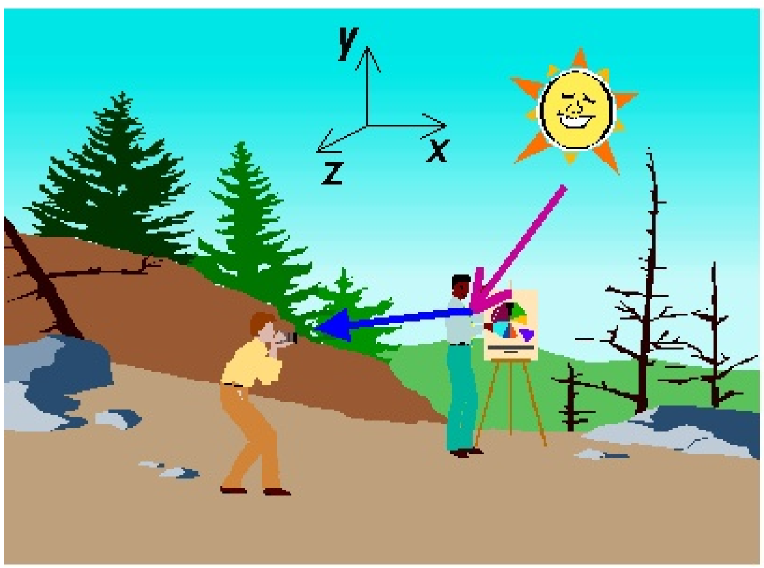
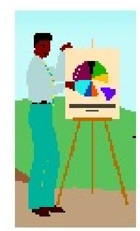
# Image Formation

- For natural images we need a light source *($\lambda$: wavelength of the source)* .
  - $E(x, y, z, \lambda)$: incident light on a point *($x, y, z$ world coordinates of the point)*
- Each point in the scene has a reflectivity function.
  - $r(x, y, z, \lambda)$: reflectivity function
- Light reflects from a point and the reflected light is captured by an imaging device.
  - $c(x, y, z, \lambda) = E(x, y, z, \lambda) \times r(x, y, z, \lambda)$: reflected light.

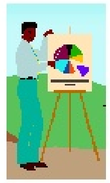



$$E(x, y, z, \lambda)$$

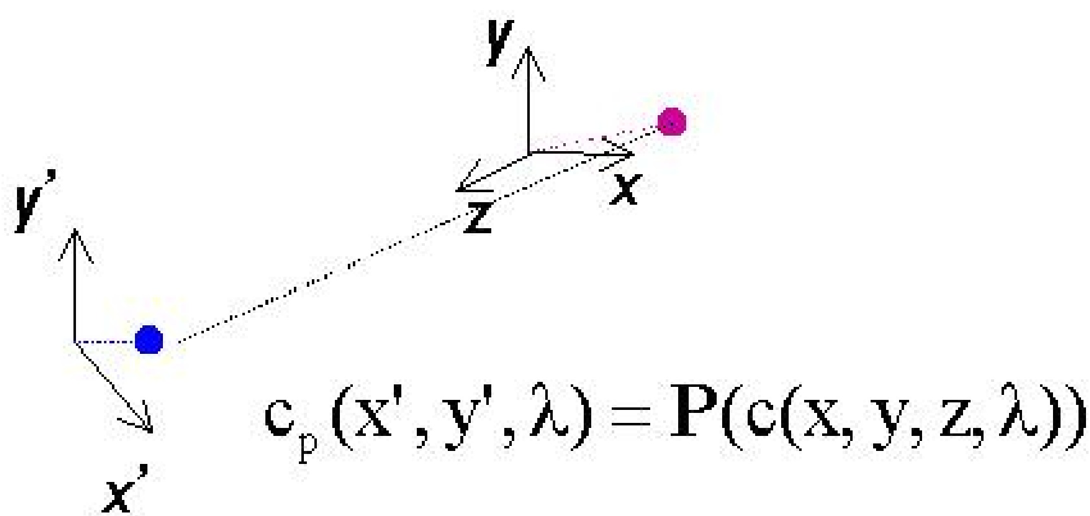
$$c(x, y, z, \lambda) = E(x, y, z, \lambda) \cdot r(x, y, z, \lambda)$$

