IMAGE PROCESSING

- NAME : JITENDRA KAILAS WAGH
- CLASS :SEIT
- ROLL NO :82
- BATCH : S4

Color images

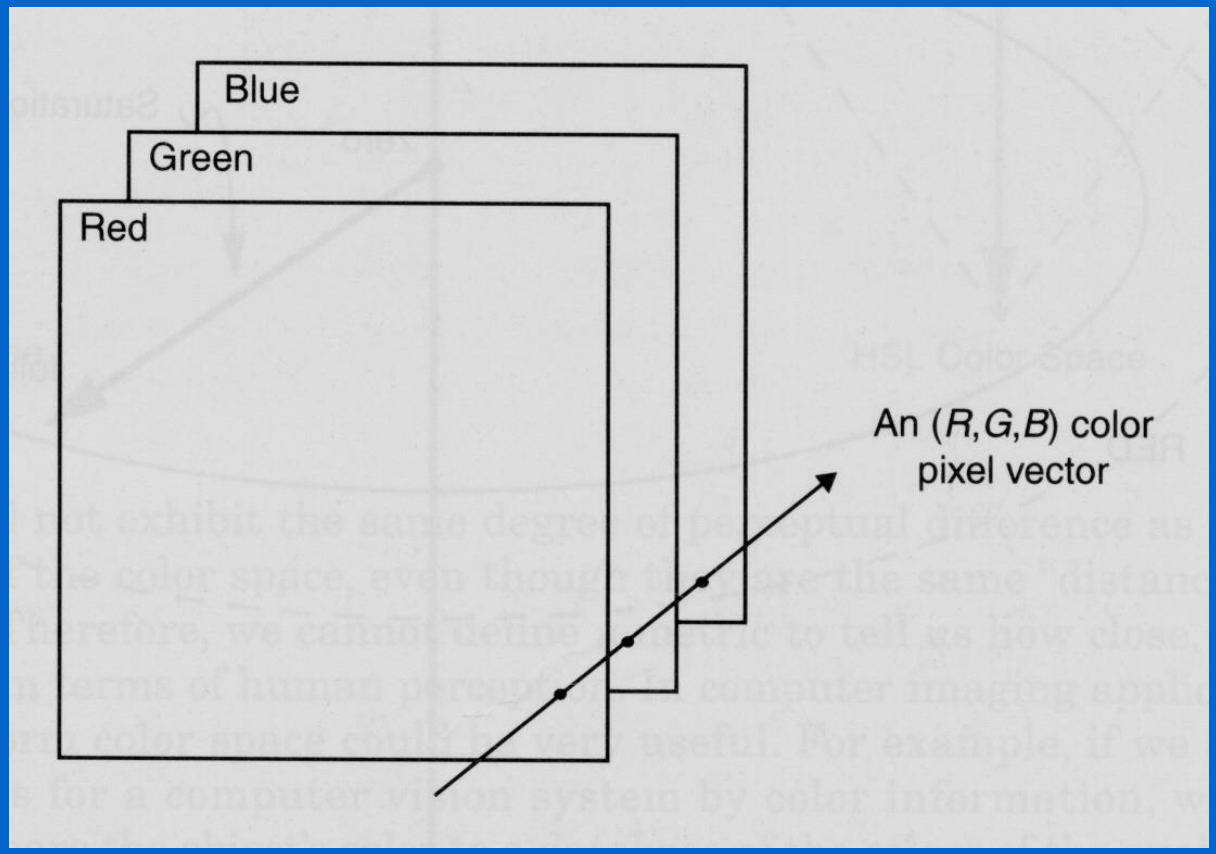
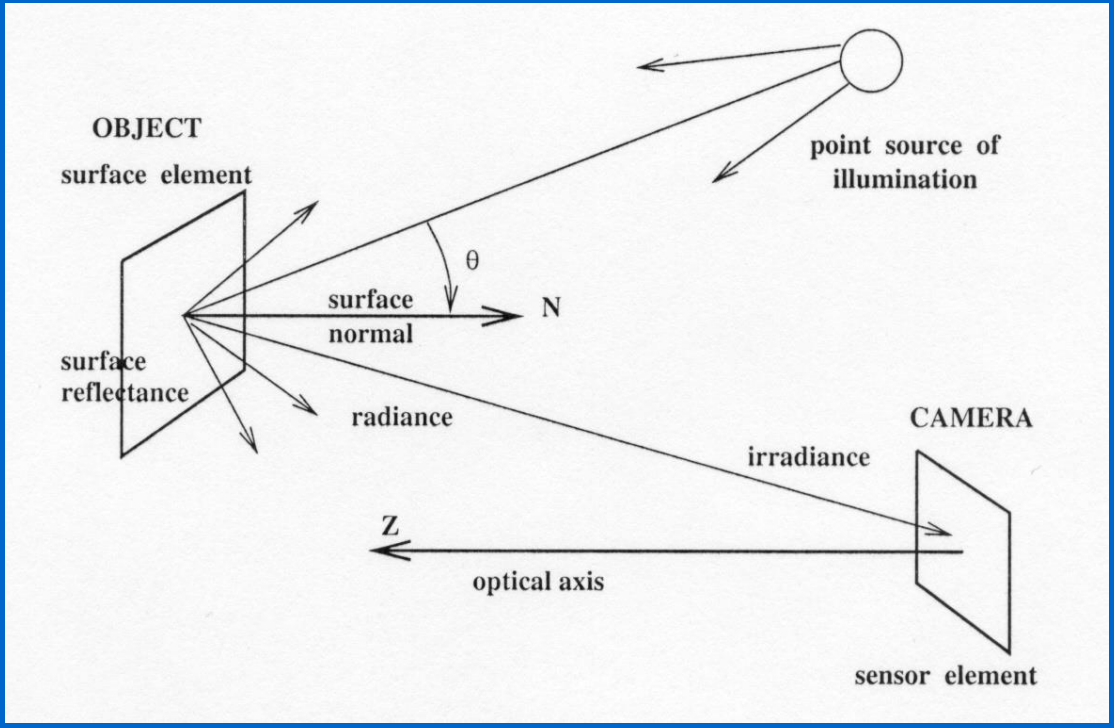


