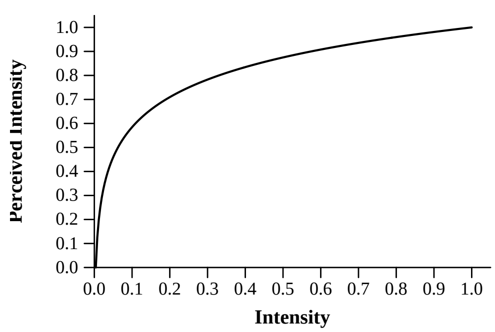
Intensity

Intensity curves
- Stimulus/Response - Perception
- Input/Output - Gamma

Quantization
- Uniform quantization
- Non-uniform quantization

Dithering and halftoning
- Random dither
- Ordered dither
- Clustered dither or halftone
- Error diffusion

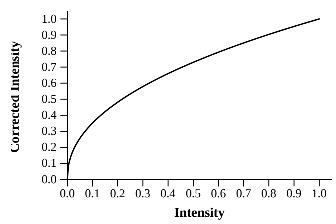
Perception of Intensities

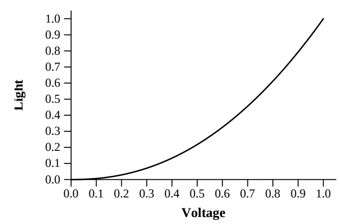


$$\Delta PI = \frac{\Delta I}{I} = \text{constant}$$

Weber-Fechner Law
Logarithmic Response

Gamma





Camera γ=0.6 **Monitor γ=2.2**

$$L = V^{\gamma}$$

Gamma Correction

Goal: Operate in linear light intensity space

Input	Output
0	0
...	0
11	0
12	1
...	1
19	1
...	...
251	247
252	249
253	251
254	253
255	255

$$L = \left(V^{1/\gamma}\right)^{\gamma} = V$$

Vidicon (γ=0.5) to Monitor (γ=2.8) = System γ=1.4

Graphics system: γ=1.2 or 1.7 or 2.4

Quantization

Definition: The process of turning a high precision number into a low precision number

Why?

- **Conserve memory**
- **Binary output devices**

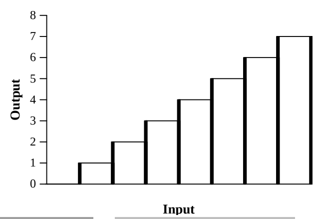
How?

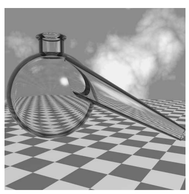
- **Uniform perceptual steps**
- **Minimal error**

Uniform Quantization

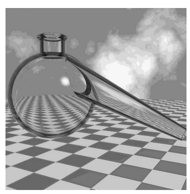
Uniform map:

$$pv = trunc(K \times v)$$
$$0 \le v < 1$$
$$0 \le pv < K$$

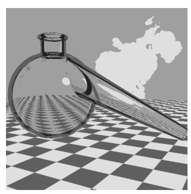




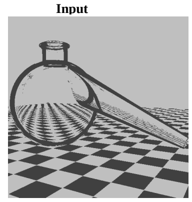
K=256



K=16



K=4



K=2

Notice contouring

CS248 Lecture 3
Copyright © Pat Hanrahan

Quantization Error

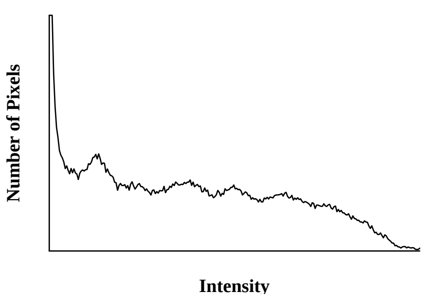
Quantization introduces error

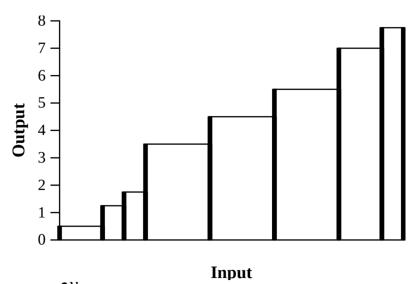
$$E^2 = \sum_{(x,y)} (pv(x,y) - K \times v(x,y))^2$$

To reduce error:

1. **Optimal quantization (minimize error)**
2. **Dithering (trade-off intensity/space error)**

Non-uniform Quantization





$$\hat{v}_k = \frac{\int_{v_{k-1}}^{v_k} v p(v) dv}{\int_{v_{k-1}}^{v_k} p(v) dv} \qquad \{v_k : 0 \le k < K\}$$

$$E^2 = \sum_{k=1}^{K} \int_{v_{k-1}}^{v_k} (v - \hat{v}_k)^2 p(v) dv$$

$$\bar{v}_k = \frac{1}{2}(\hat{v}_{k-1} + \hat{v}_k)$$

K-Means Algorithm

1. **Pick initial means** $\{\hat{v}_k\}$
2. **Compute interval** $\bar{v}_k = \frac{1}{2}(\hat{v}_{k-1} + \hat{v}_k)$
3. **Quantize the input**
4. **Compute new means** $\hat{v}_k = \frac{\int_{v_{k-1}}^{v_k} v\, p(v)\, dv}{\int_{v_{k-1}}^{v_k} p(v)\, dv}$

