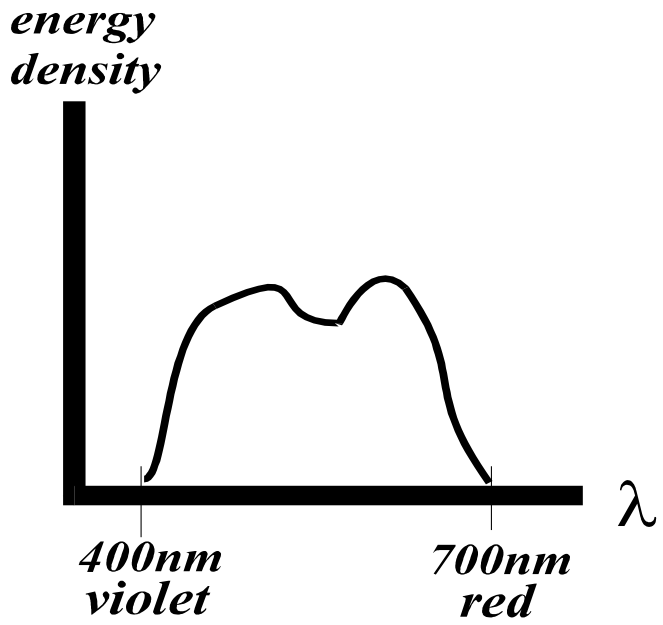
Chapter 3

Media Representations:
Visual Media (Discrete)

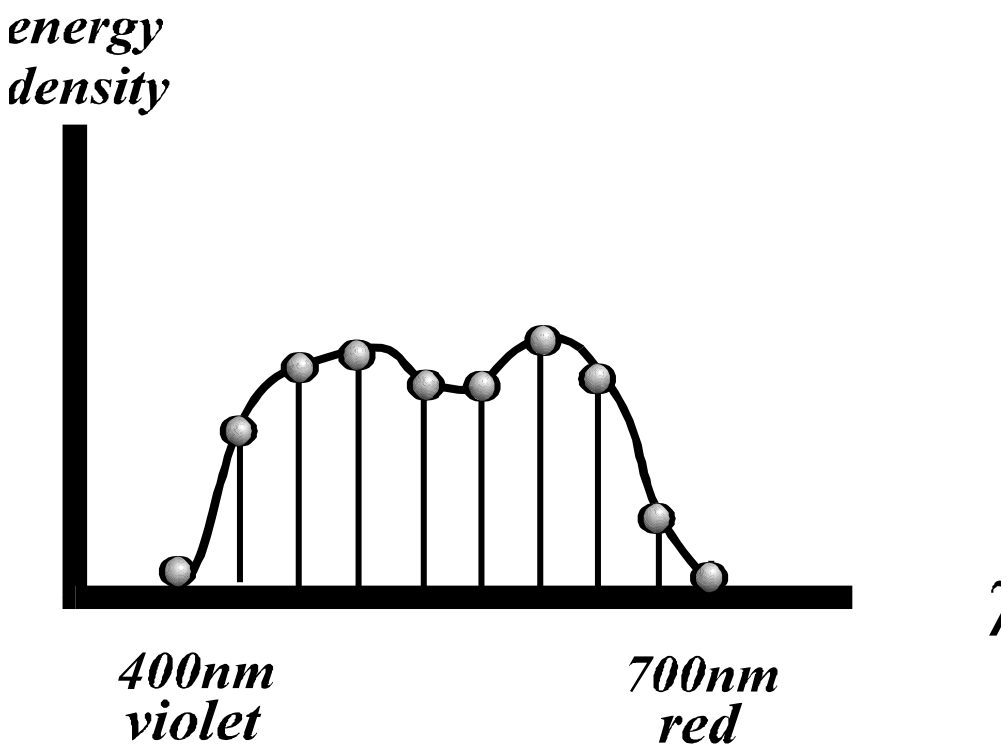
How do we represent light?

- RGB?
- No!
- Light spectrum: EM wave in [400nm, 700nm]



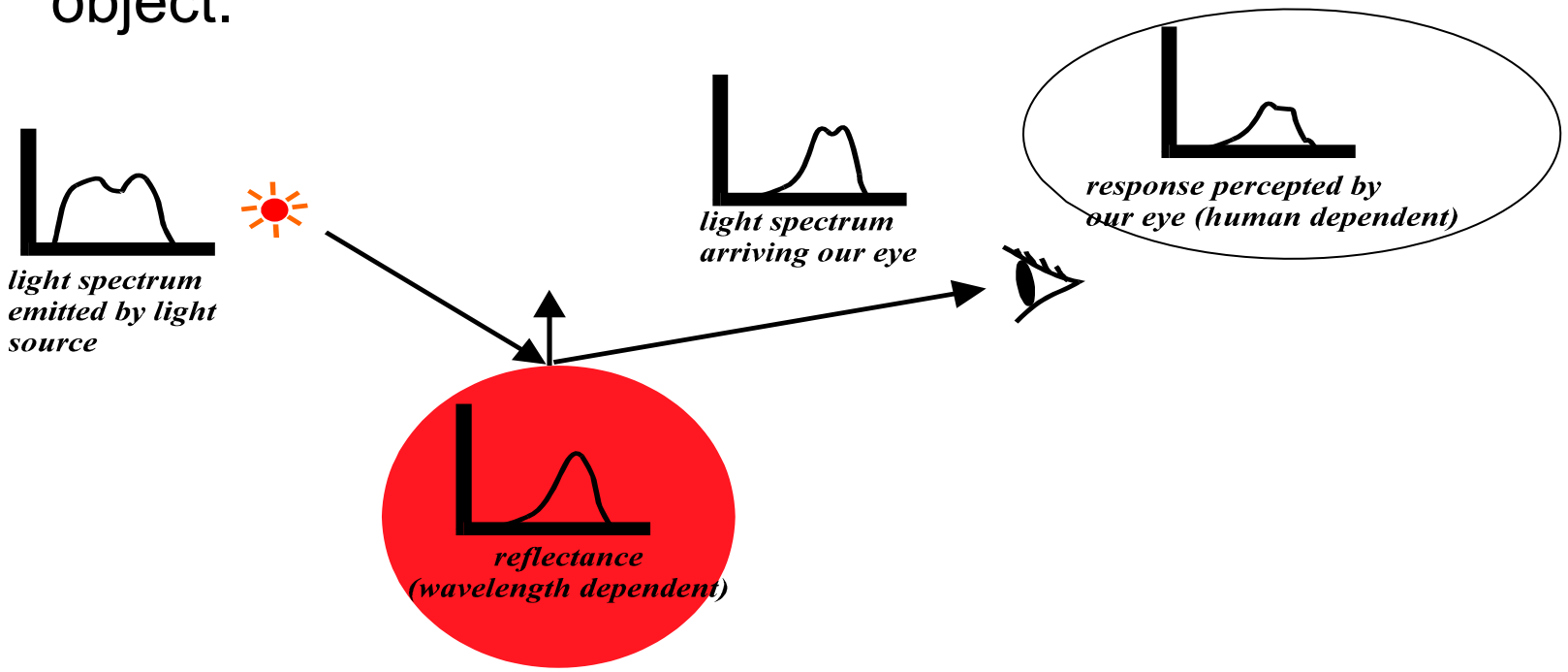
How do we represent light? (2)

- How? Taking samples on the spectrum.
- [Hall89] proposed to take 9 samples on the curves.



How do we represent color?

- The diagram below tells us how can we observe a red object.



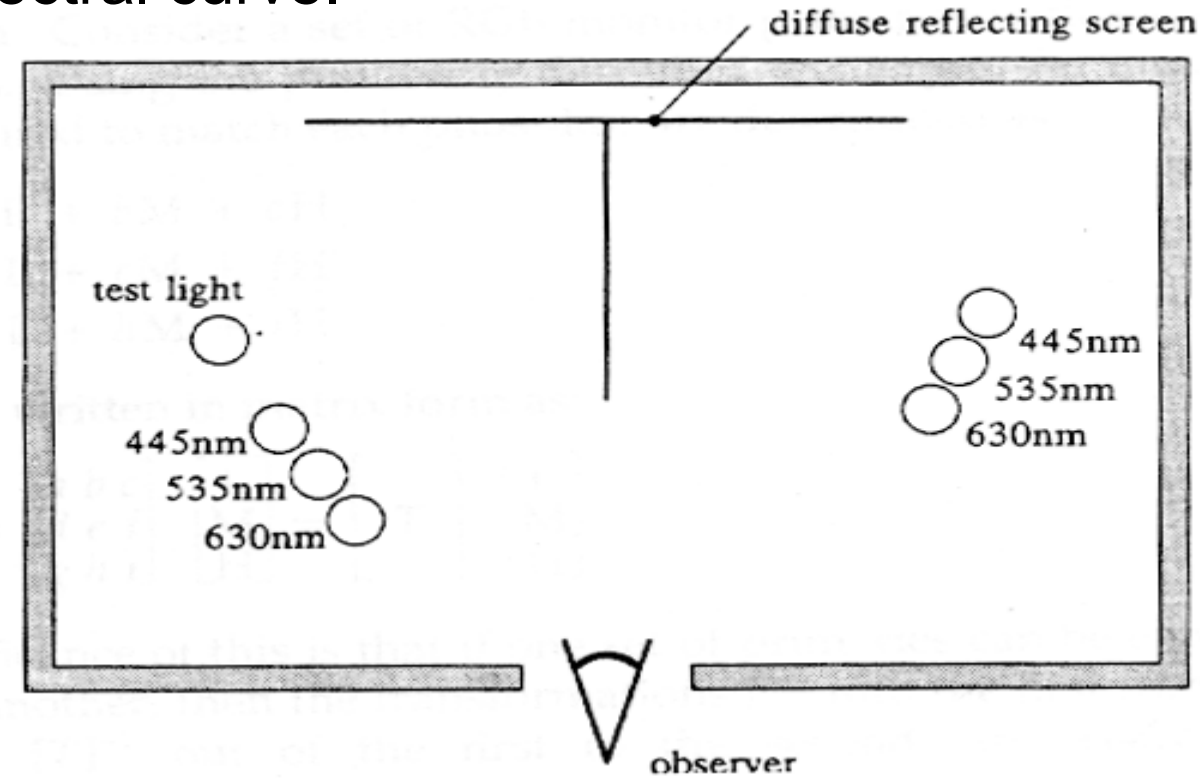
- But “How can we display the final light spectrum on the RGB monitor?”

How do we represent color? (2)

- Different spectrums may produce same response in our eyes.
- Hence no need to reproduce the exact light spectrum.
- But reproduce another spectrum that gives us the "same" perceptual color.
- The color matching experiment tells us which two spectrums produce "same" color in our mind.

Color Matching Experiment

- Three types of color receptors (cones) on our retina: responsible for short, middle and long wavelengths
- A matching experiment of Sensations, not matching of spectral curve.

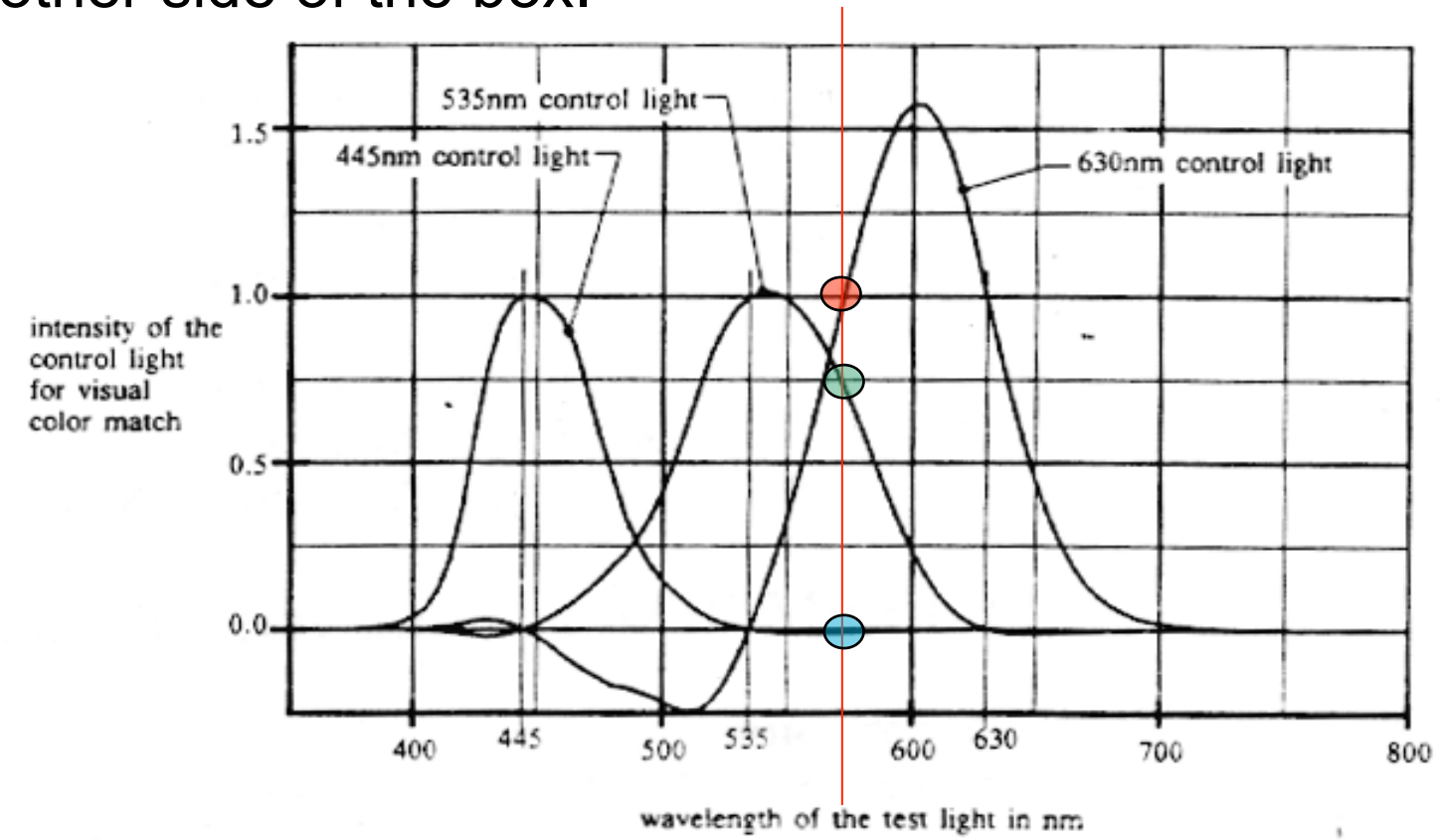


Color Matching Experiment (2)

- A statistical, psychological experiment. May vary for different individual.
- 3 lights are chosen
  - L: 445nm
  - M: 535nm
  - H: 630nm
- Not necessary equal to RGB on your monitor

Color Matching Experiment (3)

- e.g. wavelength 570nm has the same response as 0 L + 0.7 M + 1.0 H
- Negative value means “move the primitive light to the other side of the box.