Image formation

- There are two parts to the image formation process:
  - The **geometry of image formation**, which determines where in the image plane the projection of a point in the scene will be located.
  - The **physics of light**, which determines the brightness of a point in the image plane as a function of illumination and surface properties.

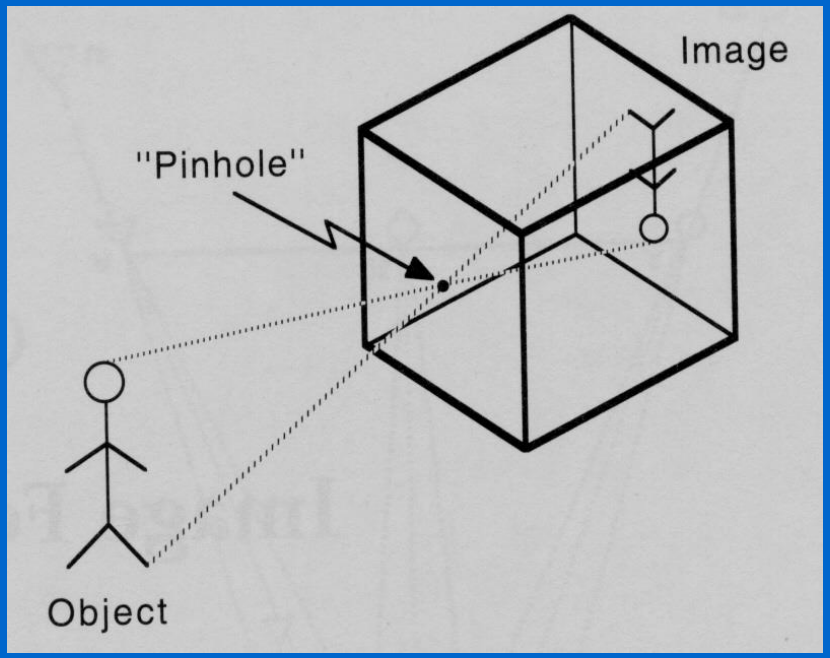
A Simple model of image formation

- The scene is illuminated by a single source.
- The scene reflects radiation towards the camera.
- The camera senses it via chemicals on film.



