27 - Image Warping dan Image Morphing

Bahan Kuliah IF4073 Interpretasi dan Pengolahan Citra

Oleh: Rinaldi Munir

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
ITB

SUMBER (REFERENSI):

1. Alexei Efros, *Image Warping, 15-463: Computational Photography*, CMU, Fall 2008
2. Connelly Barnes, *Image Warping / Morphing, Computational Photography*.
3. Yao Wang, *EL512 Image Processing, Geometric Transformations: Warping, Registration, Morphing*, Polytchnic University, Brooklyn
4. Image Processing, VICOS Sualgnitiveystemslab

Image Warping

http://www.jeffrey-martin.com

15-463: Computational Photography
Alexei Efros, CMU, Fall 2008

Some slides from Steve Seitz

Image Warping / Morphing

[Wolberg 1996, Recent Advances in Image Morphing]

Computational Photography

Connelly Barnes

Some slides from Fredo Durand, Bill Freeman, James Hays

EL512 --- Image Processing

Geometric Transformations: Warping, Registration, Morphing

Yao Wang
Polytechnic University, Brooklyn, NY 11201

With contribution from Zhu Liu, Onur Guleryuz, and
Partly based on
A. K. Jain, Fundamentals of Digital Image Processing

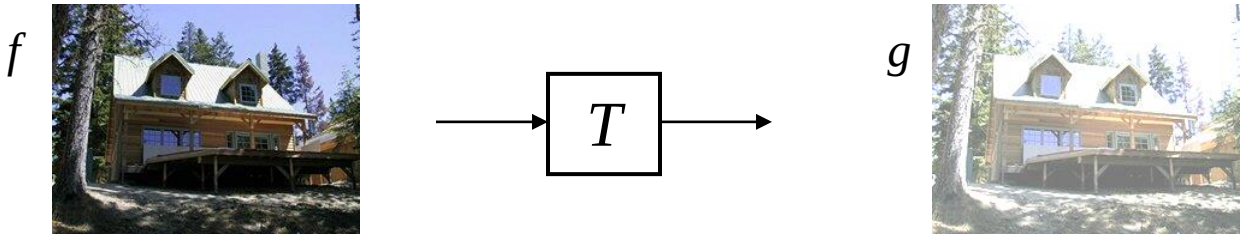
Image Transformation

- So far, the image processing operations we have discussed modify the color values of pixels in a given image → image filtering
- With geometric transformation, we modify the positions of pixels in a image, but keep their colors unchanged
  - To create special effects
  - To register two images taken of the same scene at different times
  - To morph one image to another

Image Transformations

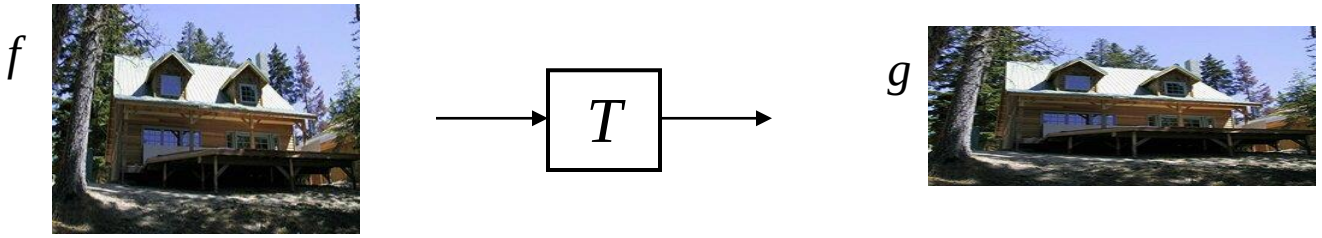
image filtering: change ***range*** (or intensity) of image

