Image warping/morphing

Digital Visual Effects, Spring 2005

Yung-Yu Chuang

2005/3/9

with slides by Richard Szeliski, Steve Seitz and Alexei Efros

Announcements

DigiVFX

- Class time: 1:30-4:20 (with a 20-minute break)
- Last call: send cyy@csie.ntu.edu.tw to subscribe vfx
- Course forum is set up (see course page)
- Scribe volunteers for today and next week
- A schedule for scribes will be posted in forum soon. Please fill in the schedule.

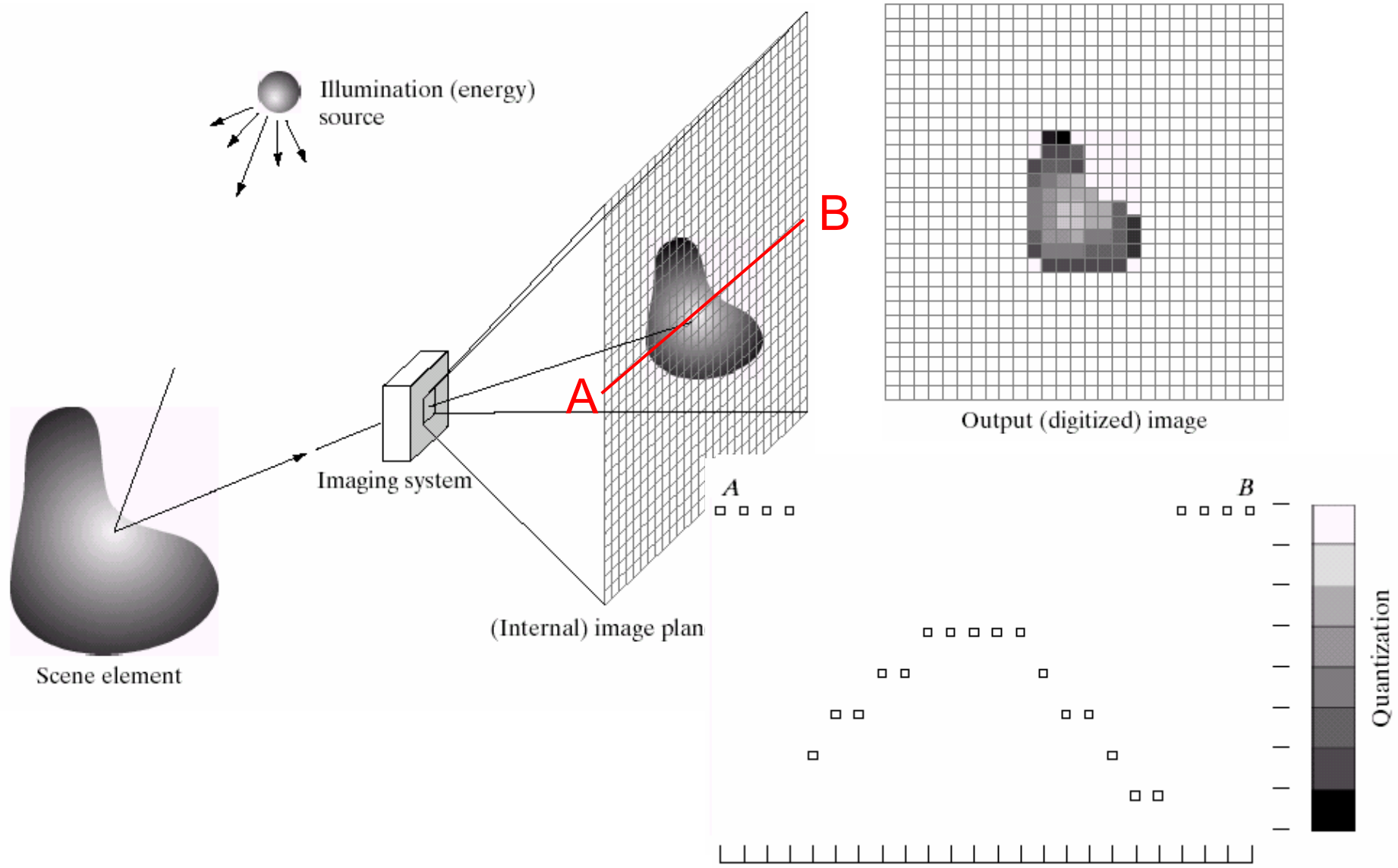
Outline

DigiVFX

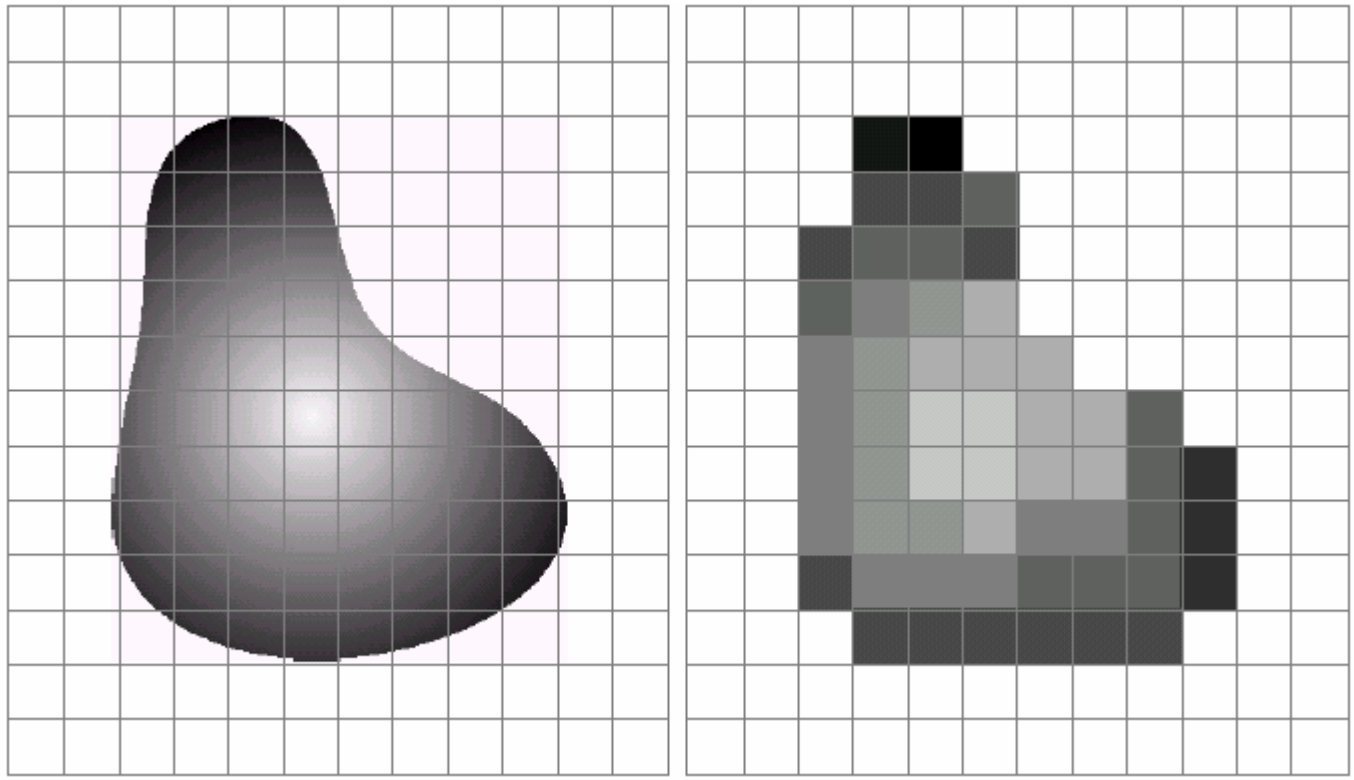
- Images
- Image warping
- Image morphing
- Project #1

Image fundamentals

Image formation

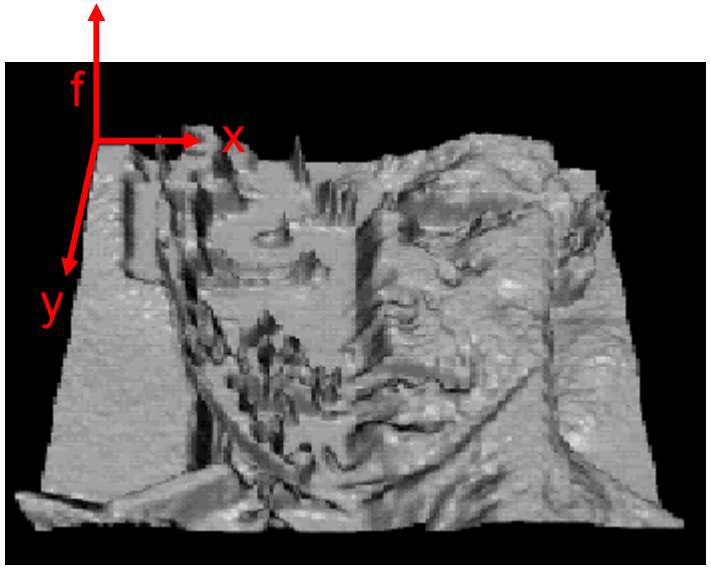


Sampling and quantization



What is an image

- We can think of an **image** as a function, f: $R^2 \rightarrow R$:
  - $f(x, y)$ gives the **intensity** at position (x, y)
  - defined over a rectangle, with a finite range:
    - f: $[a,b]x[c,d] \rightarrow [0,1]$



- A color image

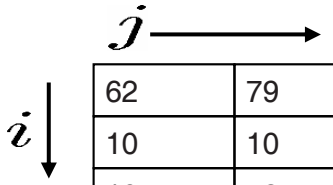
$$f(x, y) = \begin{bmatrix} r(x, y) \\ g(x, y) \\ b(x, y) \end{bmatrix}$$

A digital image

- We usually operate on **digital** (**discrete**) images:
  - **Sample** the 2D space on a regular grid
  - **Quantize** each sample (round to nearest integer)
- If our samples are D apart, we can write this as:

$$f[i, j] = \text{Quantize}\{ f(i\,D, j\,D) \}$$

- The image can now be represented as a matrix of integer values

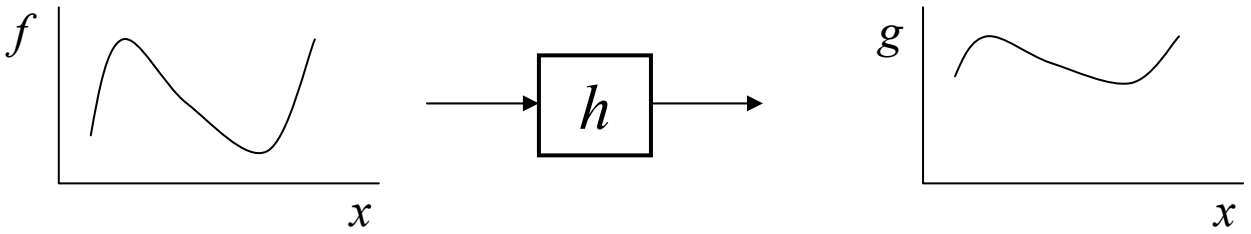


62	79	23	119	120	105	4	0
10	10	9	62	12	78	34	0
10	58	197	46	46	0	0	48
176	135	5	188	191	68	0	49
2	1	1	29	26	37	0	77
0	89	144	147	187	102	62	208
255	252	0	166	123	62	0	31
166	63	127	17	1	0	99	30