Pinhole camera

- This is the simplest device to form an image of a 3D scene on a 2D surface.
- Straight rays of light pass through a “pinhole” and form an inverted image of the object on the image plane.



$$x = \frac{fX}{Z}$$

$$y = \frac{fY}{Z}$$

Image formation (cont'd)

- Optical parameters of the lens
  - lens type
  - focal length
  - field of view
- Photometric parameters
  - type, intensity, and direction of illumination
  - reflectance properties of the viewed surfaces
- Geometric parameters
  - type of projections
  - position and orientation of camera in space
  - perspective distortions introduced by the imaging process

Frame grabber

- Usually, a CCD camera plugs into a computer board (**frame grabber)**.
- The frame grabber digitizes the signal and stores it in its memory (**frame buffer**).

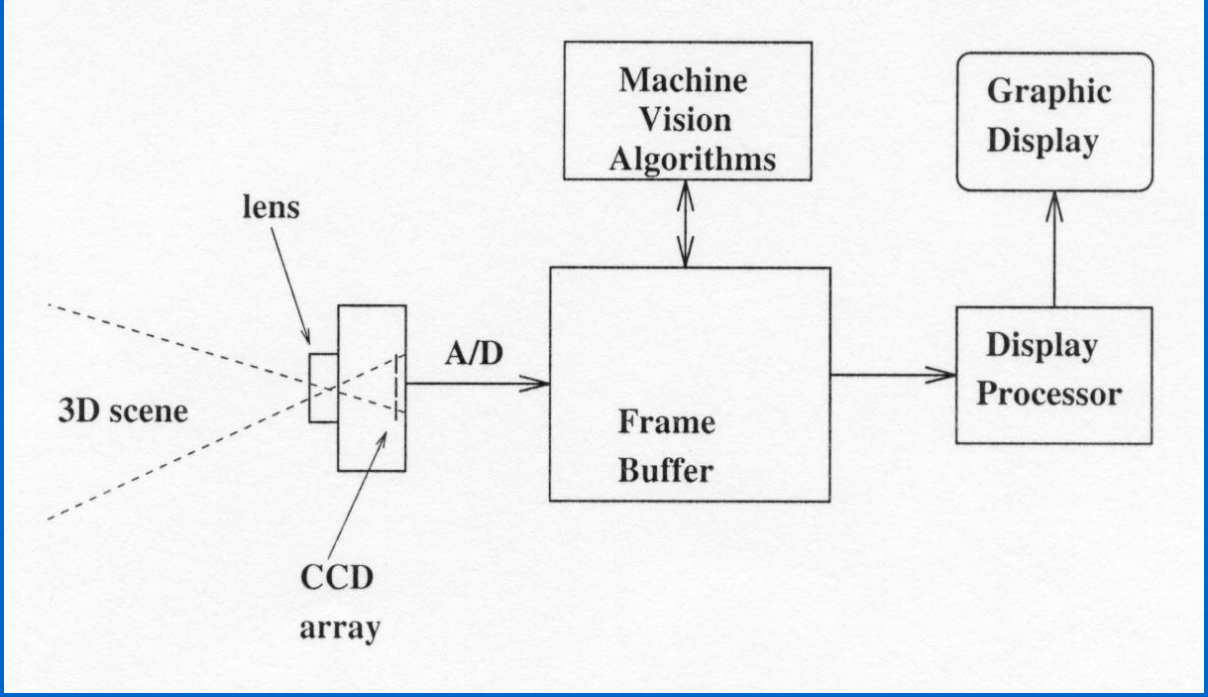
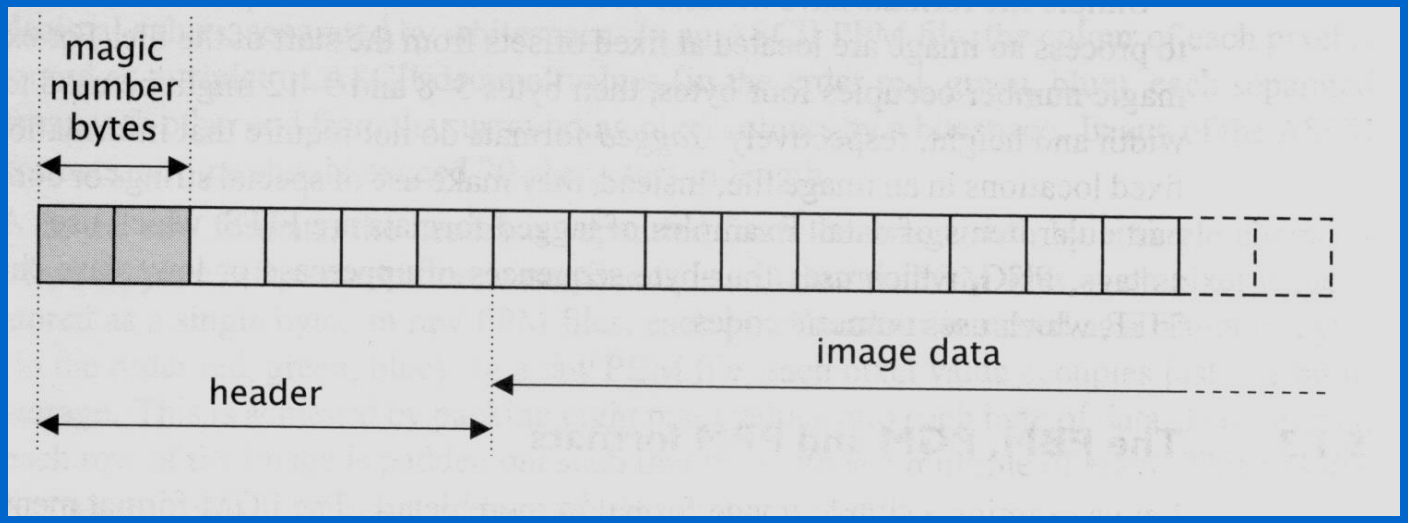


