Applied Computer Vision

David Vernon
Carnegie Mellon University Africa

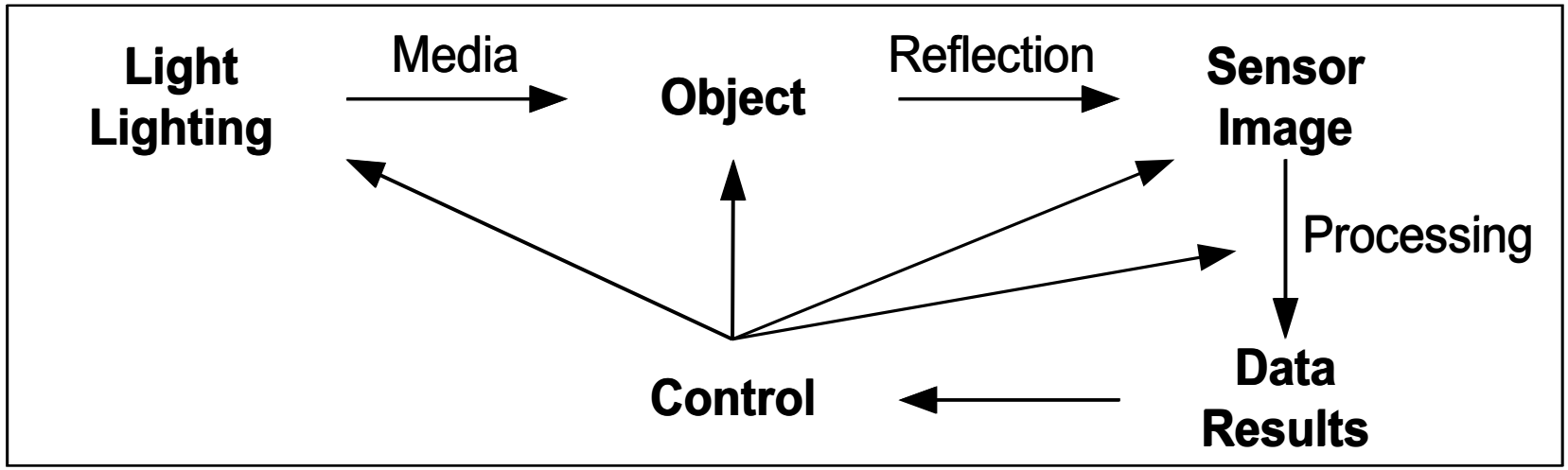
vernon@cmu.edu
www.vernon.eu

Lecture 2

Optics, Sensors, Image Formation

Machine Vision

- Application of computer vision to factory automation
- A MV system is a computer that makes decisions based on the analysis of digital images
- Take into account full chain of system components

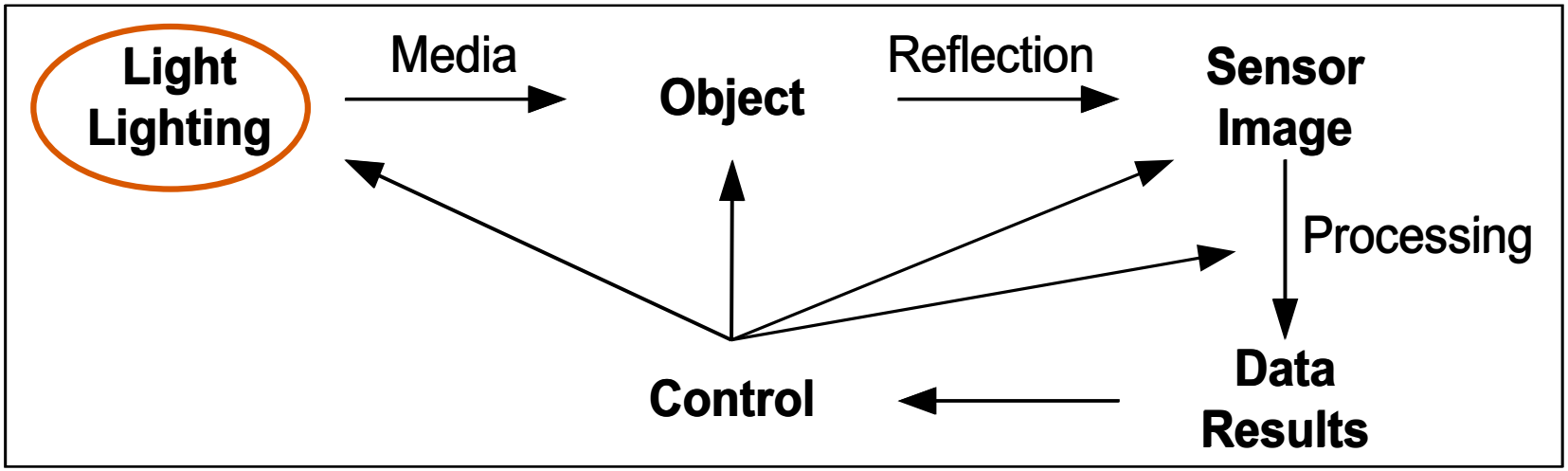


Components of a machine vision system

Machine Vision

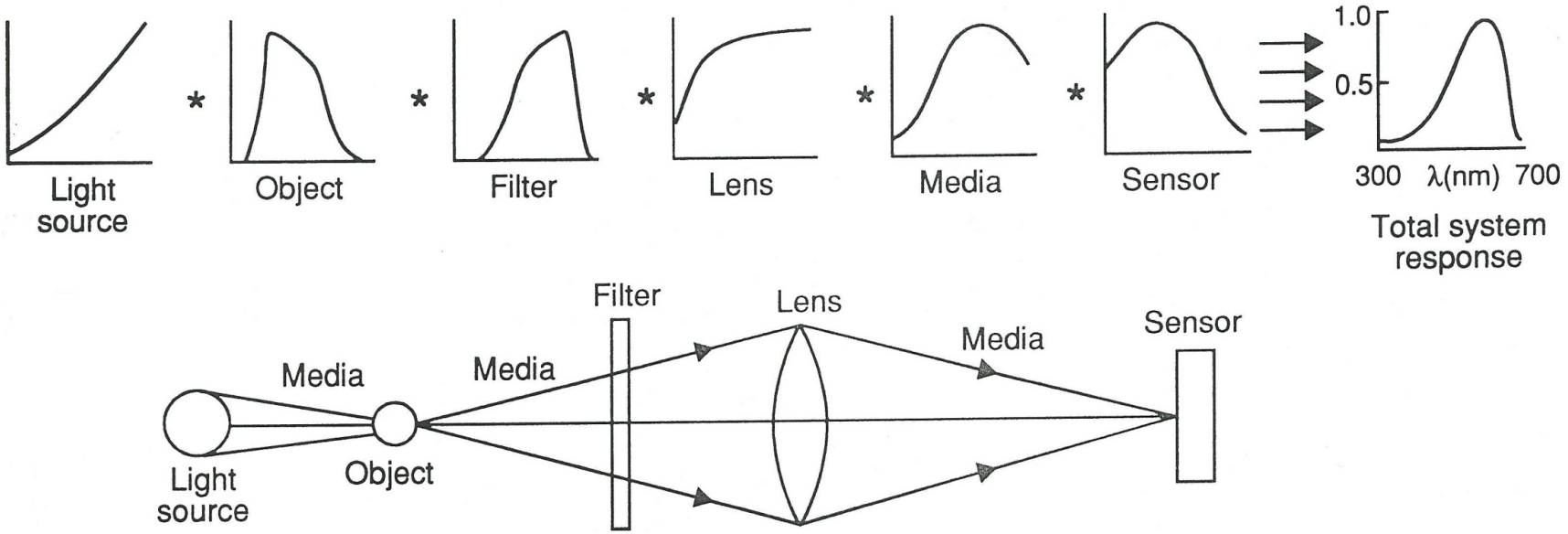
Machine Vision in industry is based on controlled light

