type of techniques

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

random dither

add noise to camouflage quantization artifacts


8 bits per pixel per channel


1 bits per pixel per channel

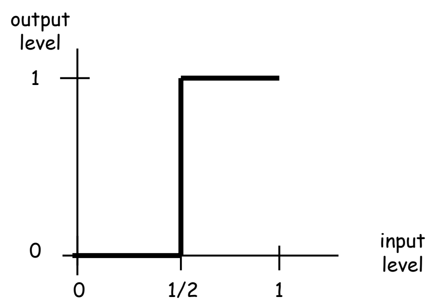

1 bits per pixel per channel noisy

quantization algorithm

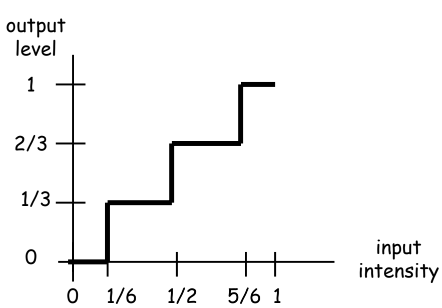

8 bits per pixel per channel


1 bits per pixel per channel

uniform quantization: 1 channel, 1 bit per pixel = 2 levels



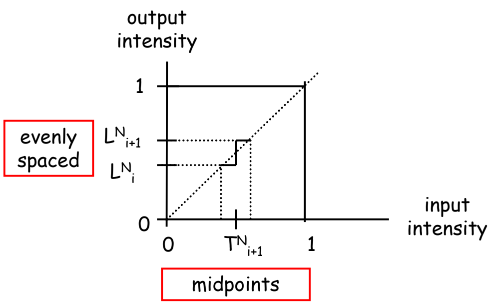
uniform quantization: 1 channel, 2 bits per pixel = 4 level



uniform quantization: n level

- output levels:
  evenly spaced
- thresholds:
  midpoints

n level uniform quantization



uniform quantization: n level

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

quantization error

8 bits per channel per pixel

1 bits per channel per pixel

dithering

- technique for camouflaging error
- algorithms
  - random
  - ordered
  - error diffusion

comparison

original
8 bits/pixel/
channel

random

ordered

error-diffusion

1 bit/pixel/channel

random dither

add noise to camouflage quantization artifacts

8 bits/pixel/channel

1 bits/pixel/channel

1 bits/pixel/channel
dithered

random dither

for each pixel in the input image
- add random noise to pixel value
- uniformly quantize new value

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

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

quantized blocks

every neighborhood is quantized in one of two ways

0	0
0	0

1	1
1	1

ordered dither: intuition

suppose all 2x2 neighborhoods were uniform

we could quantize neighborhoods together

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

neighborhood blocks

0	0
0	0

0	0
1	0

0	1
1	0

0	1
1	1

1	1
1	1

intuition

0.5	0.5
0.5	0.5

→

0	1
1	0

try to preserve average intensity over neighborhoods

more intuition

0.25	0.25
0.25	0.25

→

0	0
1	0

0.75	0.75
0.75	0.75

→

0	1
1	1

average intensity simulates 5 output 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

threshold-midpoints

0 **1/8** 1/4 **3/8** 1/2

5/8 .75 **7/8** 1

quantization

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

→

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

but we don't have uniform neighborhoods...

0.32	0.57
0.97	0.21

→ ?

or do we?

image coherence

ordered dither intuition

D	B
A	C

choose thresholds for each neighborhood position that preserve, as well as possible, average intensity over uniform neighborhoods