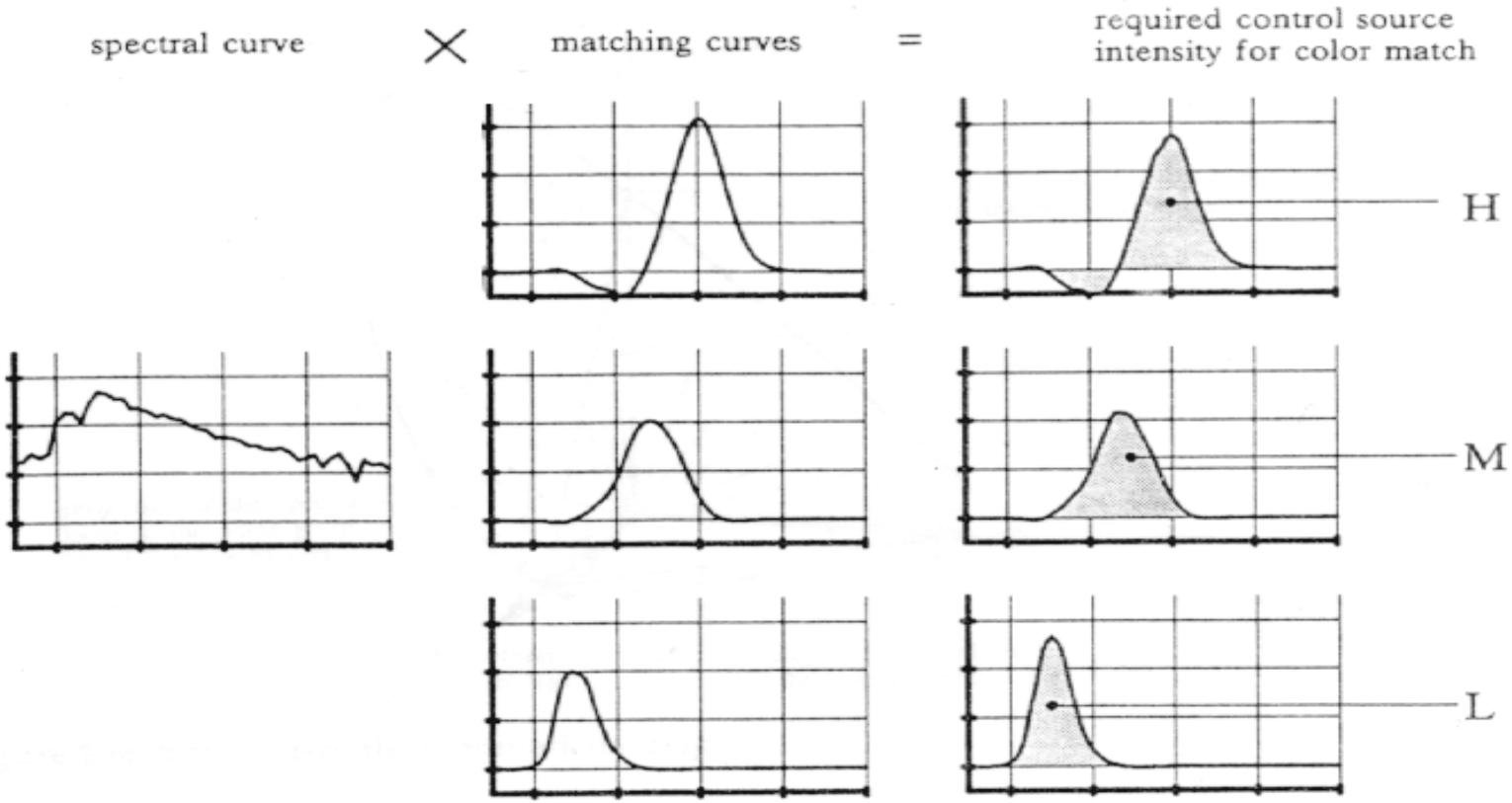


Color Matching Experiment (4)

- Because of the negative values, CIE (Commission Internationale d'Eclairage) defines three hypothetical primary light sources.
- Their matching curves are always positive.

From Light Spectrum to RGB

- Convert the light spectrum to XYZ



From Light Spectrum to RGB (2)

- From XYZ to RGB: spectral curve of RGB primitives can also be expressed as,

  $R = a\,X + b\,Y + c\,Z$

  $G = d\,X + e\,Y + f\,Z$

  $B = g\,X + h\,Y + i\,Z$

- In other words,

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} \quad & T & \quad \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

From Light Spectrum to RGB (2)

- Two monitors may not display the same image equally.
- The reason of the popularity of RGB color model is due to the wide availability of CRT
- CRT has 3 phosphors (RGB) which produce a combination of light when excited by electrons.
- The color space of RGB is smaller than that of CIE XYZ

Reference

- [Hall89] Roy Hall, *Illumination and Color in Computer Generated Imagery*, Springer-Verlag 1989.

Color Models

- RGB model is one of the color models.
- There exists other color models, different color models are used in different domain with different purpose.

- RGB is mostly used in computer graphics/image processing
- YIQ and YUV are commonly used in TV broadcasting
- CMY model is used in printing industry
- HSV (Hue, Saturation & Value) and HLS (Hue, Lightness & Saturation) are used by artists to retouch the images (e.g. Photoshop)

YIQ Color Model

- YIQ is used in TV broadcasting
- Y is the luminance (grayness or lightness) component same as Y primitive in CIE
- I and Q are the chrominance components (color components)
- Roughly speaking, I is red-orange color, and Q is its complement.
- It can be easily transformed from RGB by

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.275 & -0.321 \\ 0.212 & -0.528 & 0.311 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

YIQ Color Model (2)

- Why YIQ for TV broadcasting?
  - By ignoring the I and Q channels and only handling Y (luminance component), color TV is easily converted to B/W TV. In other words, backward compatible.
  - Our eye is more sensitive to luminance than chrominance components. By separating luminance from chrominance and reducing the bandwidth of chrominance channels, we can reduce the bandwidth without much noticeable artifact.
- NTSC adopts YIQ.
- The ratio of bandwidth allocated for Y:I:Q is 4:2:2

YIQ Color Model (3)

- Visual comparison: 5 bits per pixel (both images)

RGB coded

YIQ coded

- Note the red and green “clouds” in the left image

YUV Color Model

- Similar to YIQ.

$$Y = 0.299R + 0.587G + 0.114B$$

$$U = (B - Y) \times 0.493$$

$$V = (R - Y) \times 0.877$$

- This color model is adopted in CD-I (CD-Interactive) and DVI video.
- Just like YIQ, the bandwidth of chrominance components are reduced.

CYM and CYMK Color Models

- Cyan, Magenta and Yellow (CMY) are the complementary colors of RGB. They are subtractive primaries, i.e.

  $C = 1.0 - R$

  $M = 1.0 - G$

  $Y = 1.0 - B$

- Used in printing industry because color pigments absorbs color light (subtractive) instead of emitting light
- Sometimes an extra channel is added, the black K. So the CYMK model is modified to

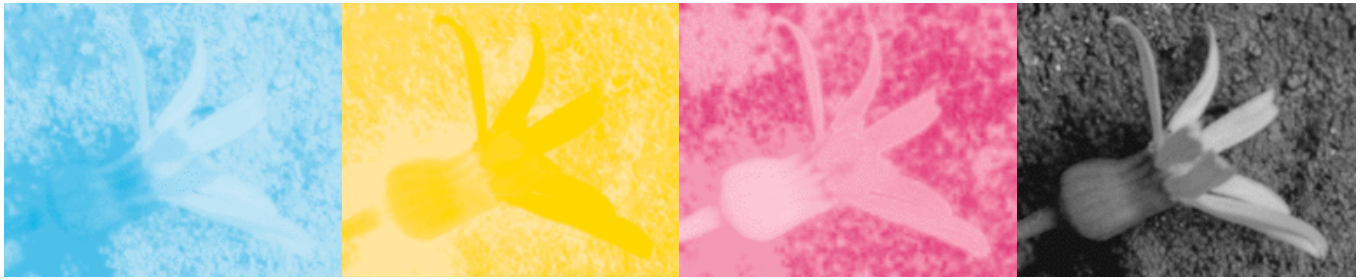
  $K = \min(C, M, Y)$

  $C = C - K$

  $M = M - K$

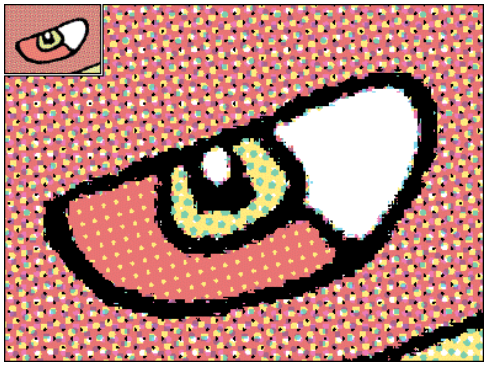
  $Y = Y - K$

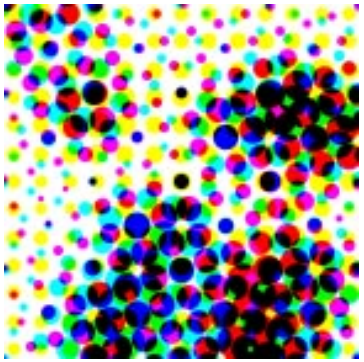
CYM and CYMK Color Models (2)



Cyan Yellow Magenta Black

Apply (print) order:
from light to dark
Y, C, M, B





Close up

Digital Image

- An image is a 2D continuous function of light intensity values.
- In computer, we can only store the discrete version of this function.
- Just like audio, the 2D function is sampled at discrete intervals yielding a 2D matrix of discrete values.
- A sample in the digital image is called *pixel*.

Continuous 2D function

Digital 2D image

Digital Image (2)

- Just like digital audio, there are 3 steps: sampling, quantization and coding.
- Image resolution actually specifies the sampling rate. e.g. 320 x 200, 640 x 480, 1024 x 768,
- Again, we cannot represent any color intensity values, only discrete color values can be represented.
- The no of quantization levels is referred as color depth.
- Naively quantizing the color intensity will give poor image due to the huge color space. Tricky for color image.
- A clever coding scheme (color index and color lookup table) is used due to the lack of memory.

Grayscale Image

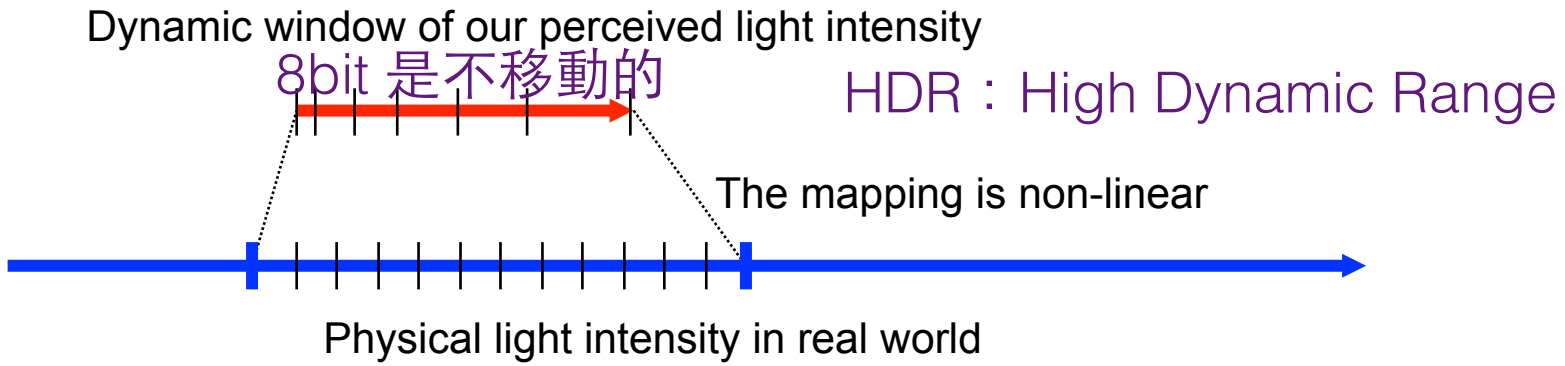
- For B/W image, a 256-level quantization is usually enough. Pixel values range from 0 to 255.
- Therefore 1 byte (8 bits) is needed to code the value of a pixel.
- For example, (try to find quantization artifact on the right)
- In fact, the Y component is actually a grayscale image

24-bit Color Image

- In previous slide, the values in 1 channel (B/W image) can be finely represented by 256-level quantization.
- Therefore, a color image requires three channels, each with 256-level quantization, altogether $(256)^3$ quantization levels. Or in other words, 3 bytes are needed for each pixel.
- Obviously, this is a huge storage requirement, especially for computer before 90's.
- So, people come up with a clever solution, color indexing.

Is 8-bit Really Enough?

- When we say 8-bit per channel is usually sufficient. We actually mean “sufficient for representing perceived light intensity”.
- Is 8-bit really sufficient for representing the physical light intensity?
- No. Our vision is only a **small moving window** in the long light intensity range in real world.



Is 8-bit Really Enough? (2)

- Have you even experienced the problem of over-exposure and under-exposure in your photographs?

Captured image

Obviously the object inside the chamber also reflects certain level of light intensity. It is hard to observe/acquire when we stand outside

High dynamic range image

- We will see the object if we are inside that chamber
- Our eyes (automatically) and cameras (manually) adjust exposure during capture
- We acquire a **relative** light intensity and the perceived light intensity is **not linear** to physical light intensity

High Dynamic Range Image

- For serious applications, we need to acquire quantity closer to physical light intensity
- This motivates the development of high dynamic range image. That is, 16-bit per color channel
- 16-bit per channel digital cameras are now available in the market but usually expensive
- But our monitors are still 8-bit per channel. Therefore high dynamic range images are still not popular
- CAVE demo

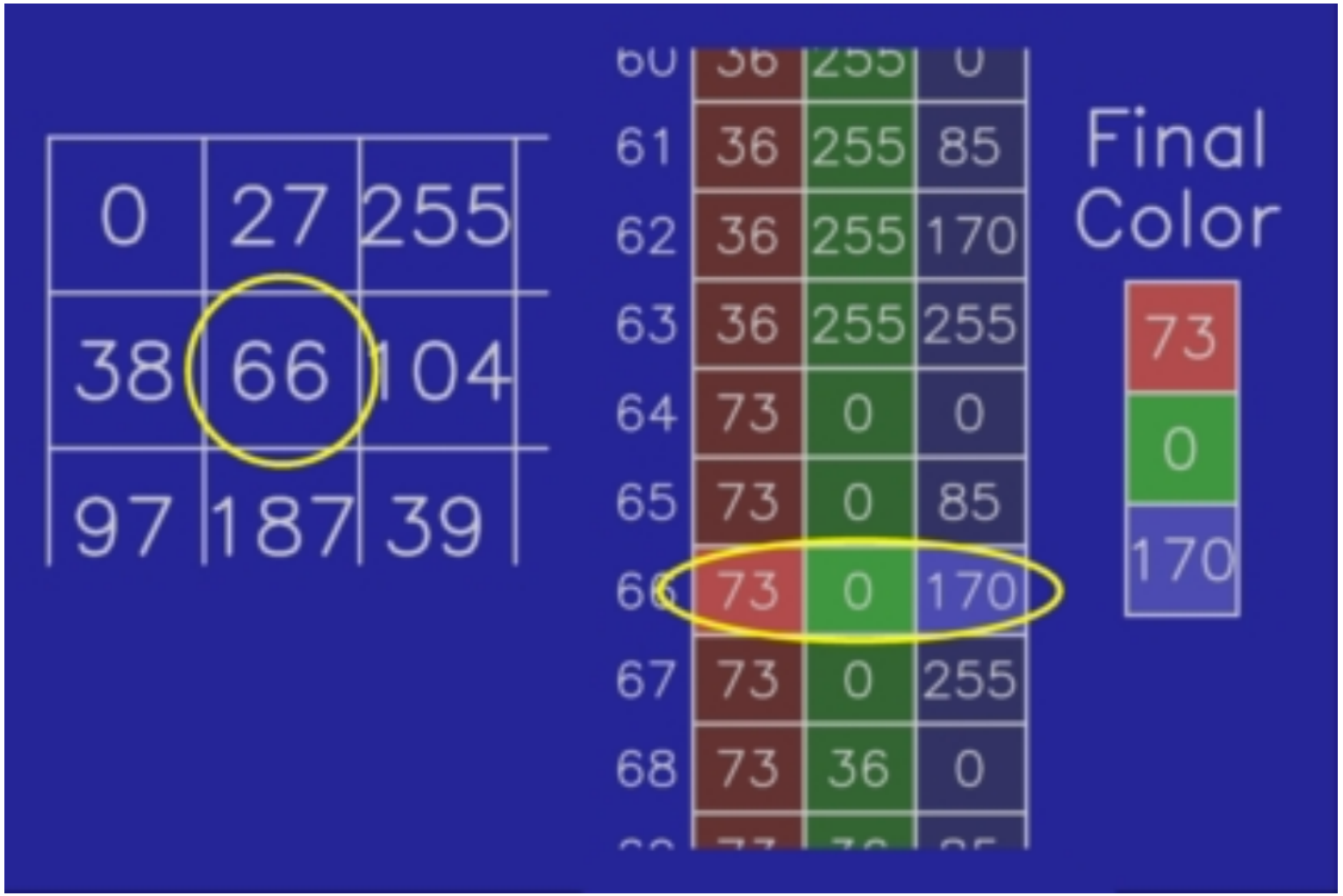
  http://www.cs.columbia.edu/CAVE/tomoo/RRHomePage/rrgallery.html