$$g(x) = T(f(x))$$

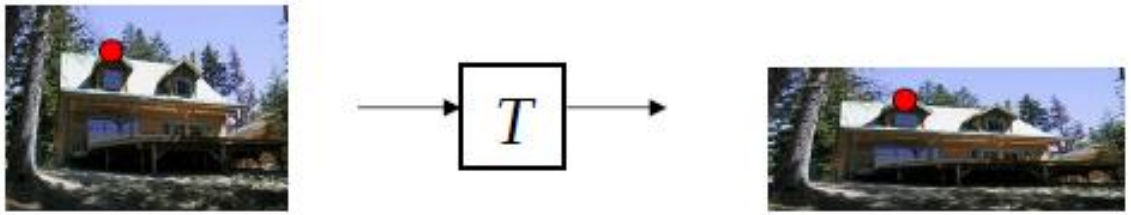


geometric transformation: change ***domain*** (or geometry) of image

$$g(x) = f(T(x))$$



Parametric transformations



$\mathbf{p} = (x,y)$ $\quad \mathbf{p}' = T(\mathbf{p})$ $\quad \mathbf{p'} = (x',y')$

- Transformation T changes coordinates of pixel p
- Global transformation changes all pixels in the same way

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \mathbf{M} \begin{bmatrix} x \\ y \end{bmatrix} \qquad \mathbf{p}' = \mathbf{M} \cdot \mathbf{p}$$

Beberapa Transformasi Geometri

- Rotation:

$$x' = \cos\Theta x - \sin\Theta y \qquad \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} cos\Phi & -sin\Phi \\ sin\Phi & cos\Phi \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$
$$y' = \sin\Theta x + \cos\Theta y$$

- Shear:

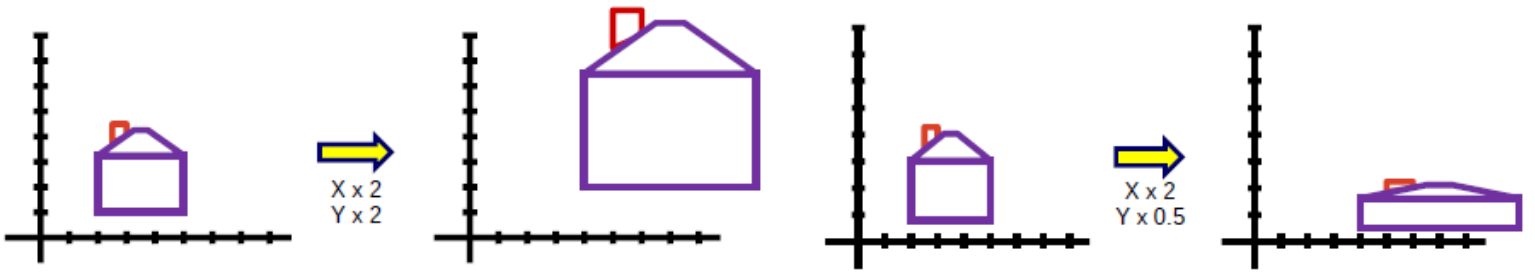
$$x' = x + \alpha_x y \qquad \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & \alpha_x \\ \alpha_y & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$
$$y' = \alpha_y x + y$$

- Mirroring:

$$x' = -x \qquad \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$
$$y' = -y$$

- Scalling
  - Multiply coordinates with a scalar
    - **Uniform** - same scalar for all axes
    - **Non-uniform** - different scalar

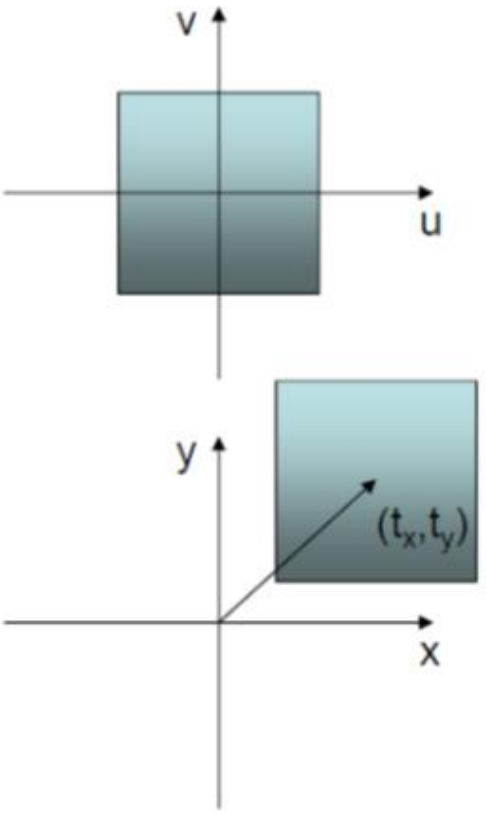
$$x' = \alpha x \qquad \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \alpha & 0 \\ 0 & \beta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$
$$y' = \beta y$$