- Set lighting conditions to simplify the task
- 50% of a machine vision solution is good illumination



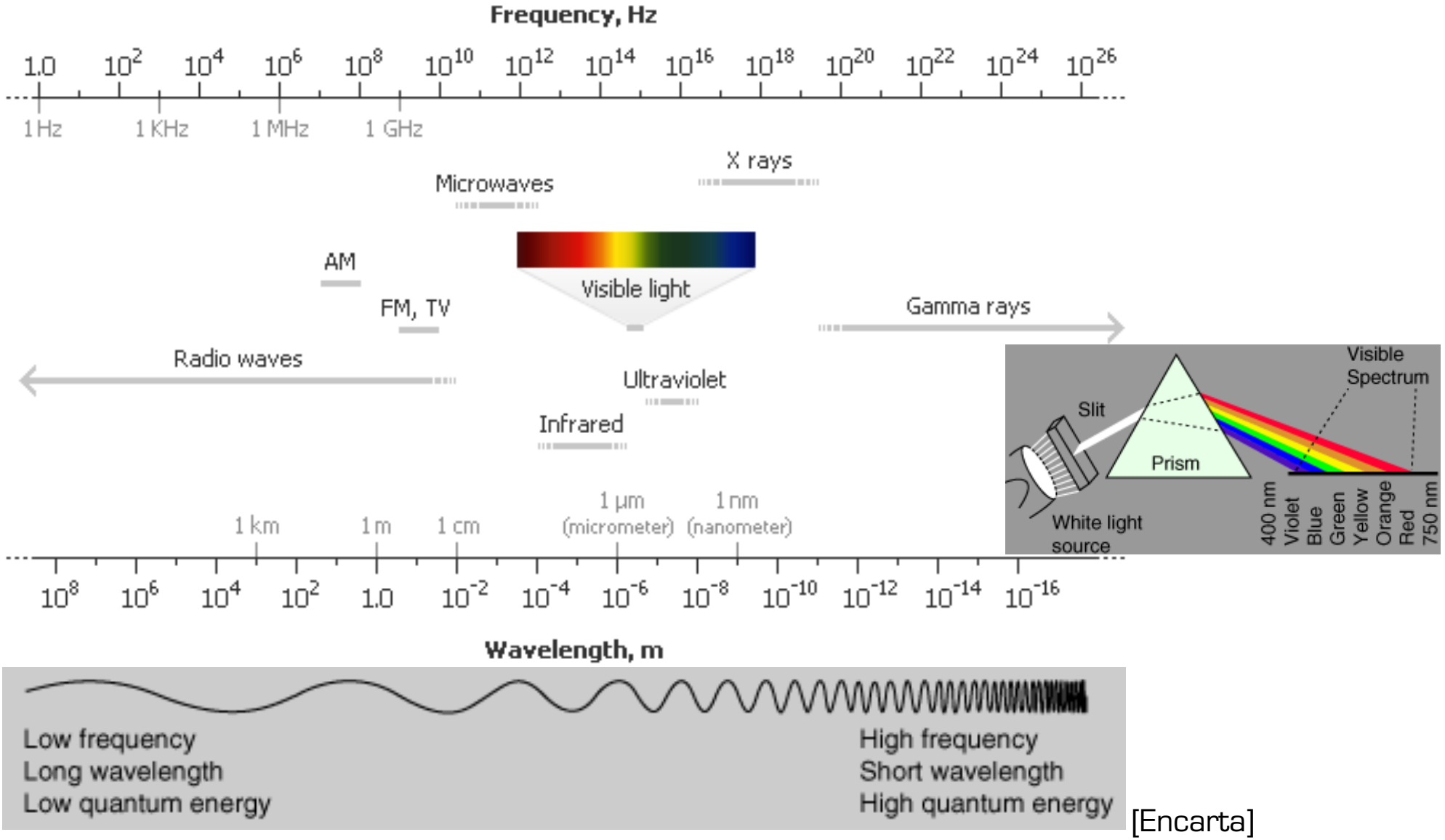
Components of a machine vision system

From Light Source to Image



- Many different influencing factors
- Complete modeling is impossible, so far
- Solutions only under very constrained conditions

Electromagnetic Spectrum



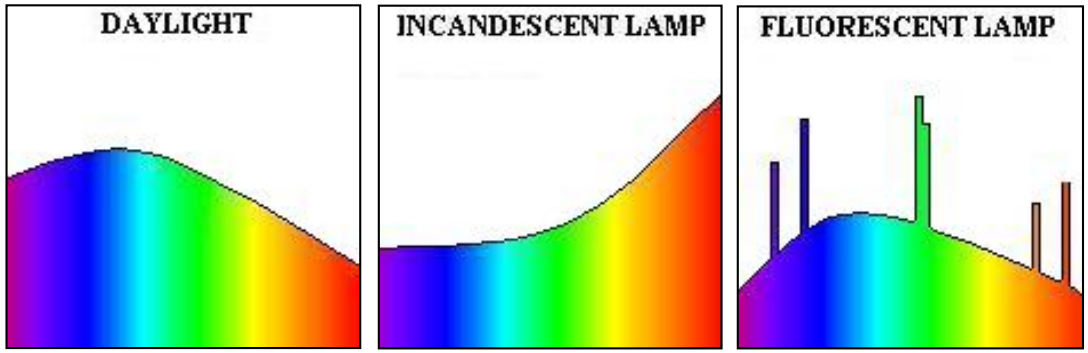
Low frequency
Long wavelength
Low quantum energy