Aliasing

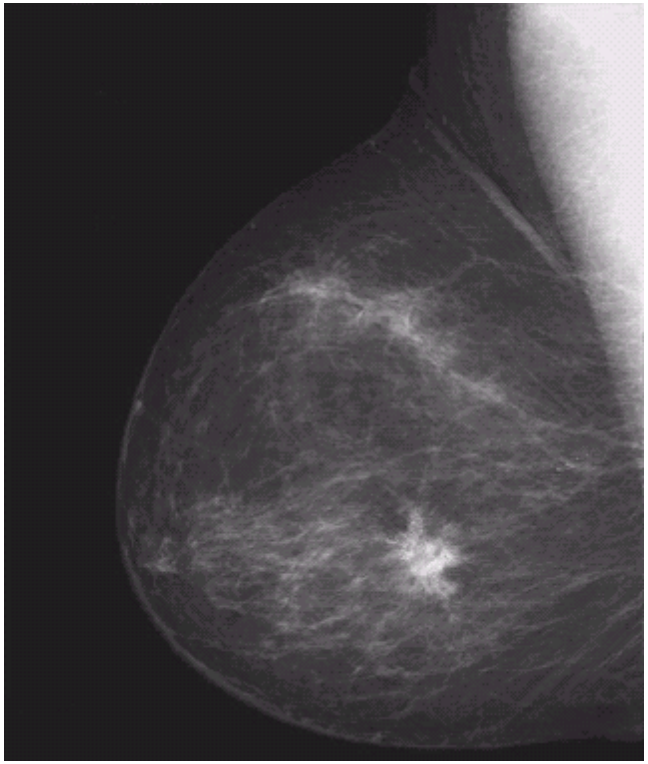
DigiVFX

Image processing

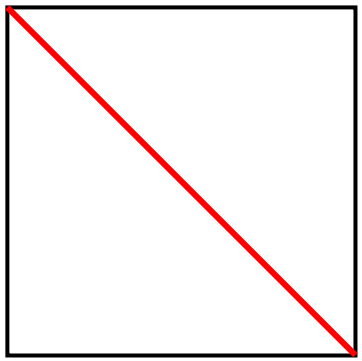


$$g(x) = h(f(x))$$

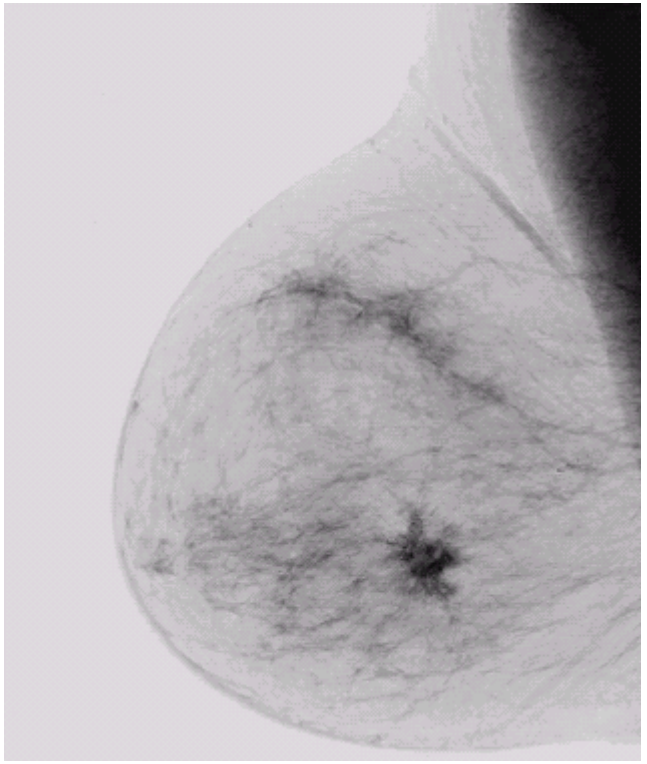
Point processing



f



h(a)=1-a
negative

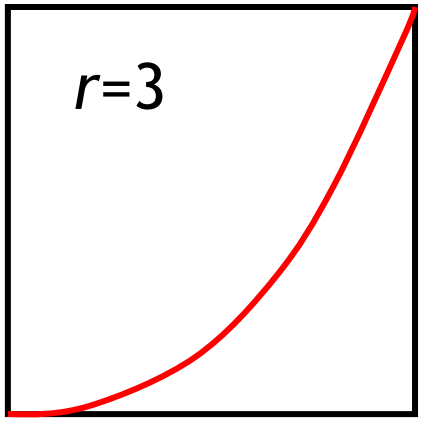


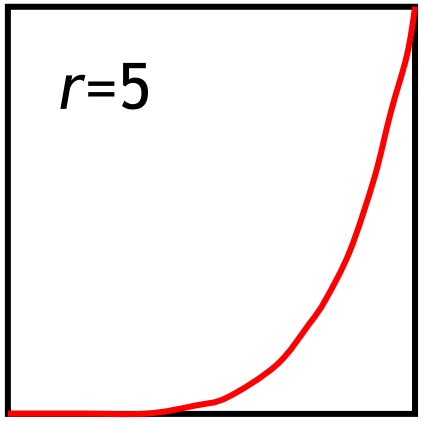
g

Image enhancement

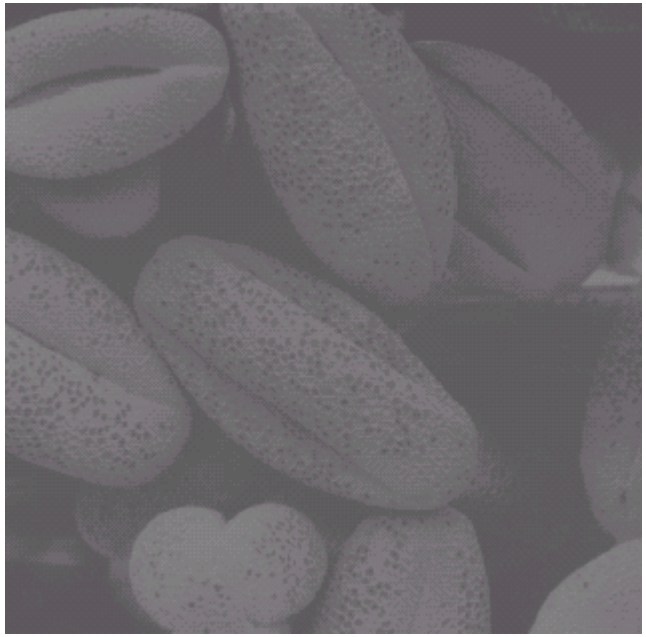
gamma mapping

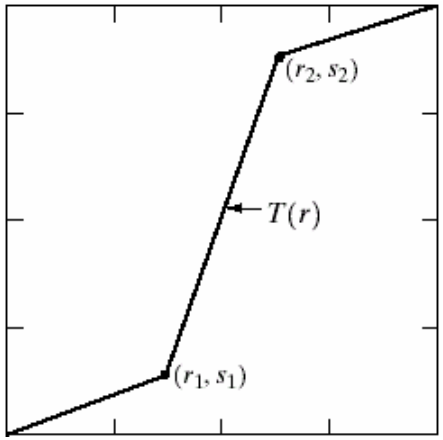
$h(a)=a^r$

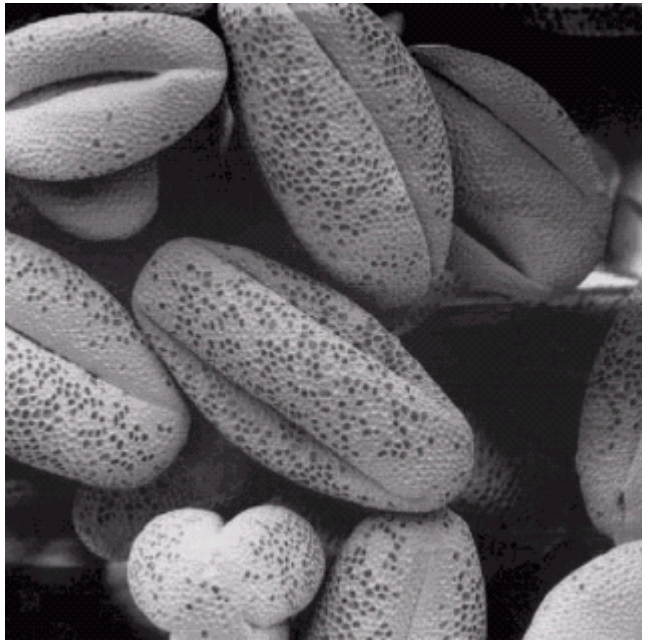




Contrast stretching

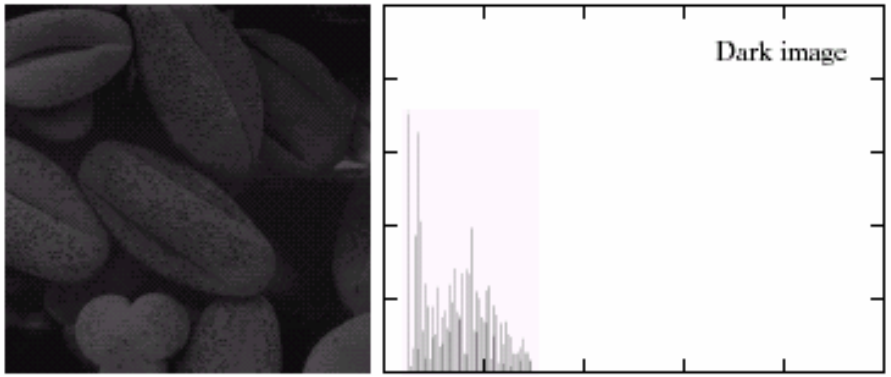




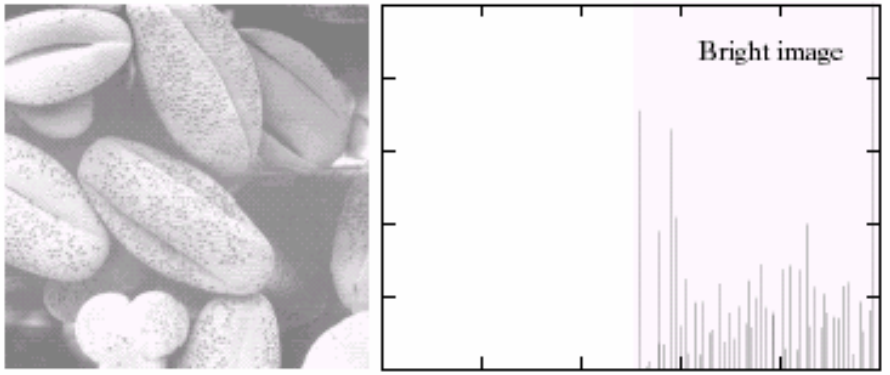


Histogram

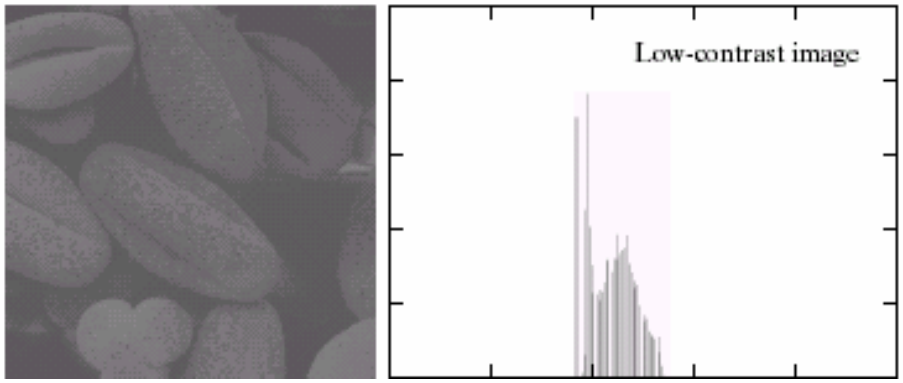
(1)



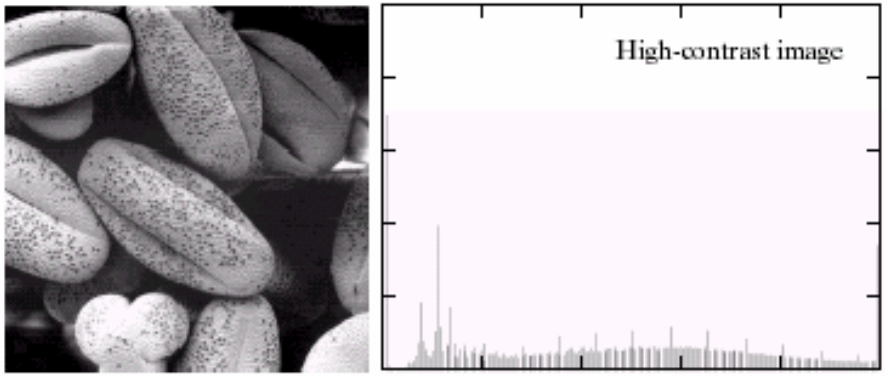
(2)



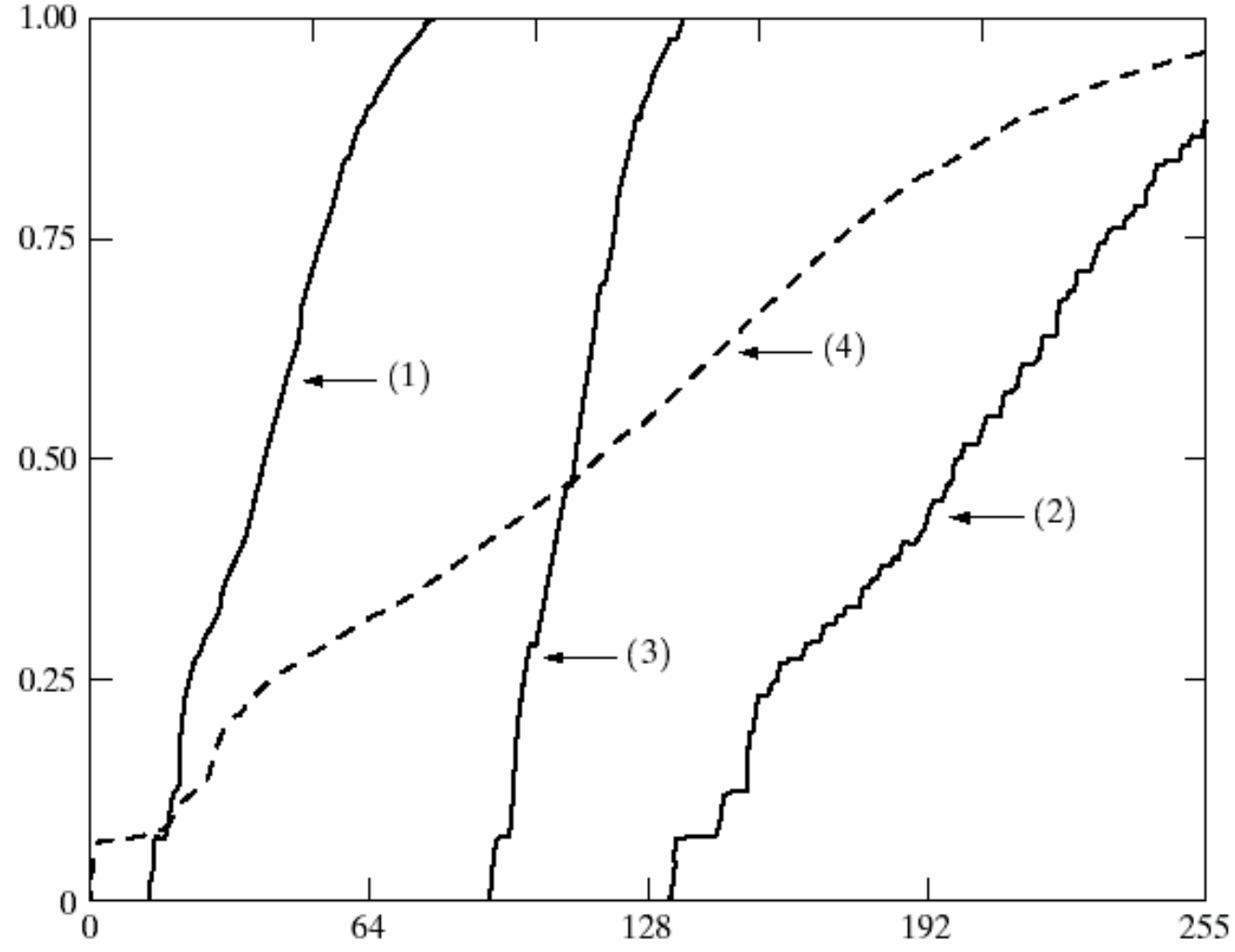
(3)



(4)