- Translation

Translation is defined by the following mapping functions:

$$\begin{cases} x = u + t_x \\ y = v + t_y \end{cases} \quad and \quad \begin{cases} u = x - t_x \\ v = y - t_y \end{cases}$$



In matrix notation

$$\mathbf{x} = \mathbf{u} + \mathbf{t}, \quad \mathbf{u} = \mathbf{x} - \mathbf{t}$$

where

$$\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}, \quad \mathbf{u} = \begin{bmatrix} u \\ v \end{bmatrix}, \quad \mathbf{t} = \begin{bmatrix} t_x \\ t_y \end{bmatrix}.$$

Rotation

- Rotation by an angle of θ is defined by

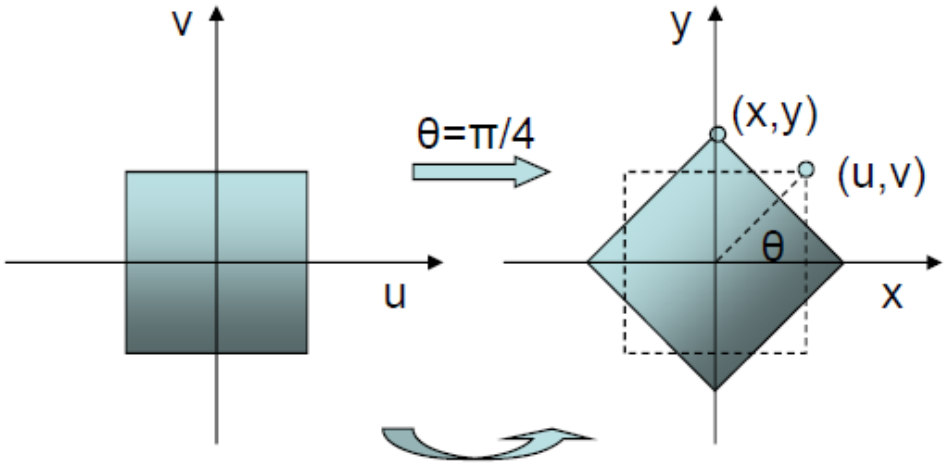
$$\begin{cases} x = u\cos\theta - v\sin\theta \\ y = u\sin\theta + v\cos\theta \end{cases} \quad and \quad \begin{cases} u = x\cos\theta + y\sin\theta \\ v = -x\sin\theta + y\cos\theta \end{cases}$$

- In matrix format

$$\mathbf{x} = \mathbf{R}\mathbf{u}, \quad \mathbf{u} = \mathbf{R}^T\mathbf{x}$$

where

$$\mathbf{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$



- **R** is a unitary matrix: $\mathbf{R}^{-1}=\mathbf{R}^{T}$

B translation

B rotation

Transformasi Affine

- Setiap transformasi geometri (rotasi, shear, *scalling*, translasi) dapat dinyatakan dalam bentuk umum sebagai *affine mapping* atau *affine transformation:*

$$\begin{cases} x' = a_1 x + a_2 y + a_0 \\ y' = b_1 x + b_2 y + b_0 \end{cases} \text{ atau } \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a_1 & a_2 \\ b_1 & b_2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} a_0 \\ b_0 \end{bmatrix}$$

$$\mathbf{x} = \mathbf{Au} + \mathbf{b} \quad , \mathbf{x} = \begin{bmatrix} x' \\ y' \end{bmatrix} , \mathbf{A} = \begin{bmatrix} a_1 & a_2 \\ b_1 & b_2 \end{bmatrix}, \mathbf{u} = \begin{bmatrix} x \\ y \end{bmatrix}, \mathbf{b} = \begin{bmatrix} a_0 \\ b_0 \end{bmatrix},$$

- Jika hanya rotasi saja, maka **b** = 0, jika hanya translasi saja, maka **A** = 0, dan seterusnya