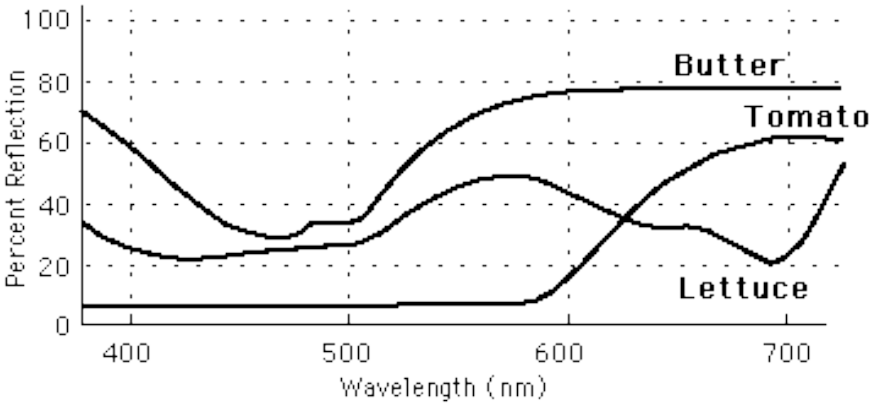
High frequency
Short wavelength
High quantum energy

[Encarta]

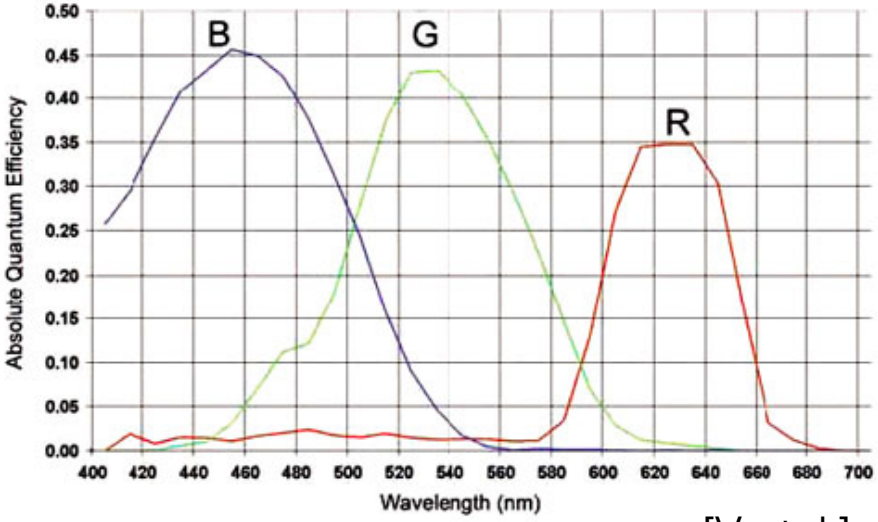
Illumination and Object Spectra

Spectra of different sources and objects

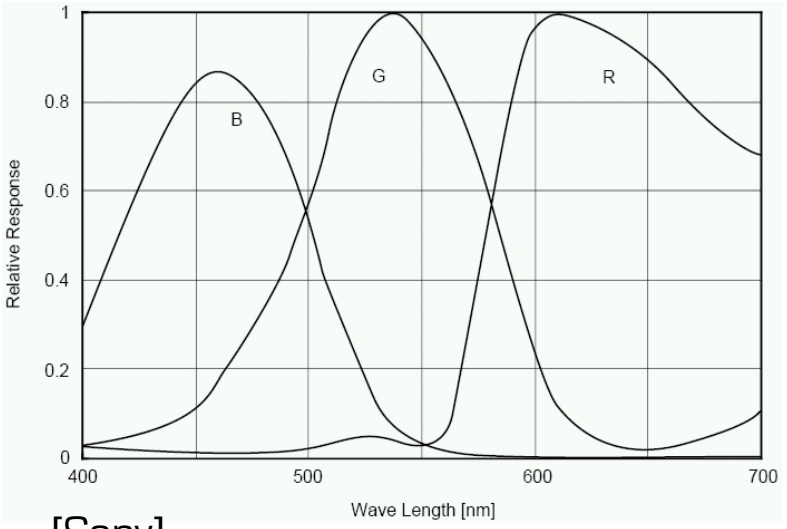




Spectral sensitivity of two colour cameras



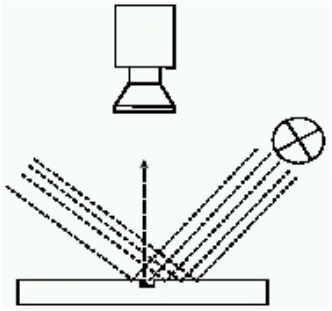
[Vaytek]



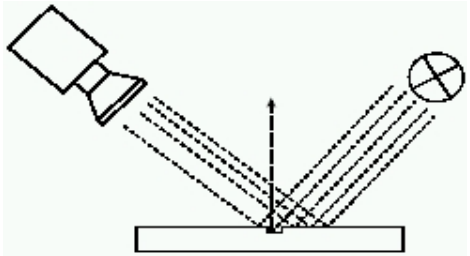
[Sony]

Illumination

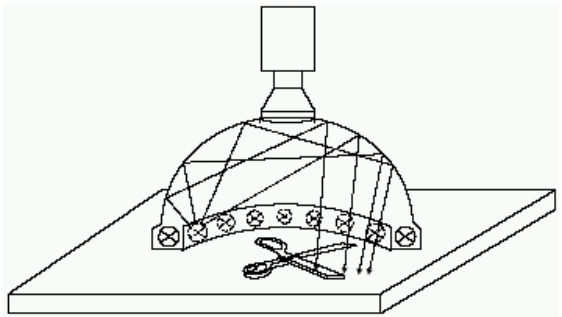
Dark field

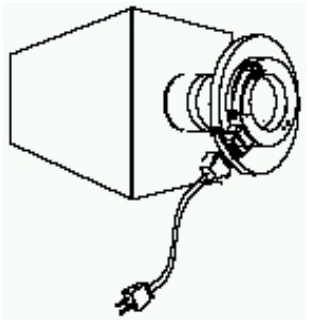


Bright field

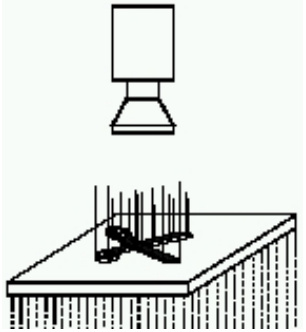


Diffuse: bell or ring light

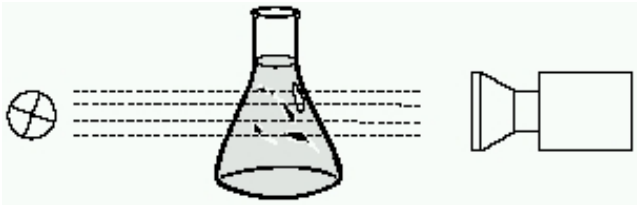




Through light



Transparent object



Cast shadow

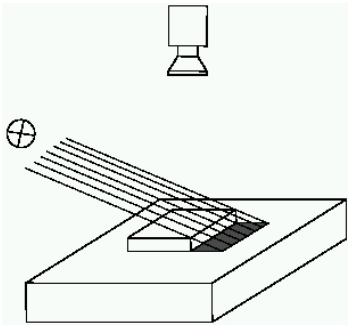


Image Formation

- Orthgraphic Projection
  - Normal or parallel projection: x = X, y = Y
  - Simplification when object is far away

