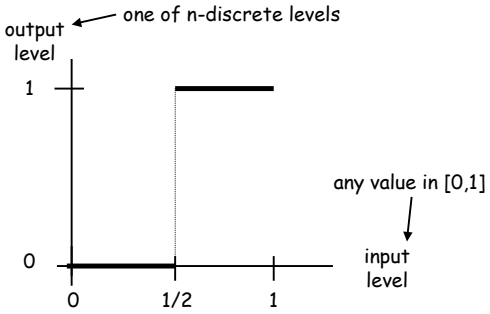
type of techniques

- simple pixel modification
- interpolation/extrapolation
- compositing
- convolution
- **dithering**
- warping
- morphing
- misc. effects

digital image

- sample at points on grid
- quantize color at sample points

n-level quantization (per channel)

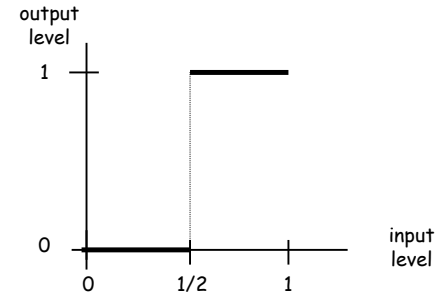


uniform quantization

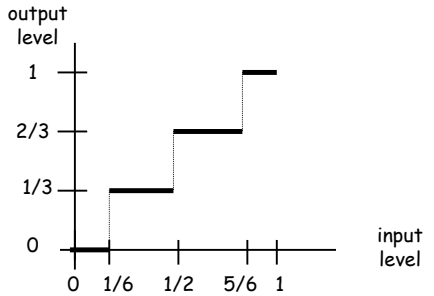
Guiding principles:

- 0 and 1 are valid output levels
- For any input level x, the output $y=Q_n(x)$ should be chosen so as to minimize $|y-x|$.

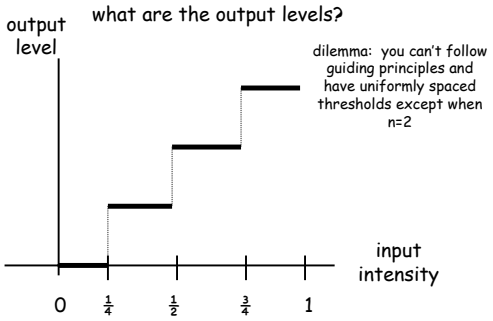
2-level uniform quantization



uniform quantization: 4 level



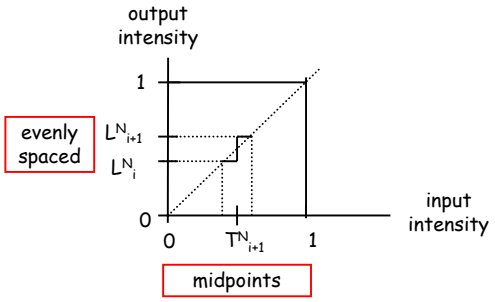
4-level uniformly spaced thresholds



n-level uniform quantization

- output levels:
  evenly spaced
- thresholds:
  midpoints

n- level uniform quantization



n-level uniform quantization

- output levels:
  $L^n_i = i/(n-1)$, $i = 0, \ldots, n-1$
- thresholds:
  $T^n_i = (L^n_{i-1} + L^n_i)/2 = (2i-1)/2(n-1)$, $i = 1, \ldots, n-1$
- quantization function:
  $Q_n:[0,1] \rightarrow \{0, 1/(n-1), 2/(n-1), \ldots, 1\}$
  $Q_n(v) = \lfloor v(n-1) + 0.5 \rfloor/(n-1)$

quantization error

8 bits per pixel per channel

1 bits per pixel per channel

dithering

add "noise" to camouflage quantization error

random dither

add random noise to camouflage quantization artifacts

8 bits per pixel per channel | 1 bits per pixel per channel | 1 bits per pixel per channel noisy