Homogeneous Coordinates

Q: How can we represent translation as a 3x3 matrix?

$$x' = x + t_x$$

$$y' = y + t_y$$

Homogeneous Coordinates

Homogeneous coordinates

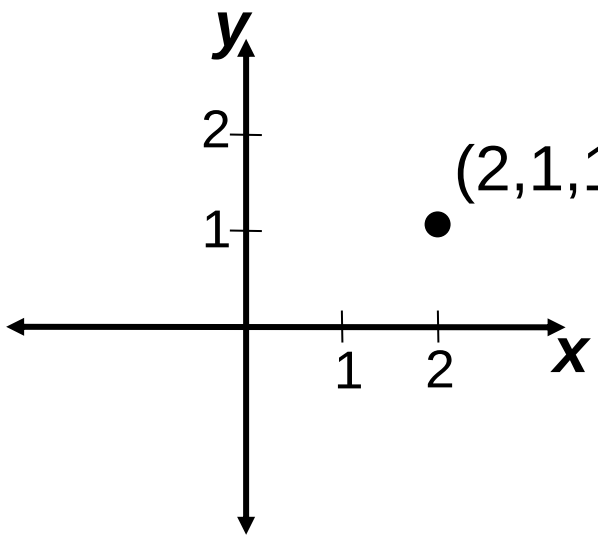
- represent coordinates in 2 dimensions with a 3-vector

$$\begin{bmatrix} x \\ y \end{bmatrix} \Longrightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Homogeneous Coordinates

Add a 3rd coordinate to every 2D point

- (x, y, w) represents a point at location (x/w, y/w)
- (x, y, 0) represents a point at infinity
- (0, 0, 0) is not allowed



(2,1,1) or (4,2,2) or (6,3,3)

Convenient coordinate system to represent many useful transformations

Homogeneous Coordinates

Q: How can we represent translation as a 3x3 matrix?

$$x' = x + t_x$$

$$y' = y + t_y$$

A: Using the homogeneous coordinate:

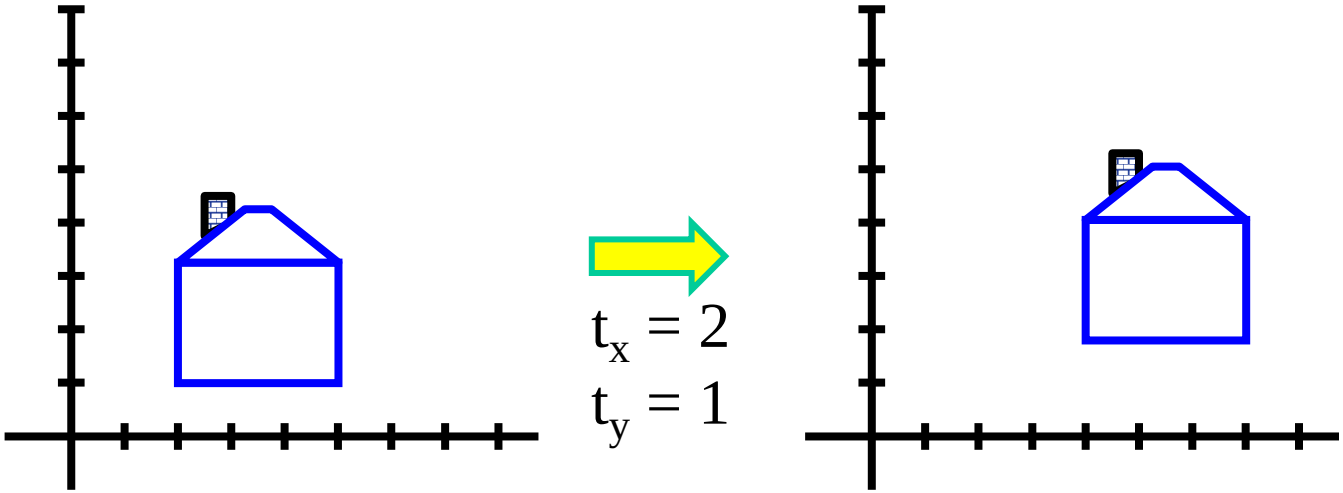
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix}$$