**Camera**$(c(x, y, z, \lambda))$ **=** 

# Inside the Camera - Projection

**Camera**$(c(x, y, z, \lambda))$ **=** 

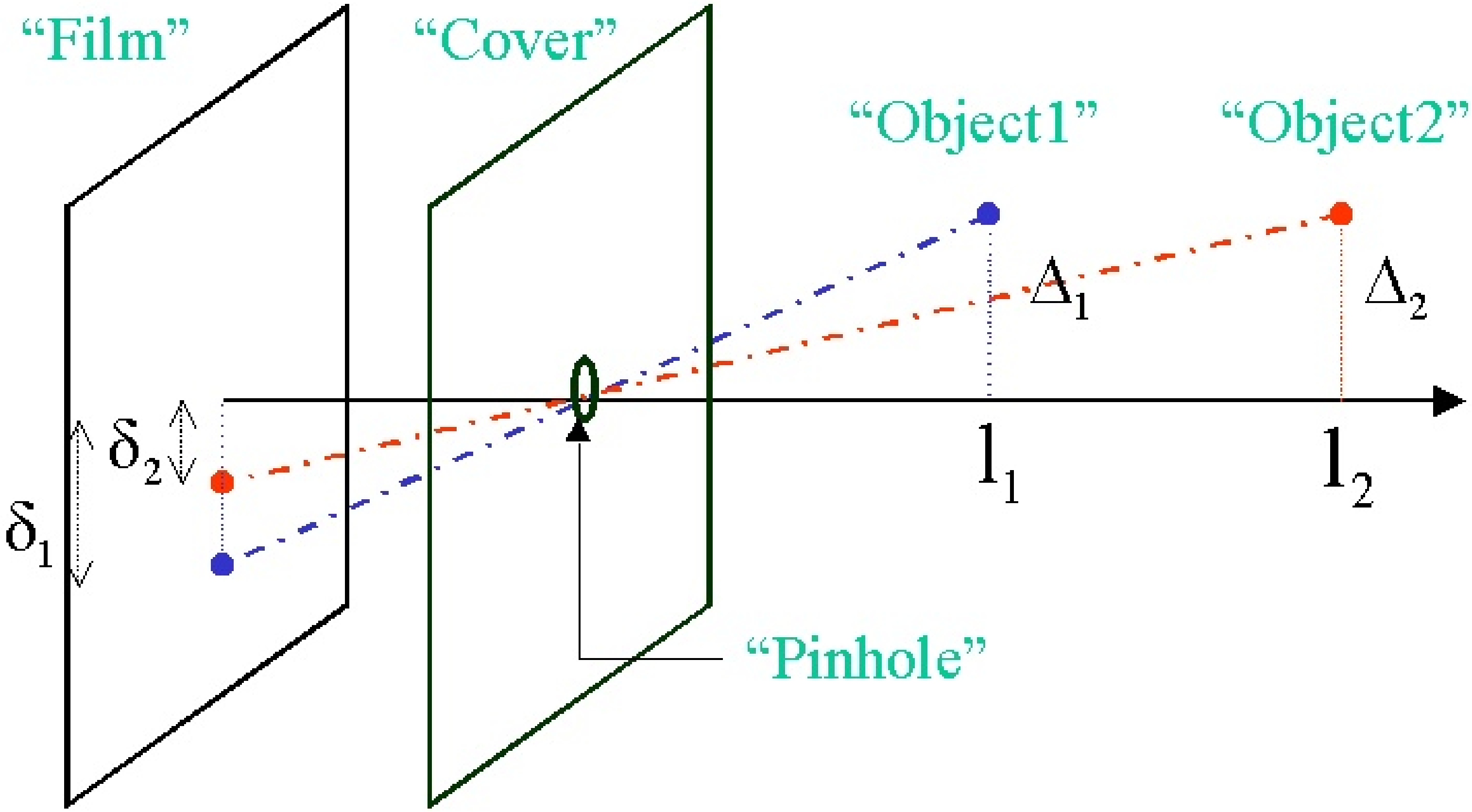
- Projection ($\mathcal{P}$) from world coordinates $(x, y, z)$ to camera or image coordinates $(x', y')$ $\left[c_p(x', y', \lambda) = \mathcal{P}(c(x, y, z, \lambda))\right]$.