dithering

- random
- ordered
- error diffusion

comparison

original 8 bits/pixel/channel | random | ordered | error-diffusion

1 bit/pixel/channel

random dither

for each pixel in the input image

- add random noise in $[-\varepsilon, \varepsilon]$ to pixel value
- clamp value to [0,1]
- uniformly quantize new value

for project 1 ask user to choose ε

random dither

add noise to camouflage quantization artifacts

8 bits/pixel/channel | 1 bits/pixel/channel | 1 bits/pixel/channel dithered

ordered dither

add pseudo-random noise to camouflage quantization artifacts

8 bits/pixel/channel | 1 bits/pixel/channel | 1 bits/pixel/channel dithered

ordered dither

intuition

image coherence

adjacent pixels are usually similar

(or perceived as similar in high-resolution images)

0.32	0.57
0.97	0.21

0.52	0.52
0.52	0.52

ordered dither intuition: 2 level output

suppose all 2x2 neighborhoods were uniform

.2	.2	.6	.6	.3	.3
.2	.2	.6	.6	.3	.3
.4	.4	.9	.9	.2	.2
.4	.4	.9	.9	.2	.2

2 level output: quantized blocks

every neighborhood is quantized in one of two ways

0	0
0	0

1	1
1	1

is there a better way?

we can simulate 5 levels:

0

0	0
0	0

1/4

0	0
1	0

1/2

0	1
1	0

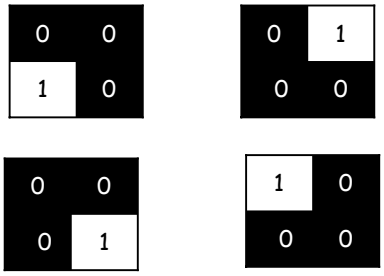
0	1
1	1

3/4

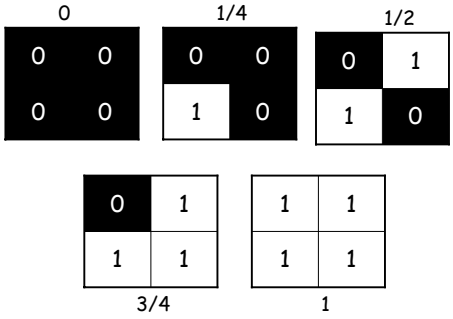
1	1
1	1

1

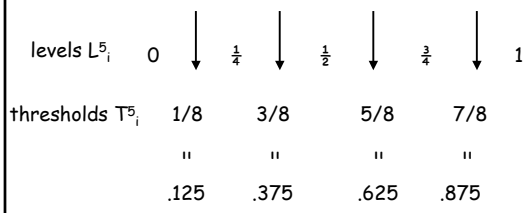
note: these are the same shade of gray



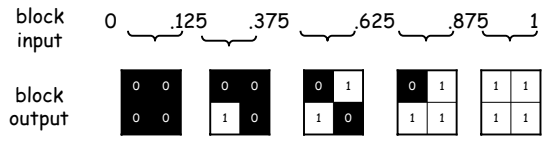
what are corresponding thresholds?