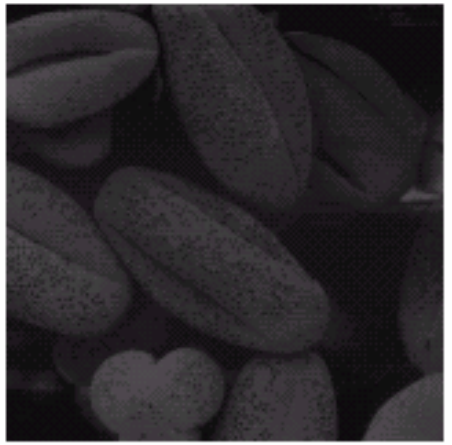
Accumulated histogram



Histogram equalization

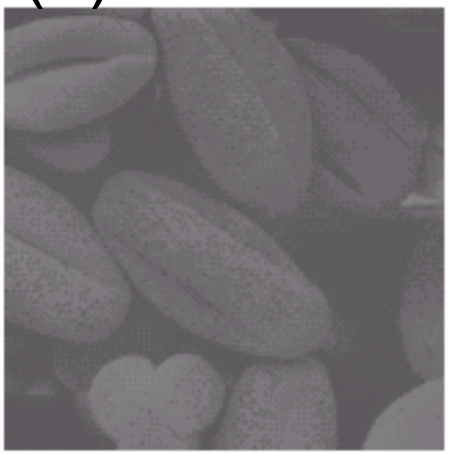
DigiVFX

(1)

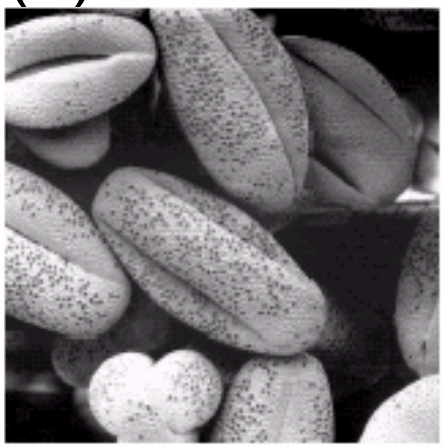


(2)

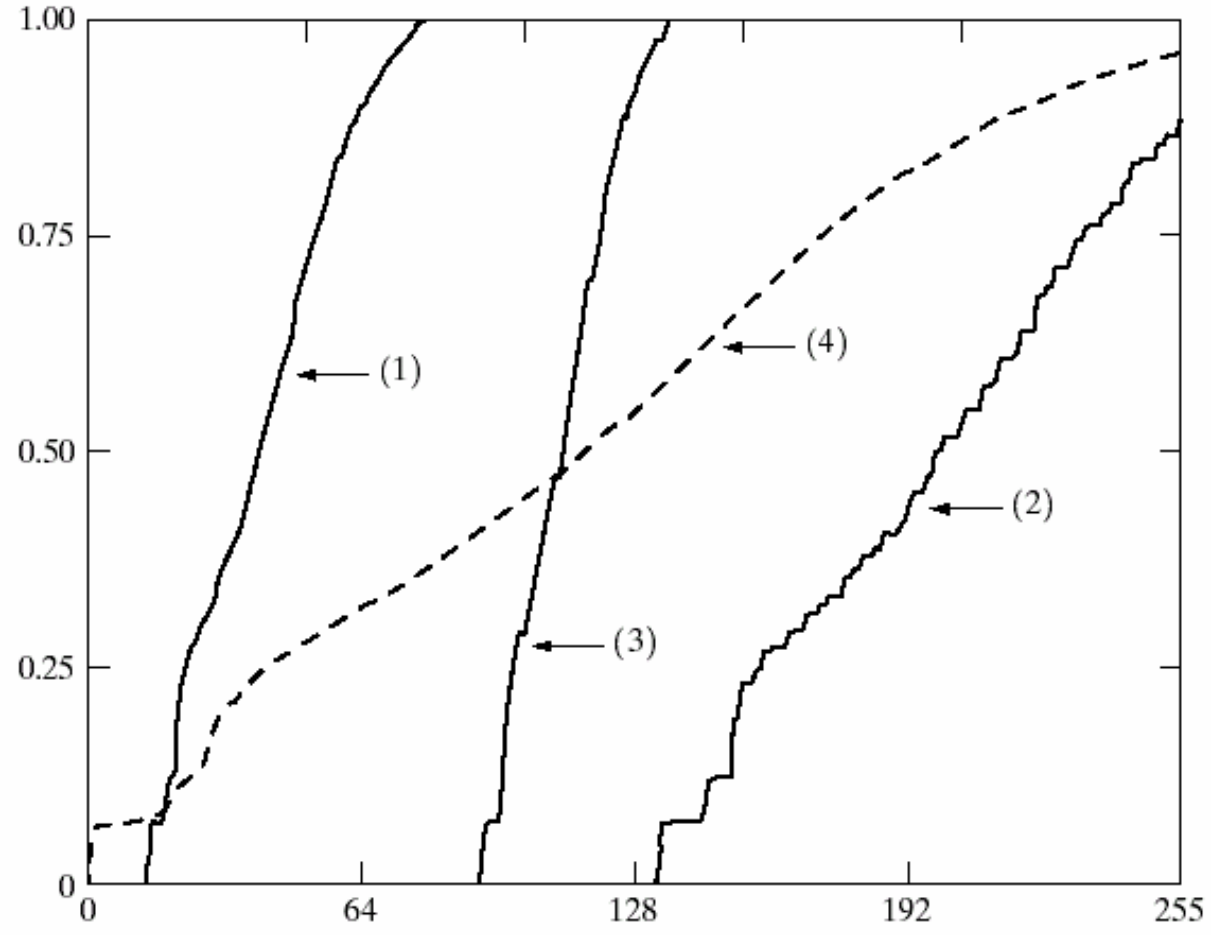
(3)



(4)



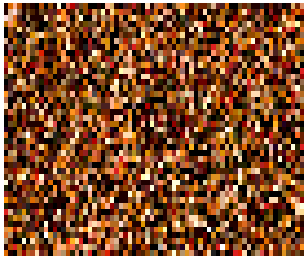
Histogram matching



It is useful for calibrating exposure.

Neighborhood Processing (filtering)

- Q: What happens if I reshuffle all pixels within the image?



- A: It's histogram won't change. No point processing will be affected…

- Need spatial information to capture this.

Noise

Original

Salt and pepper noise

Impulse noise

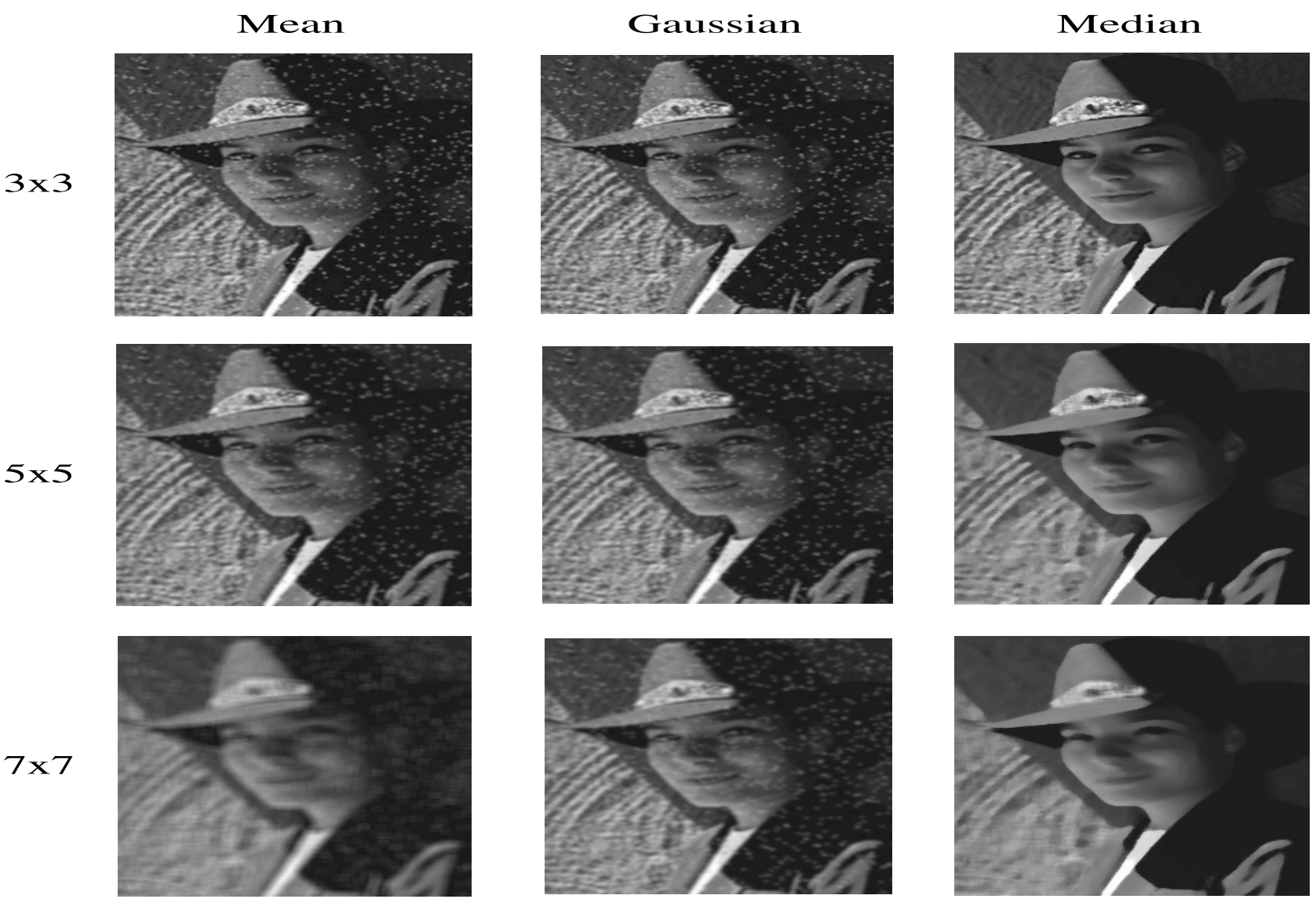
Gaussian noise

Noise reduction

- Mean filter
- Median filter
- Gaussian filter

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	100	130	110	120	110	0	0
0	0	0	110	90	100	90	100	0	0
0	0	0	130	100	90	130	110	0	0
0	0	0	120	100	130	110	120	0	0
0	0	0	90	110	80	120	100	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Comparison: salt and pepper noise