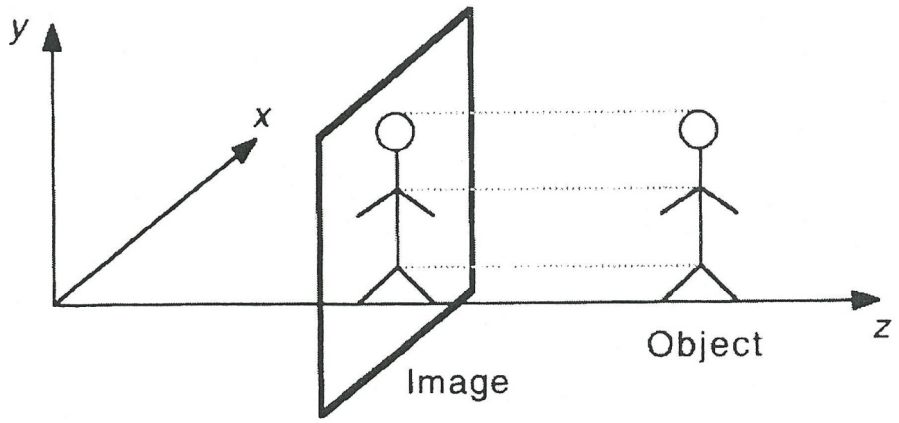
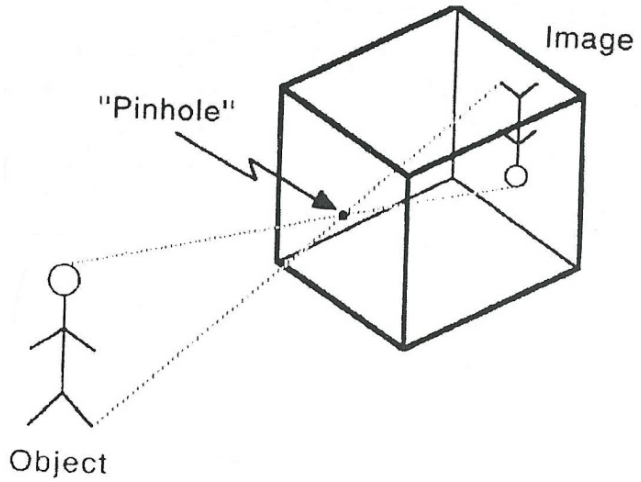
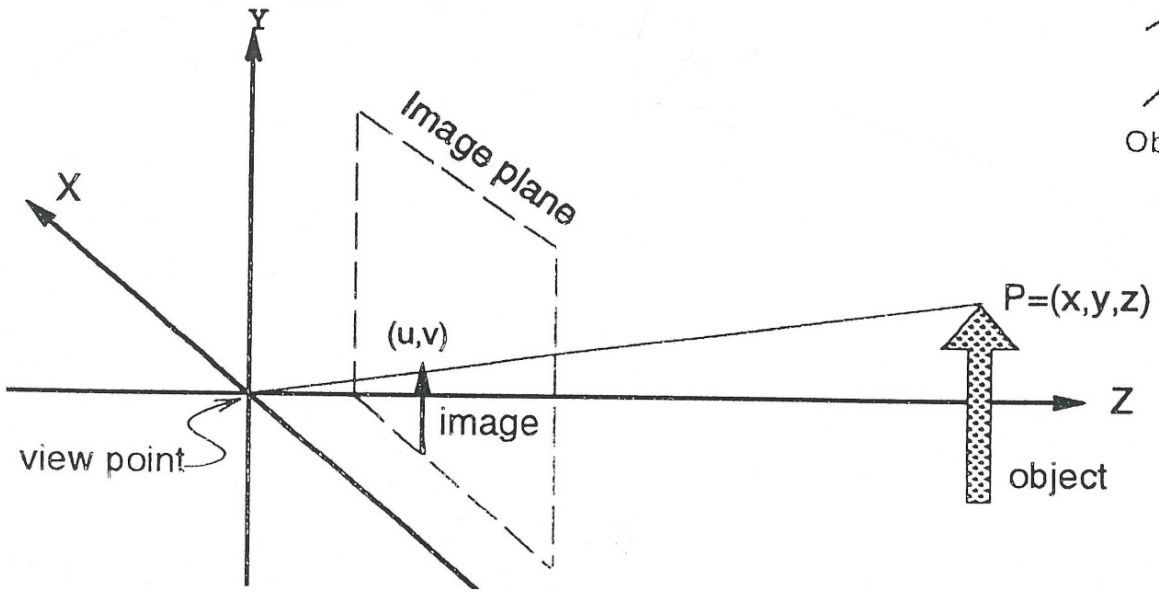


Image Formation

- Perspective Projection
- Pin-hole model of a camera





Basic model of imaging process

Lenses

Lenses are required to focus part of the visual environment onto the image sensor

Lenses are defined by:

- their Focal Length (quoted in millimetres)
- their Aperture (the f number)

These parameters determine the performance of the lens in terms of light gathering power and magnification, and it often has a bearing on its physical size

Lenses

The focal length of a lens is a guide to the magnification it effects and its field of view

Selecting the focal length which is appropriate to a particular application is simply a matter of applying the basic lens equation