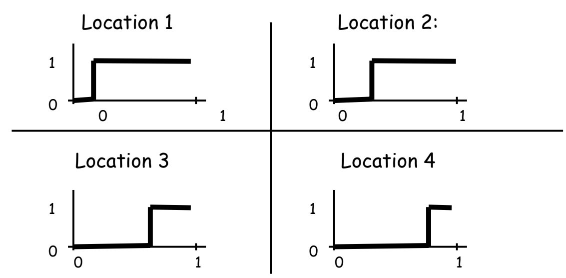
ordered dither: step 1

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

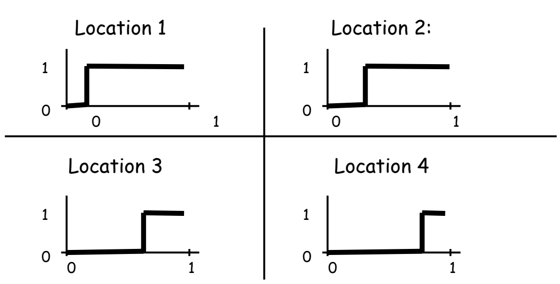
ordered dither: step 2

quantize based on threshold for pixel location



what are thresholds?

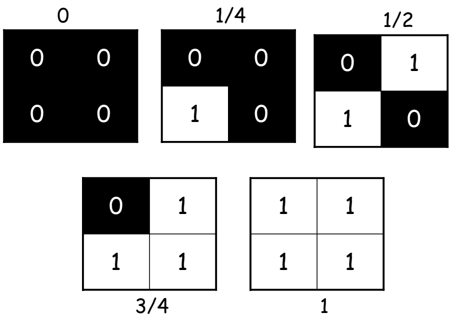
quantize based on threshold for pixel location



intuitively

- output levels:
  evenly spaced
- thresholds:
  midpoints

simulated output levels

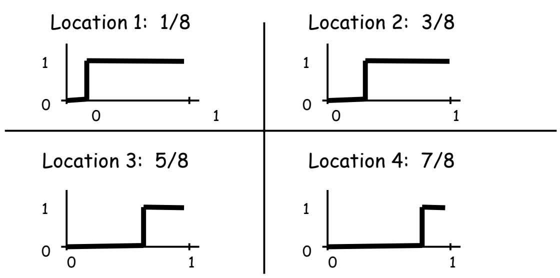


threshold-midpoints

0 **1/8** 1/4 **3/8** 1/2

5/8 .75 **7/8** 1

thresholds



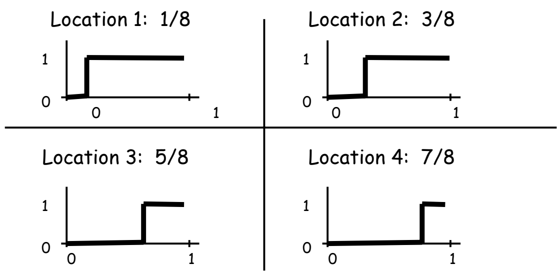
ordered dither recap: step 1

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



generalizations

- **simulate more output levels**
- use more bits/pixel/channel

ordered dither (1 bit/pixel/channel)

- to simulate $M=m^2+1$ levels
  - assign pixel locations in mXm neighborhoods to m^2 classes
  - use T^M_i as threshold to quantize pixel in i^{th} class

we just did m=2, M=5

m=4, M=17

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

Bayer's ordered 4x4

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

bayer's 2x2 ordered dither

4	2
1	3

Bayer's ordered 4x4 (recursive)