Comparison: Gaussian noise

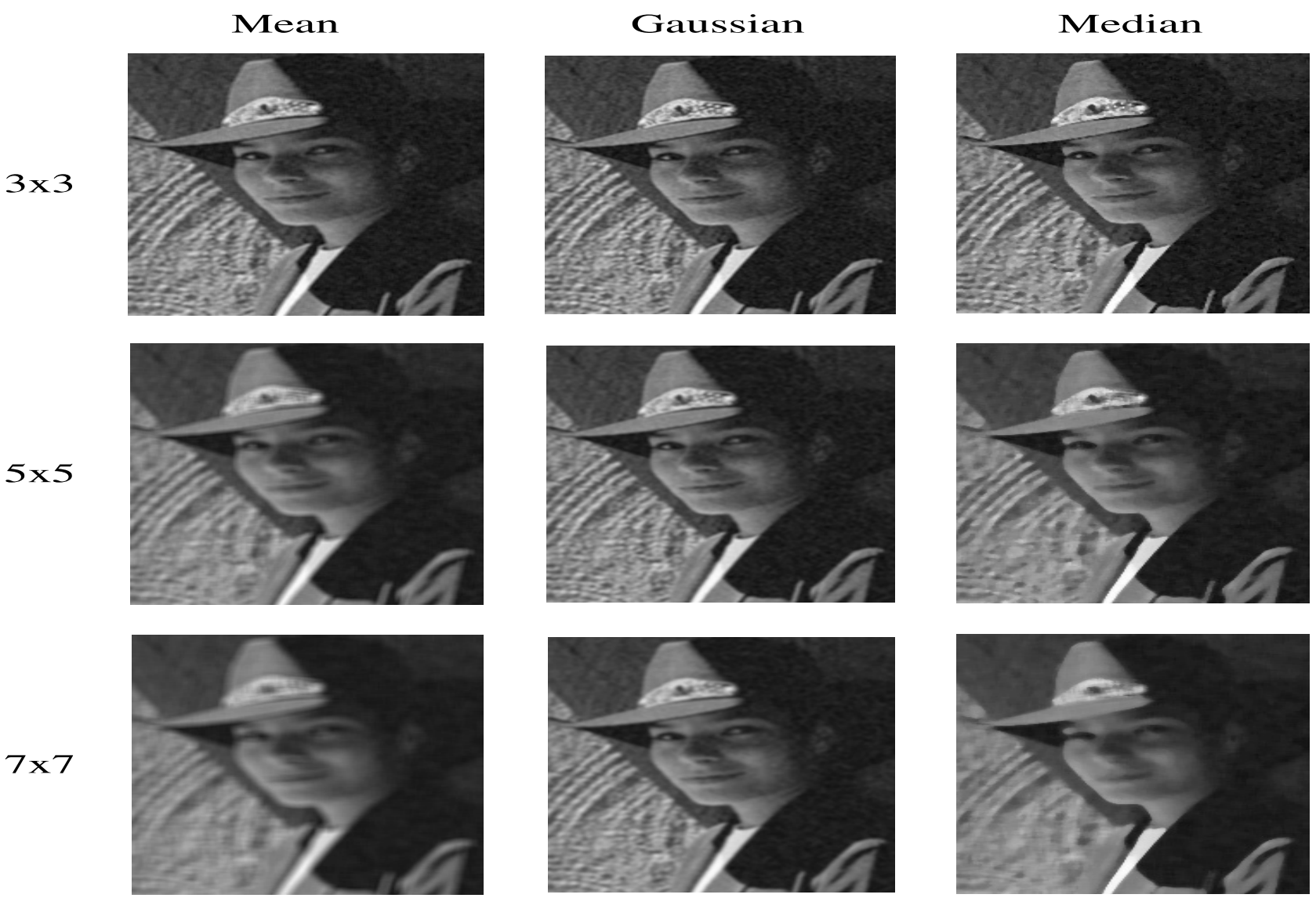


Image warping

Image warping

image filtering: change ***range*** of image

$$g(x) = h(f(x))$$

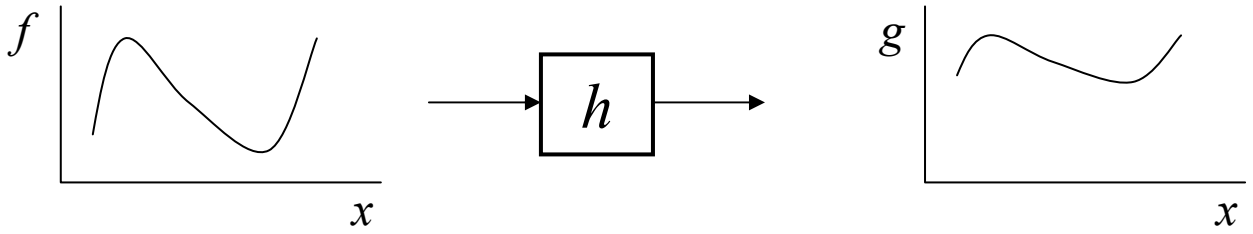


image warping: change ***domain*** of image

$$g(x) = f(h(x))$$

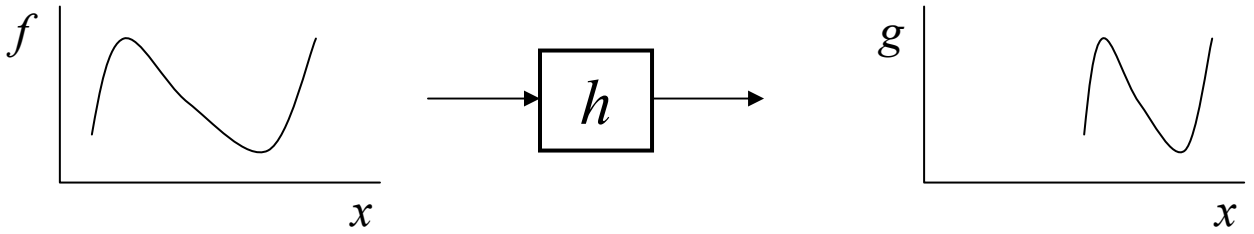


image filtering: change ***range*** of image

$$f(x) = h(g(x))$$

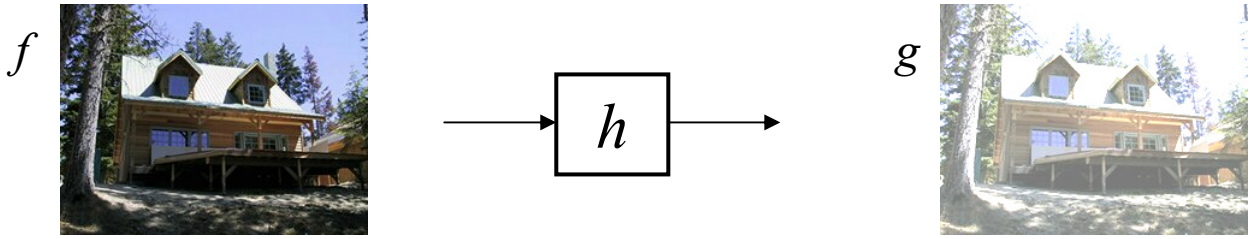
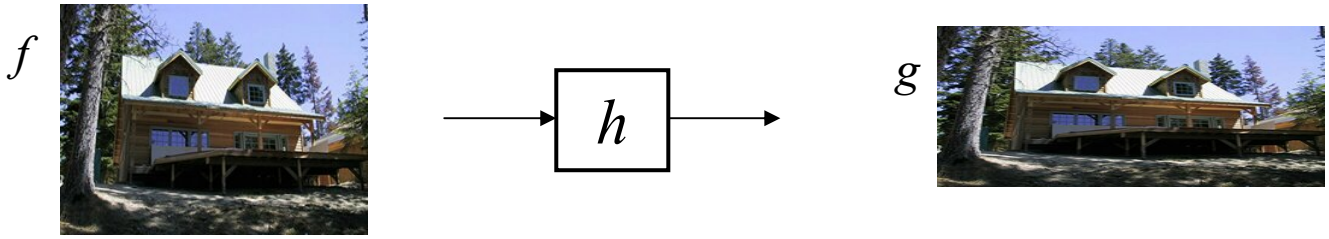


image warping: change ***domain*** of image

$$f(x) = g(h(x))$$



Parametric (global) warping

DigiVFX

Examples of parametric warps:

translation

rotation

aspect

affine

perspective

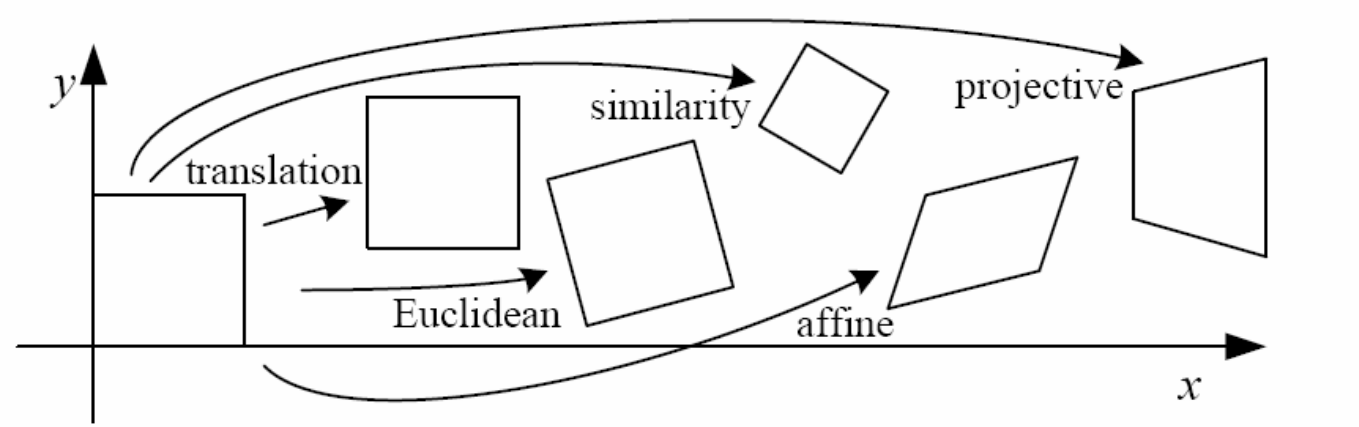
cylindrical

2D coordinate transformations

DigiVFX

- translation: $\boldsymbol{x}' = \boldsymbol{x} + \boldsymbol{t}$ $\boldsymbol{x} = (x,y)$
- rotation: $\boldsymbol{x}' = \boldsymbol{R}\,\boldsymbol{x} + \boldsymbol{t}$
- similarity: $\boldsymbol{x}' = s\,\boldsymbol{R}\,\boldsymbol{x} + \boldsymbol{t}$
- affine: $\boldsymbol{x}' = \boldsymbol{A}\,\boldsymbol{x} + \boldsymbol{t}$
- perspective: $\underline{\boldsymbol{x}}' \cong \boldsymbol{H}\,\underline{\boldsymbol{x}}$ $\underline{\boldsymbol{x}} = (x,y,1)$
  ($\underline{\boldsymbol{x}}$ is a *homogeneous* coordinate)
- These all form a nested *group* (closed under composition w/ inv.)

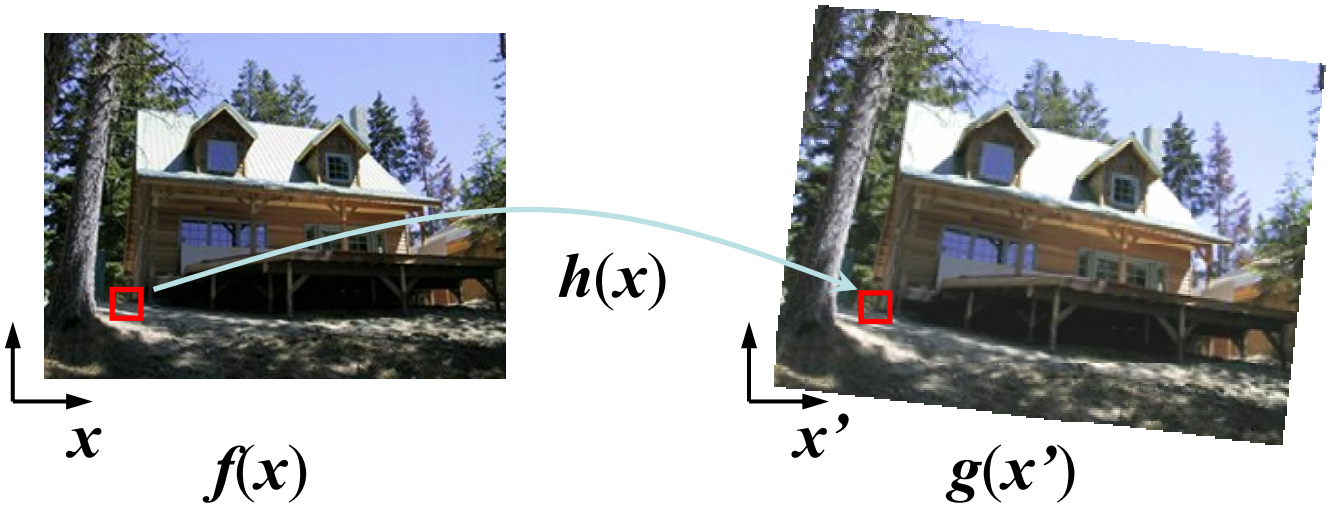
2D image transformations



Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\left[\begin{array}{c\|c} \boldsymbol{I} & \boldsymbol{t} \end{array}\right]_{2\times 3}$	2	orientation $+\cdots$	
rigid (Euclidean)	$\left[\begin{array}{c\|c} \boldsymbol{R} & \boldsymbol{t} \end{array}\right]_{2\times 3}$	3	lengths $+\cdots$	
similarity	$\left[\begin{array}{c\|c} s\boldsymbol{R} & \boldsymbol{t} \end{array}\right]_{2\times 3}$	4	angles $+\cdots$	
affine	$\left[\begin{array}{c} \boldsymbol{A} \end{array}\right]_{2\times 3}$	6	parallelism $+\cdots$	
projective	$\left[\begin{array}{c} \tilde{\boldsymbol{H}} \end{array}\right]_{3\times 3}$	8	straight lines	

Image warping

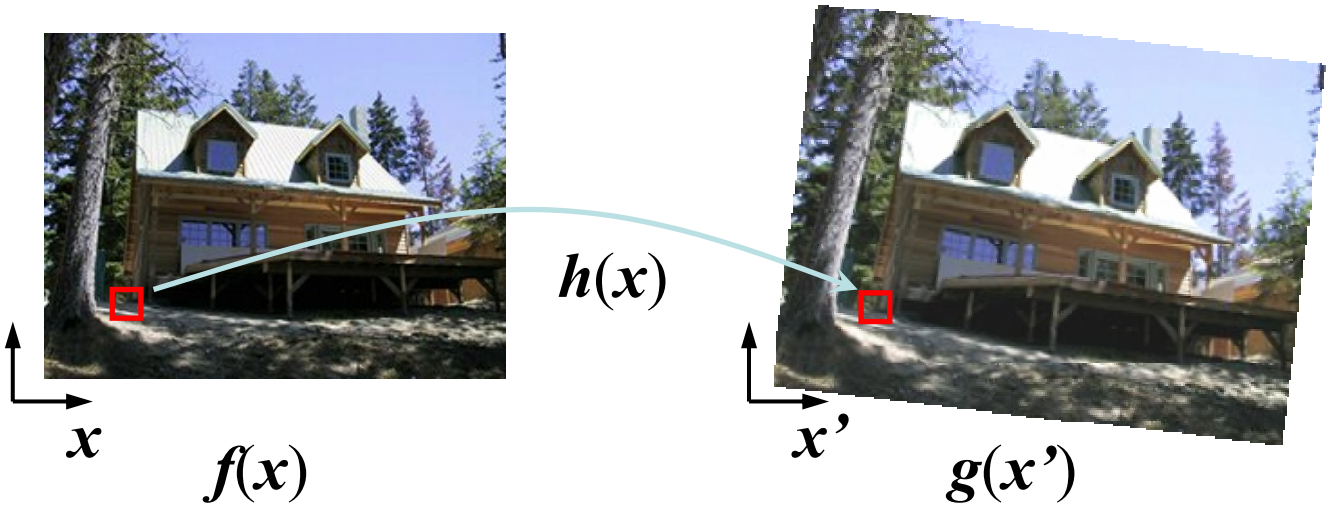
- Given a coordinate transform $x' = h(x)$ and a source image $f(x)$, how do we compute a transformed image $g(x') = f(h(x))$?



Forward warping

- Send each pixel $f(x)$ to its corresponding location $x' = h(x)$ in $g(x')$

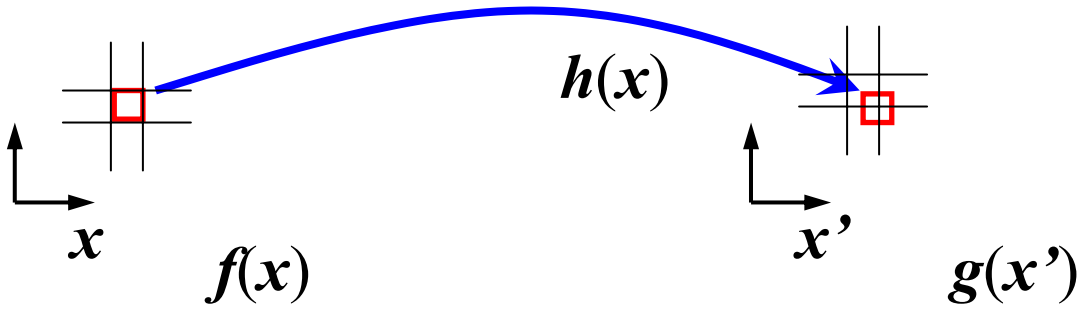
  - What if pixel lands "between" two pixels?