$$\frac{1}{u}+\frac{1}{v}=\frac{1}{f}$$

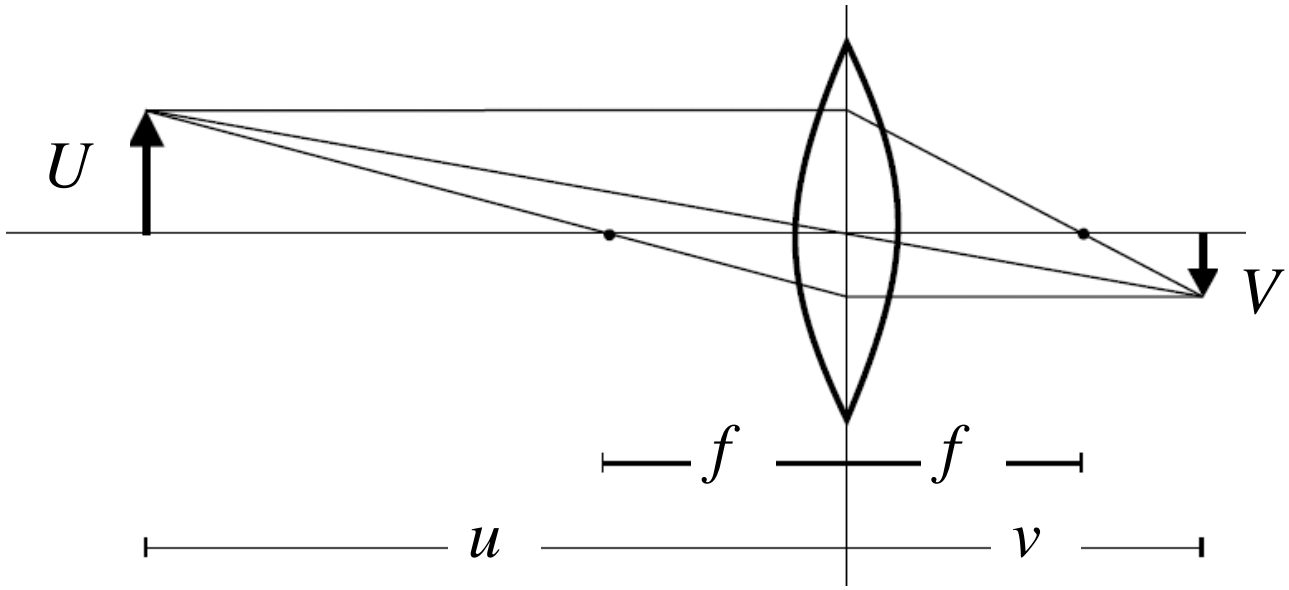
where

- v is the distance from the lens to the image
- u is the distance from the lens to the object
- f is the focal length

Lenses

Gaussian lens equations: $\frac{1}{u} + \frac{1}{v} = \frac{1}{f}$ and $\frac{V}{U} = \frac{v}{u}$

Focal length: $f = \frac{u\ V}{U\ +\ V}$



Lenses

Noting the Magnification Factor M is

$$M = \frac{image\_size}{object\_size} \frac{V}{U}$$

Lenses

Thus

$$f = \frac{uM}{M+1}$$

Hence if we know the required magnification factor and the distance from the object to the lens, we can compute the required focal length

Lenses

For example, if a 10cm wide object is to be imaged on a common 8.8 * 6.6mm sensor from a distance of 0.5m, this implies a magnification factor of :

$$M = \frac{8.8}{100} = 0.088$$

So,

$$f = \frac{500 * 0.088}{1.088} = 40.44$$

Thus, we require a 40.44 mm focal length lens

Typically, one would use a slightly shorter focal length (*e.g.* 35 mm) and accept the slight loss in resolution due to the larger field of view

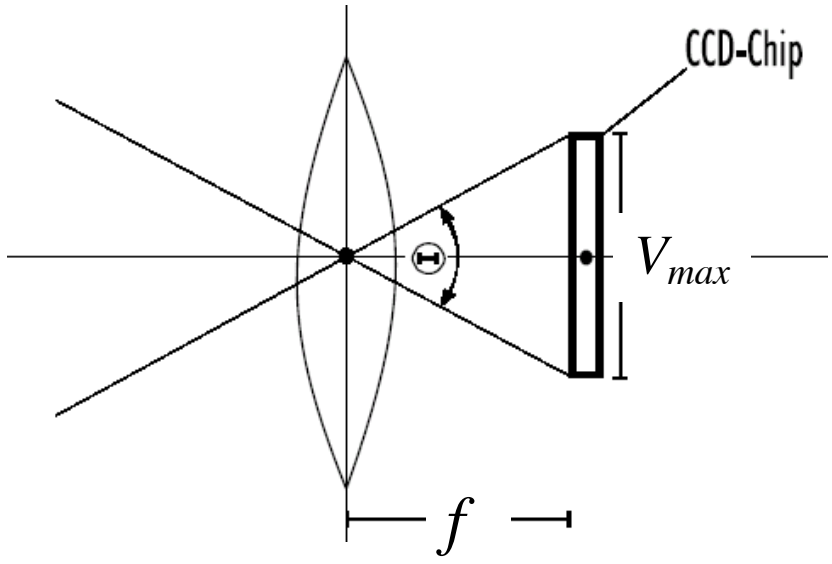
Field of View FOV

Small focal length → wide FOV

$$\theta = 2\,\mathrm{atan}\left(\frac{V_{max}}{2f}\right)$$

Large Depth of Field

- Small aperture
- Short focus length
- Large object distance



Lenses

The minimum focal distance is the minimum distance between the front of the lens and the object at which the object can still be focused

Generally speaking, lenses with short focal lengths have smaller minimum focal distances

Lenses

If the lens is required to focus a relatively small object at short distances, the minimum focal distance (typically 300mm) may be too large

In that case, an extension tube can be used to increase the distance, v, of the lens from the sensor and hence decrease the distance, u, from the lens to the object

For a given focal length, f, the lens equation stipulates that
u decreases as v increases,
if the image is to remain in focus $\frac{1}{u}+\frac{1}{v}=\frac{1}{f}$

Lenses

The *f*-number is measure of the amount of the light allowed to pass through the lens and is a normalised measure of lens aperture

It is defined by the focal length divided by the diameter of the aperture

The standard scale is 1.4, 2, 2.8, 4, 5.6, 8, 11, 16; each increase reduces the amount of light passing through the lens by one half

Lenses

The depth of field is the distance between the nearest and the furthest points of a scene which remain in acceptable focus at one aperture setting

In general, the depth of field gets larger as the focused point moves away from the lens (given constant aperture and focal length)

Also, the depth of field increases significantly as the aperture closes (*i.e.* as the f number increases) for any given focusing distance

Optical Filters

Many of the visual problems which cause difficulty interpreting a scene can often be solved by the use of simple filters on the camera lens

Some sensors are sensitive to segments of the electro-magnetic spectrum which do not convey visual data, *e.g.* IR radiation

The use of a simple IR blocking filter can solve this problem in a simple and effective manner

Sensors

Most solid-state cameras are based on

Charge-Coupled Device (CCD) or

CMOS

technology

- The basic structure of CCD technology is that of an analogue shift register consisting of a series of closely spaced capacitors
- Charge integration (accumulation) by the capacitors, photosites, caused by the photons comprising the incident light, provides the analogue representation of light intensity
- At the end of the integration period (exposure time) these charges are read out of the sensor

Sensors

CCD sensors most commonly use one of three addressing strategies:

- Interline transfer
- Frame transfer
- Column-row transfer

Sensors

Interline Transfer CCD

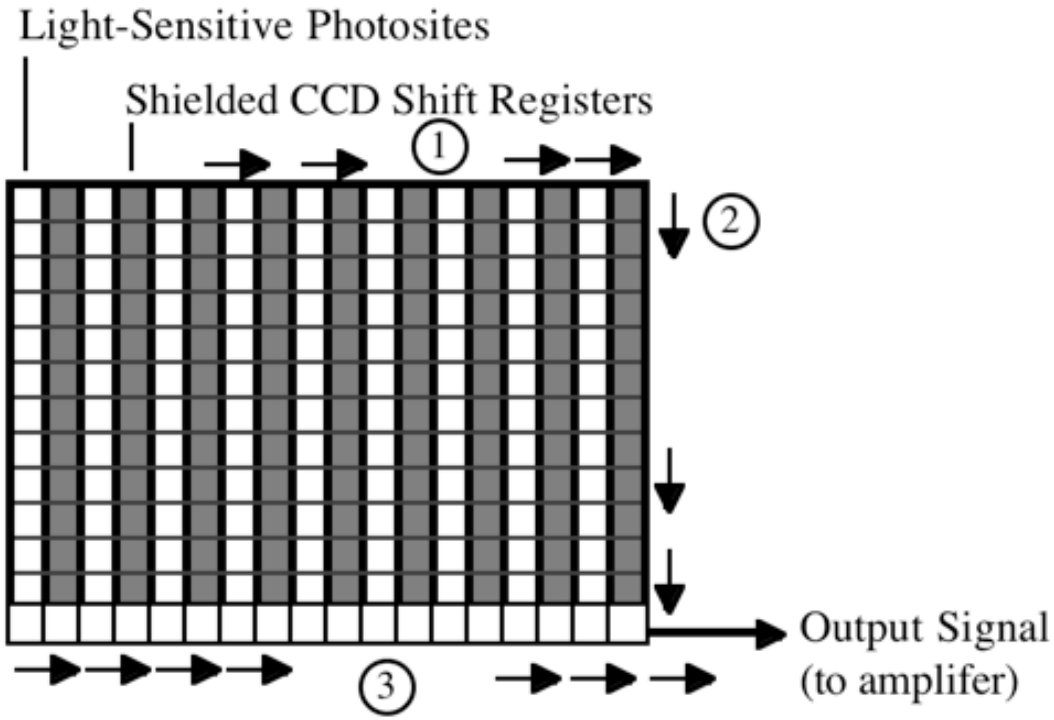
- Organised into column pairs of devices
- An imaging column of photosensors is adjacent to an opaque vertical shift register
- Charge accumulates in the imaging column until the end of the integration period.
- When it is transferred to the opaque column
- The signal then shifts vertically into a horizontal shift register that represents the picture sequentially line by line.

Sensors

① Charges are shifted to the shielded area

② The charges in the column are shifted down one cell

③ A row of charge is then shifted out



The advantage of the interline transfer is that the transfer time (to opaque storage) is short compared to the integration period.

Sensors

When transfer time approaches the integration time, solid-state sensors tend to exhibit a locally-contained spreading of the image response, called smear

Thus, the Interline Transfer minimises Smear

Sensors

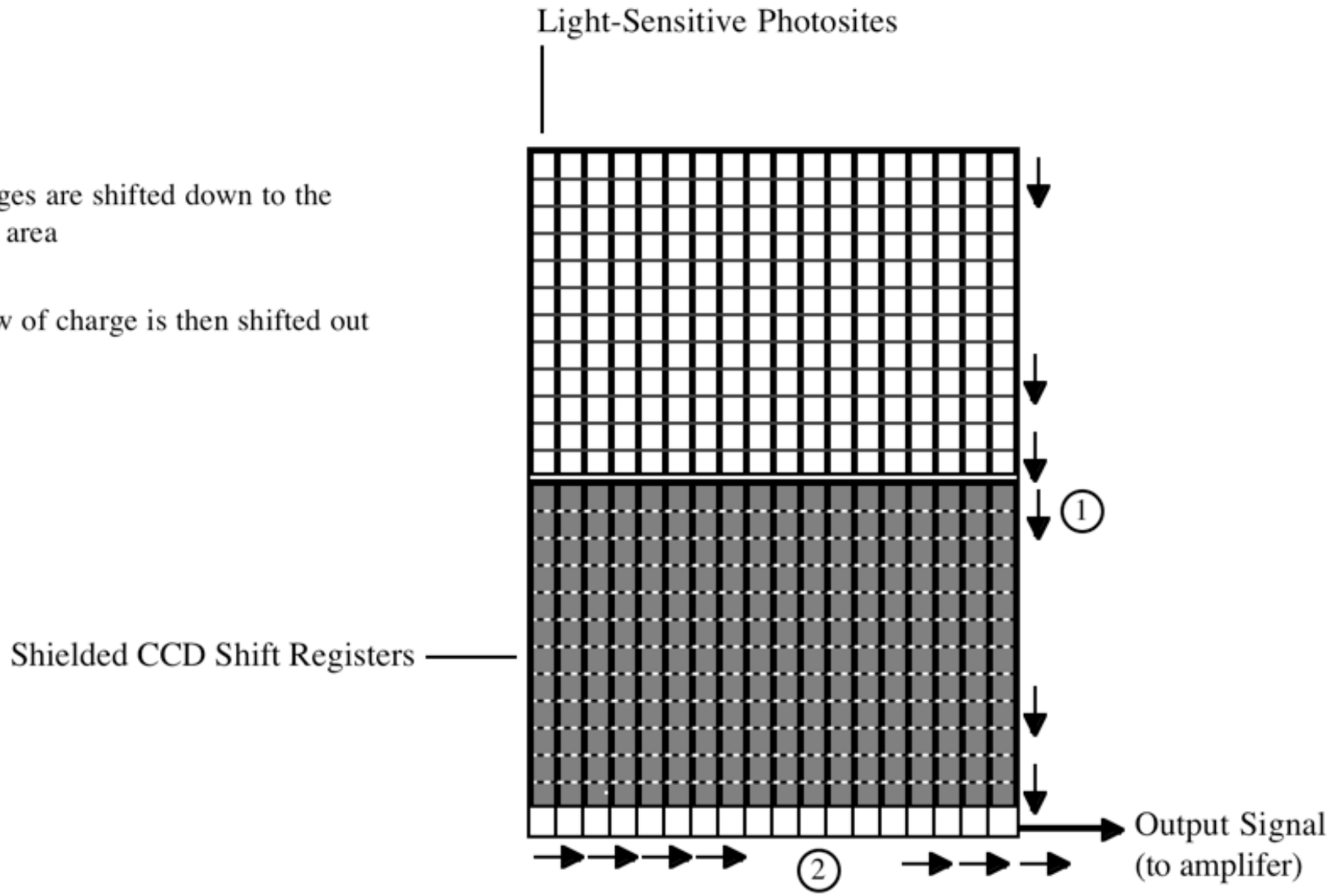
Frame Transfer

- Sensor consists of vertical columns of CCD shift registers divided into two zones
- One zone, where charge accumulates during integration time, is photosensitive
- When integration is complete, the whole array is transferred in parallel to the opaque storage area of the second zone