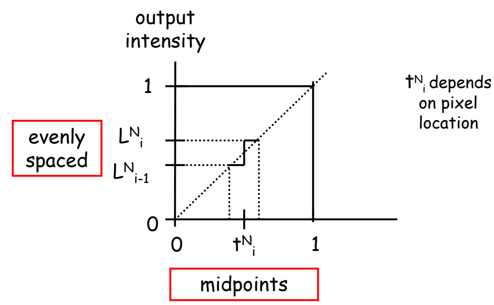
16	8	14	6
4	12	2	10
13	5	15	7
1	9	3	11

have to be careful to avoid creating artifacts

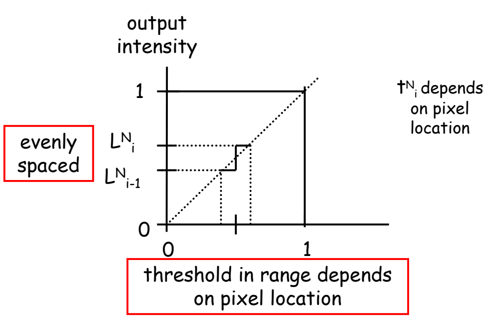
generalizations

- simulate more output levels
- **use more bits/pixel/channel**

Uniform Quantization: n bits=2^n levels



Ordered Dither: n bits=2^n levels



error-diffusion dither

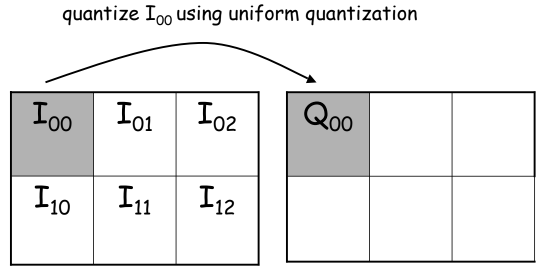
8 bits per channel per pixel | 1 bits per channel per pixel | 1 bits per channel per pixel dithered

comparison

original | ordered | error-diffusion

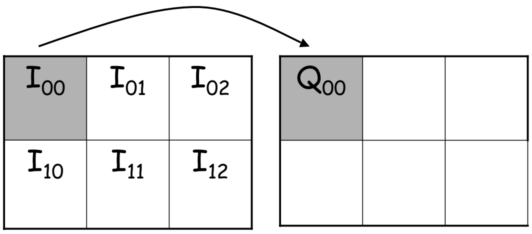
error-diffusion dither intuition

quantize I_{00} using uniform quantization



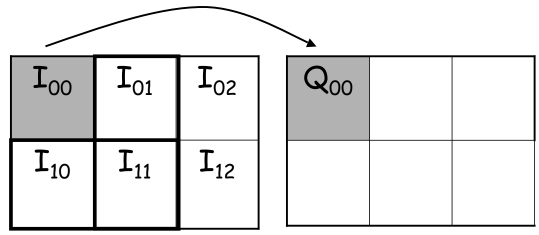
error-diffusion dither intuition

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



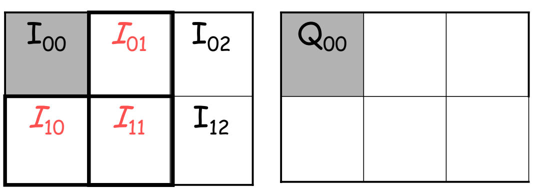
error-diffusion dither intuition

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



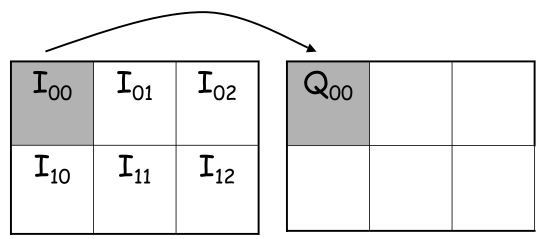
error-diffusion dither intuition

now continue quatization on modified image



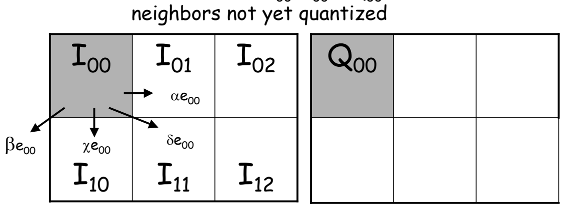
now for the details

quantize I_{00} using uniform quantization



error diffusion dither

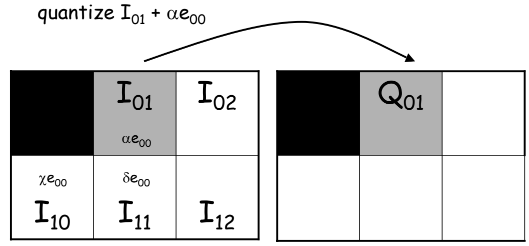
distribute error $e_{00}=I_{00} - Q_{00}$ to neighbors not yet quantized