Image file formats

- Many image formats adhere to the simple model shown below (line by line, no breaks between lines).
- The header contains at least the width and height of the image.
- Most headers begin with a **signature** or “magic number” - a short sequence of bytes for identifying the file format.



Comparison of image formats

Image File Format	No. Bytes "Hi"	No. Bytes "Cars"
PGM	595	509,123
GIF	192	138,267
TIF	918	171,430
PS	1591	345,387
HIPS	700	160,783
JPG (lossless)	684	49,160
JPG (lossy)	619	29,500

ASCII vs Raw format

- ASCII format has the following advantages:
  - Pixel values can be examined or modified very easily using a standard text editor.
  - Files in raw format cannot be modified in this way since they contain many unprintable characters.
- Raw format has the following advantages:
  - It is much more compact compared to the ASCII format.
  - Pixel values are coded using only a single character !

Image Class

```
class ImageType {
 public:
   ImageType();
  ~ImageType();

   // more functions ...

private:
   int N, M, Q; //N: # rows, M: # columns
   int **pixelValue;
};
```

An example - Threshold.cpp

```
void readImageHeader(char[], int&, int&, int&, bool&);
void readImage(char[], ImageType&);
void writeImage(char[], ImageType&);

void main(int argc, char *argv[])
{
 int i, j;
 int M, N, Q;
 bool type;
 int val;
 int thresh;

 // read image header
 readImageHeader(argv[1], N, M, Q, type);

 // allocate memory for the image array
 ImageType image(N, M, Q);

 // read image
 readImage(argv[1], image);
```

Threshold.cpp (cont'd)

```
cout << "Enter threshold: ";
cin >> thresh;

// threshold image

for(i=0; i<N; i++)
  for(j=0; j<M; j++) {
    image.getVal(i, j, val);
    if(val < thresh)
      image.setVal(i, j, 255);
    else
      image.setVal(i, j, 0);
   }

// write image
writeImage(argv[2], image);

}
```

Writing a PGM image to a file

```
void writeImage(char fname[], ImageType& image)
 int N, M, Q;
 unsigned char *charImage;
 ofstream ofp;

 image.getImageInfo(N, M, Q);

 charImage = (unsigned char *) new unsigned char [M*N];

 // convert the integer values to unsigned char

 int val;

 for(i=0; i<N; i++)
    for(j=0; j<M; j++)
     image.getVal(i, j, val);
     charImage[i*M+j]=(unsigned char)val;
   }
```