Translation

Example of translation

Homogeneous Coordinates

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix}$$



Basic 2D Transformations

Basic 2D transformations as 3x3 matrices

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Translate

$$\begin{bmatrix} \boldsymbol{x}' \\ \boldsymbol{y}' \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{s}_x & 0 & 0 \\ 0 & \boldsymbol{s}_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ 1 \end{bmatrix}$$

Scale

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\Theta & -\sin\Theta & 0 \\ \sin\Theta & \cos\Theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Rotate

$$\begin{bmatrix} \boldsymbol{x}' \\ \boldsymbol{y}' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & \boldsymbol{sh}_x & 0 \\ \boldsymbol{sh}_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ 1 \end{bmatrix}$$

Shear

Matrix Composition

Transformations can be combined by matrix multiplication

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \left(\begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\Theta & -\sin\Theta & 0 \\ \sin\Theta & \cos\Theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{bmatrix} \right) \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

$\mathbf{p}'$ = $T(t_x,t_y)$ $R(\Theta)$ $S(s_x,s_y)$ $\mathbf{p}$

Affine Transformations matrix 3 x 3

Affine transformations are combinations of

- linear transformations (rotation, scalling, rotation), and
- translations

$$\begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

Image Warping

Warping = pelengkungan

(Sumber gambar: Wikipedia)

Parametric (global) warping

Examples of parametric warps:

translation

rotation

aspect

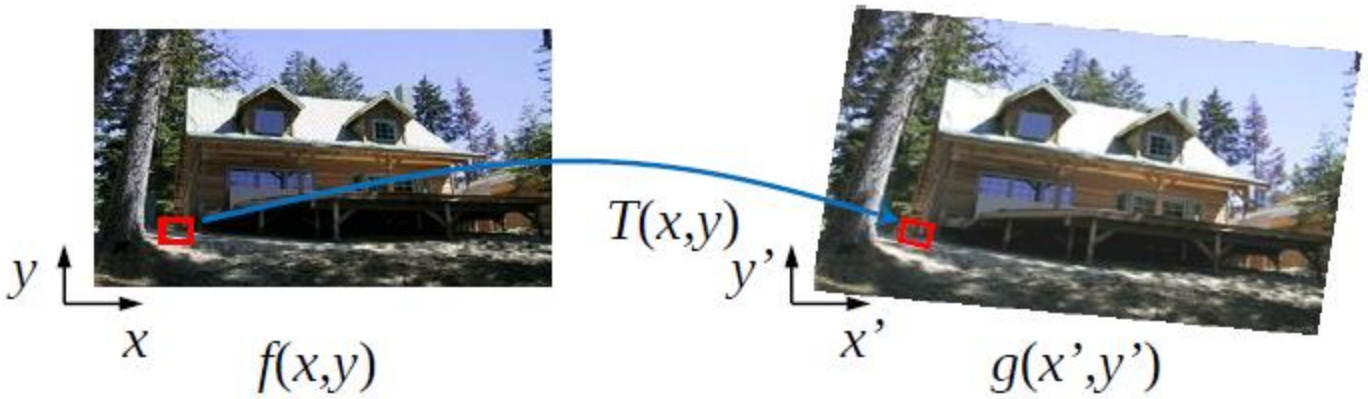
affine

perspective

cylindrical

Image Warping

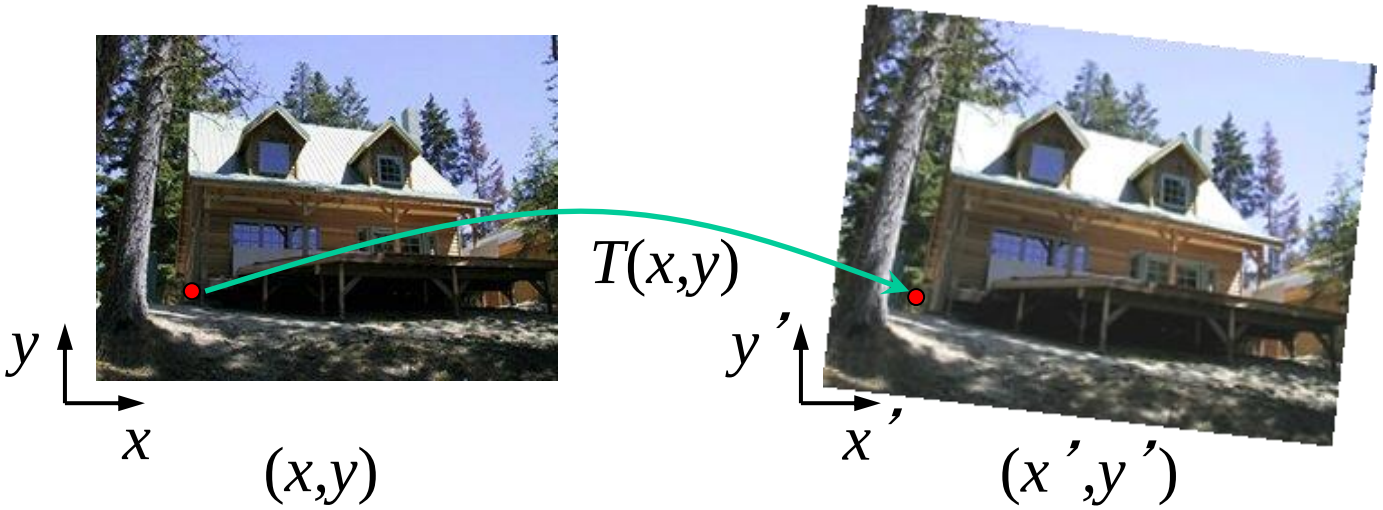
Given transform [x‘,y‘] = T(x,y) and f(x,y), how to compute g(x‘,y‘) = f(T(x,y)) ?



Naive approach → Forward warping

- For each pixel f with coordinates (x,y)
  - Compute transformed coordinates [x‘,y‘]=T(x,y).
  - Copy color of f(x,y) to new image at coordinates g(x‘,y‘)
- Why is it naive?
  - We visit all pixels in f(x,y)
  - Do we visit all pixels of g(x,y)?