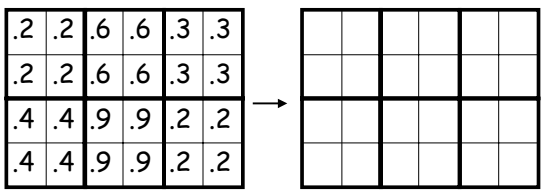
midpoint thresholds for 5 output levels



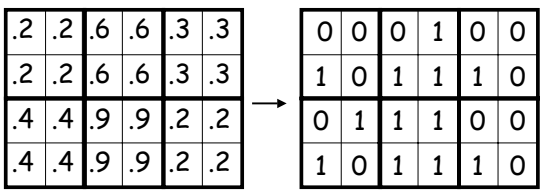
quantization rule



exercise



quantization



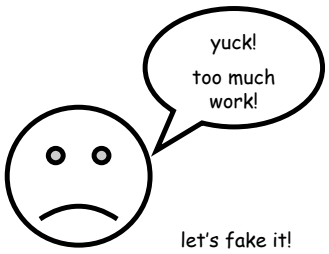
non-uniform neighborhoods

- average neighborhoods
- then use previous algorithm

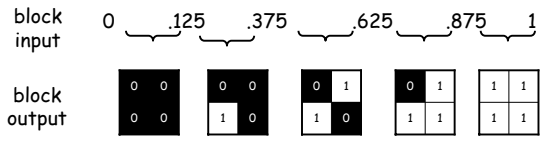
exercise

.3	.3	.4	.4	.8	.2
.3	.2	.6	.6	.2	.8
.5	.5	.9	.5	.2	.2
.4	.5	.9	.9	.9	.2

→



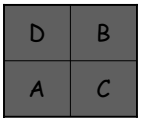
quantization rule



Question: What is the quantization rule for the lower left pixel of the block?

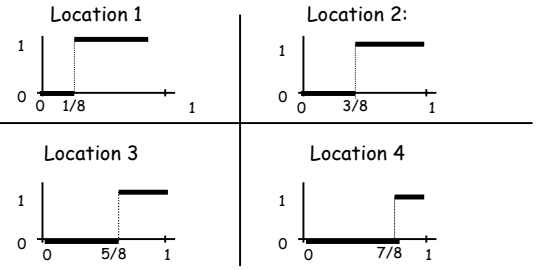
equivalent quantization rule

1. find pixel location in block



equivalent quantization rule

2. quantize based on threshold for pixel location



now we can use our algorithm even if we don't have uniform neighborhoods!

exercise

.3	.3	.4	.4	.8	.2
.3	.2	.6	.6	.2	.8
.5	.5	.9	.5	.2	.2
.4	.5	.9	.9	.9	.2

→

exercise

.3	.3	.4	.4	.8	.2
.3	.2	.6	.6	.2	.8
.5	.5	.9	.5	.2	.2
.1	.5	.9	.9	.9	.2

→

0	0	0	1	0	0
1	0	1	0	1	1
0	1	1	1	0	0
0	0	1	1	1	0

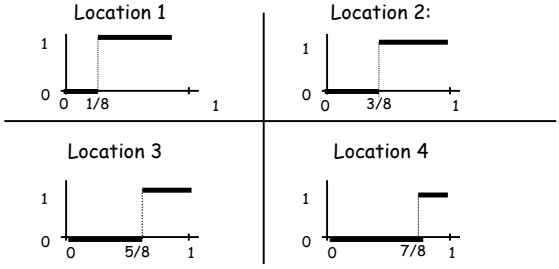
ordered dither recap: step 1

determine pixel location in neighborhood

4	**2**	4	2	4	2	4
1	**3**	1	3	1	3	1
4	2	4	2	4	2	4
1	3	1	3	1	3	1
4	2	4	2	4	2	4

ordered dither: step 2

quantize based on threshold for pixel location



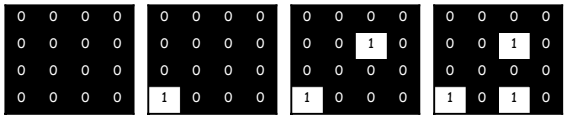
bottom line

ordered dither can simulate 5 output levels when only 2 output levels are available

can we simulate more than 5 output levels?

with window size kxk we can simulate k^2+1 levels

k=4, $k^2+1=17$



0 1 2 3

etc.