Color Lookup Table

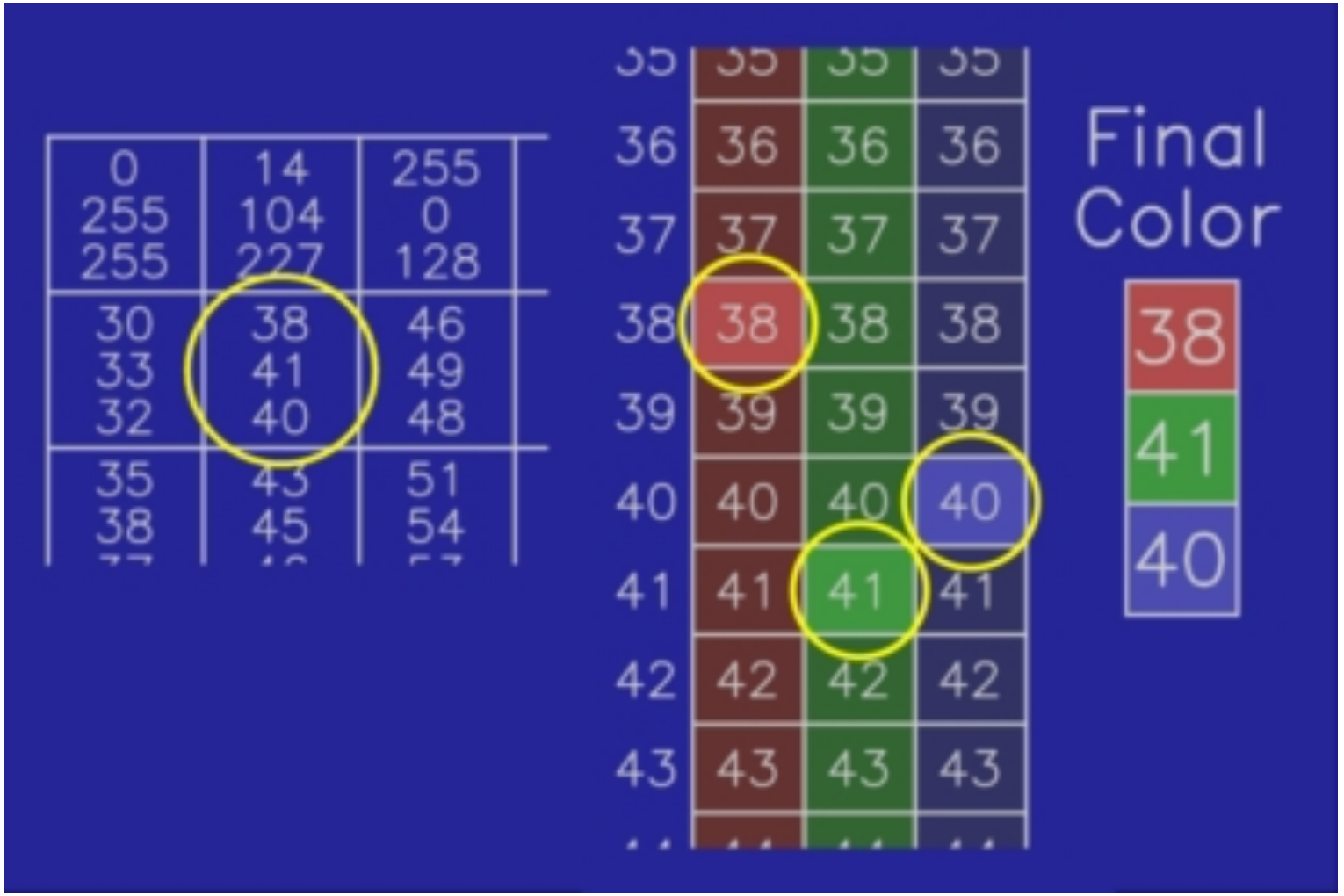
- Instead of directly storing the pixel color in the frame buffer, the actual colors are stored in a color lookup table.
- The frame buffer only stores the color index (indexing to the color table).
- Therefore the quantization is divided into two steps, one in the color table (24 bits) and one in the frame buffer (can be 1, 2, 4, 8, 16 or 24 bits)

先HDR再用 tone mapping 變成 8bit

8-bit Color Lookup



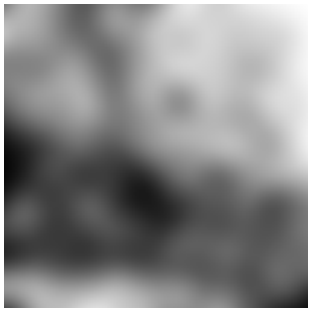
True Color


0
255
255
14
104
227
255
0
128
30
33
32
38
41
40
46
49
48
35
38
43
45
51
54
36
37
38
39
40
41
42
43
Final Color
38
41
40

Spatial Integration

- Sometimes, the quantization is too bad and there is no more memory. What can we do?
- Visual media offer an extra method to increase the *perceptual* color depth besides quantization, the spatial integration.
- When a small region is viewed from a sufficiently large distance, our eyes can only perceive the overall intensity of that region.
- To make use of this deficiency of our eye, dithering is performed on the image.

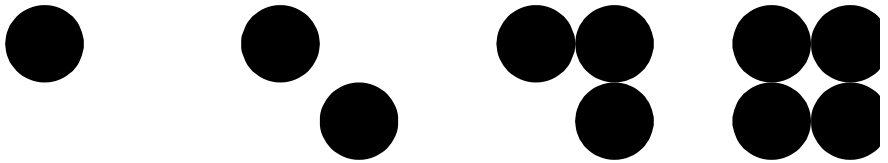
Actual details



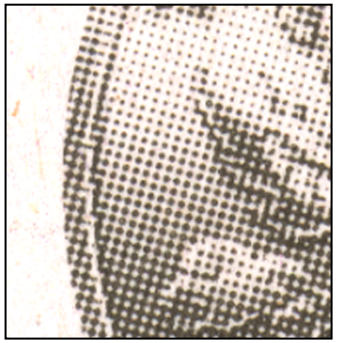
What we perceive

Digital Halftoning

- To illustrate the idea, we study an extreme situation: Given a grayscale image, how can I display it on a device with only 2 colors (■ and □).
- For example, a 2x2 region can produce 5 different intensity levels if viewed from far distance:



- It is actually the real problem in printing.
- Below is the blowup of a small piece of newspaper.



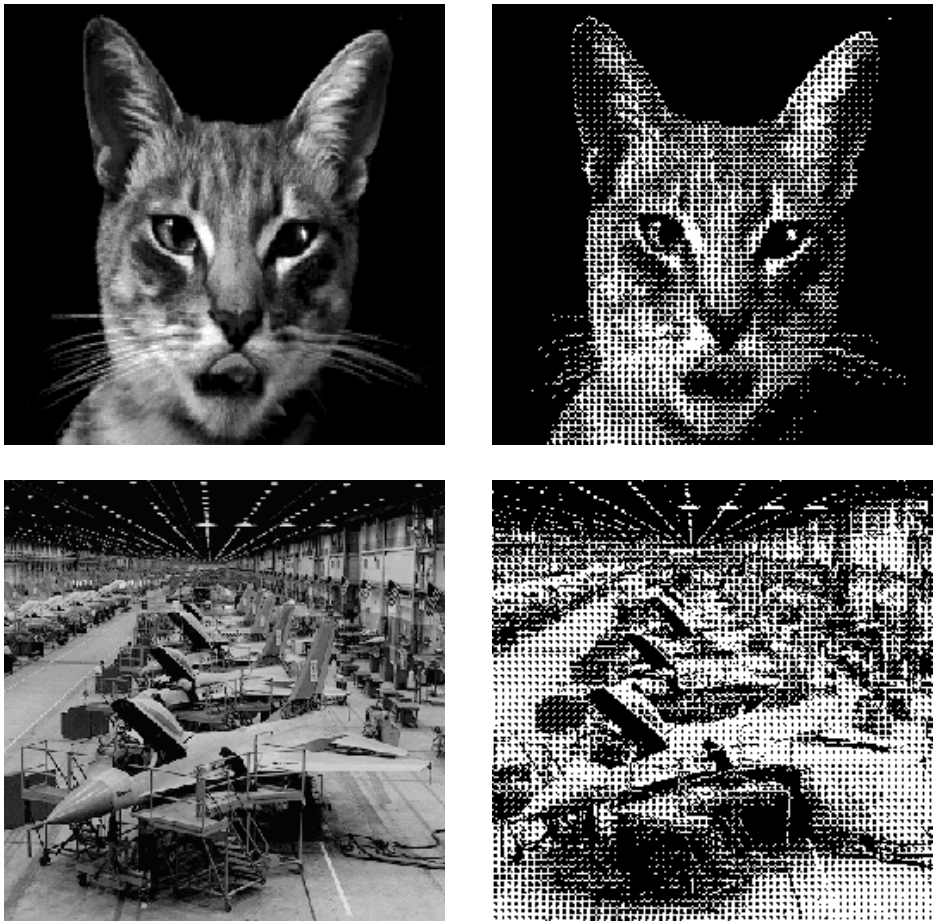
Ordered Dither

- Usually used in printing (e.g. laser printer, newspaper)
- Fast, easy and parallel
- The image is tiled with a threshold matrix (see below), the pixel turns black iff the pixel value underlined exceeds the corresponding threshold value.

$$D = \begin{bmatrix} 1/32 & 17/32 & 5/32 & 21/32 \\ 25/32 & 9/32 & 29/32 & 13/32 \\ 7/32 & 23/32 & 3/32 & 19/32 \\ 31/32 & 15/32 & 27/32 & 11/32 \end{bmatrix}$$

Ordered Dither (2)

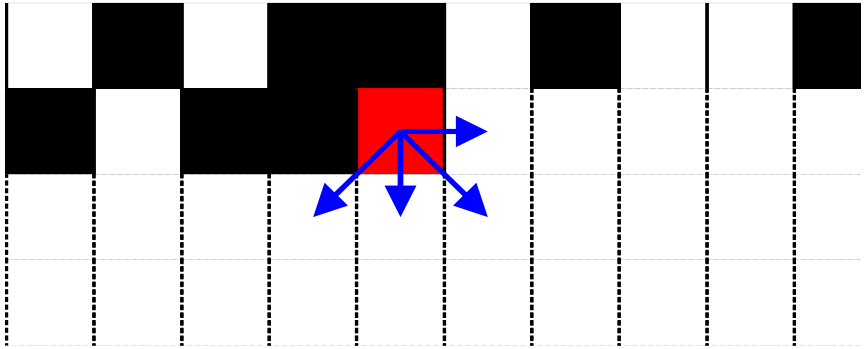
- There is noticeable pattern due to the threshold matrix.
- The artifact will be unnoticeable if the dots are very tiny.
- Both input and output image are same in resolution.



Original grayscale Ordered-dithered image

Error Diffusion

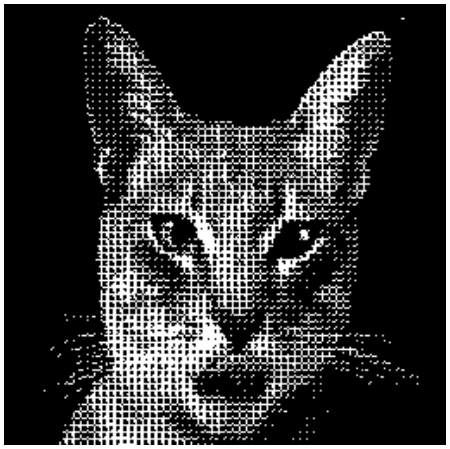
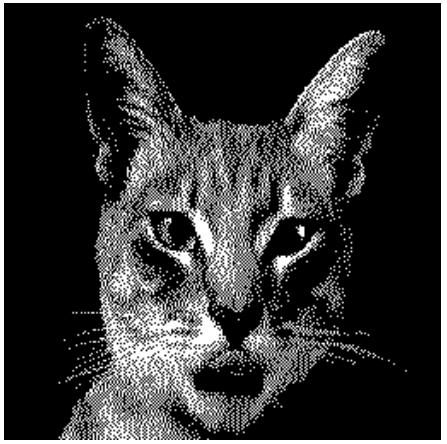
- A substantial improvement can be achieved when the image is dithered with Floyd-Steinberg algorithm.

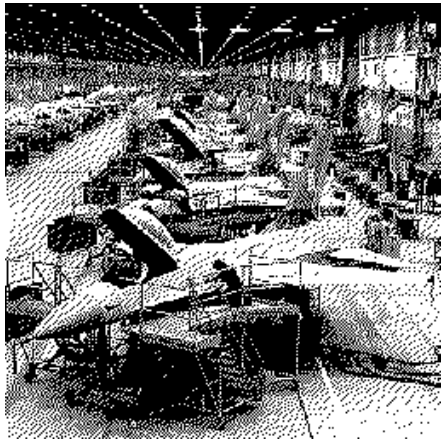


- The quantization error can be "diffused" to 4 neighboring pixels.
- Algorithm:

```
for y = 1 to height
    for x = 1 to width
        if p[x,y] < 0.5
            image[x,y] = 0
        else
            image[x,y] = 1
        err = p[x,y] - image[x,y]
        p[x+1,y]   += err * 7/16
        p[x-1,y+1] += err * 3/16
        p[x,y+1]   += err * 5/16
        p[x+1,y+1] += err * 1/16
```

Error Diffusion (2)

Original grayscale

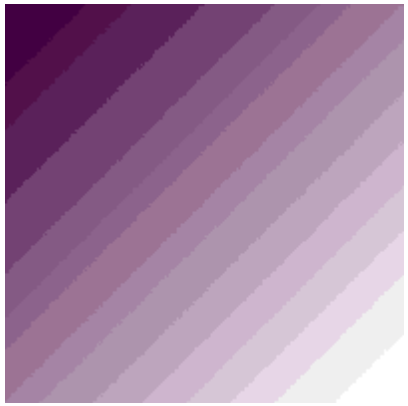
Ordered dither

Error Diffusion

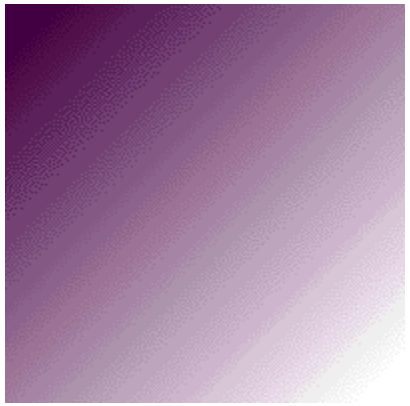
Dithering

- Error diffusion produces darker hardcopy than ordered dither due to ink smudging.
- Ordered dither is frequently used in printing
- Error diffusion is usually used in screen display.
- Dithering not just applied to B/W images, but also color images. It reduces the contouring problem
- e.g.

24-bit image



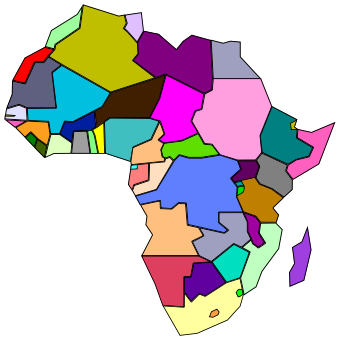
4-bit image
(contour appears)

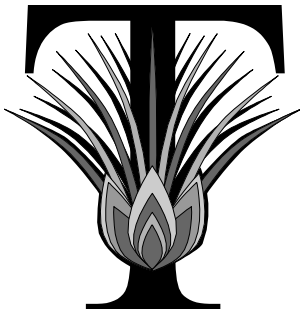


4-bit with dithering

Image and Graphics

- As you see, storing image is sometimes expensive.
- Why don't we store geometric primitives and render them in real time during display?
- Image and graphics is analogous to digital audio and music (MIDI).
- Two components: geometric primitives and synthesis process.
- Geometric primitives can be 2D or 3D.
- For 2D, the synthesis is simple and usually referred as rasterization.