# Projections

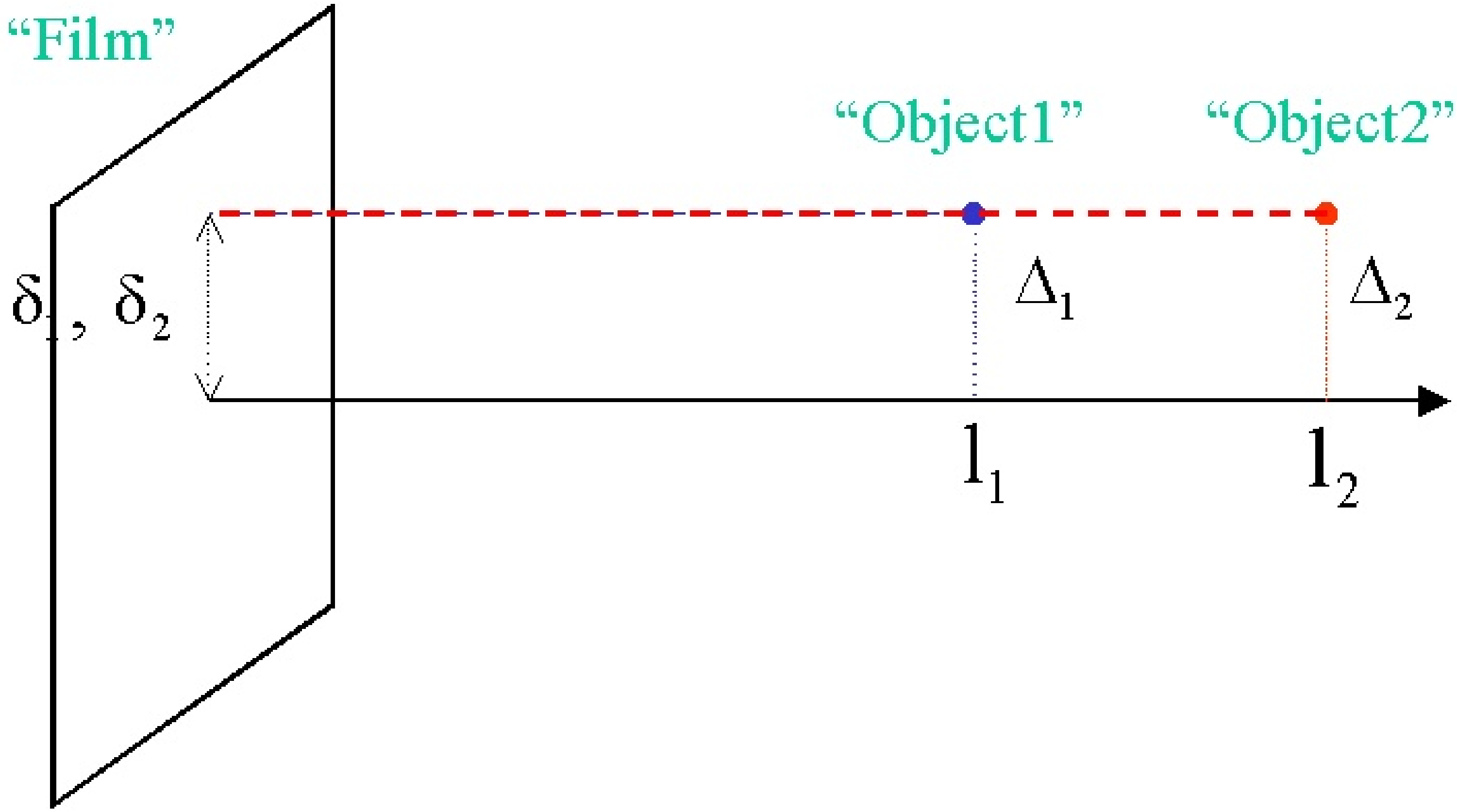
- There are two types of projections ($\mathcal{P}$) of interest to us:
    1. Perspective Projection
        - Objects closer to the capture device appear bigger. Most image formation situations can be considered to be under this category, including images taken by camera and the *human eye*.
    2. Ortographic Projection
        - This is "unnatural". Objects appear the same size regardless of their distance to the "capture device".
- Both types of projections can be represented via mathematical formulas. Ortographic projection is *easier* and is sometimes used as a mathematical convenience. For more details see [1].

# Example - Perspective




- Perspective Projection: $\Delta_1 = \Delta_2$, $l_1 < l_2 \rightarrow \delta_2 < \delta_1$.

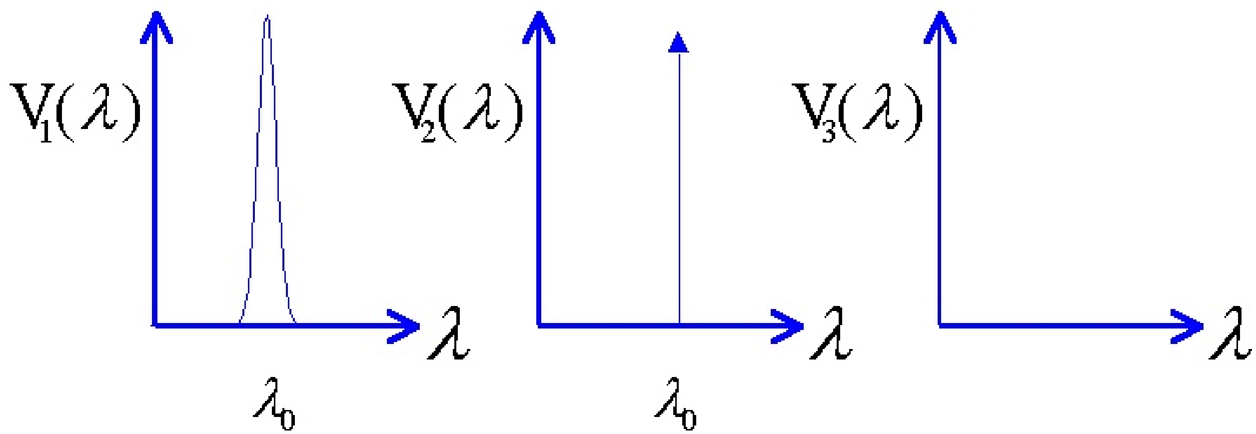
# Example - Ortographic




- Ortographic Projection: $\Delta_1 = \Delta_2$, $l_1 < l_2 \rightarrow \delta_2 = \delta_1$.