Forward warping

- Send each pixel $f(x)$ to its corresponding location $x' = h(x)$ in $g(x')$

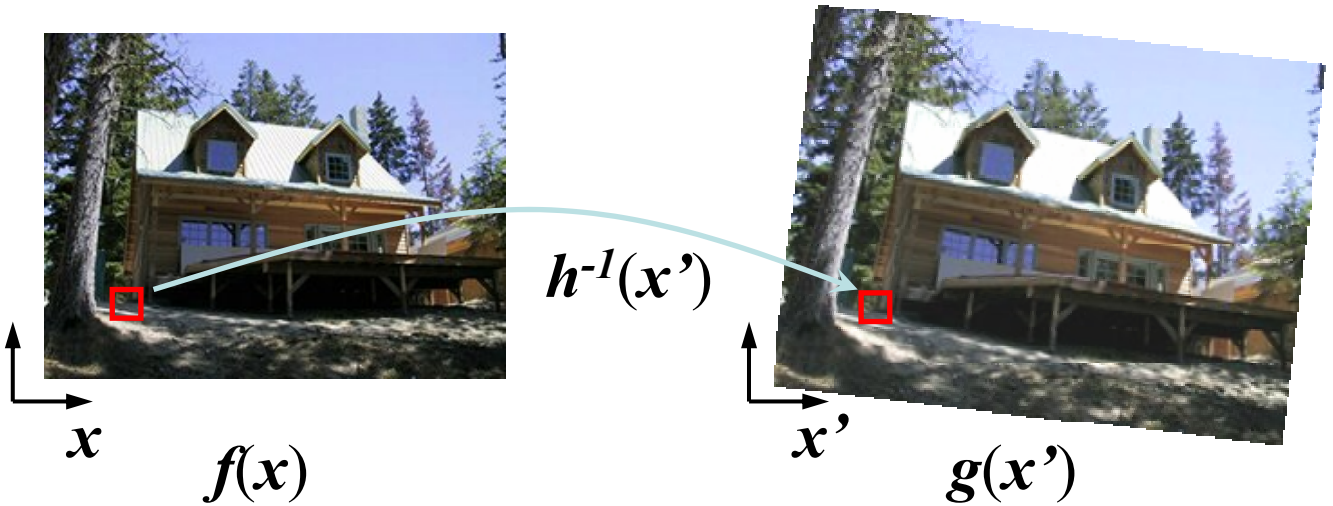
  - What if pixel lands "between" two pixels?
  - Answer: add "contribution" to several pixels, normalize later (*splatting*)



Inverse warping

- Get each pixel $g(x')$ from its corresponding location $x = h^{-1}(x')$ in $f(x)$

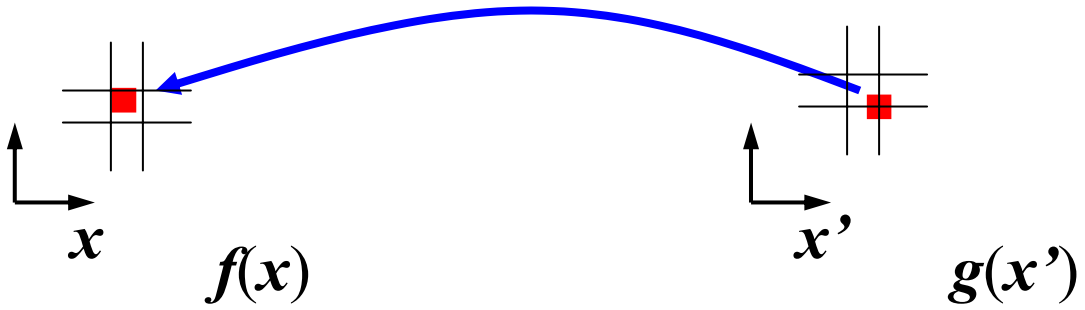
  - What if pixel comes from "between" two pixels?



Inverse warping

- Get each pixel $g(x')$ from its corresponding location $x = h^{-1}(x')$ in $f(x)$

  - What if pixel comes from "between" two pixels?
  - Answer: *resample* color value from *interpolated* (*prefiltered*) source image



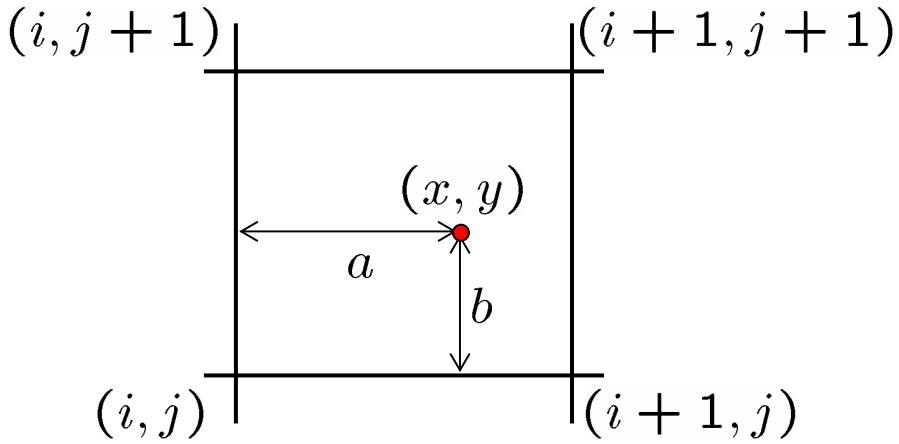
Interpolation

- Possible interpolation filters:
  - nearest neighbor
  - bilinear
  - bicubic
  - sinc / FIR

Bilinear interpolation

DigiVFX

- A simple method for resampling images



$$
\begin{aligned}
f(x,y) = \quad & (1-a)(1-b) \quad f[i,j] \\
& +a(1-b) \quad f[i+1,j] \\
& +ab \quad f[i+1,j+1] \\
& +(1-a)b \quad f[i,j+1]
\end{aligned}
$$

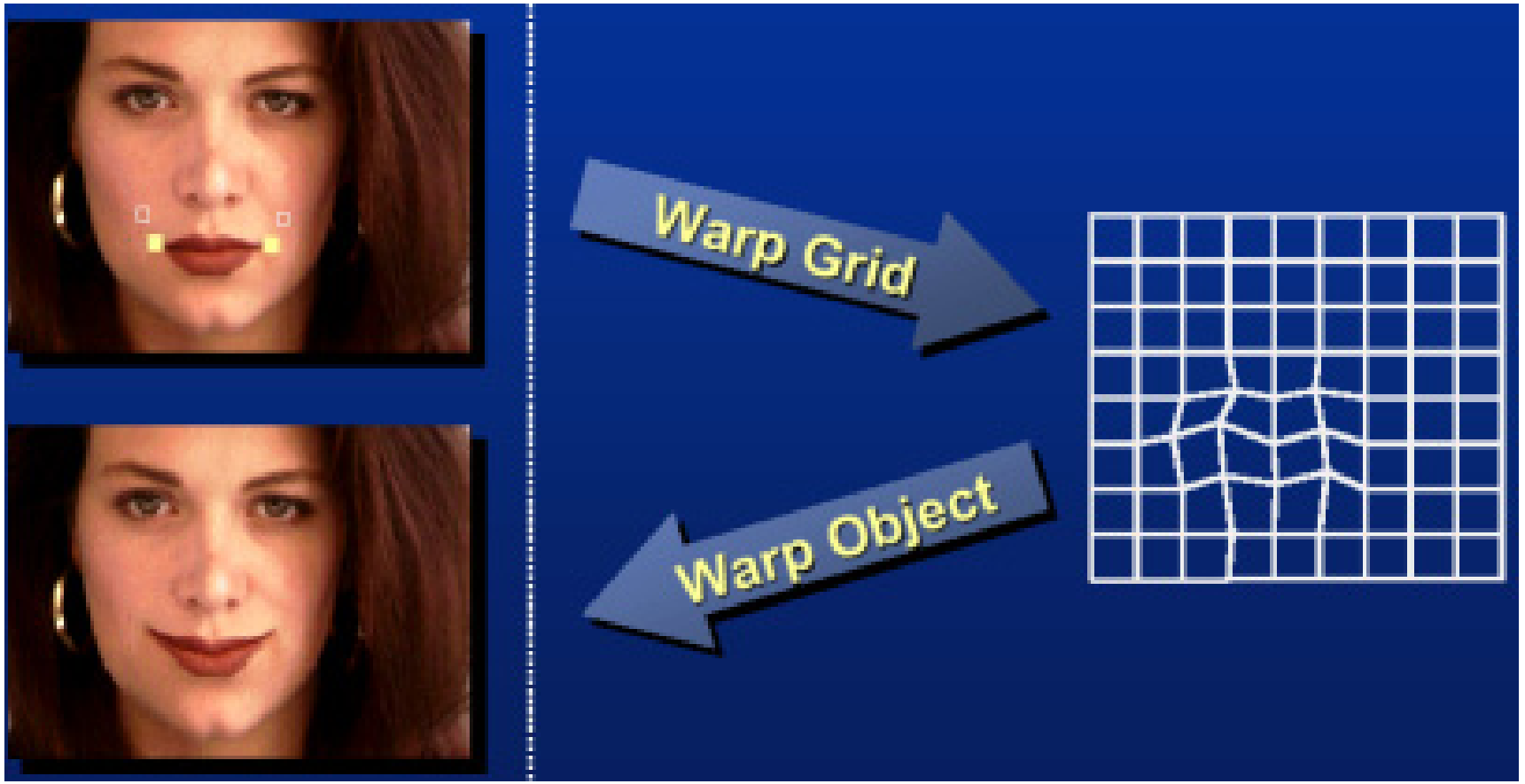
Bicubic interpolation

DigiVFX

http://astronomy.swin.edu.au/~pbourke/colour/bicubic/

Non-parametric image warping

- Specify a more detailed warp function
- Splines, meshes, optical flow (per-pixel motion)



Demo

DigiVFX

- http://www.colonize.com/warp/
- Warping is a useful operation for mosaics, video matching, view interpolation and so on.

Image morphing

Image morphing

- The goal is to synthesize a fluid transformation from one image to another.
- Cross dissolving is a common transition between cuts, but it is not good for morphing because of the ghosting effects.

image #1

dissolving

image #2

Image morphing

DigiVFX

- Why ghosting?
- Morphing = warping + cross-dissolving

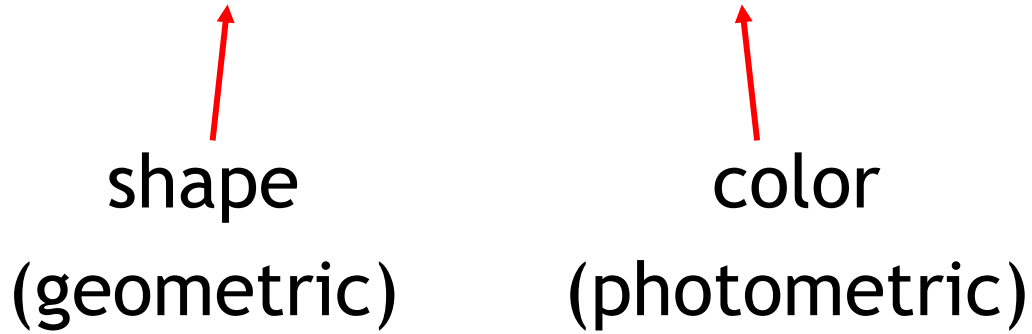
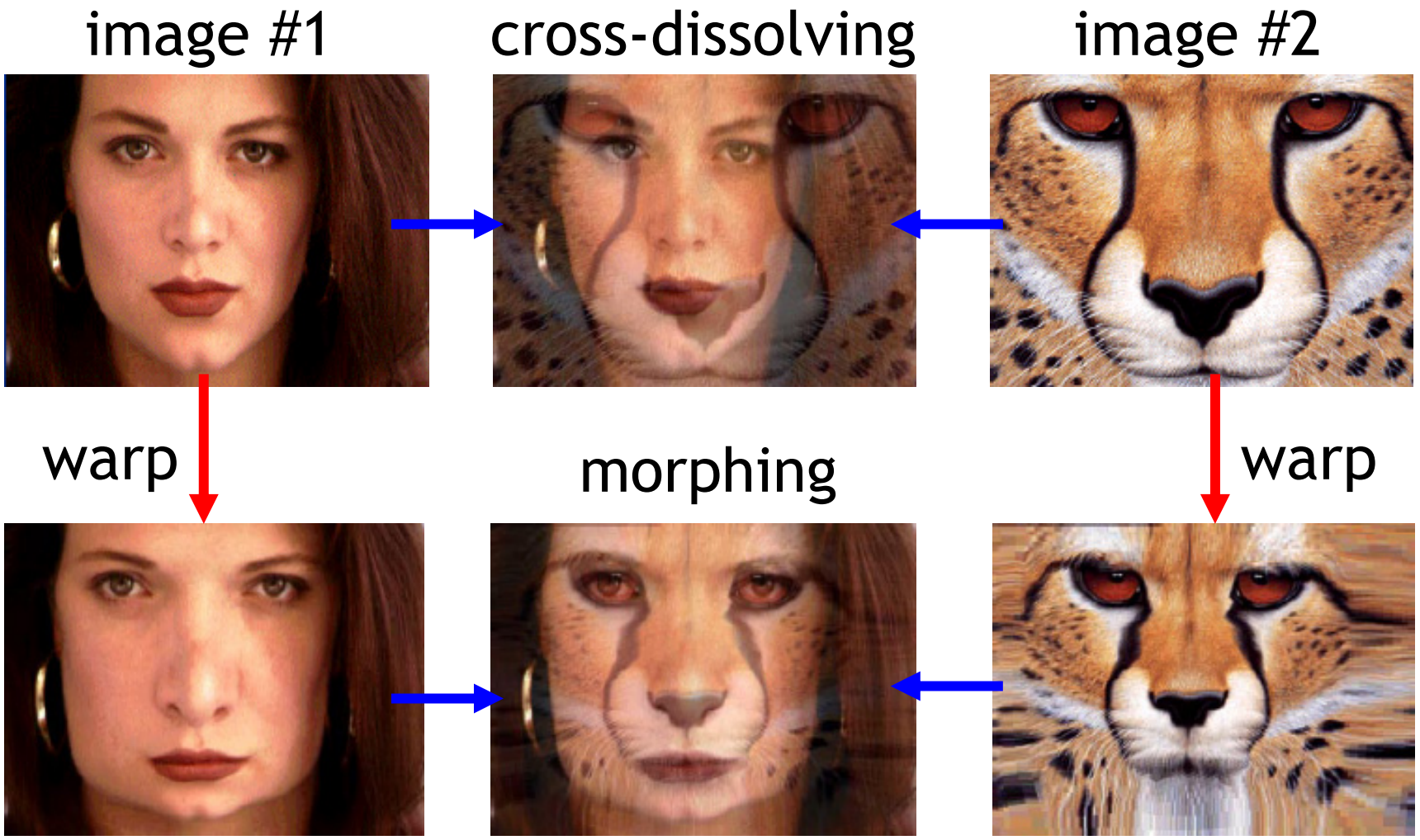