Sensors

① All charges are shifted down to the shielded area

② Each row of charge is then shifted out



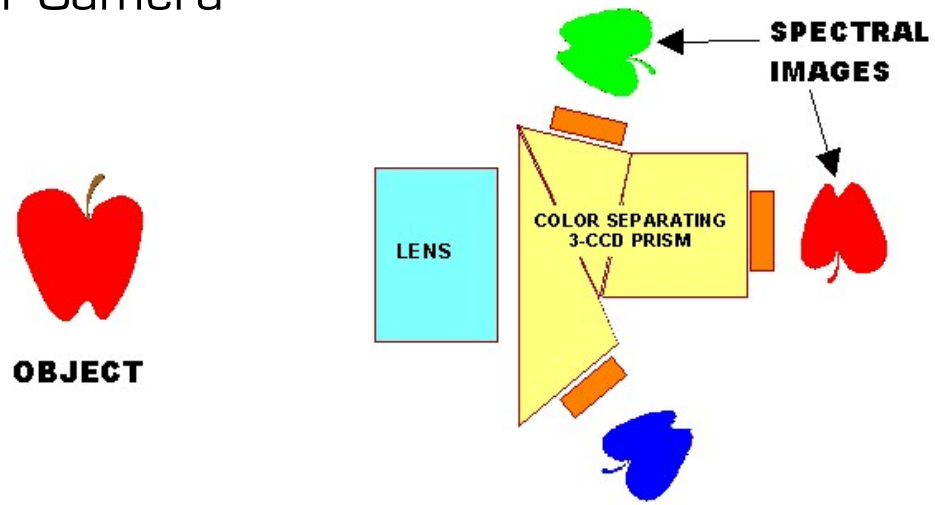
Sensors

X-Y Addressing

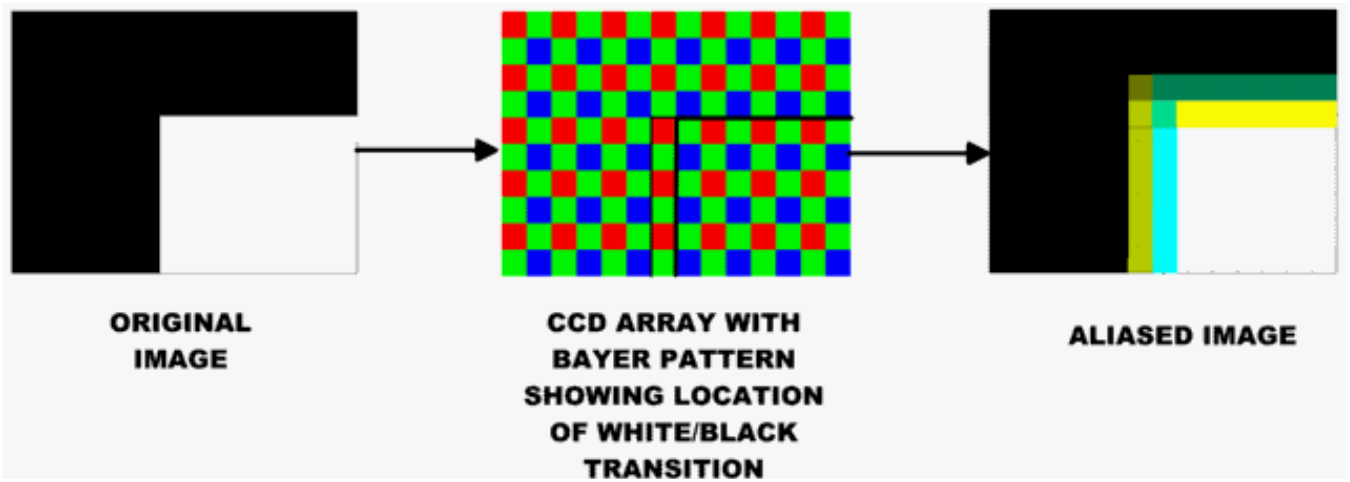
- The sensor elements are addressed by selecting individual column and row electrodes
- Charge collected under the column electrode is transferred to the row electrode and amplified for output

Sensors

3-Chip Colour Camera



1-Chip Colour Camera



Sensors

Resolution

- the purpose of the camera sensor is to convert an incident analogue optical image to some electrical signal
- the resolution of a sensor can be defined as the number of optical image elements which can be discriminated by that sensor and represented by the resulting signal
- The resolution is limited by the number of photosites in the sensor since this defines the frequency with which the optical image is sampled

Sensors

Note that in solid-state cameras the effective resolution of the sensor is approximately half that of the number of photosites in any given direction (Nyquist sampling; see later)

Resolution is also limited by the array geometry and by how much opaque material separates the photosites

> Interline transfer sensors have poorer effective resolution than frame transfer sensors due to the presence of shielded buffers between each photosensitive line

Sensors

Noise

- Noise, unwanted interference on the video signal, is defined quantitatively by the signal–to-noise ratio (S/N)
- the ratio of the amplitude of the wanted signal to the average amplitude of the interference

Sensors

Noise

- Noise, unwanted interference on the video signal, is defined quantitatively by the signal–to-noise ratio (S/N):
  - the ratio of the amplitude of the wanted signal to the average amplitude of the interference
- If one follows general practice in telecommunications and expresses the ratio of electrical variables of the same unit in logarithmic terms
  - one can compute the level, defined as 20 times the decimal log of the ratio of the linear variables and expressed in decibels (dB)

Sensors

Noise

- Thus, a signal to noise ratio of 20dB means that the picture quality is very bad (20dB implies that $\log_{10}(S/N)$ equals 1 and, thus, the signal to noise ratio is 10:1)

- For satisfactory quality, a signal to noise ratio of 40dB or more is required

Demos

- Optics
  - Focussing & extension tubes
  - Focal length
  - Depth of field

- Image acquisition
  - `imageAcquisition.exe`

OpenCV

version 2.4

For documentation on OpenCV data structures, functions, and methods, see

http://docs.opencv.org/2.4/index.html

Demos

The following code is taken from the imageAcquisition project in the lectures directory of the ACV repository