Bayer's ordered 4x4: pixel locations

this can be derived recursively from 2x2

16	8	14	6
4	12	2	10
13	5	15	7
1	9	3	11

Bayer's ordered 2x2: pixel locations

4	2
1	3

Bayer's ordered 4x4: pixel location

16	8	14	6
4	12	2	10
13	5	15	7
1	9	3	11

Bayer's ordered 4x4: pixel location

16	8	14	6
4	12	2	10
13	5	15	7
1	9	3	11

etc.

k^2+1 levels (where k is a power of 2)

- compute bayer kxk matrix
- compute thresholds for k^2+1 levels
- use threshold based on pixel location

bottom line

ordered dither can simulate 5 output levels when only 2 output levels are available

can we simulate more than 5 output levels?

what if we have more than 2 output levels to works with?

Ordered Dither: n levels, kxk neighborhood

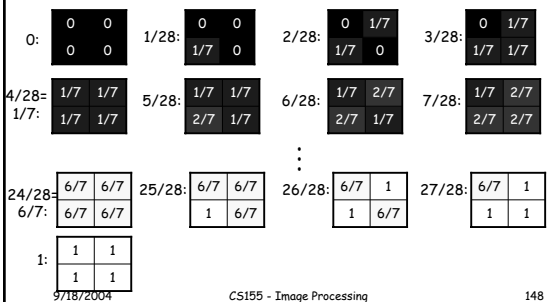
if we have n output levels and use kxk dithering neighborhoods we can simulate a total of $(n-1)*k^2+1$ output levels

Example n=8, k=2: $(8-1)*2^2+1=29$ simulated levels

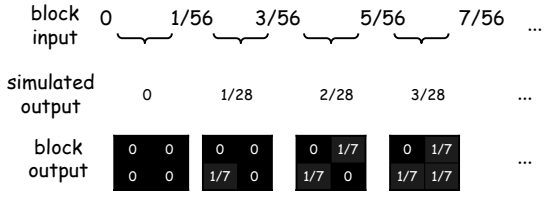
- 8 output levels: 0=0/7, 1/7, 2/7, ... , 6/7, 7/7=1
- simulated levels: 0=0/28, 1/28, ..., 27/28, 28/28=1

Example n=8, k=2: $(8-1)*2^2+1=29$ simulated levels

- We simulate the level by a corresponding 2x2 block:



Example n=8, k=2: Thresholds



Example n=8, k=2: Thresholds

Pixel Location 1 Thresholds:

1/56, 9/56, 17/56, 25/56, 33/56, 41/56, 49/56

Pixel Location j Thresholds:

j/56, (j+8)/56, (j+16)/56, (j+24)/56, (j+32)/56, (j+40)/56, (j+48)/56

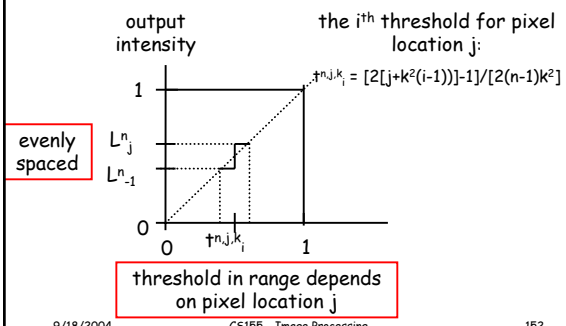
For general n, k: Thresholds

The ith threshold (i∈[1,n-1]) for pixel location j (j ∈[1,k²]) is

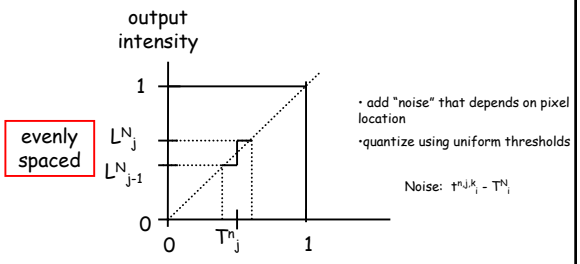
$[2[j+k^2(i-1)]-1]/[2(n-1)k^2]$

Ordered Dither: Quantization for position j

$(j+n(i-1))/[2(n-1)k^2]$.