2D Graphics

- e.g. to represent a circle, you only have to store the center, radius, colors of interior and outline.
- 2D graphics are scalable (can be in any scale) while digital image (samples) is not good for scaling.
- One example is the font:
  - Last generation OS uses bitmap fonts.
    Aliasing (staircase) appears when scaled up
  - Modern OS uses scalable font such as Truetype fonts and Postscript fonts.
    The outline is represented as Bezier curves. (read your graphics textbook for details on Bezier curve)

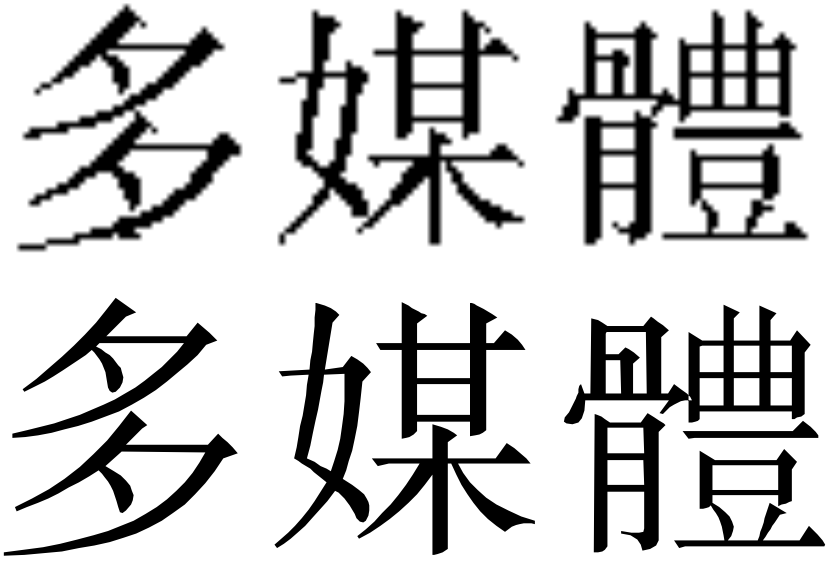


Image vs. Graphics

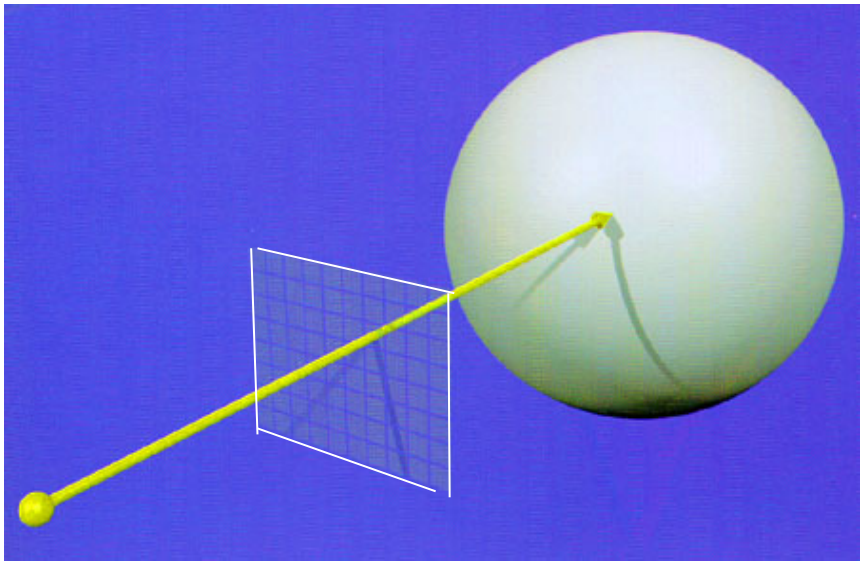
- Let's compare digital images and graphics.

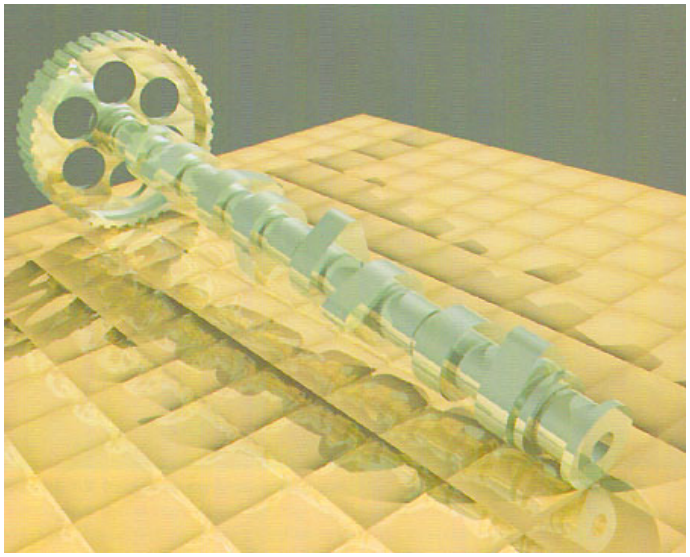
- Image is actually a 2D array of color samples.
- Once recorded, cannot be changed.
- Usually large storage needed
- Aliasing may happen if sampling rate is not enough
- Display is straightforward.

- Graphics consists of geometric primitives and rendering process.
- Can be easily modified.
- Usually more compact, but not always.
- No aliasing as the analytic formulae are stored and rasterization is done when display.
- Display requires rendering which may be computational intensive, e.g. 3D graphics

3D Graphics

- One nice thing of graphics is that it can be used to express photorealistic scenes of virtual world.
- 3D graphics consists of 3D geometric primitives and rendering process.
- The rendering process simulates how light propagates in the 3D space. (Obviously, extremely expensive)





3D Graphics (2)

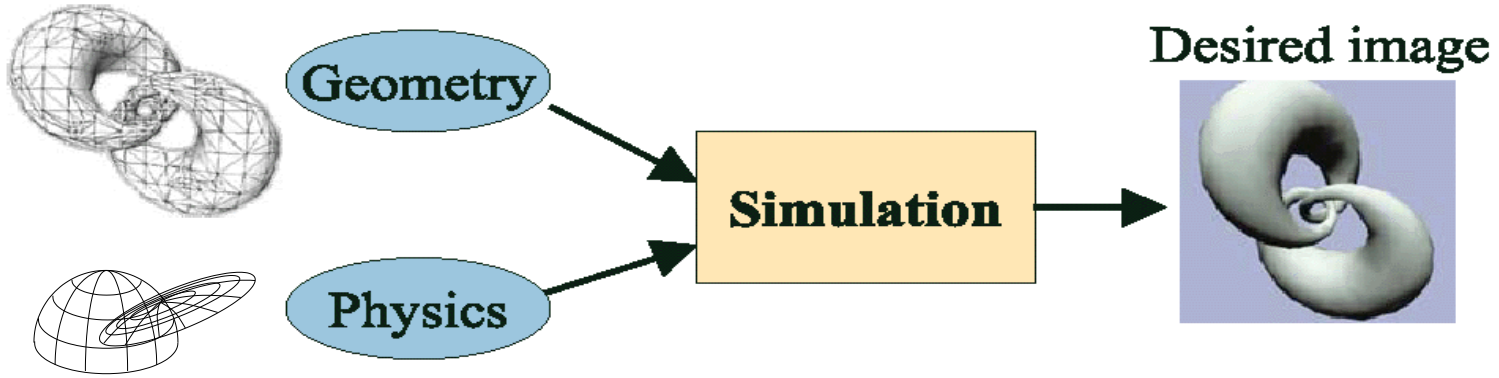
- 3D graphics is more interactive than 2D digital images as you can manipulate.
- Some examples: OpenGL, Direct3D, MacDraw 3D, VRML, MetaStream.

- 3D Graphics is a big topic, will not be covered in this course.
- I recommend you to take a graphics course if you have not yet done so.

Image-based Modeling & Rendering

Geometry-based Rendering: (traditional 3D graphicss)

- Traditionally, we model the scene with geometry and physical laws.
- Images are generated by simulating the propagation of light in space.



- As the computer capacity and power increase, more realistic image of more complex scene can be synthesized.

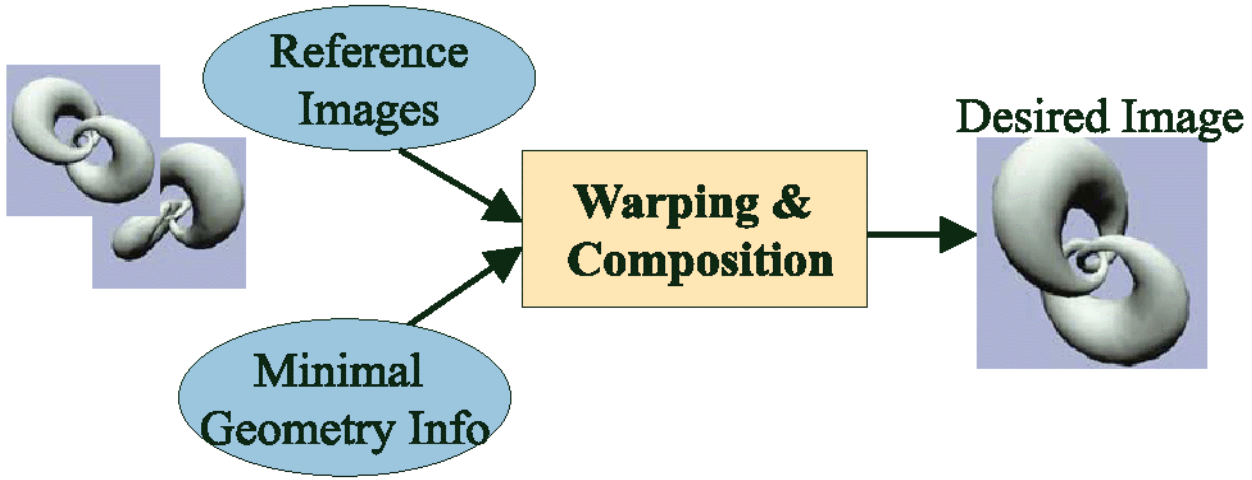
IBMR (2)

- The order of time complexity of geometry-based rendering a scene containing *N* objects is at least *O(N)*.
- The problem is *N* can be *arbitrarily large*.
- Imagine a scene of forest.
- More believe that no matter how fast is a computer, we can never display arbitrarily complex scene in real time.
- Modeling complex world is tedious.

IBMR (3)

Image-based Rendering:

- Instead of modeling the world with geometry, *new images can be generated by making use of prerecorded images*.



- New image is called the *desired* image.
- Prerecorded images are called *reference* images.

IBMR (4)

- No longer needed to model the scene with geometry. Scene can be modeled by taking photographs.
- Unlike computer vision, image-based rendering does not recover the geometry from images, it directly makes use of image pixels.
- The time complexity no longer depends on scene complexity, but on the image resolution.
- Image synthesis is no longer a simulation, but image warping and composition.

IBMR (5)

- e.g. QuickTime VR, a panorama is warped to generate perspective snapshots
- e.g. See my video.

inside looking out: QuickTime VR (Panorama, cubemap)
outside looking in : QuickTime VR Object Movie

- Morphing is another example of IBMR. The transformation may or may not be physically correct.

Google Street View

- Google put this into extreme by recording panorama snapshot along the streets worldwide

Google Street View (2)

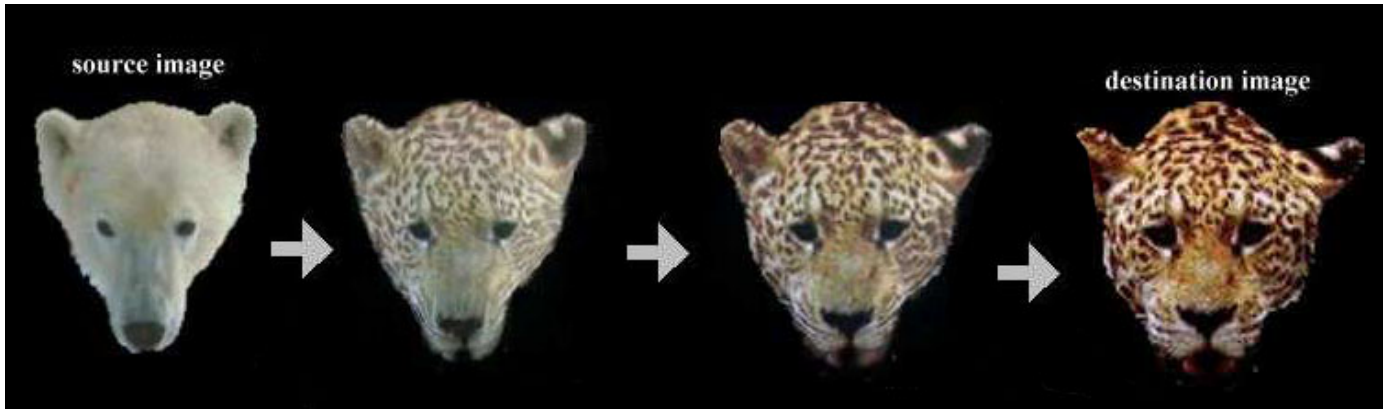
- They mount a panoramic video camera on a car to capture

Image Morphing

- MTV “Black and White”, read the morphing paper
  Thaddeus Beier and Shawn Neely, “Feature-Based Image Metamorphosis”, SIGGRAPH’92 Proceedings, July 1992, pp. 35-42.

fade in fade out 變形

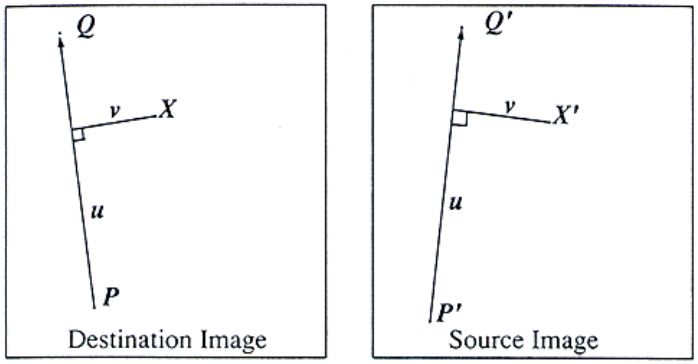
- Blending and Warping



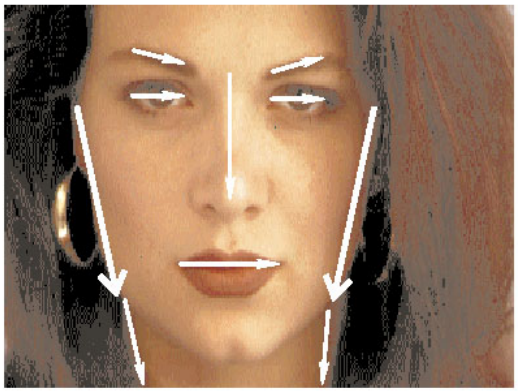
- Blending is easy, just weight the pixel color
- How about warping?

Image Morphing (2)

- Coordinate of a pixel with respective to the user-defined lines:



- Inverse mapping (destination to source) prevents holes.
- Pairs of lines define correspondences, they tell us how the image should be distorted (warped).



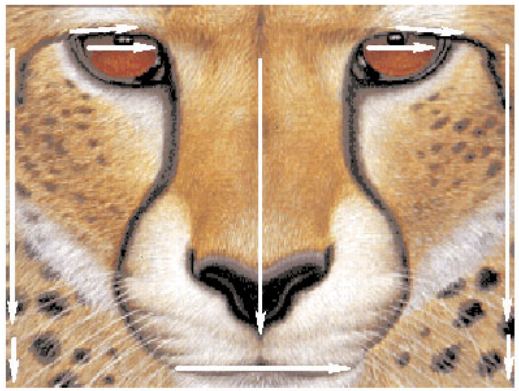
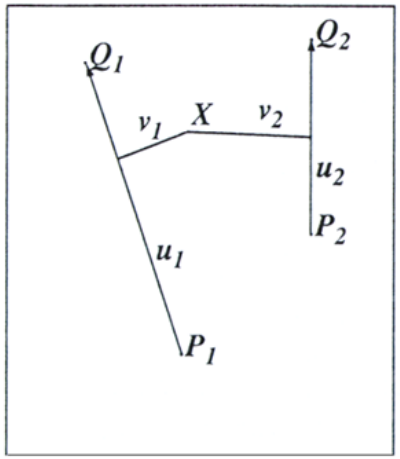


Image Morphing (3)

- Transformation with multiple pairs of lines is affected by the distance between pixel and lines. (or weights)

$$\text{weight} = \left(\frac{\text{length}^p}{a + \text{distance}} \right)^b$$



- Algorithm:

```
For each pixel X in the destination
   disp = (0,0)
   weightsum = 0
   for each line PiQi
       calculate u, v
       calculate X'I based on u, v and P'iQ'i
       calculate Di = X'i - X
       dist = shortest distance from X to PiQi
       calculate weight (using equation above)
       disp += Di * weight
       weightsum += weight
   X' = X + disp/weightsum
```

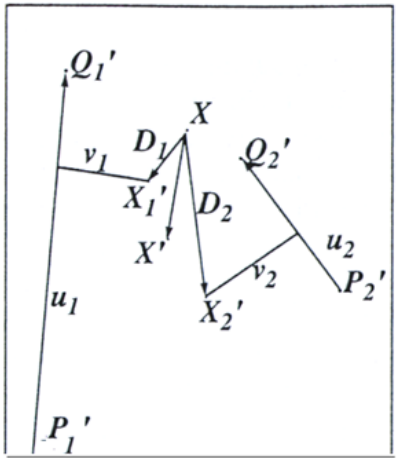
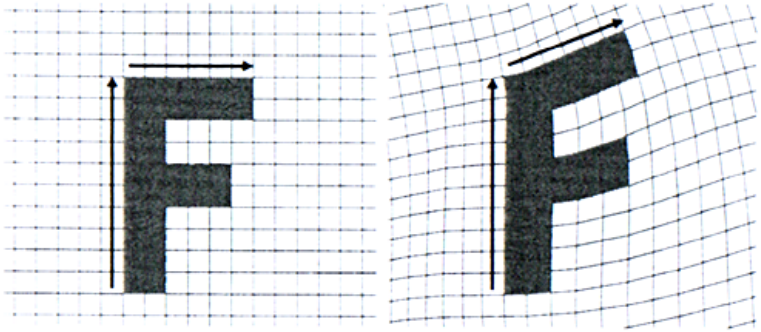


Image Morphing (4)

- Some results:

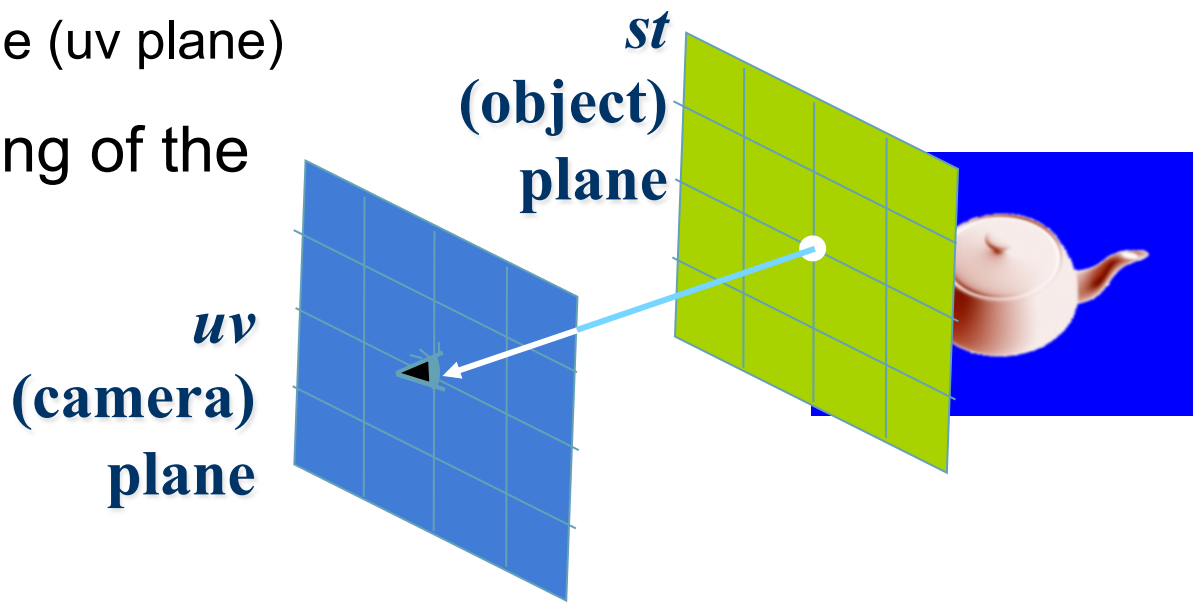


IBMR

- People are more interested in physically correct image synthesis based on images.
- Hard Problems:
  - Once the images are captured, the viewing position is not allowed to change. How to interpolate view?
  - The lighting cannot be changed. How to control illumination?
  - Some portion of the scenes are missing due to occlusion.
- Morphing is one of the solutions (probably the simplest).

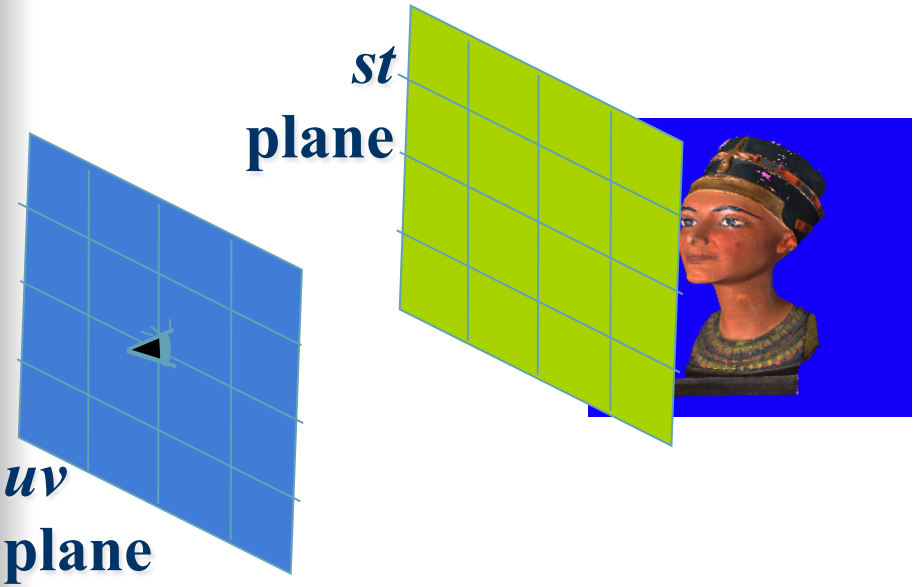
Light Field

- How can we change the viewpoint?
- Take multiple reference images from different viewpoints
- Light field representation:
  - Object plane (st plane)
  - Camera plane (uv plane)
- Dense sampling of the environment.