Writing a PGM image... (cont'd)

```
ofp.open(fname, ios::out);
if (!ofp) {
  cout << "Can't open file: " << fname << endl;
  exit(1);
 }

 ofp << "P5" << endl;
 ofp << M << " " << N << endl;
 ofp << Q << endl;

 ofp.write( reinterpret_cast<char *>(charImage), (M*N)*sizeof(unsigned char));

 if (ofp.fail()) {
  cout << "Can't write image " << fname << endl;
  exit(0);
 }
 ofp.close();
}
```

Reading a PGM image from a file

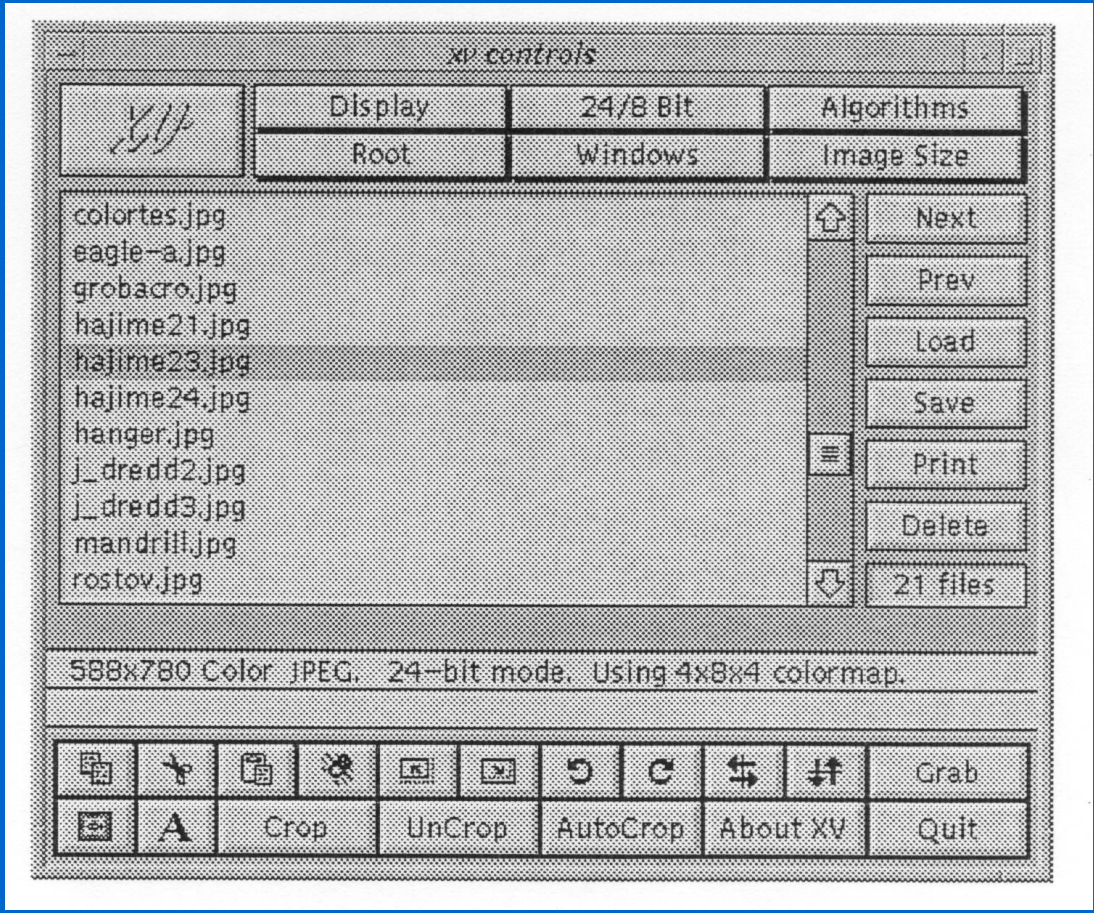
```
void readImage(char fname[], ImageType& image)
{
 int i, j;
 int N, M, Q;
 unsigned char *charImage;
 char header [100], *ptr;
 ifstream ifp;

 ifp.open(fname, ios::in);
 if (!ifp) {
   cout << "Can't read image: " << fname << endl;
   exit(1);
 }

 // read header
 ifp.getline(header,100,'\n');
  if ( (header[0]!=80) ||    /* 'P' */
      (header[1]!=53) ) {    /* '5' */
      cout << "Image " << fname << " is not PGM" << endl;
      exit(1);
}
```