Image morphing



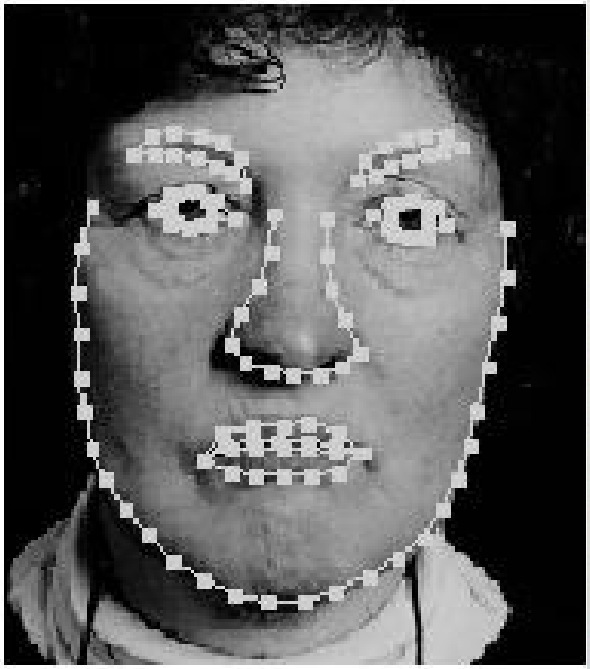
Morphing sequence

DigiVFX

Artifacts of cross-dissolving

http://www.salavon.com/

Face averaging by morphing



average faces

Image morphing

DigiVFX

create a morphing sequence: for each time t

1. Create an intermediate warping field (by interpolation)
2. Warp both images towards it
3. Cross-dissolve the colors in the newly warped images

An ideal example

t=0

morphing

t=1

An ideal example

t=0

middle face (t=0.5)

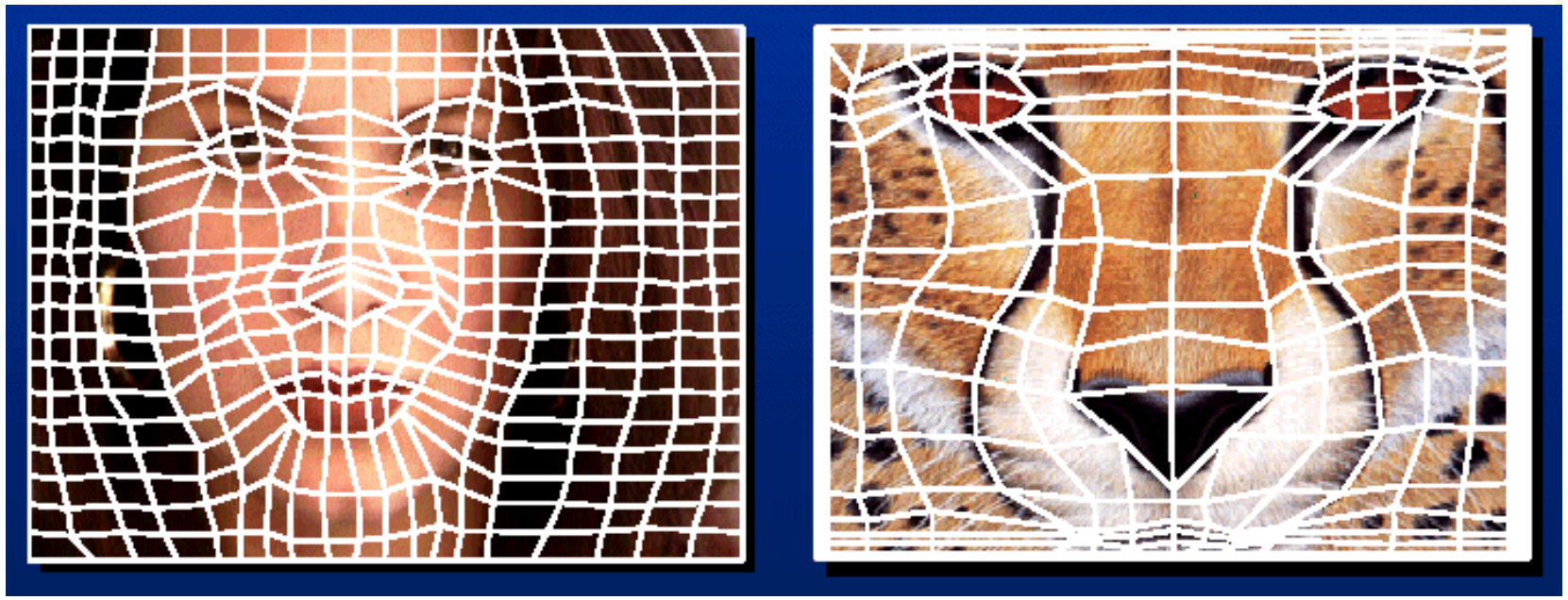
t=1

Warp specification (mesh warping)

DigiVFX

- How can we specify the warp?
  1. Specify corresponding *spline control points*

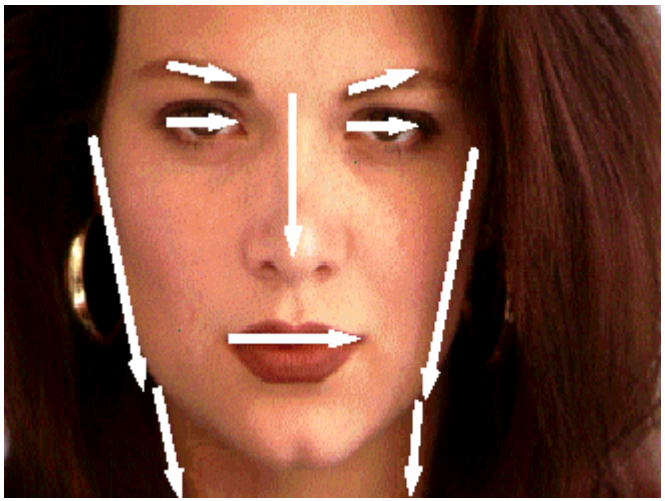
     interpolate to a complete warping function

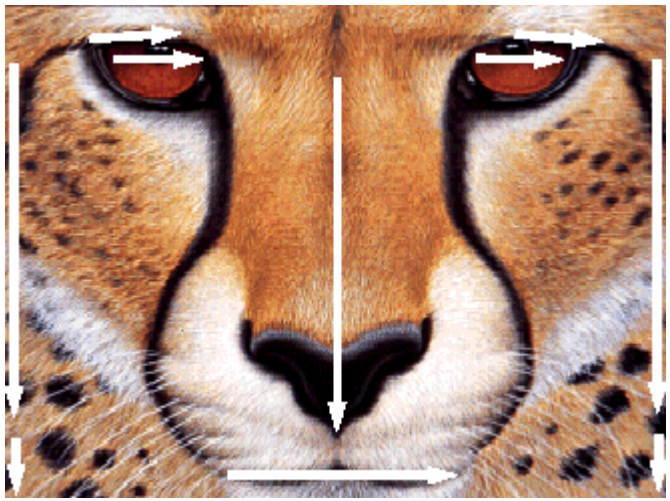


easy to implement, but less expressive

Warp specification (field warping)

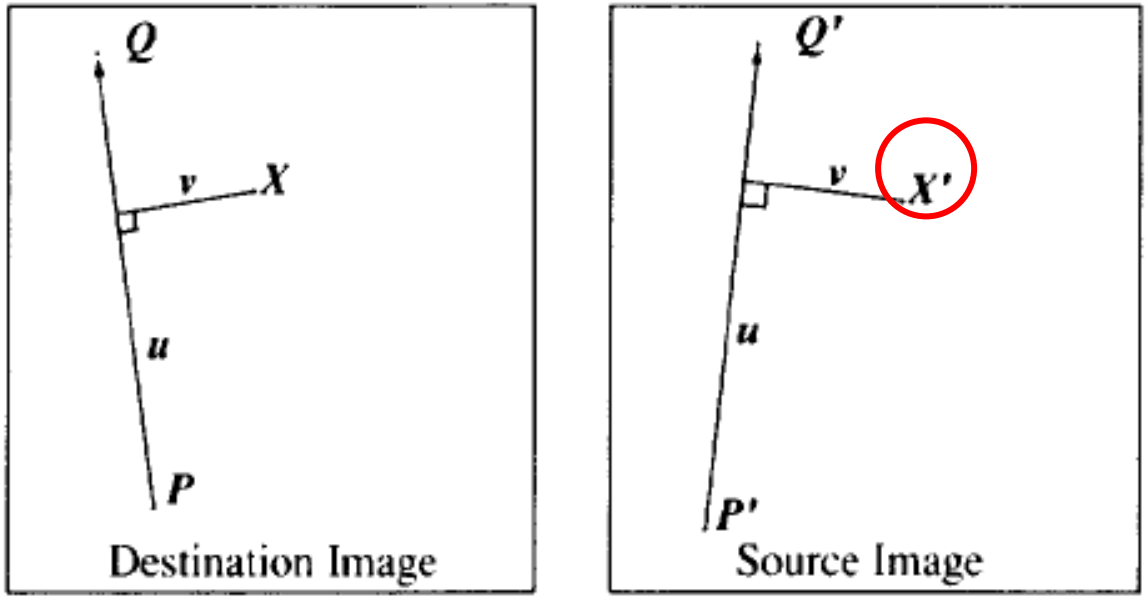
- How can we specify the warp?
  2. Specify corresponding *vectors*
     - *interpolate* to a complete warping function
     - The Beier & Neely Algorithm





Beier&Neely (SIGGRAPH 1992)

- Single line-pair PQ to P'Q':



$$u = \frac{(X-P)\cdot(Q-P)}{\|Q-P\|^2} \quad (1)$$

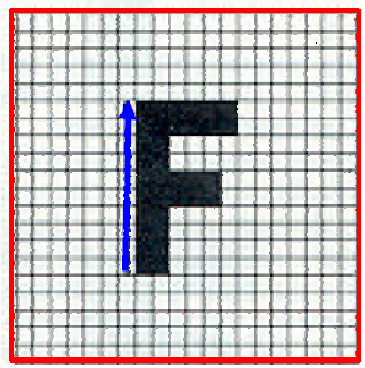
$$v = \frac{(X-P)\cdot Perpendicular(Q-P)}{\|Q-P\|} \quad (2)$$

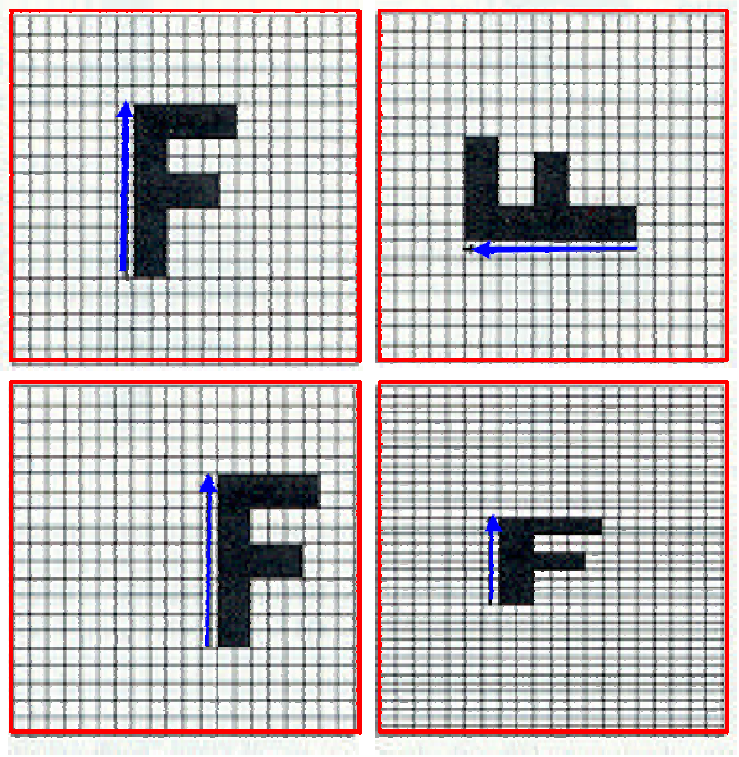
$$X' = P' + u\cdot(Q'-P') + \frac{v\cdot Perpendicular(Q'-P')}{\|Q'-P'\|} \quad (3)$$

Algorithm (single line-pair)

DigiVFX

- For each X in the destination image:
  1. Find the corresponding u,v
  2. Find X’ in the source image for that u,v
  3. destinationImage(X) = sourceImage(X’)
- Examples:

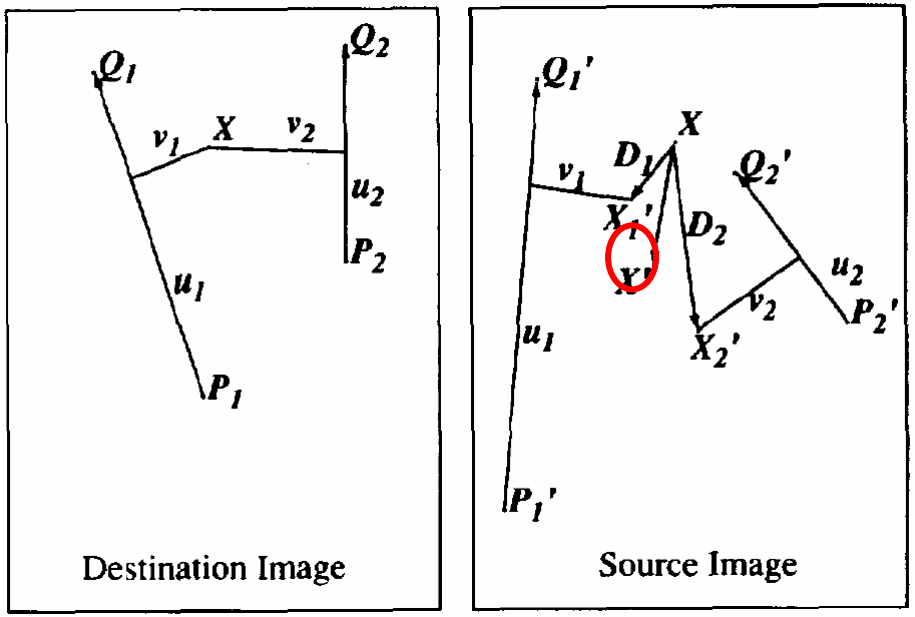




Affine transformation

Multiple Lines

$$D_i = X_i' - X_i$$



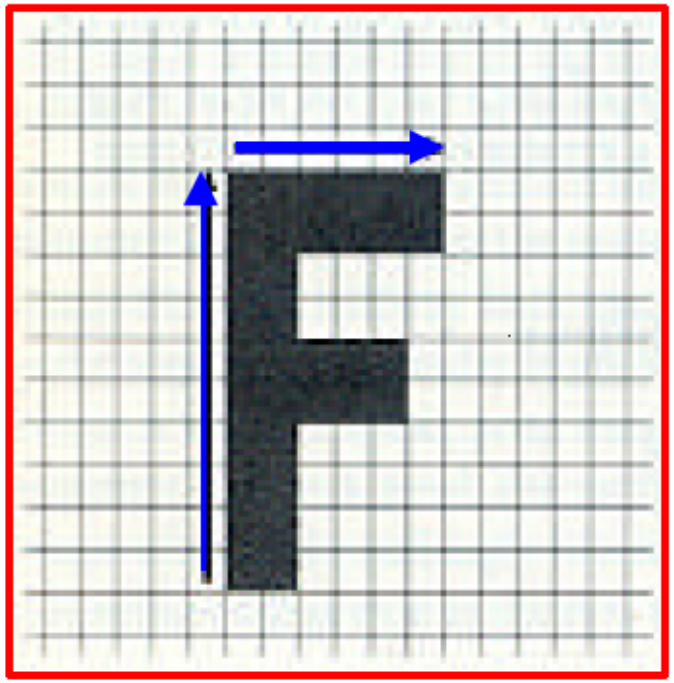
$$weight = \left(\frac{length^p}{(a + dist)} \right)^b$$

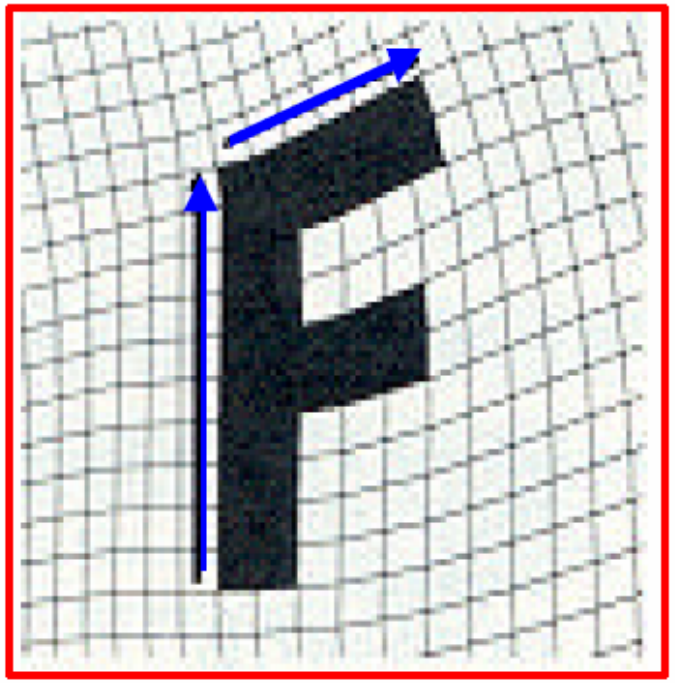
length = length of the line segment,
dist = distance to line segment
The influence of a, p, b. The same as the average of X_i'

Full Algorithm

For each pixel X in the destination
    $DSUM = (0,0)$
    $weightsum = 0$
    For each line $P_i Q_i$
        calculate u,v based on $P_i Q_i$
        calculate X'_i based on u,v and $P_i' Q_i'$
        calculate displacement $D_i = X_i' - X_i$ for this line
        $dist$ = shortest distance from X to $P_i Q_i$
        $weight = (length^p / (a + dist))^b$
        $DSUM \mathrel{+}= D_i * weight$
        $weightsum \mathrel{+}= weight$
    $X' = X + DSUM / weightsum$
    destinationImage(X) = sourceImage(X')

Resulting warp





Animated sequences

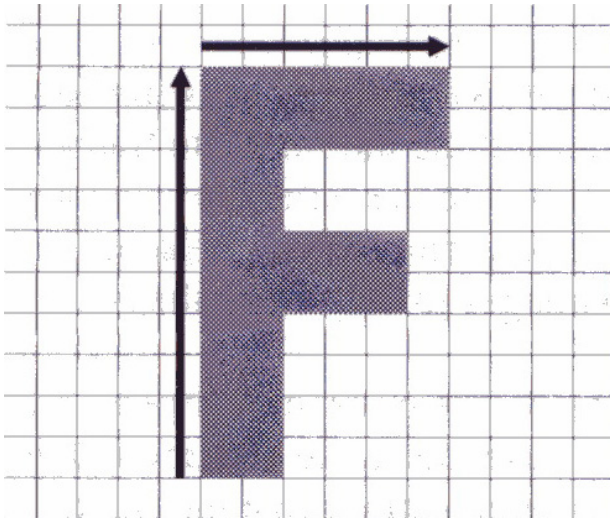
DigiVFX

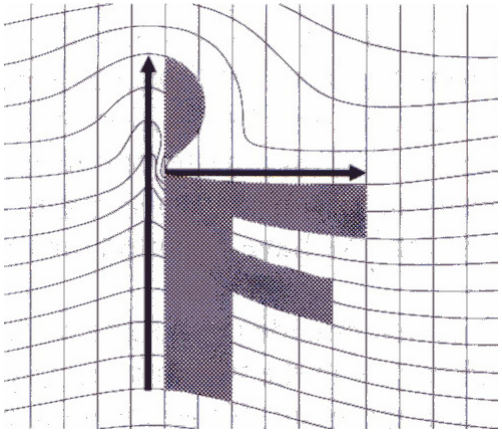
- Specify keyframes and interpolate the lines for the inbetween frames
- Require a lot of tweaking

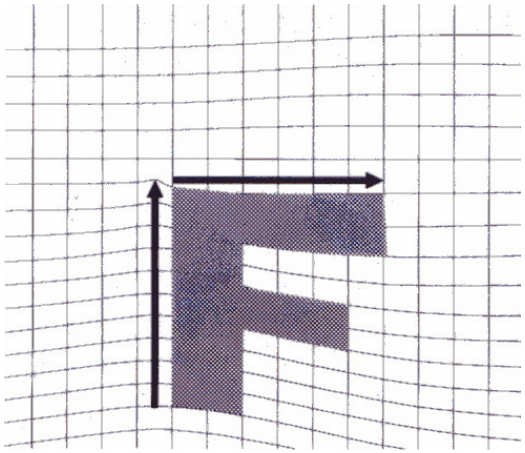
Comparison to mesh morphing

DigiVFX

- Pros: more expressive
- Cons: speed and control





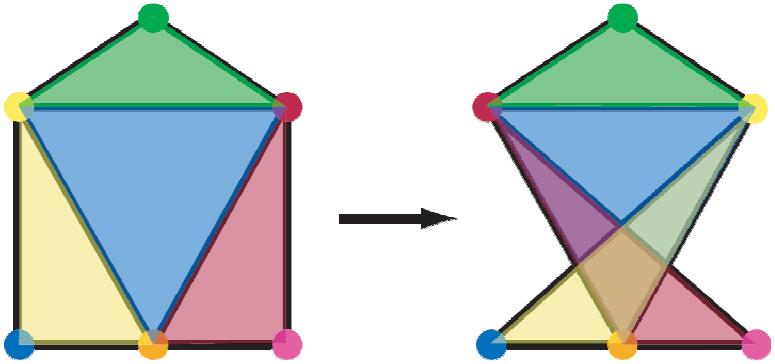


Warp interpolation

DigiVFX

- How do we create an intermediate warp at time t?
- For optical flow:
  - Easy. Interpolate each flow vector
- For feature point methods:
  - linear interpolation of each feature pair
- For Beier-Neely:
  - Can do the same for line end-points
  - But, a line rotating 180 degrees will become 0 length in the middle
  - One solution is to interpolate line mid-point and orientation angle
  - Not very intuitive

Other Issues



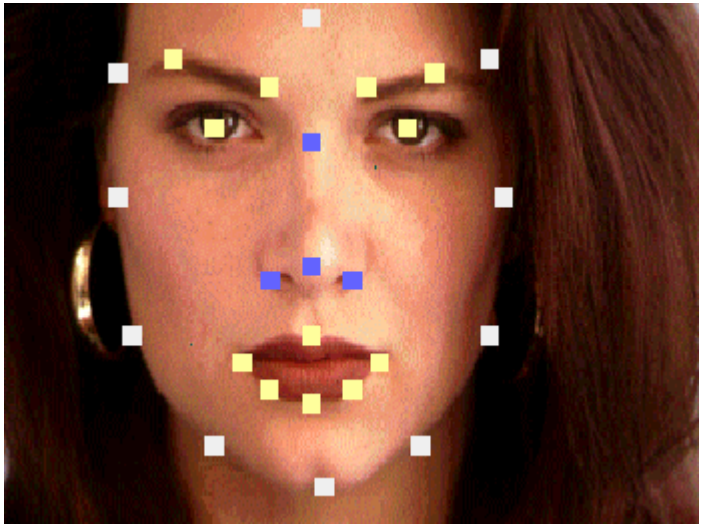
- Beware of folding
  - Can happen in any of the methods
  - You are probably trying to do something 3D-ish
- Extrapolation can sometimes produce interesting effects
  - Caricatures

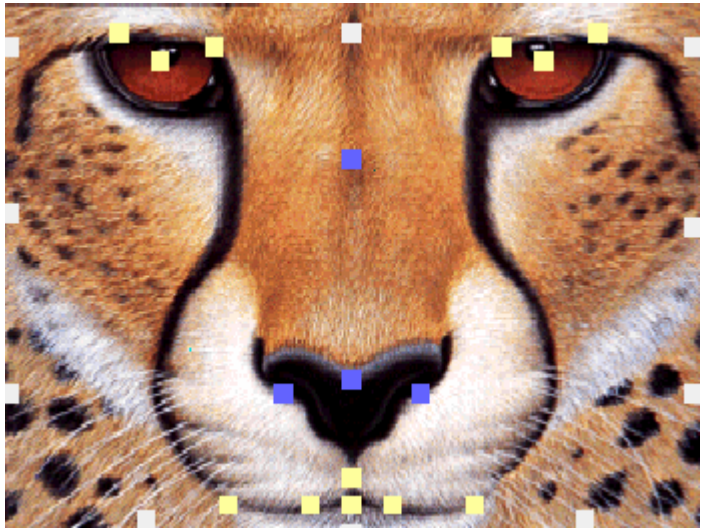
Results

Michael Jackson's MTV "Black or White"

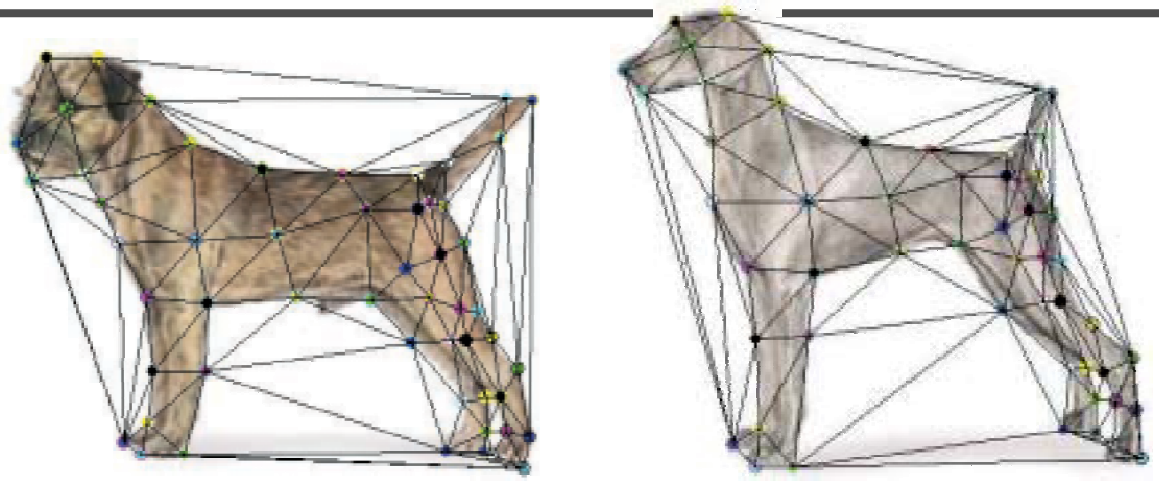
Warp specification

- How can we specify the warp
  3. Specify corresponding *points*
     - *interpolate* to a complete warping function