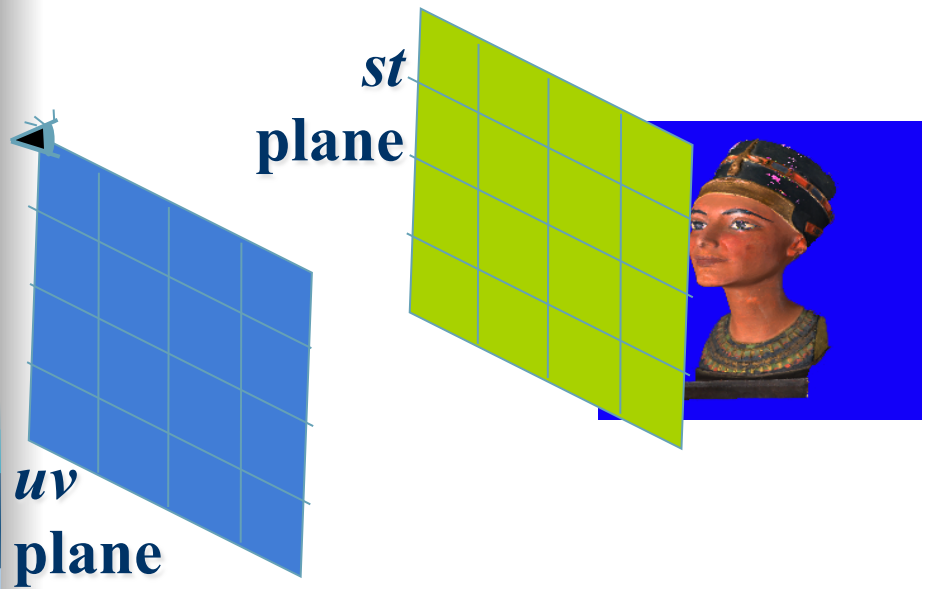


Light Field

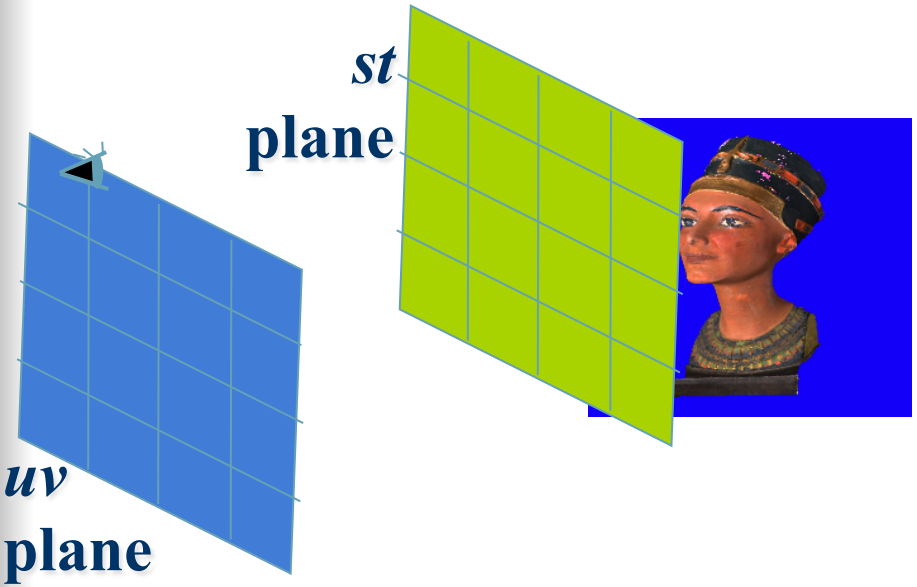
- Let's capture the reference images



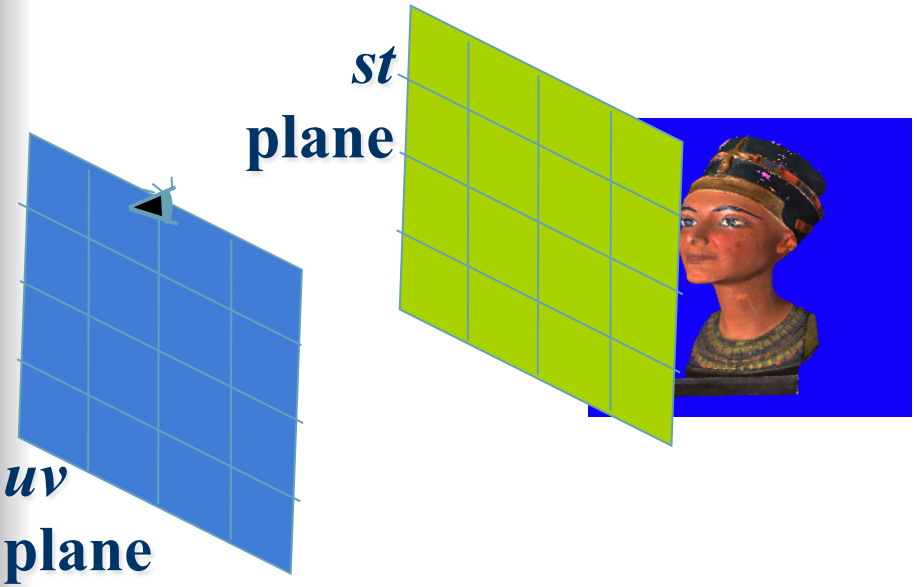
Light Field

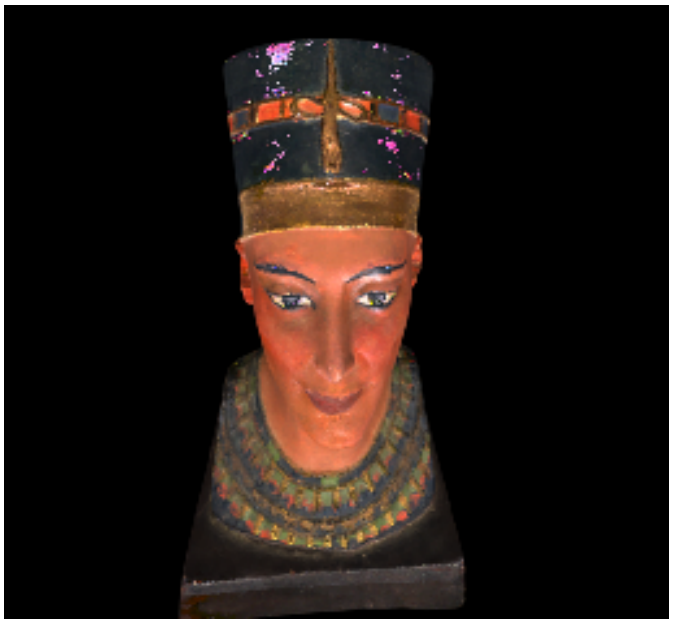


Light Field

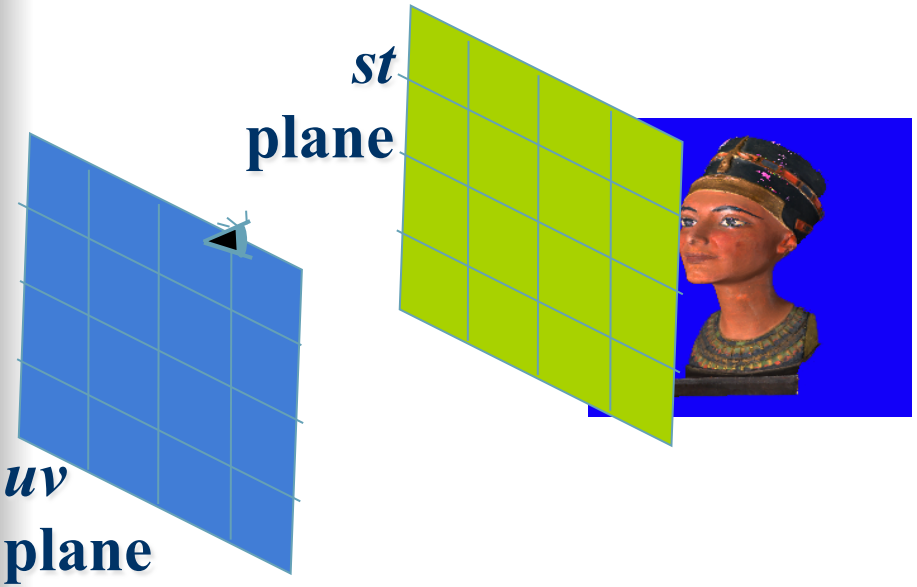


Light Field

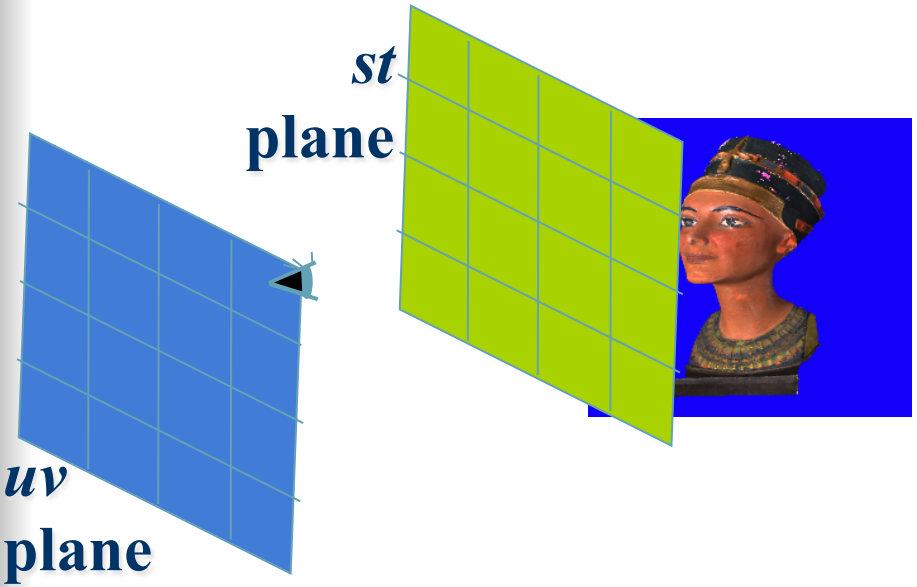




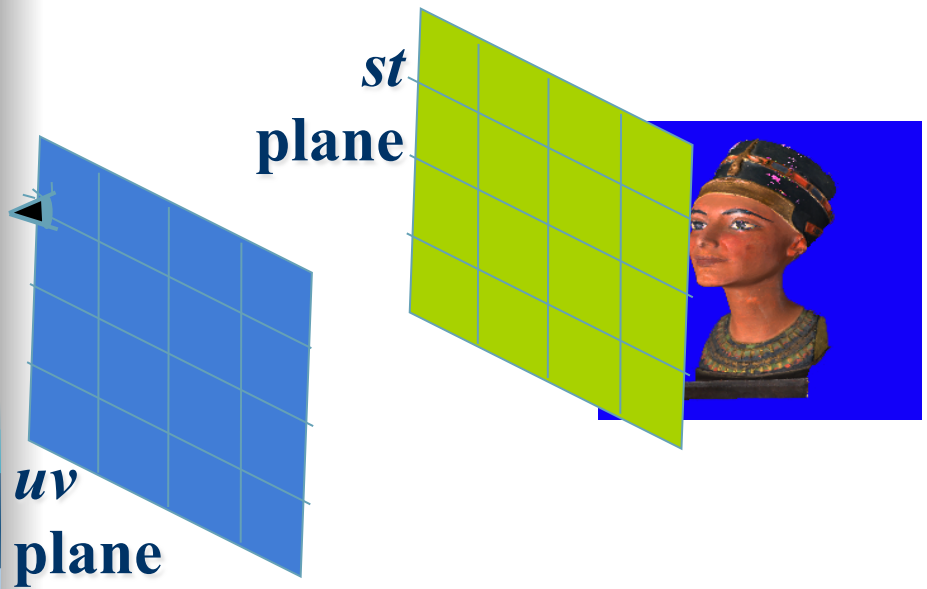
Light Field



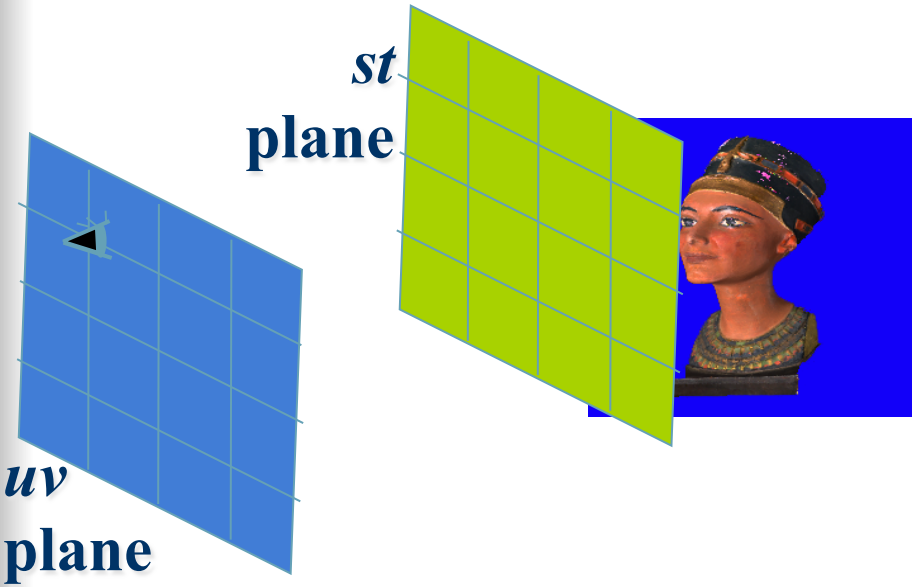
Light Field



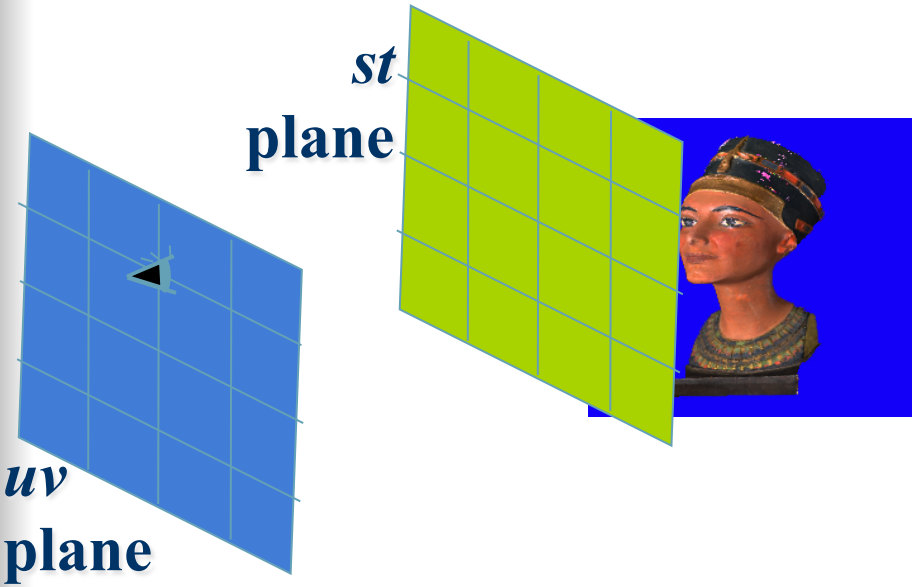
Light Field



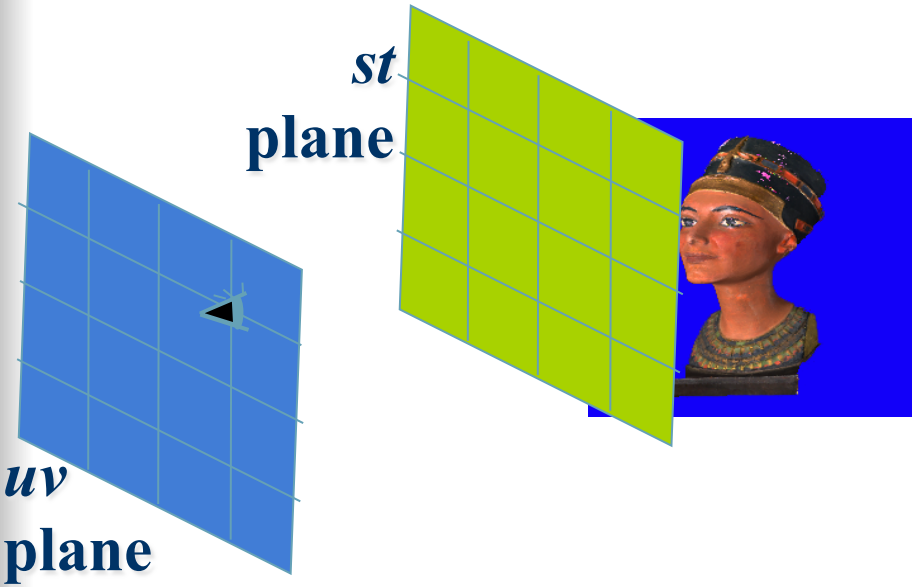
Light Field



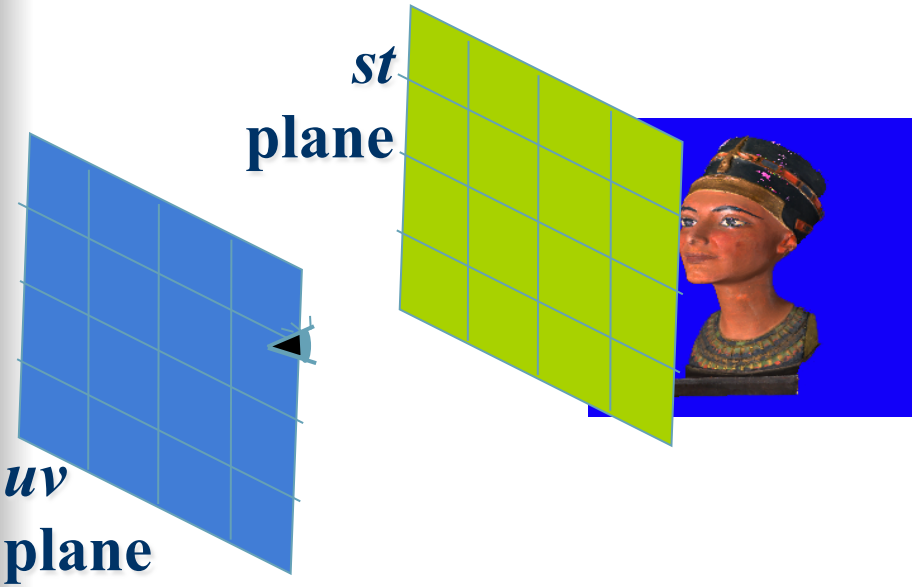
Light Field



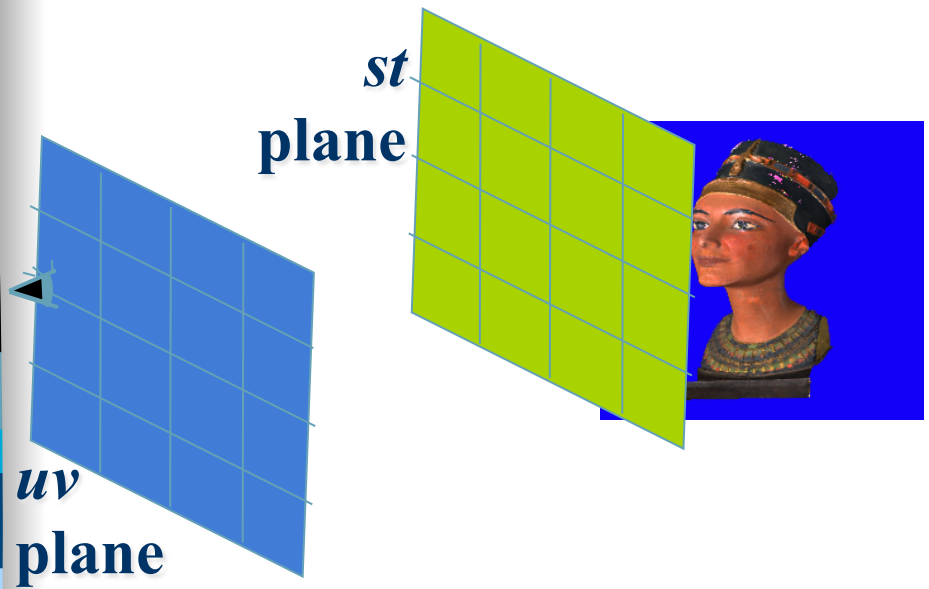
Light Field



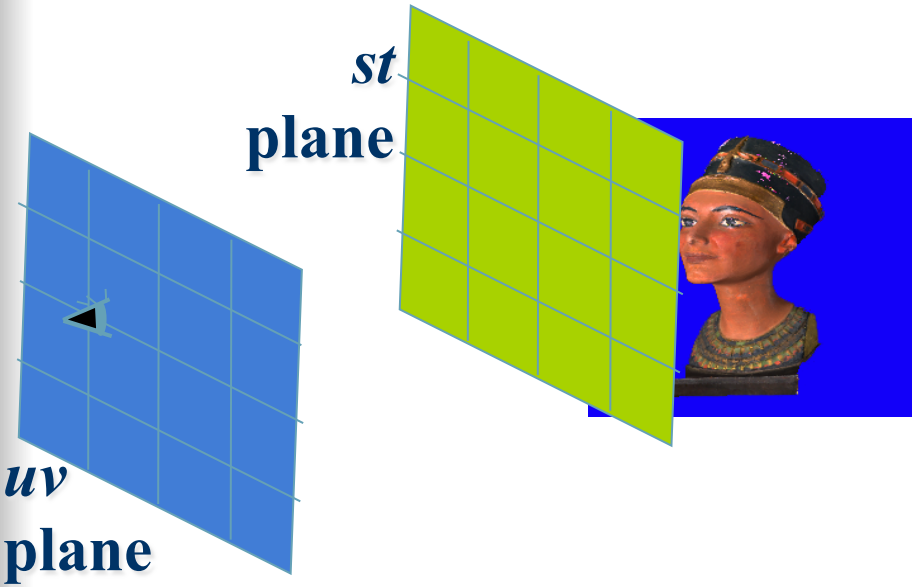
Light Field



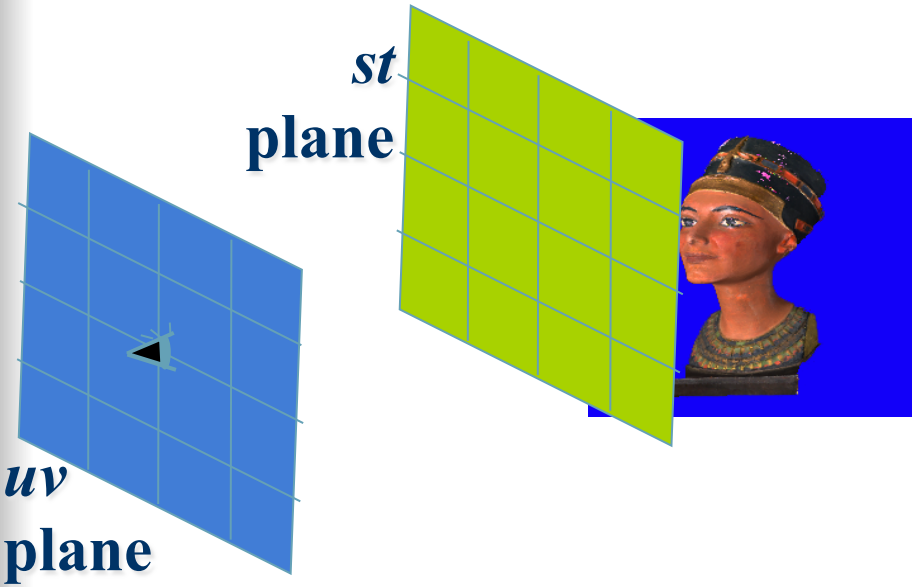
Light Field



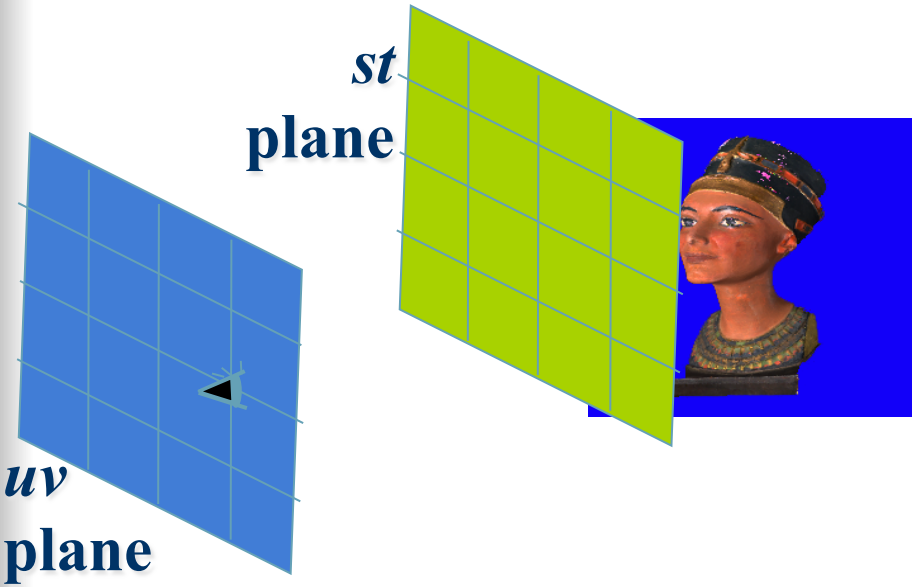
Light Field



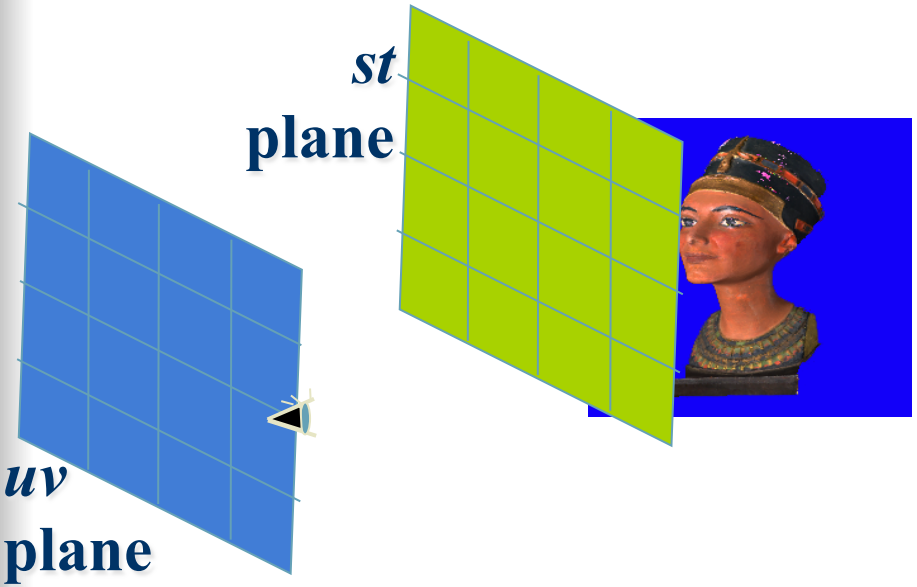
Light Field



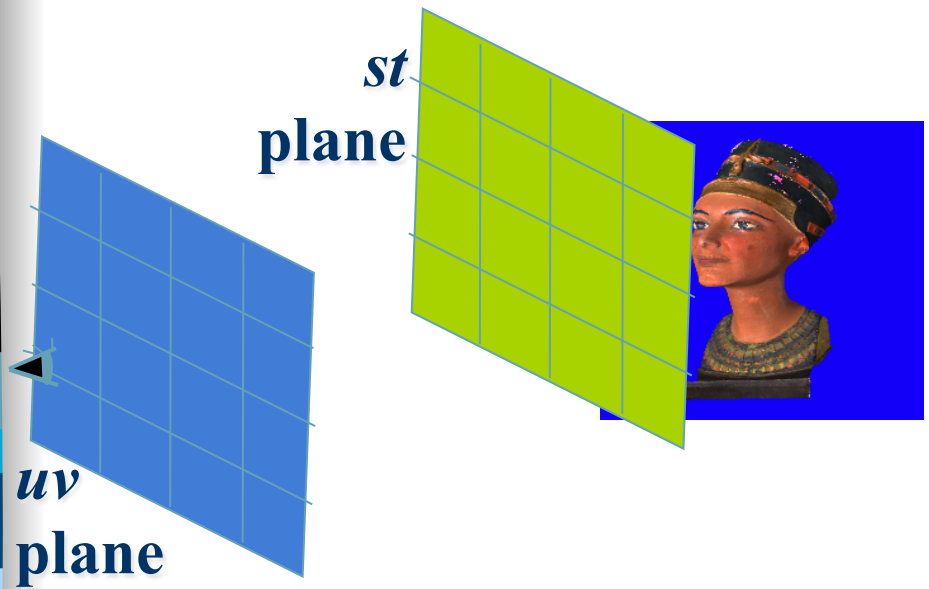
Light Field



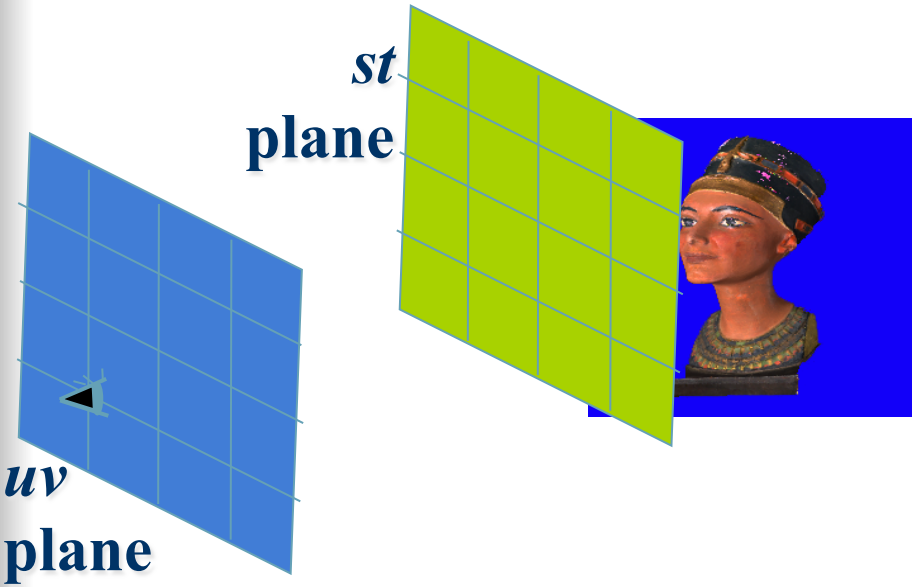
Light Field



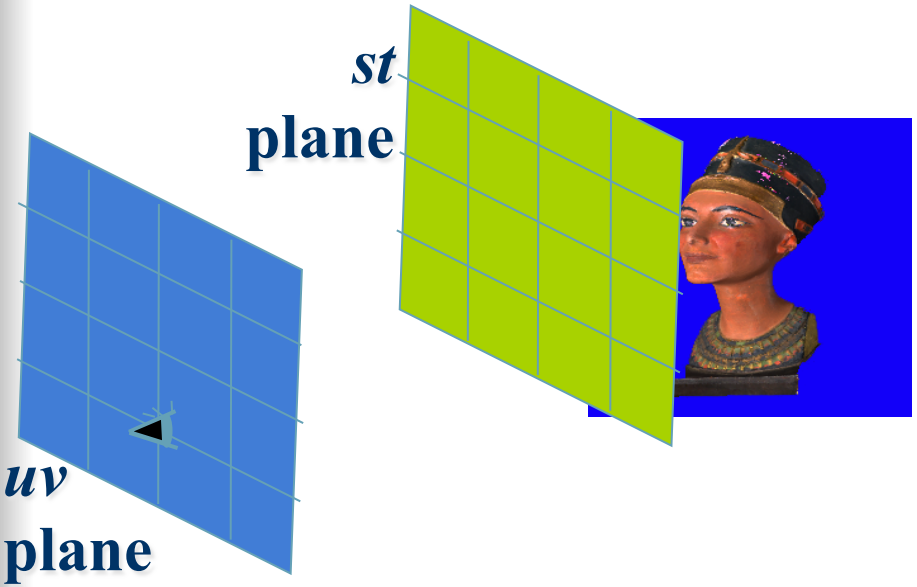
Light Field

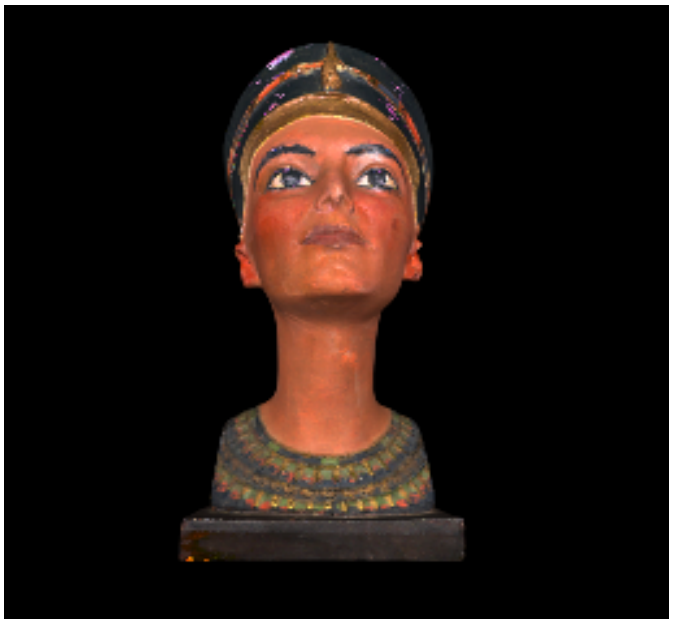


Light Field

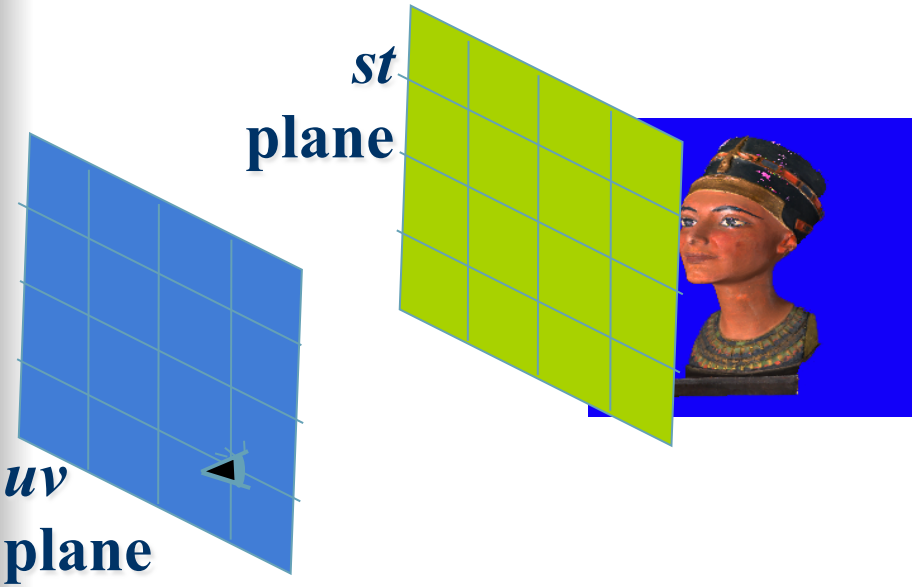


Light Field

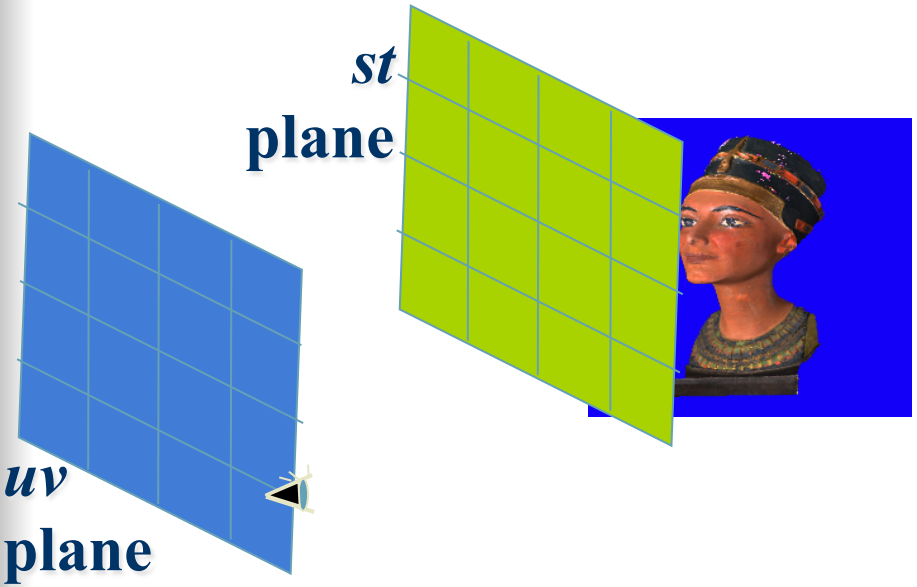




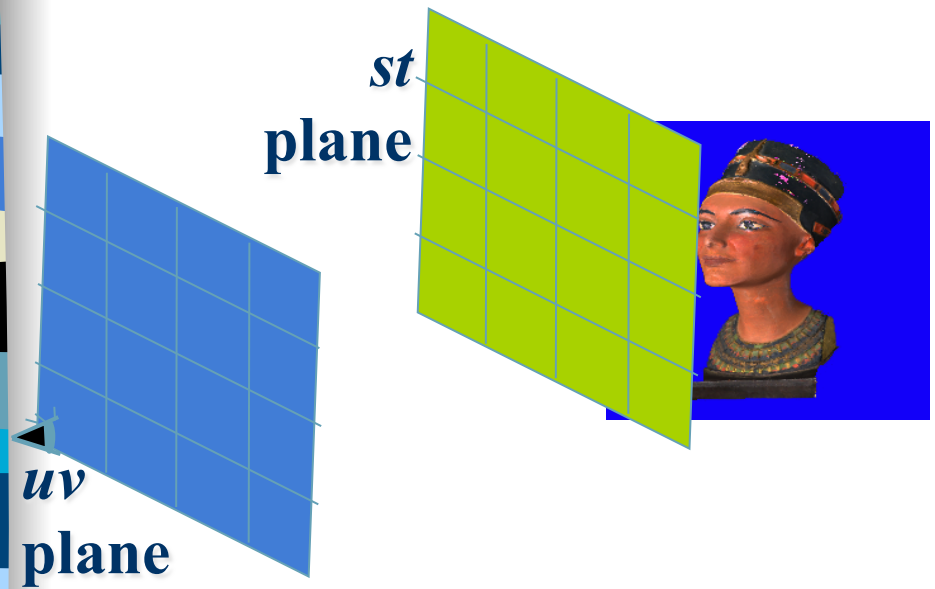
Light Field



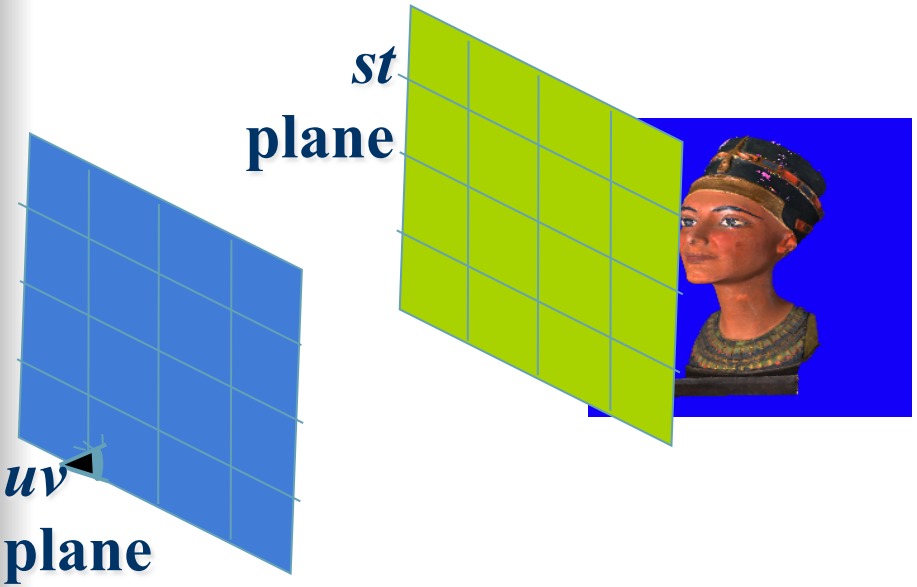
Light Field



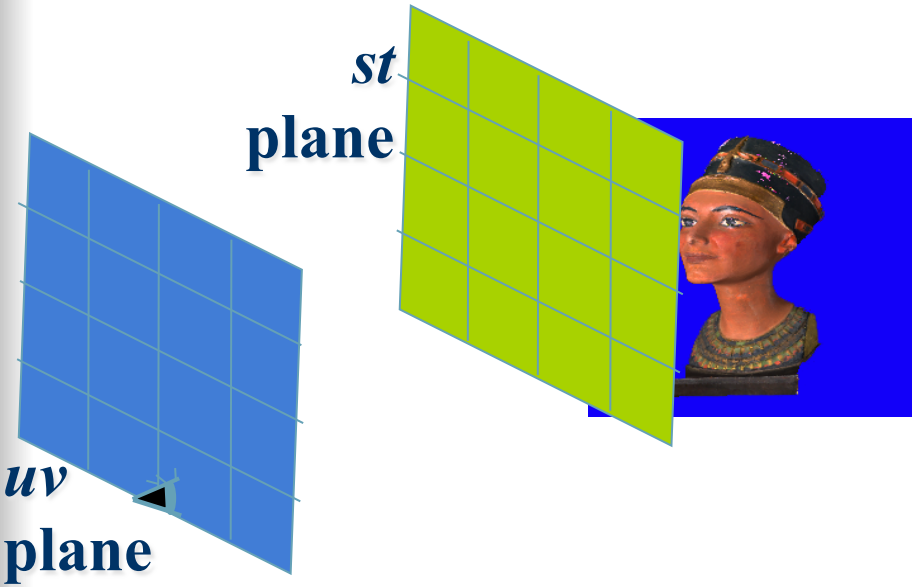
Light Field



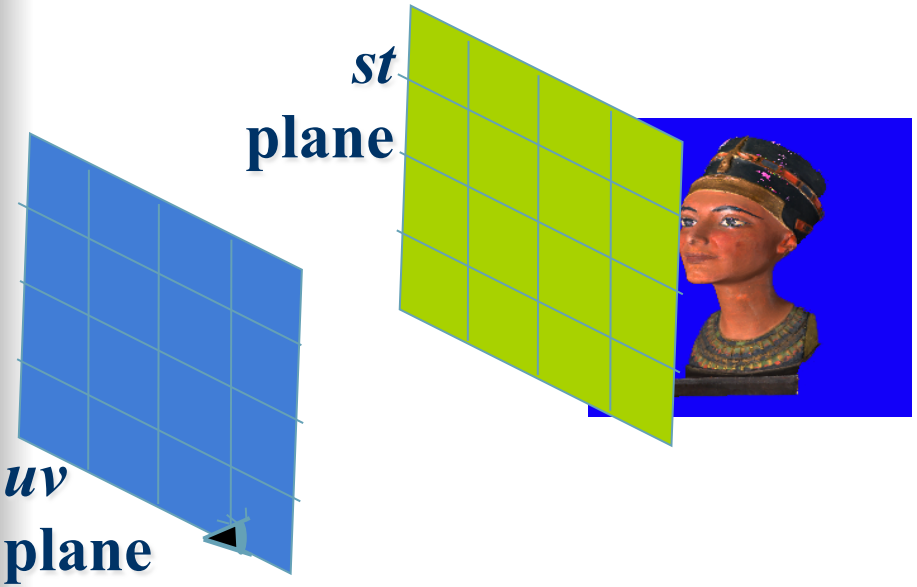
Light Field



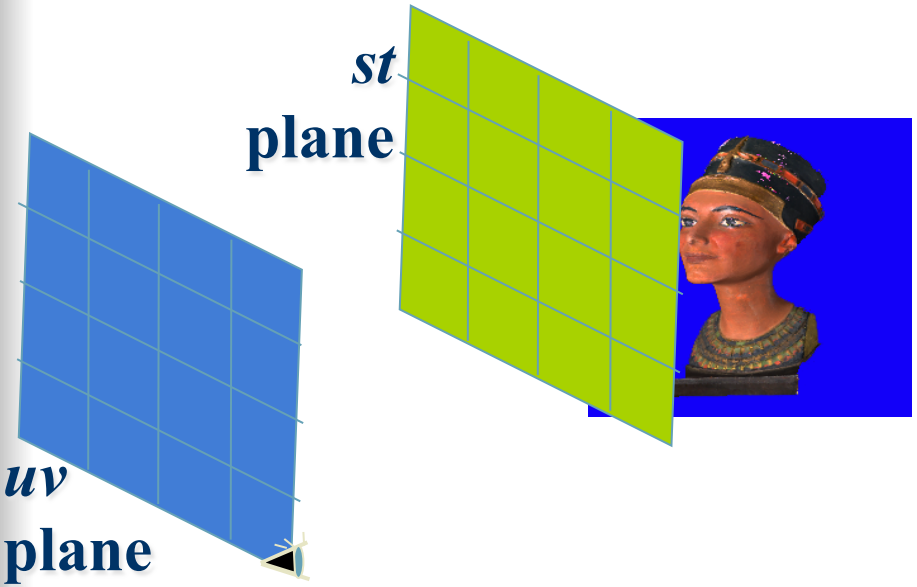