# Inside the Camera - Sensitivity

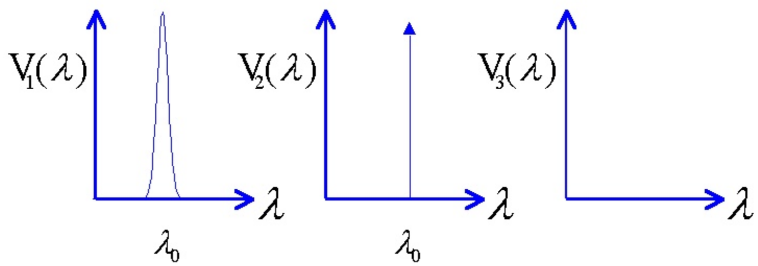
- Once we have $c_p(x', y', \lambda)$ the characteristics of the capture device take over.
- $V(\lambda)$ is the *sensitivity function* of a capture device. Each capture device has such a function which determines how sensitive it is in capturing the range of *wavelengths* $(\lambda)$ present in $c_p(x', y', \lambda)$.




- The result is an "image function" which determines the amount of reflected light that is captured at the camera coordinates $(x', y')$.

$$f(x', y') = \int c_p(x', y', \lambda) V(\lambda) d\lambda \qquad (1)$$

# Example




Let us determine the image functions for the above sensitivity functions imaging the same scene:

1. This is the most realistic of the three. Sensitivity is concentrated in a band around $\lambda_0$.

$$f_1(x', y') = \int c_p(x', y', \lambda) V_1(\lambda) d\lambda$$

2. This is an unrealistic capture device which has sensitivity only to a single wavelength $\lambda_0$ as determined by the delta function. However there are devices that get close to such "selective" behavior.

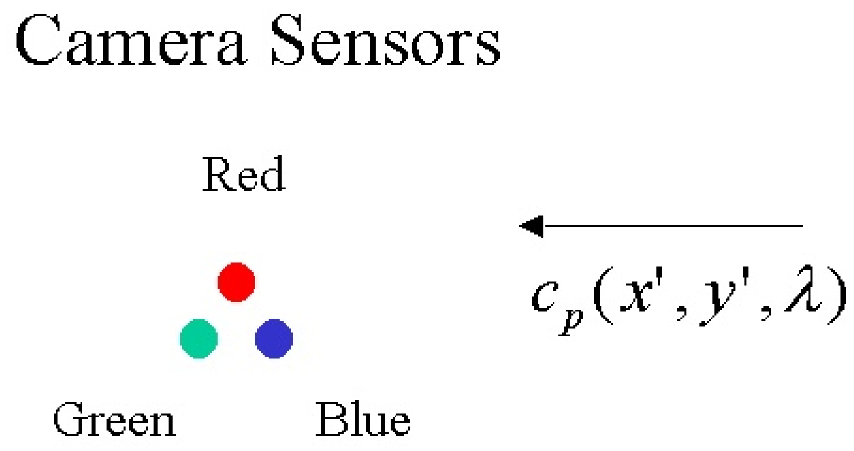
$$\begin{aligned} f_2(x', y') &= \int c_p(x', y', \lambda) V_2(\lambda) d\lambda = \int c_p(x', y', \lambda) \delta(\lambda - \lambda_0) d\lambda \\ &= c_p(x', y', \lambda_0) \end{aligned}$$

3. This is what happens if you take a picture without taking the cap off the lens of your camera.

$$\begin{aligned} f_3(x', y') &= \int c_p(x', y', \lambda) V_3(\lambda) d\lambda = \int c_p(x', y', \lambda)\ 0\ d\lambda \\ &= 0 \end{aligned}$$

## Sensitivity and Color




- For a camera that captures color images, imagine that it has *three sensors* at each $(x', \ y')$ with sensitivity functions tuned to the colors or wavelengths red, green and blue, outputting *three* image functions:

$$
\begin{aligned}
f_{\mathrm{R}}(x', y') &= \int c_p(x', y', \lambda) V_{\mathrm{R}}(\lambda) d\lambda \\
f_{\mathrm{G}}(x', y') &= \int c_p(x', y', \lambda) V_{\mathrm{G}}(\lambda) d\lambda \\
f_{\mathrm{B}}(x', y') &= \int c_p(x', y', \lambda) V_{\mathrm{B}}(\lambda) d\lambda
\end{aligned}
$$

- These three image functions can be used by display devices (such as your monitor or your eye) to show a "color" image.

## Summary

- The image function $f_{\mathrm{C}}(x', y')$ $(C = R,\ G,\ B)$ is formed as:

$$f_{\mathrm{C}}(x', y') = \int c_p(x', y', \lambda) V_{\mathrm{C}}(\lambda) d\lambda \tag{2}$$

- It is the result of:

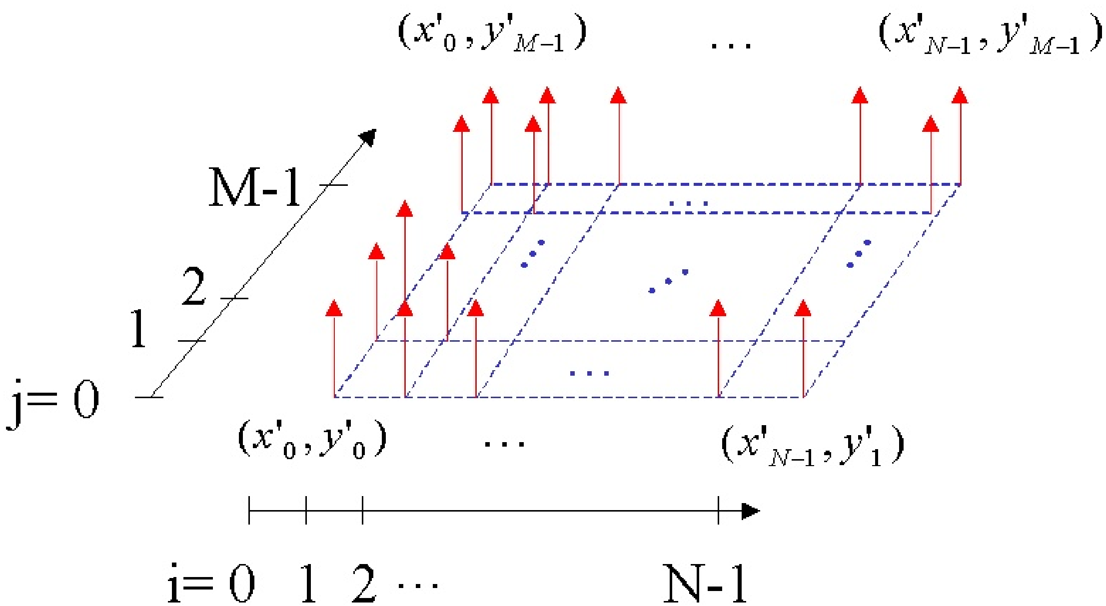
  1. Incident light $E(x, y, z, \lambda)$ at the point $(x, y, z)$ in the scene,
  2. The reflectivity function $r(x, y, z, \lambda)$ of this point,
  3. The formation of the reflected light $c(x, y, z, \lambda) = E(x, y, z, \lambda) \times r(x, y, z, \lambda)$,
  4. The projection of the reflected light $c(x, y, z, \lambda)$ from the *three* dimensional world coordinates to *two* dimensional camera coordinates which forms $c_p(x', y', \lambda)$,
  5. The sensitivity function(s) of the camera $V(\lambda)$.

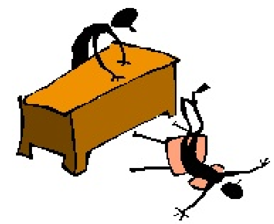
# Digital Image Formation

- The image function $f_{\mathrm{C}}(x', y')$ is still a function of $x' \in [x'_{min}, x'_{max}]$ and $y' \in [y'_{min}, y'_{max}]$ which vary in a continuum given by the respective intervals.
- The values taken by the image function are real numbers which again vary in a continuum or interval $f_{\mathrm{C}}(x', y') \in [f_{min}, f_{max}]$.
- Digital computers cannot process parameters/functions that vary in a continuum.
- We have to *discretize*:
    1. $x', \ y' \Rightarrow x'_i, \ y'_j \quad (i = 0, \ldots, N-1, \ j = 0, \ldots, M-1)$: Sampling
    2. $f_{\mathrm{C}}(x'_i, y'_j) \Rightarrow \hat{f}_{\mathrm{C}}(x'_i, y'_j)$: Quantization.

# Sampling

comb(x',y')