Solution#1: convert to mesh warping



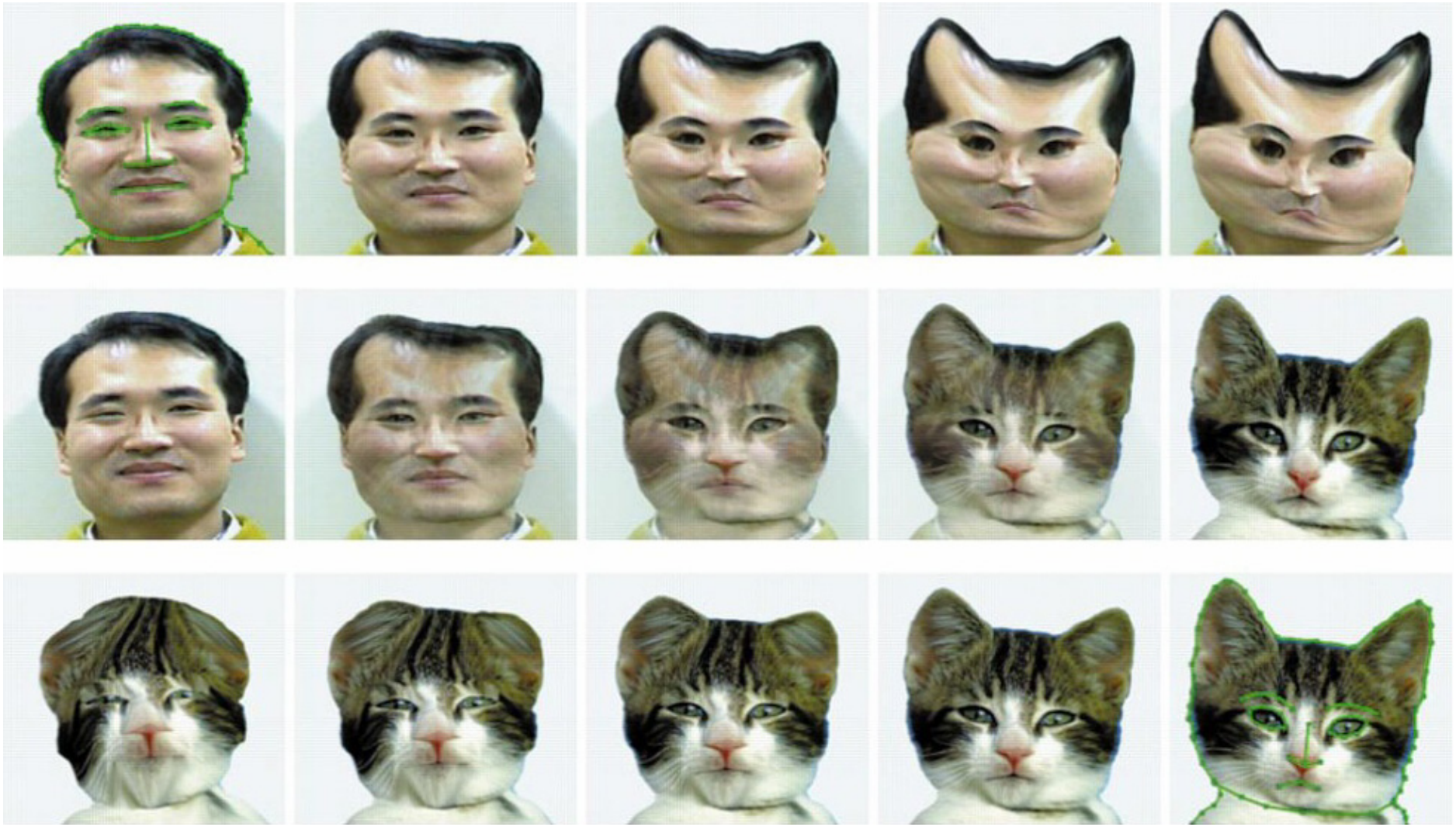
1. Define a triangular mesh over the points
   - Same mesh in both images!
   - Now we have triangle-to-triangle correspondences
2. Warp each triangle separately from source to destination
   - How do we warp a triangle?
   - 3 points = affine warp!
   - Just like texture mapping

Solution#2: scattered point interpolation

DigiVFX

- RBF
- Work minimization

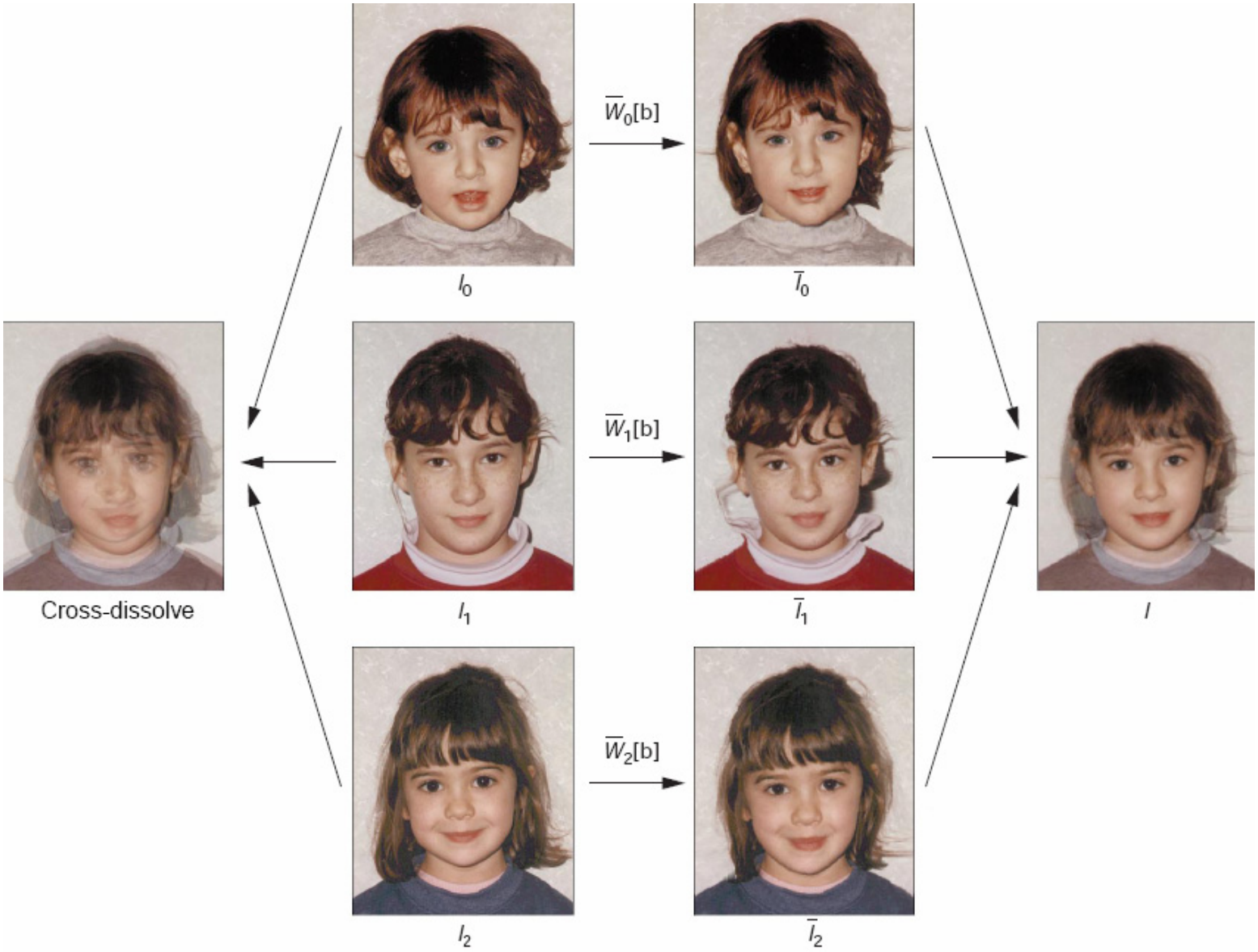
Transition control



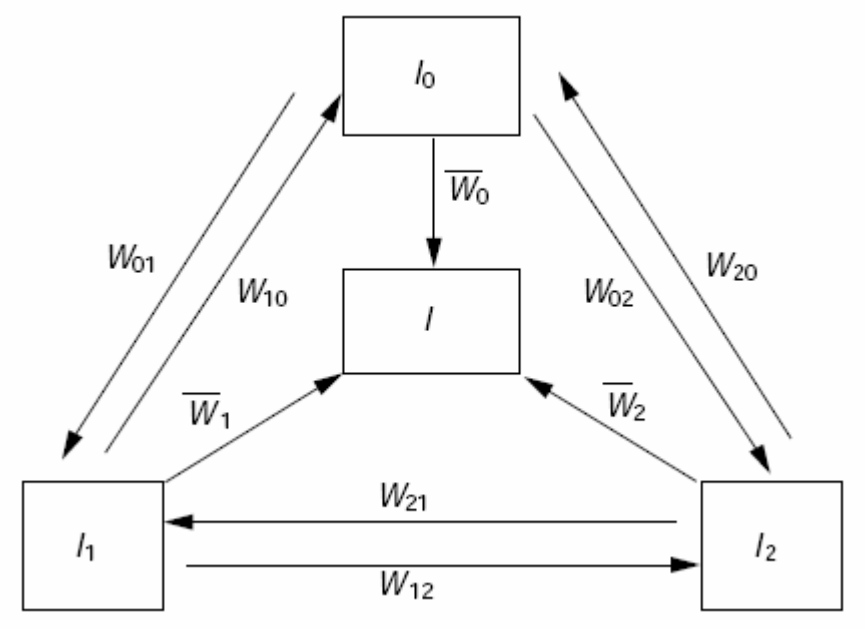
Transition control

Transition control

Multi-source morphing



Multi-source morphing

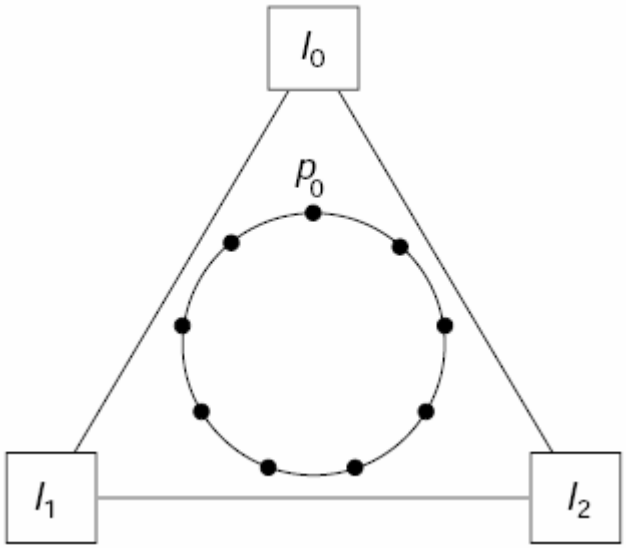


$$\overline{W}_i(p) = \sum_{j=1}^{n} b_j W_{ij}(p)$$

$$\bar{I}_i(r) = \overline{W}_i(p) \bullet b_i I_i(p)$$

$$I(r) = \sum_{i=1}^{n} \bar{I}_i(r)$$

Multi-source morphing



Multi-source morphing

Project #1: image morphing

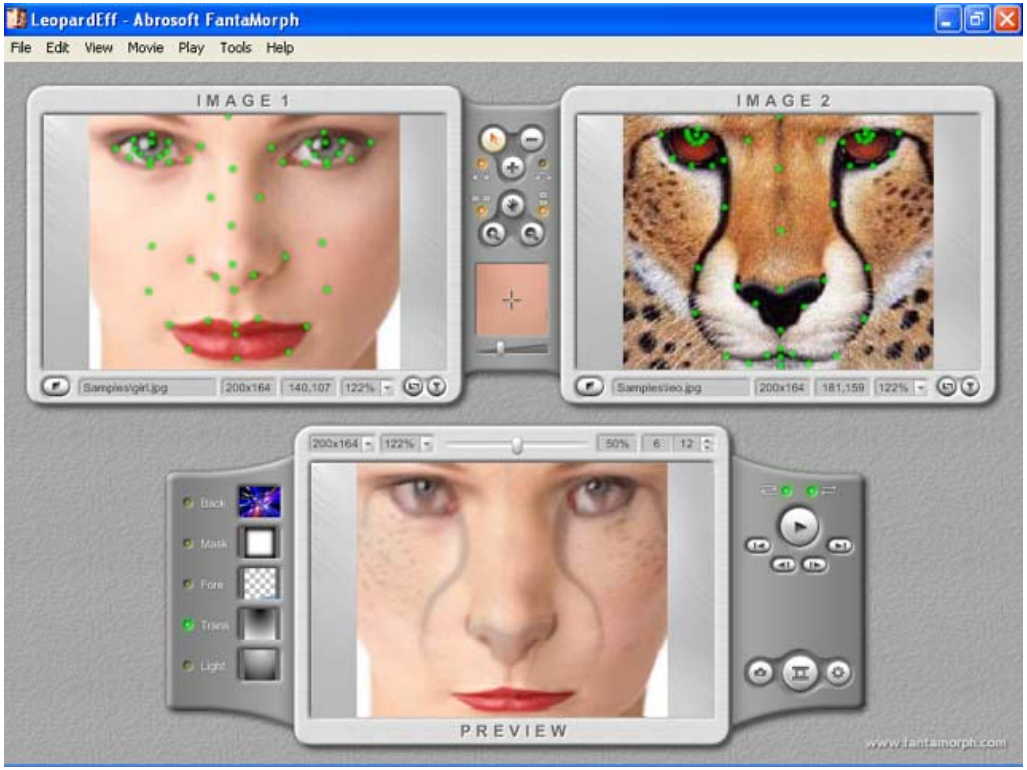
Project #1 image morphing

DigiVFX

- Assigned: 3/9
- Due: 11:59pm 3/29
- Work in pairs
- Handout will be online by tomorrow noon. I will send a mail to vfx when it is available.
- We will provide a generic image library, gil.

Reference software

- Morphing software review
- I used FantaMorph 30-day evaluation version. You can use any one you like.



Morphing is not only for faces

DigiVFX

Morphing is not only for faces

 Bells and whistles

DigiVFX

- Multi-source morphing
- Automatic morphing
- Morphing for animated sequences

Submission

DigiVFX

- You have to turn in your complete source, the executable, a html report and an artifact.
- Report page contains:

  description of the project, what do you learn, algorithm, implementation details, results, bells and whistles…
- Artifacts must be made using your own program.

  artifacts voting on forum
- Submission mechanism will be announced later.