See:

```
imageAcquisition.h
imageAcquisitionImplementation.cpp
imageAcquisitionApplication.cpp
```

```
/*
  Example use of openCV to acquire and display images from file and from a video camera
  ------------------------------------------------------------------------------------

  This application reads a sequence lines from an input file imageAcquisitionInput.txt.
  Every triplet of lines contains

  1.  a filename of an image to be displayed
  2.  a filename of a video to be displayed
  3.  a camera number (typically 0 for the internal webcam and 1 for an external USB camera)

  After the video is displayed, images from the video camera are displayed.

  When display from the camera is terminated by the user (by pressing any key),
  the last image to be acquired is written to a file 'camera_image.png'
  and the cycle repeats until all filenames from the input file have been read.

  It is assumed that the input file is located in a data directory given by the path ../data/
  defined relative to the location of executable for this application.

  (This is the application file: it contains the client code that calls dedicated functions to implement the application.
  The code for these functions is defined in the implementation file. The functions are declared in the interface file.)

  David Vernon
  24 November 2017

  Added camera_number as an input
  DV 27 February 2019
*/
```

```c
#include "imageAcquisition.h"

int main() {

   int end_of_file;
   bool debug = true;
   char filename[MAX_FILENAME_LENGTH];
   int  camera_number;
   FILE *fp_in;

   if ((fp_in = fopen("../data/imageAcquisitionInput.txt","r")) == 0) {
      printf("Error can't open input imageAcquisitionInput.txt\n");
      prompt_and_exit(1);
   }

   printf("Example of how to use openCV to acquire and display images\n");
   printf("from three sources: image file, video file, and USB camera.\n\n");

   do {
      end_of_file = fscanf(fp_in, "%s", filename);

      if (end_of_file != EOF) {
         printf("\nDisplaying image from image file %s \n",filename);
         display_image_from_file(filename);

         end_of_file = fscanf(fp_in, "%s", filename);
         if (end_of_file != EOF) {
            printf("\nDisplaying image from video file %s \n",filename);
            display_image_from_video(filename);
         }

         end_of_file = fscanf(fp_in, "%d", &camera_number);
         if (end_of_file != EOF) {
            printf("\nDisplaying image from USB camera ... \n");
            display_image_from_camera(camera_number);
         }
      }
   } while (end_of_file != EOF);

   fclose(fp_in);
   return 0;
}
```

```
/*
  Example use of openCV to acquire and display images
  ---------------------------------------------------
  Interface file

  David Vernon
  14 December 2016

  Audit Trail
  -----------
  6 May 2017 Updated to process multiple images and read image filenames from an input file (imageAcquisitionInput.txt)

*/

#include "stdio.h"
#include "stdlib.h"
#include "string.h"
#include <ctype.h>
#include <iostream>
#include <string>
#include <conio.h>

//opencv
#include <cv.h>
#include <highgui.h>
#include <opencv2/opencv.hpp>

#define TRUE  1
#define FALSE 0
#define MAX_STRING_LENGTH   80
#define MAX_FILENAME_LENGTH 80

using namespace std;
using namespace cv;

/* function prototypes go here */

void display_image_from_file(char *filename);
void display_image_from_video(char *filename);
void display_image_from_camera(int camera_number);
void prompt_and_exit(int status);
void prompt_and_continue();
```

```
/*=======================================================*/
/* display images from a video file in an openCV window  */
/* pass the filename of the video as a parameter         */
/*=======================================================*/

void display_image_from_video(char *filename) {

   VideoCapture video;     //  the video device
   Mat frame;              //  an image read from a camera
   Mat processedImage;     //  a processed image
   string inputWindowName  = "Input Image";

   namedWindow(inputWindowName, CV_WINDOW_AUTOSIZE); // create the window

   video.open(filename);                    // open the video input
   if (video.isOpened()){
      printf("Press any key to stop image display\n");

      do {
         video >> frame;                    // read a frame from the video

         if (!frame.empty()) {
            imshow(inputWindowName, frame);  // show our image inside it.
            waitKey(30);                     // this is essential as it allows openCV to handle the display event ...
                                             // the argument is the number of milliseconds to wait
         }
      } while ((!_kbhit()) && (!frame.empty()));

      getchar(); // flush the buffer from the keyboard hit
      destroyWindow(inputWindowName);
   }
   else {
      printf("Failed to open video file\n");
      prompt_and_continue();
      return;
   }
}
```

```
/*================================================*/
/* Display images from a file in an openCV window */
/* pass the filename as a parameter               */
/*================================================*/

void display_image_from_file(char *filename) {

   string inputWindowName     = "Input Image";

   Mat image;
   Mat processedImage;

   namedWindow(inputWindowName, CV_WINDOW_AUTOSIZE);// create the window

   image = imread(filename, CV_LOAD_IMAGE_COLOR);   // Read the file

   if (!image.data) {                               // Check for invalid input
      printf("Error: failed to read image\n");
      prompt_and_exit(-1);
   }

   printf("Press any key to stop image display\n");

   imshow(inputWindowName, image );                 // show our image inside it.

   do{
      waitKey(30);                                  // Must call this to allow openCV to display the images
   } while (!_kbhit());                             // We call it repeatedly to allow the user to move the windows
                                                    // (if we don't the window process hangs when you try to click and drag)

   getchar(); // flush the buffer from the keyboard hit

   destroyWindow(inputWindowName);
}
```

```
/*=================================================*/
/* display images from a camera in an openCV window  */
/* pass the index of the camera as a parameter       */
/*=================================================*/

void display_image_from_camera(int cameraNum) {

   VideoCapture camera;            //  the camera device
   Mat frame;                      //  save an image read from a camera
   vector<int> compressionParams;  // parameters for image write

   char windowName[MAX_STRING_LENGTH];
   char cameraNumber[MAX_STRING_LENGTH];

   strcpy(windowName,"Camera");
   sprintf(cameraNumber, " %d", cameraNum);

   namedWindow(windowName,     CV_WINDOW_AUTOSIZE); // create the window

   if (camera.open(cameraNum) == true) {          // open the camera input

      printf("Press any key to stop image display\n");

      camera >> frame;                            // read a frame from the camera to get the image size ... this is actually C++

      // printf("Camera image is %d x %d\n", frame.cols, frame.rows);

      do {
         camera >> frame;                         // read a frame from the camera
         imshow(windowName, frame);
         cvWaitKey(30); // this is essential as it allows openCV to handle the display event ...
                        // the argument is the number of milliseconds to wait
      } while (!_kbhit());

      getchar(); // flush the buffer from the keyboard hit
```

```
        compressionParams.push_back(CV_IMWRITE_PNG_COMPRESSION);
        compressionParams.push_back(9);                                    // 9 implies maximum compression

        imwrite("../data/camera_image.png", frame, compressionParams);    // write the image to a file just for fun

        destroyWindow(windowName);
    }
    else {
        printf("Failed to open camera %d\n",cameraNum);
        prompt_and_continue();
    }
}
```

Exercises

Run `C:\ACV\lectures\bin\imageAcquisition.exe`

Reading

R. Szeliski, *Computer Vision: Algorithms and Applications*, Springer, 2010.

Section 2.2.3 Optics