Alternate View Ordered Dither



Alternate view

$t^{n,j,k}_i - T^n_i = [2[j+k^2(i-1))]-1]/[2(n-1)k^2] - (2i-1)/2(n-1)$

$= [2j-k^2-1]/[2(n-1)k^2]$

Note: This does not depend on i!

Algorithm: Add $[2j-k^2-1]/[2(n-1)k^2]$ noise to pixel in location j then use uniform quantization rule.

dithering

- random
- ordered
- error diffusion

error-diffusion dither

8 bits per channel per pixel | 1 bits per channel per pixel | 1 bits per channel per pixel dithered

comparison

original ordered error-diffusion

error-diffusion dither intuition

quantize I_{00} using uniform quantization

I_{00}	I_{01}	I_{02}
I_{10}	I_{11}	I_{12}

Q_{00}		

error-diffusion dither intuition

this introduces some error ... suppose Q_{00} is too dark

I_{00}	I_{01}	I_{02}
I_{10}	I_{11}	I_{12}

Q_{00}		

error-diffusion dither intuition

we can compensate by brightening the neighbors of I_{00}.

I_{00}	I_{01}	I_{02}
I_{10}	I_{11}	I_{12}

Q_{00}		

error-diffusion dither intuition

now continue quantization on modified image

I_{00}	I_{01}	I_{02}
I_{10}	I_{11}	I_{12}

Q_{00}		

now for the details

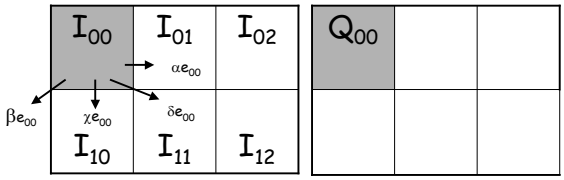
quantize I_{00} using uniform quantization

I_{00}	I_{01}	I_{02}
I_{10}	I_{11}	I_{12}

Q_{00}		

error diffusion dither

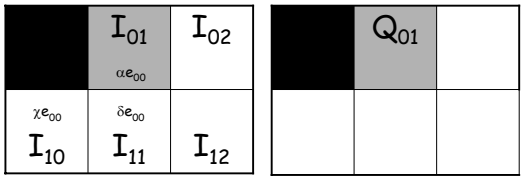
distribute error $e_{00}=I_{00} - Q_{00}$ to neighbors not yet quantized



$\alpha+\beta+\chi+\delta=1$

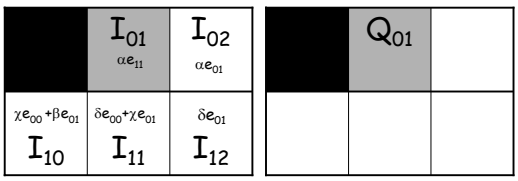
error diffusion dither

quantize $I_{01} + \alpha e_{00}$



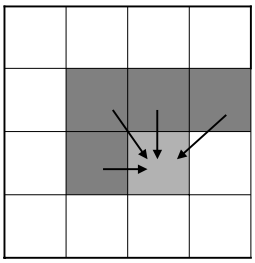
error diffusion dither

distribute error: $e_{01} = I_{01}+\alpha e_{00} - Q_{01}$



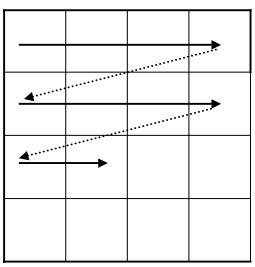
error diffusion dither

error contributions by upper & left neighbors



error diffusion dither

order of quantization is important



floyd-steinberg

$\alpha = 7/16$

$\beta = 3/16$

$\chi = 5/16$

$\delta = 1/16$