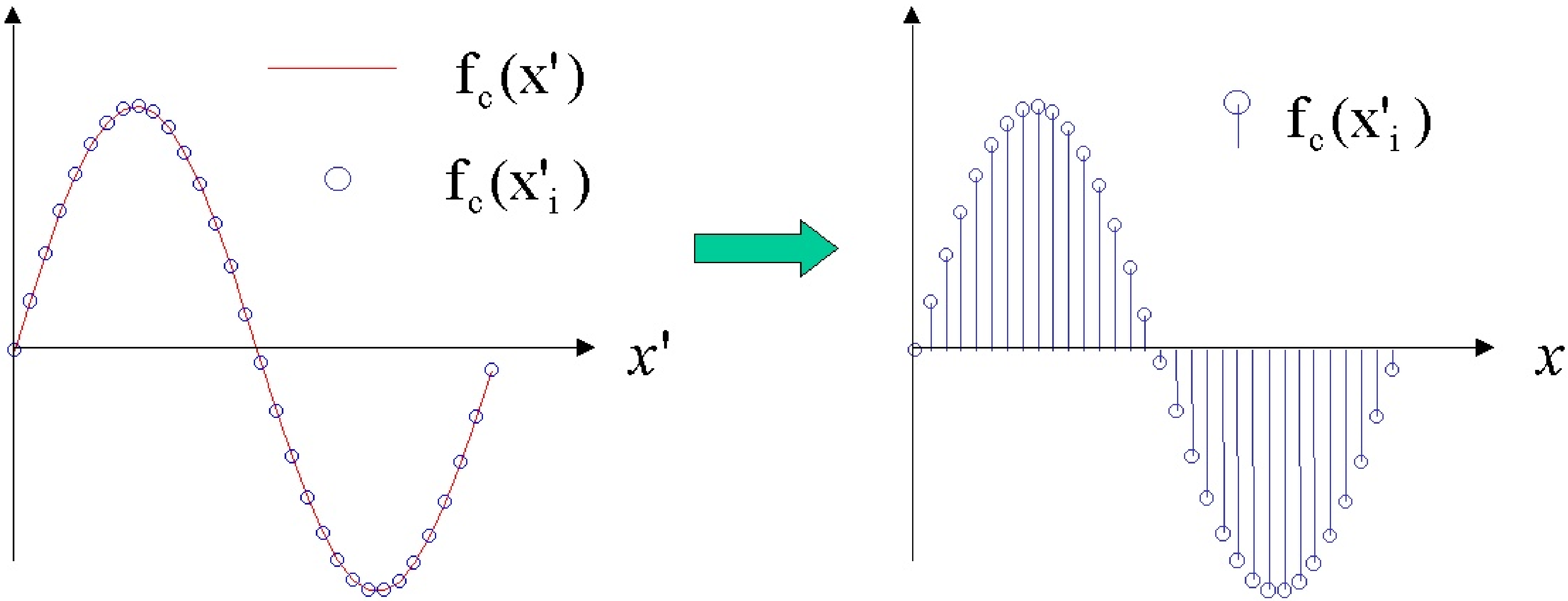


$$comb(x', y') = \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} \delta(x' - i\Delta_x, y' - j\Delta_y) \tag{3}$$

- Obtain sampling by utilizing $f_C(x', y') \times comb(x', y')$.

# Example



# Idealized Sampling

- We will later see that by utilizing $f_{\mathrm{C}}(x', y') \times comb(x', y')$, it is possible to discretize $(x', y')$ and obtain a new "image function" that is defined on the discrete grid $(x'_i, y'_j) \quad (i = 0, \ldots, N-1, \; j = 0, \ldots, M-1)$. ⓘ

- For now assume that we somehow obtain the sampled image function $f_{\mathrm{C}}(x'_i, y'_j)$.
- To denote this discretization refer to $f_{\mathrm{C}}(x'_i, y'_j)$ as $f_{\mathrm{C}}(i, j)$ from now on.

# Quantization

- $f_{\mathrm{C}}(i,j) \quad (i = 0, \ldots, N-1,\ j = 0, \ldots, M-1)$. We have the second step of discretization left.
- $f_{\mathrm{C}}(i,j) \in [f_{min}, f_{max}],\ \forall (i,j)$.
- Discretize the values $f_{\mathrm{C}}(i,j)$ to $P$ *levels* as follows:

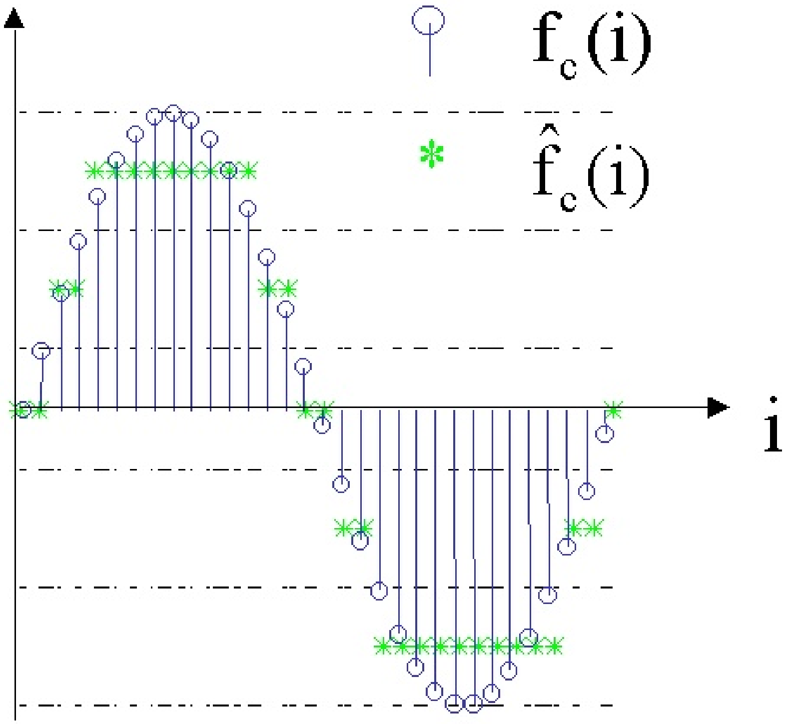
  Let $\Delta_Q = \frac{f_{max} - f_{min}}{P}$.

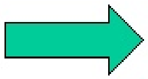
$$\hat{f}_{\mathrm{C}}(i,j) = Q(f_{\mathrm{C}}(i,j)) \tag{4}$$