Forward warping

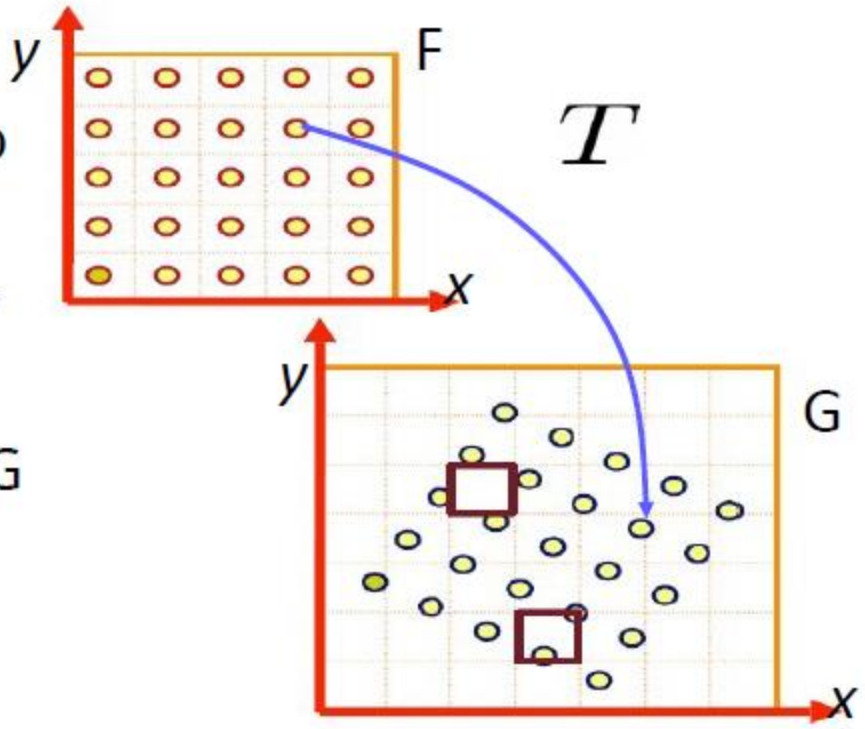


Send each pixel (x,y) to its corresponding location

(x',y') = $T(x,y)$ in the second image

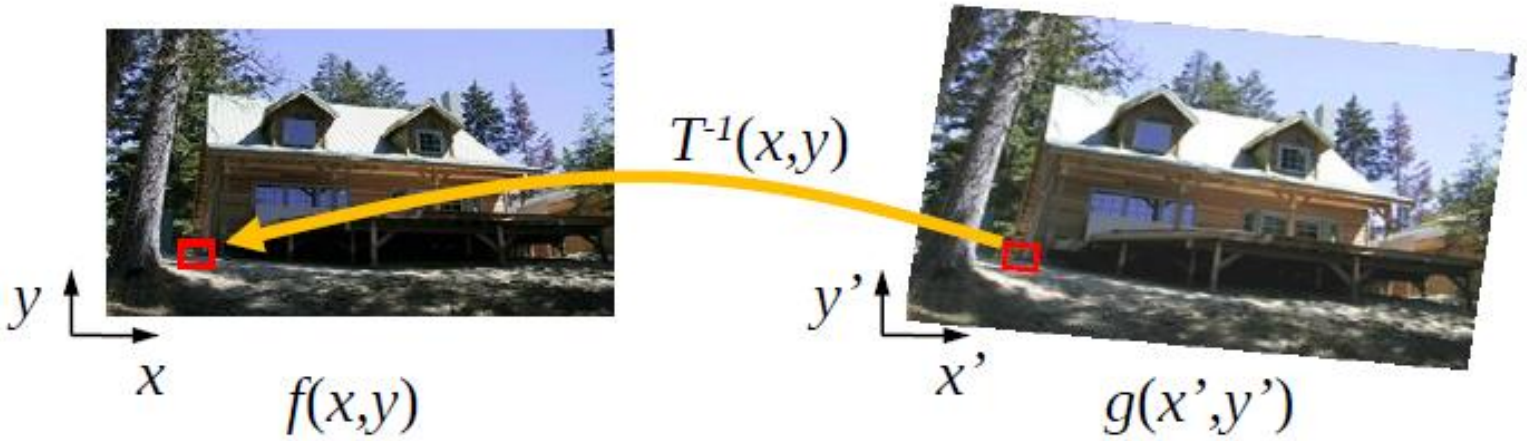
Problems

- A single pixel of F is mapped to more pixels in G.
- Pixel in F is not mapped to any pixel in G.
- Guarantee to visit all pixels in G