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

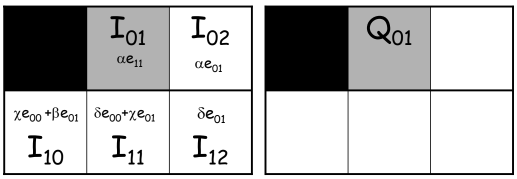
error diffusion dither

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



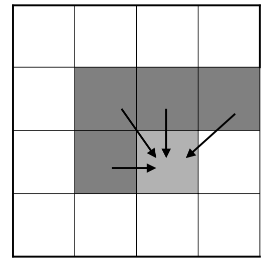
error diffusion dither

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



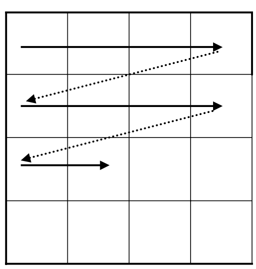
error diffusion dither

error contributions by upper & left neighbors



error diffusion dither

order of quantization is important



floyd-steinberg

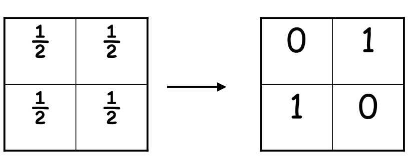
$\alpha = 7/16$

$\beta = 3/16$

$\chi = 5/16$

$\delta = 1/16$

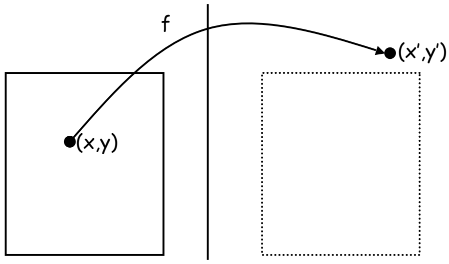
floyd-steinberg: example



types of techniques

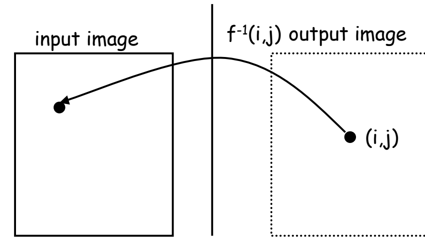
- simple pixel modification
- interpolation/extrapolation
- compositing
- convolution
- dithering
- **warping**
- morphing
- misc. effects

forward warp



f maps points in input image to the plane

forward warp



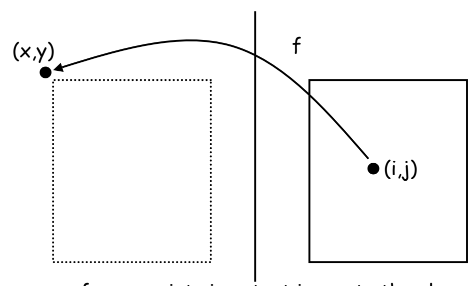
pixel at (i,j) in output image is assigned the value at location $f^{-1}(i,j)$ in input image

forward warp: problems

if f is not bijective

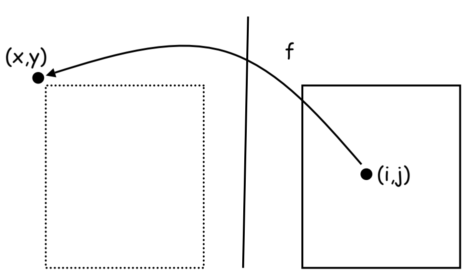
1. $f^{-1}(i,j)$ may not be defined
2. $f^{-1}(i,j)$ may not be unique

backward warp



f maps points in output image to the plane

backward warp



pixel (i,j) in output image is assigned value of input image at location f(i,j)

backward warp: problems

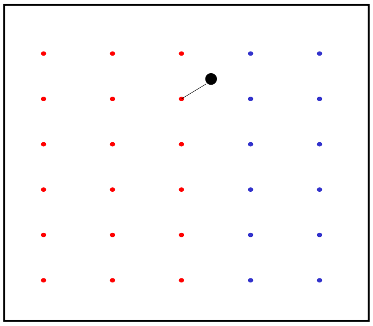
1. f(i,j) may lie outside the input image area