Reading a PGM image …. (cont'd)

```
ifp.getline(header,100,'\n');
 while(header[0]=='#')
   ifp.getline(header,100,'\n');

 M=strtol(header,&ptr,0);
 N=atoi(ptr);

 ifp.getline(header,100,'\n');
 Q=strtol(header,&ptr,0);

 charImage = (unsigned char *) new unsigned char [M*N];

 ifp.read( reinterpret_cast<char *>(charImage), (M*N)*sizeof(unsigned char));

 if (ifp.fail()) {
   cout << "Image " << fname << " has wrong size" << endl;
   exit(1);
 }

 ifp.close();
```

How do I “see” images on the computer?

- Unix: xv, gimp
- Windows: Photoshop



How do I display an image from within my C++ program?

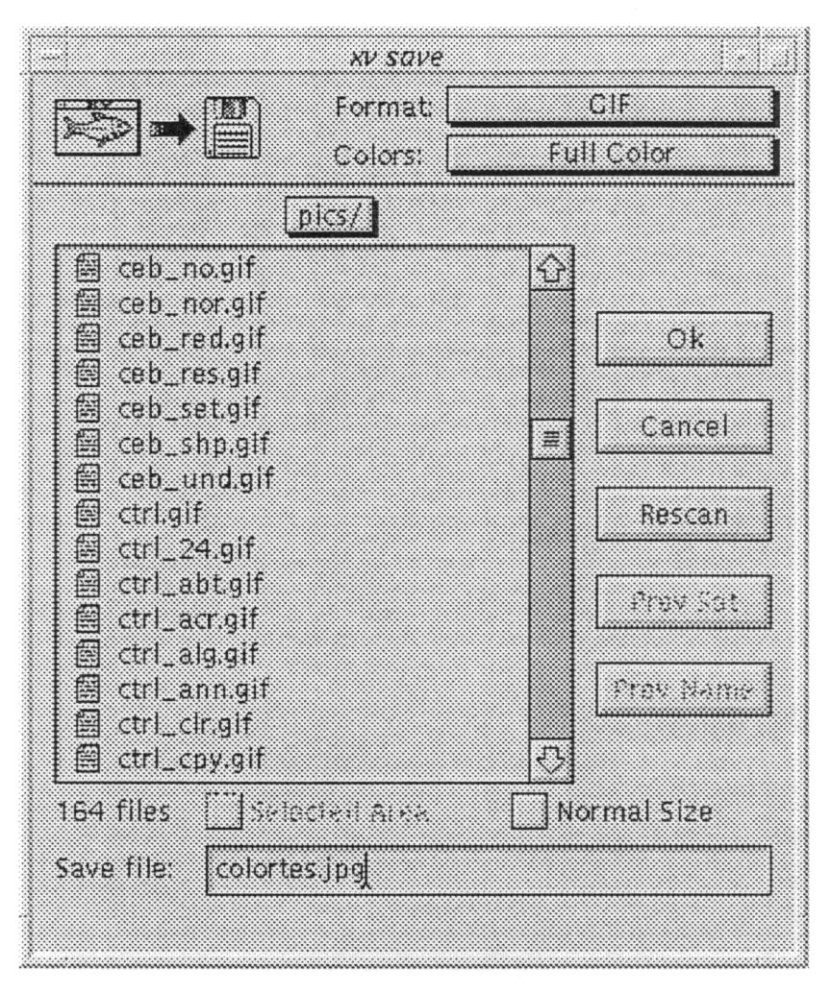
- Save the image into a file with a default name (e.g., tmp.pgm) using the WriteImage function.
- Put the following command in your C++ program:

system("xv tmp.pgm");

- This is a system call !!
- It passes the command within the quotes to the Unix operating system.
- You can execute any Unix command this way ….

How do I convert an image from one format to another?

- Use xv's "save" option
- It can also convert images



How do I print an image?

- Load the image using “xv”
- Save the image in “postscript” format
- Print the postscript file (e.g., lpr -Pname image.ps)

- THANK YOU