Dithering and Halftoning

Trade spatial for intensity resolution

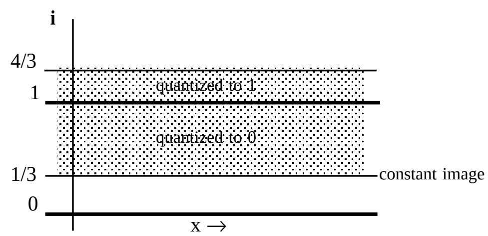
- **Random dither; Robert's algorithm**
- **Ordered dither**
- **Clustered dither or halftones**
- **Error diffusion**

Your eye will average over an area

Robert's Algorithm

First add noise $\hat{v}(x,y) = trunc(K \times v(x,y) + noise(x,y))$

Then quantize $0 \le noise < 1$



Moves low frequency (average error) to high frequency

Pink(low), *Blue (high),* White(all) frequency noise

Noise

Noise functions

- **Random function vs random sequence**
- **Some noise better than others: blue, ordered**
- **Reproducible and order independent**
- **Fast: precompute and store in a table**

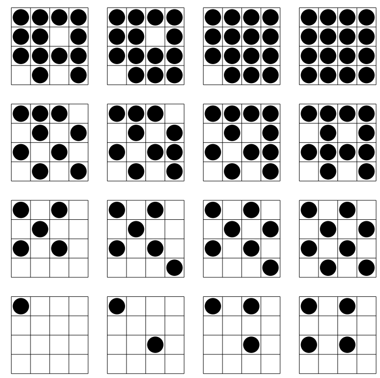
```
for(y=0; y<ny; y++)
  for(x=0; x<nx; x++)
    vq[x][y] =
      (v[x][y]<threshold[x%M][y%M]) ? 0 : 1;
```

Bayer Ordered Dither Patterns

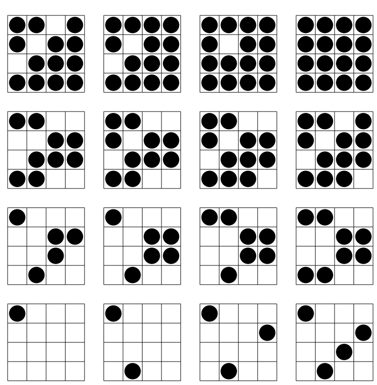
$$D_2 = \begin{bmatrix} 0 & 2 \\ 3 & 1 \end{bmatrix}$$

$$D_n = \begin{bmatrix} 4D_{n/2}+0 & 4D_{n/2}+2 \\ 4D_{n/2}+3 & 4D_{n/2}+1 \end{bmatrix}$$

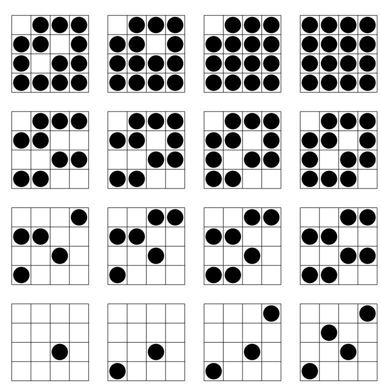
$$D_4 = \begin{bmatrix} 0 & 8 & 2 & 10 \\ 12 & 4 & 14 & 6 \\ 3 & 11 & 1 & 9 \\ 15 & 7 & 13 & 5 \end{bmatrix}$$



Magic Squares



$$M_4 = \begin{bmatrix} 0 & 6 & 13 & 11 \\ 9 & 15 & 4 & 2 \\ 14 & 8 & 3 & 5 \\ 7 & 1 & 10 & 12 \end{bmatrix}$$



$$P_4 = \begin{bmatrix} 15 & 8 & 5 & 2 \\ 4 & 3 & 14 & 9 \\ 10 & 13 & 0 & 7 \\ 1 & 6 & 11 & 12 \end{bmatrix}$$

Halftoning

Printers: *Binary Blobs* of ink

Clustered pattern

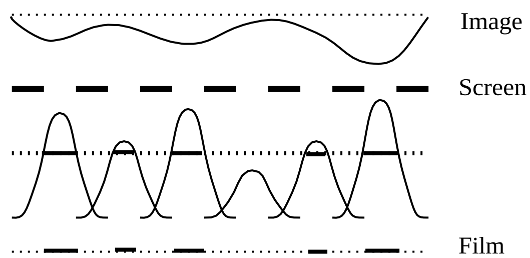
Enlarging spot

Classical printer screens

Spot function

Line frequency

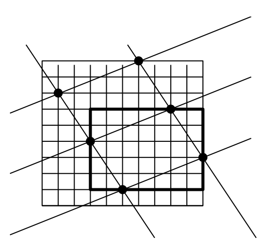
Tilt angle

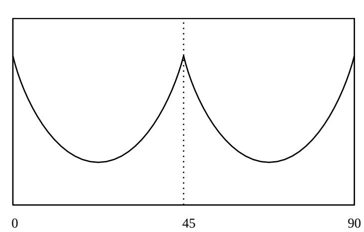


Screen Patterns

Interplay between discrete angles and discrete spacing

- **Given printer dpi implies**
  High spatial resolution means low intensity resolution
- **Angle must be ratios of integers m/n**
  45 works best; different angles for colors, moires





Error Diffusion

Idea: Quantize, then distribute error to neighbors

```
for(y=0; y<ny; y++)
  for(x=0; x<nx; x++){
    vq[x][y] = quantize(v[x][y]);

    e = v[x][y] - vq[x][y];

    i[x+1][y] += 3/8*e;
    i[x][y+1] += 3/8*e;
    i[x+1][y+1] += 1/4*e;
}
```