   solution: give image an infinite, black (or other default) border

2. f(i,j) may not lie on a sample of the input image

   solution: resample input

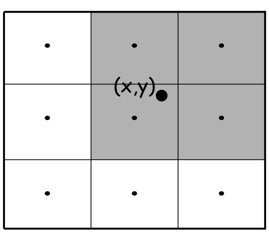
re-sample: estimate input image at arbitrary location



re-sample

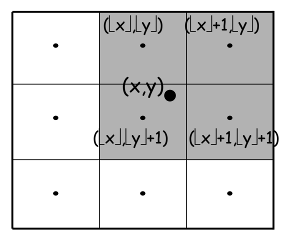
interpolate based on nearby samples
- nearest
- bilinear
- bicubic
- gaussian

which way is up?



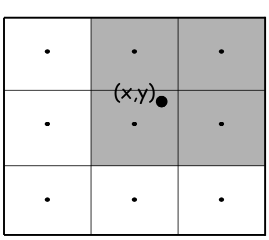
what are the coordinates of the pixels surrounding (x,y)?

which way is up?



what are the coordinates of the pixels surrounding (x,y)?

nearest

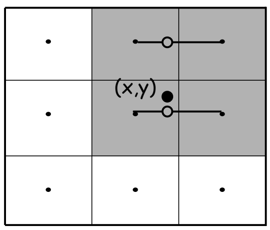


- compute distance between x,y and the locations of the neighboring samples
- set value at x,y to the value of the closest neighbor

re-sample

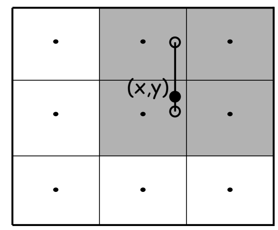
interpolate based on nearby samples
- nearest
- **bilinear**
- bicubic
- gaussian

bilinear interpolation



1. interpolate to find values at $(x,\lfloor y \rfloor)$ and $(x,\lfloor y \rfloor+1)$

bilinear interpolation



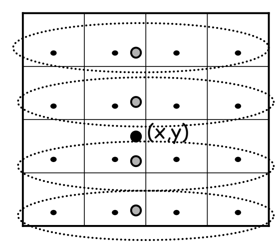
1. interpolate to find values at$(x,\lfloor y \rfloor)$ and $(x,\lfloor y \rfloor+1)$
2. interpolate to find value at x,y

re-sample

interpolate based on nearby samples

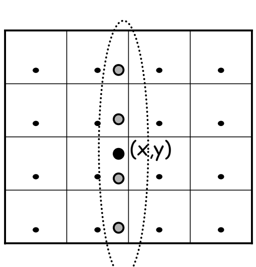
- nearest
- bilinear
- **bicubic**
- gaussian

bicubic



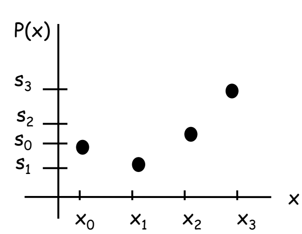
1. interpolate to find values at $(x,\lfloor y \rfloor+i)$

bicubic



1. interpolate to find values at $(x,\lfloor y \rfloor+i)$
2. interpolate to find value at (x,y)

bicubic: lagrangian



there is a unique cubic polynomial through any four distinct sample point

lagrange cubic polynomial

$$P(x) = \Sigma_{i=0,1,2,3}\, s_i \, \Pi_{j=0,1,2,3, j\neq i}\, (x-x_j)/(x_i-x_j)$$

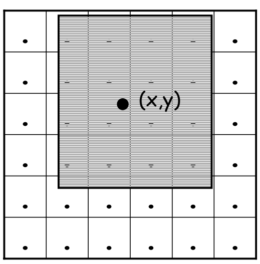
exercise: what is the value of $P(x_i)$ i=0,1,2,3

re-sample

interpolate based on nearby samples

- nearest
- bilinear
- bicubic
- **gaussian**

gaussian



interpolate nearby samples using normalized gaussian weights

unnormalized weight at (i,j) in window is

$\exp[-((x-i)^2+(y-j)^2)/\sigma^2]$