats of storing an image in a file. Essentially they all involve storing the pixels values along with other relevant information about the image. The image format we will be using is called PGM or Portable Gray Map.

The extended format for a PGM file is given in Appendix of this document. For our example images we have a simplified PGM format. A typical image file will have data as shown below

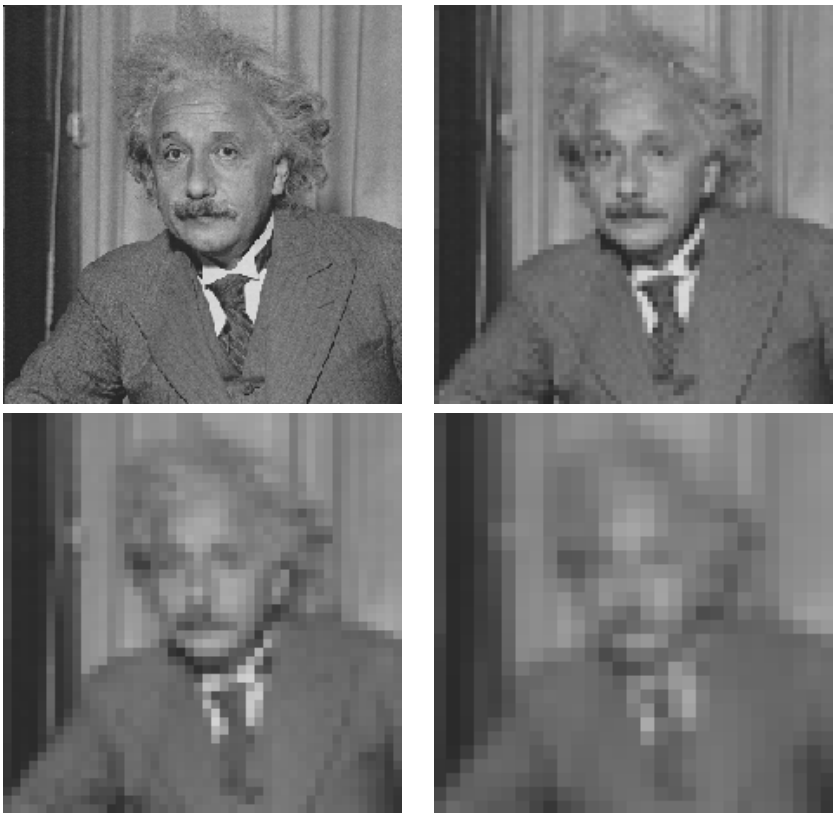

Figure 1: Images at different resolutions. Note the pixelization effect.

```
P5
401 225
255
POF[]OOMQaXZ‘kp‘YLCN^m[V^f]SKDXdbSW[bYHGOLMVUTWX[WSEHNPW]dkgggllfZXUVTOPOE?
FDLSLC<>@D=JTPNQZZSX[]ZUOQPDAEGEDHNPTGAT[Zeklrj‘ZNCAA?CGKK[c_PUeXPY_loswoln
WFLY]TWVV‘_ZOZgjXW^]e_QU^UJHLKLPHGDCKA;9EQPLHJU[TUWULVQF[[[[NJMPQLFAKVSJOW_
fd_VSNS^‘_Z]fh}~xra‘rtqqhfhqjeno‘^Xb_[UVtj][SPXjqbkyn[QYb^]UQXd
efebWKLcmXZ^XXSKKZZOP‘r{k^e^T_vyokyx
...
```

The first line is a "magic number" for identifying the file type. A pgm file's magic number is the two characters "P5".

The second line contains two numbers denoting the image width and image height, respectively.

The third line has the maximum number of gray levels which is, in our case, 255.

The data in fourth line onwards are the pixel values in binary format in row scan order. That is,

each byte corresponds to a an image pixel values. There are a total of (width * height) gray values. The pixels are stored in row-scanned order, starting at the top-left corner of the image, proceeding in normal English reading order. Note that the whole image is stored as one big line!

How do I "see" images on the computer?

There are three options. You might be able to mix and match these options to view the images.

1. You can use the ImageMagick tool, which is available on suntan or babbage. However, you have to have remote X-window access to these machines. To invoke the tool, just type `display` on suntan or babbage.
2. You can use Netscape to open an image file and it will try to pull up the image, assuming that you have the right viewer plug-in installed. However, most Netscapes can handle GIF formatted image files. So, you can first convert the image file from PGM to GIF format, using the `convert` function that is available on suntan and babbage, e.g. by typing `convert block.pgm block.gif`. This will convert the PGM image in the file `block.pgm` into a GIF formatted image file `block.gif`. Then, you can pull this GIF file up in Netscape.
3. You can also use standard PC utilities such as Paintbrush or Word to view the image but first converting from PGM to PCX format using, as above, the convert routine that is available on suntan or babbage. And then, pulling the PCX formatted image file up in Paintbrush or Word.

How do I get a print out of the image?