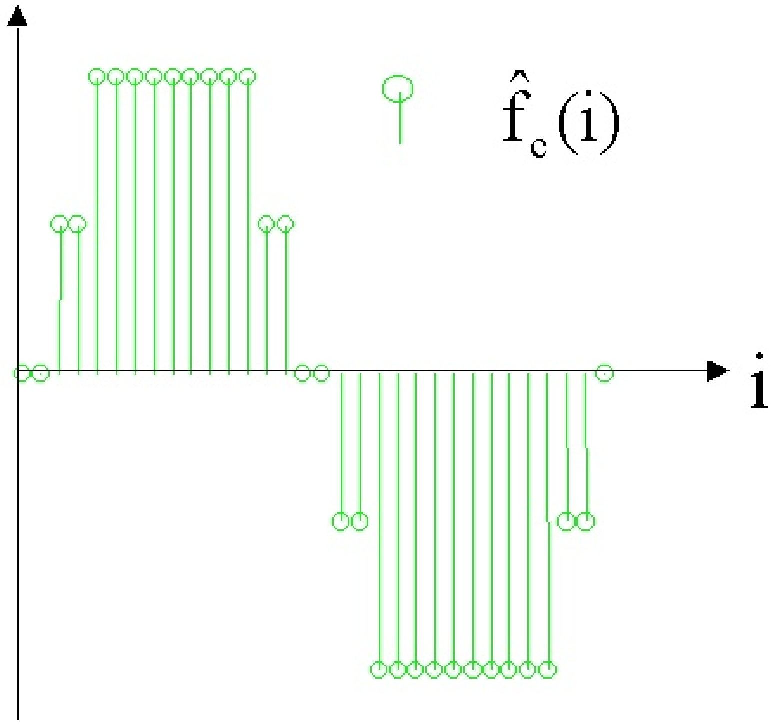
where

$$\begin{aligned} Q(f_{\mathrm{C}}(i,j)) &= (k + 1/2)\Delta_Q + f_{min} \\ & \textit{if and only if } f_{\mathrm{C}}(i,j) \in [f_{min} + k\Delta_Q, f_{min} + (k+1)\Delta_Q) \\ & \textit{if and only if } f_{min} + k\Delta_Q \leq f_{\mathrm{C}}(i,j) < f_{min} + (k+1)\Delta_Q \\ & \text{for } k = 0, \ldots, P-1 \end{aligned}$$

# Example





# Quantization to P levels

- Typically $P = 2^8 = 256$ and we have $log_2(P) = log_2(2^8) = 8$ **bit** quantization.
- We have thus achieved the second step of discretization.
- From now on omit references to $f_{min},\ f_{max}$ and unless otherwise stated assume that the original digital images are quantized to $8$ bits or $256$ levels.
- To denote this refer to $\hat{f}_{\mathrm{C}}(i,j)$ as taking integer values $k$ where $0 \leq k \leq 255$, i.e., let us say that

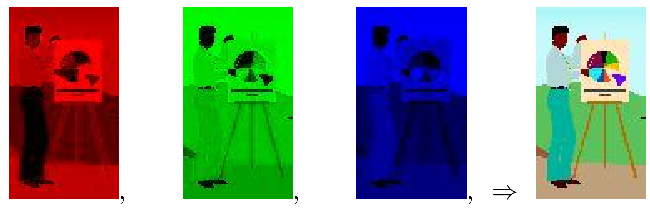
$$\hat{f}_{\mathrm{C}}(i,j) \in \{0, \ldots, 255\} \tag{5}$$

# Summary

- "Let there be light" $\rightarrow$ incident light $\rightarrow$ reflectivity $\rightarrow$ reflected light $\rightarrow$ projection $\rightarrow$ sensitivity $\rightarrow$ $f_{\mathrm{C}}(x', y')$.
- Sampling: $f_{\mathrm{C}}(x', y') \rightarrow f_{\mathrm{C}}(i, j)$.
- Quantization: $f_{\mathrm{C}}(i, j) \rightarrow \hat{f}_{\mathrm{C}}(i, j) \in \{0, \ldots, 255\}$.

# (R,G,B) Parameterization of Full Color Images

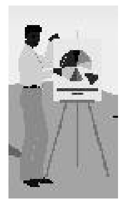
$\hat{f}_{\mathrm{R}}(i,j),\ \hat{f}_{\mathrm{G}}(i,j),\ \hat{f}_{\mathrm{B}}(i,j)\ \rightarrow$ full color image



- $\hat{f}_{\mathrm{R}}(i,j)$, $\hat{f}_{\mathrm{G}}(i,j)$ and $\hat{f}_{\mathrm{B}}(i,j)$ are called the $(R, G, B)$ parameterization of the "color space" of the full color image.
- There are other parameterizations, each with its own advantages and disadvantages (see chapter 3 of the textbook [2]).

# Grayscale Images

"Grayscale" image $\hat{f}_{\text{gray}}(i, j)$



- A grayscale or luminance image can be considered to be $one$ of the components of a different parameterization.
- Advantage: It captures most of the "image information".
- Our emphasis in this class will be on general processing. Hence we will mainly work with grayscale images in order to avoid the various nuances involved with different parameterizations.

# Images as Matrices

- Recalling the image formation operations we have discussed, note that the image $\hat{f}_{\text{gray}}(i,j)$ is an $N \times M$ *matrix* with integer entries in the range $0,\ldots,255$.
- From now on suppress $(\hat{\ })_{\text{gray}}$ and denote an image as a matrix "$\mathbf{A}$" (or $\mathbf{B}$,..., etc.) with elements $A(i,j) \in \{0,\ldots,255\}$ for $i = 0,\ldots,N-1$ , $j = 0,\ldots,M-1$.
- So we will be processing matrices!
- Warning: Some processing we will do will take an image $\mathbf{A}$ with $A(i,j) \in \{0,\ldots,255\}$ into a new matrix $\mathbf{B}$ which may *not* have integer entries!