Light Field



Light Field

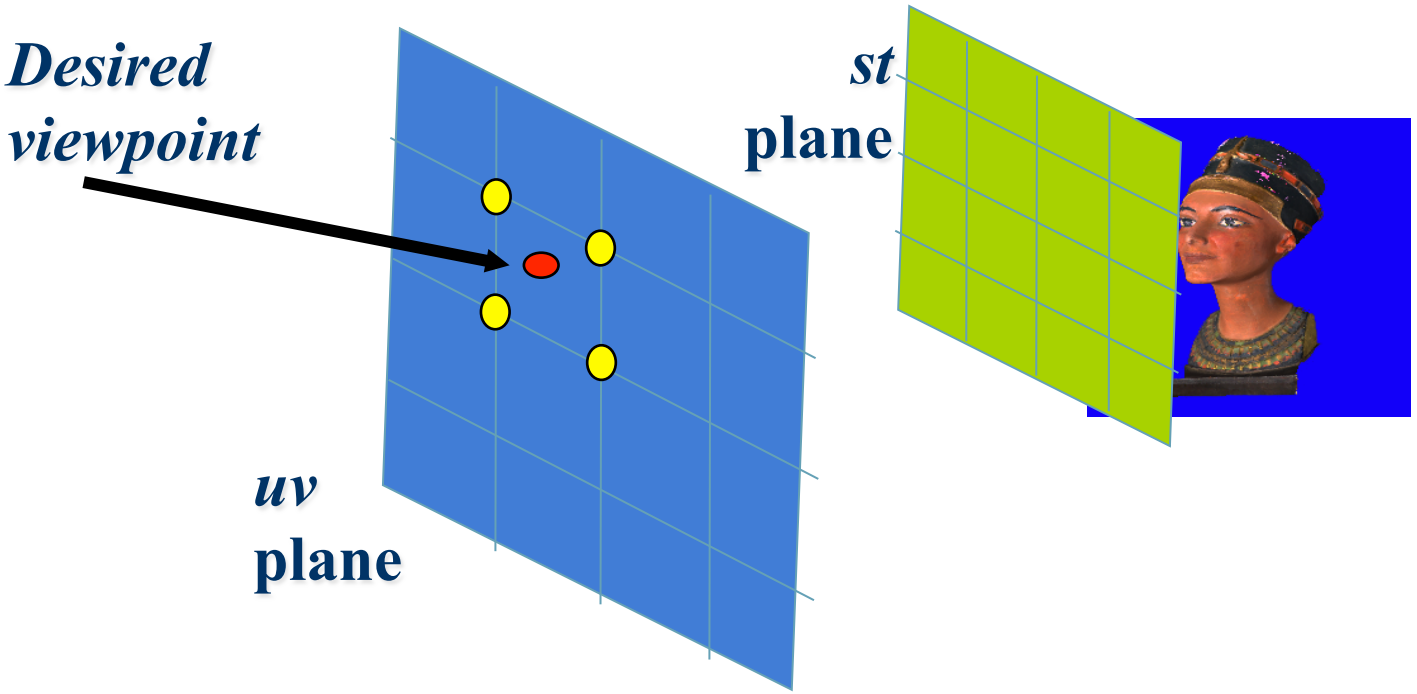


Light Field



View Interpolation

- Any viewpoint that is not on the sampling grid can be interpolated
  - Nearest neighbor - no good - 1 neighbor sample
  - Bilinear interpolation - acceptable - 4 neighbor samples
  - Quadratic interpolation - best - 9 neighbor samples



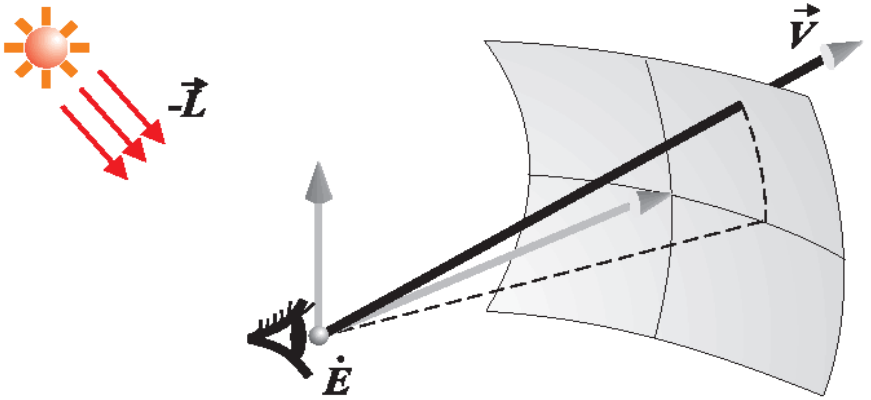
Light Field Camera Array

- To capture real world light field, researchers build camera array (Stanford Multi-Camera Array)

The Plenoptic Illumination Function

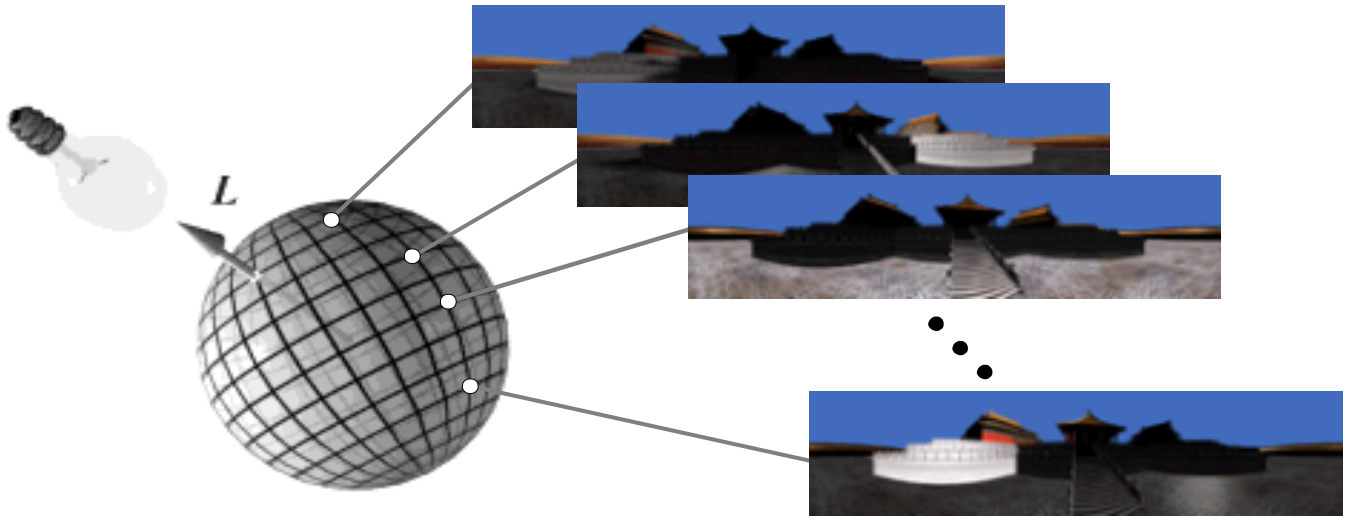
- By taking many images we can interpolate view, how about the modification of lighting?
- *Relighting* (change of lighting based on images)
- We defined *the plenoptic illumination function*
- It describes the radiance received along the viewing direction V, at viewpoint E, under the illumination of a directional light source from L

$$P_I(\vec{L},\vec{V},\dot{E},t',\lambda)$$



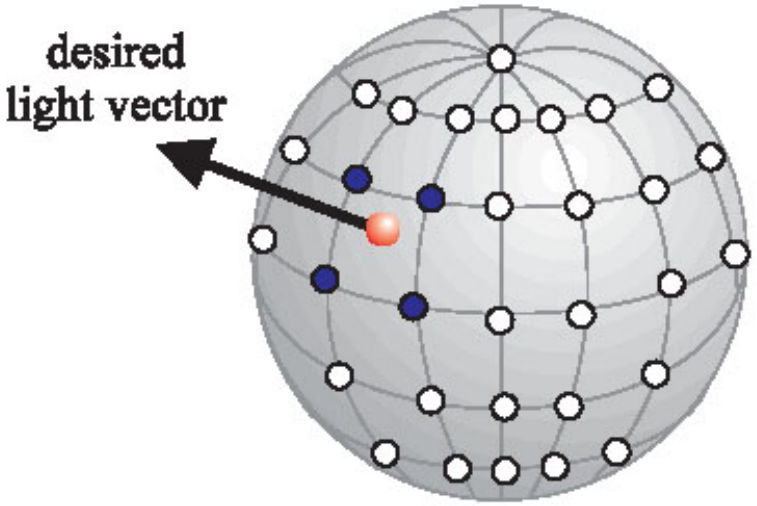
Sampling the Plenoptic Illumination Function

- Firstly, we have to sample the illumination dimension
- A directional light is put on the spherical grid & then render
- Then we have many reference images, e.g. 30 x 40



Relighting

- Just like view interpolation, relighting is basically as a process of
  - look-up of color values
  - Interpolate the color values



Relighting (2)

- A nice property of illumination: linearity
  ⇒ multiple-source relighting is simply superposition