The `.pcx` formatted image can be imported into a Microsoft Word or a WordPerfect document and printed along with a document.

If you want independent print out of the images, then you can again use suntan to convert the PGM file into a Postscript file, which can be sent to the printer. The steps are: (all on suntan)

```
convert block.pgm block.eps

lp -d lw118 block.eps
```

What should be submitted for each assignment?

All assignments are to be completed individually

For each assignment you should submit the following by the due dates and times.

1. *On Paper:* Submit, on paper (to me) by the due date and time:
   (a) A written five (maximum) page report discussing your results and experience. The report should include the following: (i) introduction, (ii) flowchart of the algorithm, (iii) description of how would one run and use your code, (iv) some typical results or outputs, (v) a section describing the programming bugs and errors you ran into (vi) a section describing what you learned from the assignment.

   Note that you are expected to submit a professionally appearing report and it should be well organized, clear, neat, with adequate visual aids such as figures and images. Writing is a very important part of the education process and should not be ignored.
   (b) A print out of the source code.
2. Email the source code to Yelena Mukomel (mukomel@csee.usf.edu) by due date and time.
3. **Demonstrate the code to your TA between 2:30am – pm on Tuesday & Thursday within one week of the assignment due date. She will have a sign-up sheet put up on his door one week before assignment due date. Make sure you do not change the timestamp of the source code files on your `engineering` account before you have demonstrated your code to the TA.**

Grading Rubric

The grading of the each assignment will be along the following lines.

1. The writeup is out of 5 points. One points each for the following:
   (a) Flowchart
   - **0.0:** No flowchart or just one global level flowchart.
   - **0.5:** Flowchart for each function exists, but not detailed enough.
   - **1.0:** Detailed flowchart presented and it is clear that the flowchart was not created as an after thought.

   (b) Description of usage,
   - **0.0:** No guidance on how someone would run and execute the program.
   - **0.5:** Some guidance is provide but all possible interaction modes are not described.
   - **1.0:** Very detailed guidance is provided to an user including all failure modes.

   (c) Results
   - **0.0:** No results are presented.
   - **0.5:** Results are presented with no discussion. Only a couple of test cases presented.
   - **1.0:** Evidence of extensive testing along with comprehensive discussion on a some test cases.

   (d) Programming bugs and errors

- **0.0:** No programming bugs and errors listed.
- **1.0:** Programming bugs and errors that cropped are clearly documented.

(e) Overall presentation.

- **0.0:** Poor job. Looks likes a report prepared within the last 8 hours.
- **0.5:** Includes all parts, but lots of grammar and spelling mistakes, with poor document structure.
- **1.0:** Excellent document structure. Little or no spelling and grammar mistakes. Easy to read and understand. Documents well the assignment experience.

2. The source code is out of 5 points. One point each for the following:

(a) The existence of the code itself

- **0.0:** No code submitted.
- **0.5:** Part of the required code submitted.
- **1.0:** All components have been submitted.

(b) Proper indentation of the code

- **0.0:** Little or no proper indentation of code.
- **0.5:** Proper indentation at some places.
- **1.0:** Proper indentation everywhere. The code looks pretty.

(c) Comments

- **0.0:** Less than 10 sentences of comments throughout the program.
- **0.5:** More than 10 sentences of comments but the comments are too sparse and too little. It is not possible to understand the logical flow of the program from the comments.
- **1.0:** Well written comments at appropriate places. One can understand the logic of the program just by reading the comments. All exceptions, special cases, limitations are outlined.

(d) Proper naming of the procedures, functions, and variables,

- **0.0:** Lots of single or two letter and meaningless names used.
- **0.5:** Some of the names are meaningful and some are meaningless.
- **1.0:** Naming method used is sound. The functions and variables do and represent what they mean.

(e) Modularity of the code.

- **0.0:** Have functions that are longer than a page. Very few functions and procedures used.
- **0.5:** Some parts of the code are modular and some are not.
- **1.0:** The overall program has been broken down in small and meaningful units that are well put together.

3. The actual execution of the code is out of 10 points.

- **0.0:** The code does not compile. There are syntax errors.
- **1.0:** The code compiles but it crashes upon running.
- **2.0:** The code compiles and executes (does not crash).
- **3.0:** The code executes and does not crash upon erroneous user interaction.

- **4.0:** The code executes, does not crash upon erroneous user interaction, and it has excellent user interaction.
- **4.0 to 10.0:** Grading is proportional to the required functionalities that have been coded (and runs) correctly.

NOTE: Your TA will try to compile and run the code on the csee or eng machines (babbage/suntan,sunblast). If your submitted code does not run there, you will not get any points. So, if you write PC based code, it is YOUR responsibility to submit a version that runs on the csee machines.

Late assignments (or parts thereof) will be penalized 20% of the maximum possible grade for each extra day. Note that computer systems have a tendency go down unexpectedly. So, plan accordingly and do not wait for the last minute.

Penalty for any unethical activity is FF