  In these cases we must suitably *scale* and *round* the elements of $\mathbf{B}$ in order to display it as an image.

# Matrices and Matlab

- The software package Matlab is a very easy to use tool that specializes in matrices.
- We will be utilizing Matlab for all processing, examples, homeworks, etc.
- If you do not have access to Matlab, a copy licensed for use in EL 512 will be provided to you. See the end of this lecture for instructions and details.

# Example - I

- The image of a ramp ($256 \times 256$):

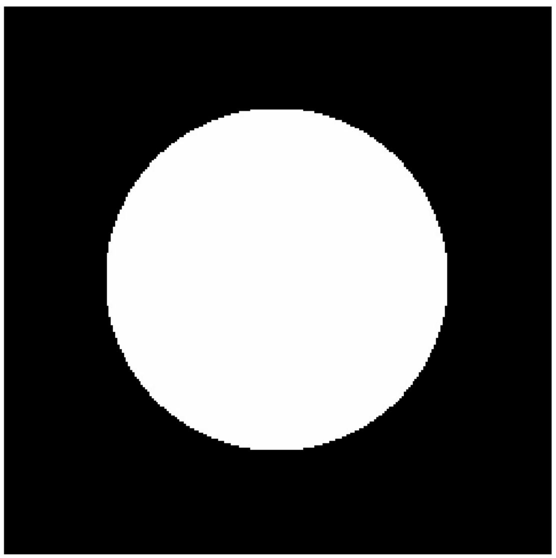
$$\mathbf{A} = \left.\begin{bmatrix} 0 & 1 & 2 & \ldots & 255 \\ 0 & 1 & 2 & \ldots & 255 \\ \vdots & & & & \\ 0 & 1 & 2 & \ldots & 255 \end{bmatrix}\right\} 256 \text{ rows}$$

```
>> for i = 1 : 256
     for j = 1 : 256
       A(i,j) = j - 1;
       end
     end
>> image(A);
>> colormap(gray(256));
>> axis('image');
```

# Example - II

- The image of a circle ($256 \times 256$) of radius $80$ pixels centered at $(128, 128)$:

$$B(i,j) = \begin{cases} 255 & \text{if } \sqrt{(i-128)^2 + (j-128)^2} < 80 \\ 0 & \text{otherwise} \end{cases}$$

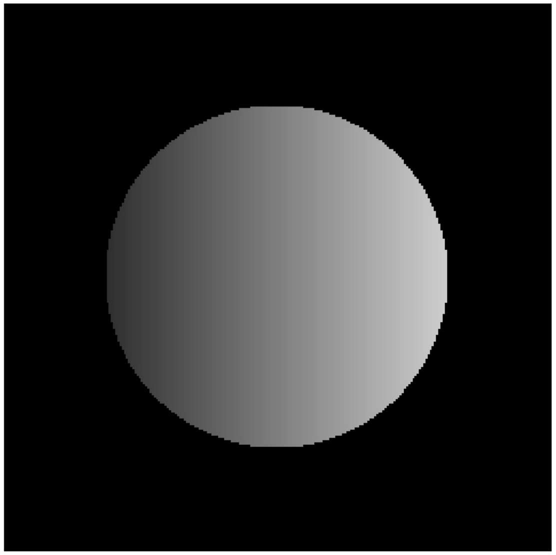


```
>> for i = 1 : 256
     for j = 1 : 256
       dist = ((i - 128)^2 + (j - 128)^2)^(.5);
       if (dist < 80)
         B(i, j) = 255;
       else
         B(i, j) = 0;
         end
       end
     end
>> image(B);
>> colormap(gray(256));
>> axis('image');
```

# Example - III

- The image of a "graded" circle ($256 \times 256$):

$$C(i,j) = A(i,j) \times B(i,j)/255$$



$$
\begin{array}{ll}
>> & \text{for } i = 1:256 \\
 & \quad \text{for } j = 1:256 \\
 & \qquad C(i,j) = A(i,j) * B(i,j)/255; \\
 & \qquad \text{end} \\
 & \quad \text{end} \\
>> & \text{image}(C); \\
>> & \text{colormap(gray(256));} \\
>> & \text{axis('image');}
\end{array}
$$

# Homework I

1. If necessary, get a copy of matlab including a handout that gives an introduction to matlab. You may do so at the Multimedia Lab. ($LC\ 008$) in the Brooklyn campus. The preferred time is Wednesday afternoon.
2. Get a picture of *yourself* taken. Make sure it is 8 bit grayscale and *learn* how to read your picture into matlab as a matrix. Again, you may do so at the Multimedia Lab. ($LC\ 008$) in the Brooklyn campus.
3. Display your image and obtain a printout. Try ">> help print" for instructions within matlab.

If you are at a different campus and have problems with the above instructions please send me email. Please note that this is a one time event, i.e., do your best.

[1] B. K. P. Horn, *Robot Vision.* Cambridge, MA: MIT Press, 1986.

[2] A. K. Jain, *Fundamentals of Digital Image Processing.* Englewood Cliffs, NJ: Prentice Hall, 1989.