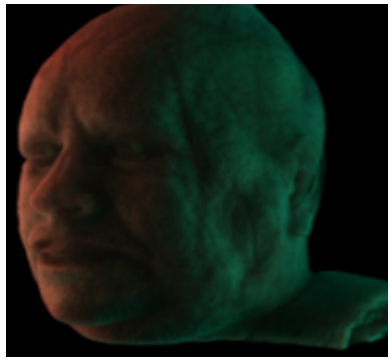


Image illuminated by red and cyan lights

=

Image illuminated by a red light

+

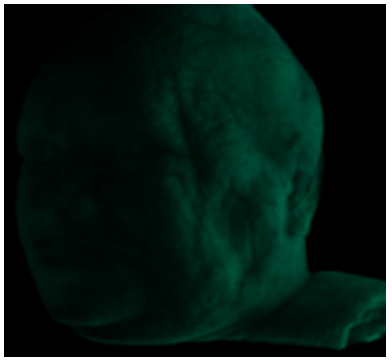


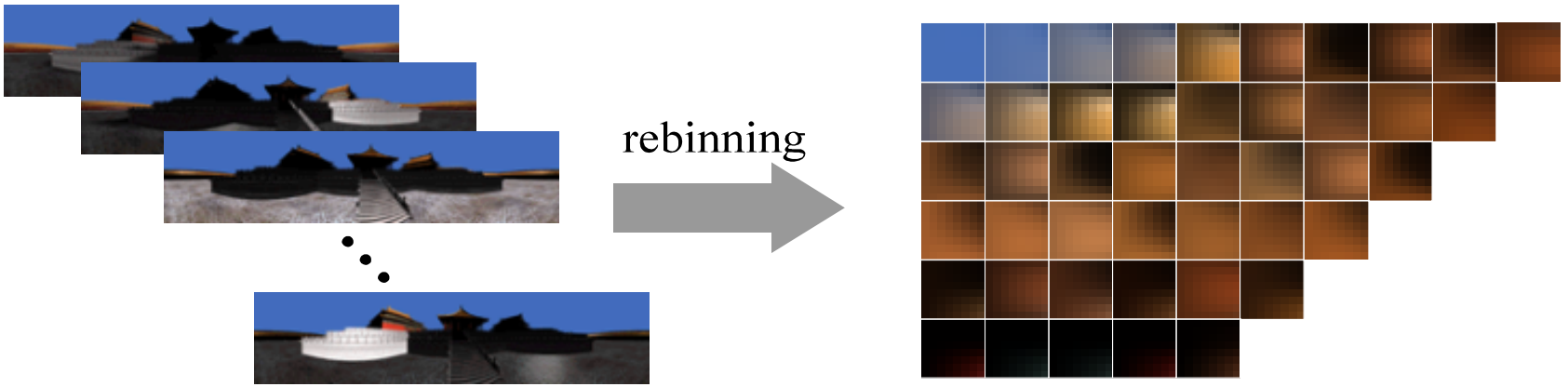
Image illuminated by a cyan light

Relighting in Film Production

- People adopts our CUHK relighting idea and put it into real film production (light stage)

Compression: Rebinning

- However, the data size is too large, e.g. 256x256, 1200 reference images => 225MB
- The key of IBMR is actually *compression*
- To facilitate the compression, we rebin the color values of multiple images
- For each pixel, we collect all color values associated with it ⇒ 1 color tile is formed
- Most of the tiles are smooth in color

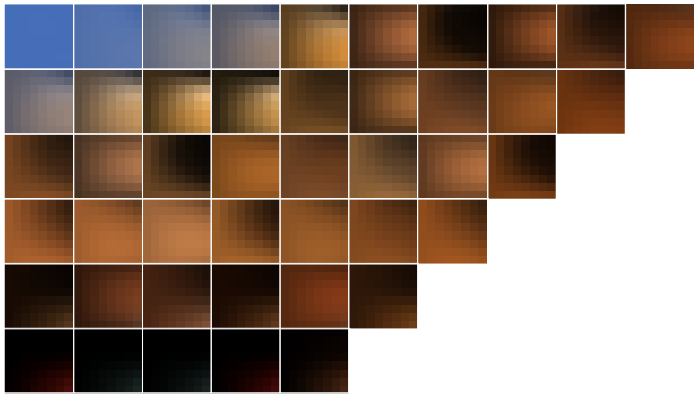


Compression: Transform Coding

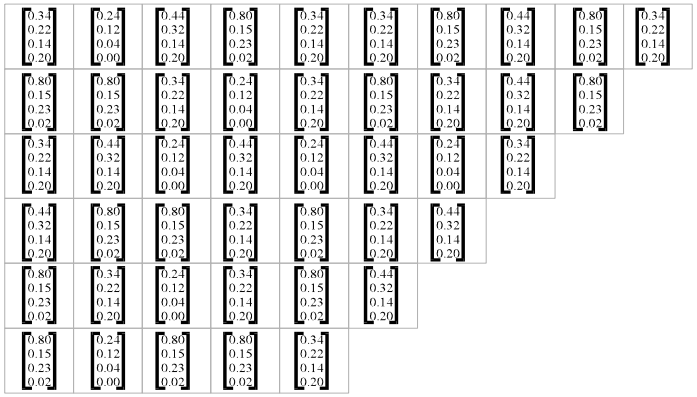
- Once you obtain the color tiles, you can easily compress them by any standard image compression techniques e.g. JPEG, JPEG2000, …
- In fact, we use a specialized transform coding: spherical harmonic transform

Compression: Transform Coding (2)

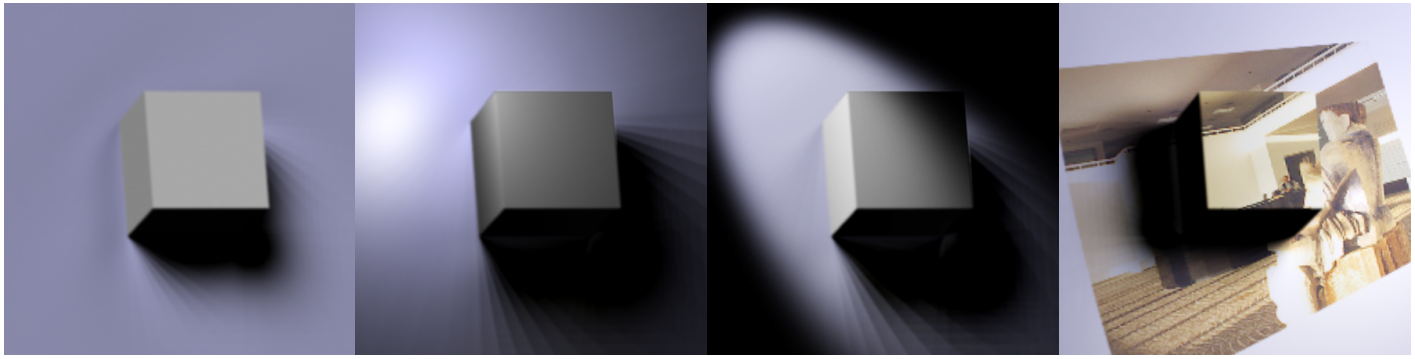
- No matter what transform coding method you used, each color tile is converted to a coefficient vector
- Storage can be largely reduced
- We shall discuss detailly on compression in later chapters



spherical harmonic transform



Examples



directional point spotlight slide projection

- You can try a demo at
  http://www.cse.cuhk.edu.hk/~ttwong/demo/panoview/panoview.html