Simple Image Processing

- Image as 2d array of pixel values
- point/pixel operations
- area operations

2D Array of Pixel Values

```java
class Pixel {
    int R, G, B;
};

Pixel image[W][H];

image[0][0].R = 0; // set the red val to 0

// fill image with black
for (int r=0; r<H; r++)
    for (int c=0; c<W; c++)
        image[c][r].R = image[c][r].G = image[c][r].B = 0;
```

Px Array

- 0<=R,G,B<256
- each image[c][r] represents a pixel value
- iterate by row and column
- Ugly sample code: Image.C (demo)
Point operations

- modify individual pixel values
- examples:
  - convert rgb to grayscale
  - convert rgb to negative
  - darken/lighten an image

RGB to grayscale

- for each pixel:
  - set R,G,B to the ave. of R, G, B
  - use weights to conform to HVR
  - e.g., ave = (r + 2*g + b) / 4
  - DEMO

RGB to negative

- for each px:
  - r' = 255 - r;  g' = 255-g;  b' = 255-b

Darken/lighten

- multiply each pixel by a positive factor
- add a constant to each pixel
- try it and see if you can find a difference
### Bayer Pattern

- Avoid boundary pixels first
- For each px at \(c,r\) in rgb image
- Its 8 nearest neighbors are \(c-1,r-1 \ldots c+1,r+1\)
- Assign the color of the bayer filter at \(c,r\)
- Calculate the averages of the other bayer px
- DEMO

### Subsample

- Reduce px dimensions, keeping resolution
- Throw away alternate pixels
- DEMO

### Add Images

- Matrix addition: adds px by px
- Add through a mask
- Add to masked area
- Dissolve - add alternate px

### Efficiency

- Need to reduce number of operations per px
- 640x480 image is 300K px
- Color image is 3 bytes per px
- Brightness operation would cost almost a million (900K) multiplications
- Exacerbated with video