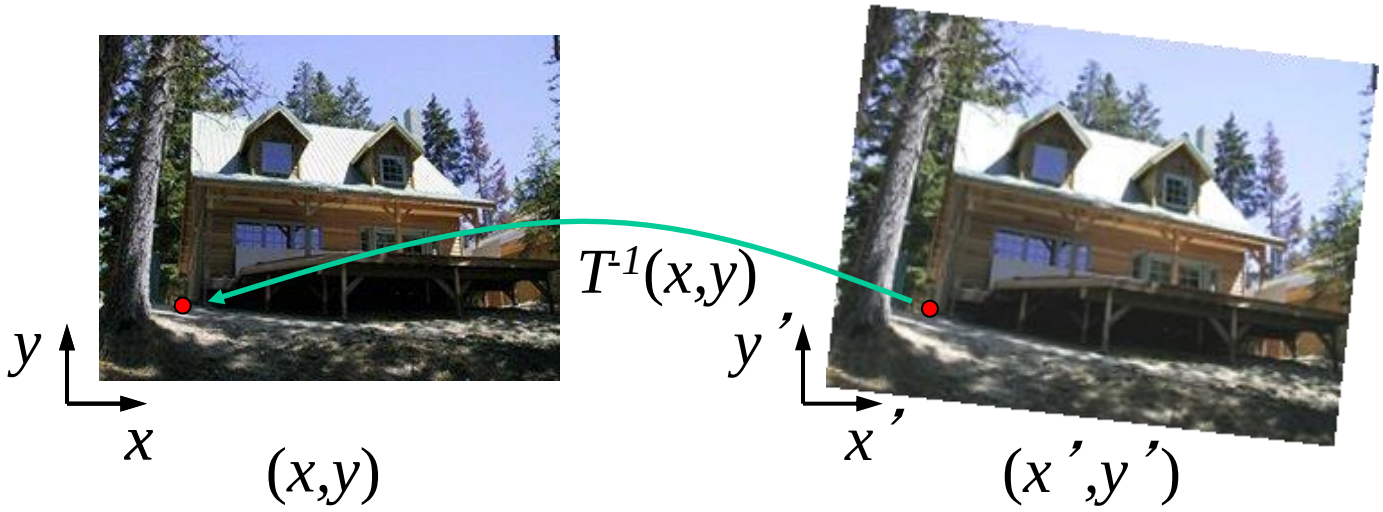


Inverse mapping approach

- For each pixel in g with coordinates (x‘,y‘)
  - Compute old coordinates using inverse transform [x,y]= $T^{-1}(x',y')$
  - We copy pixel color of f(x,y) to g(x‘,y‘)
- We visit all pixels in g
- Pixel from g can transform to more than one pixel in f

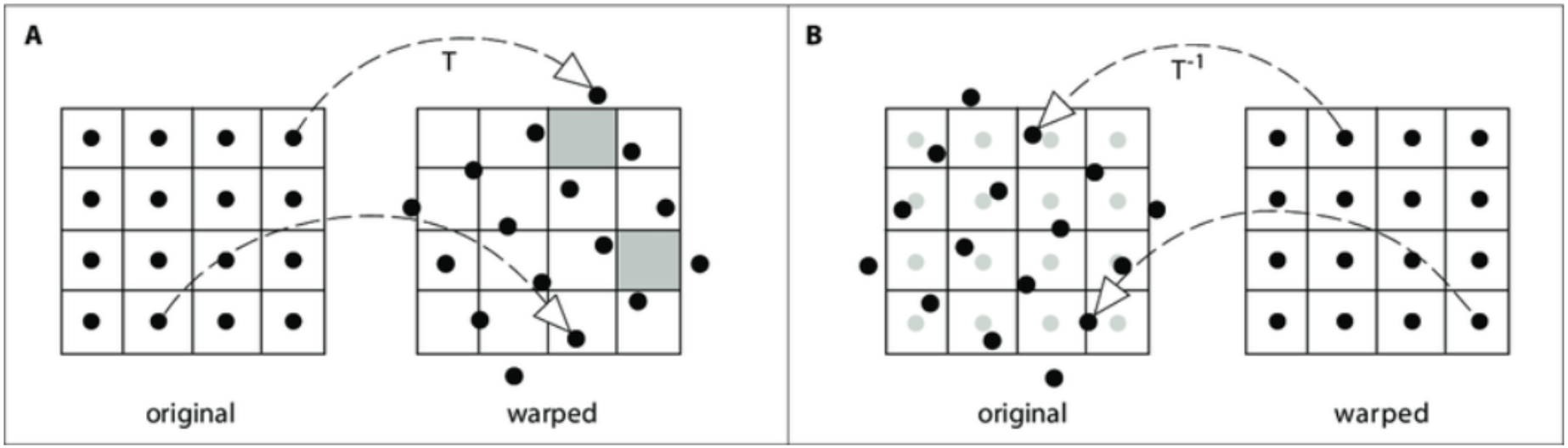


Inverse warping (backward warping)



Get each pixel color $g(x',y')$ from its corresponding location

$(x,y) = T^{-1}(x',y')$ in the first image



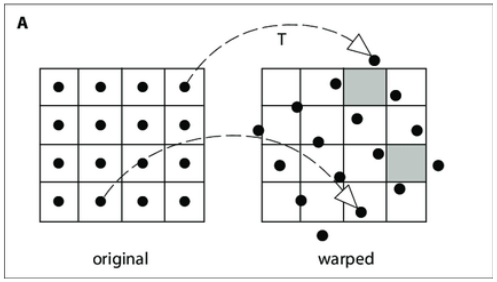
2.: Forward and backward image warping. In the case of foward warping (A), holes can occur in the warped image, marked in gray. Backward warping (B) eliminates this problem since intensities at locations that do not coincide with pixel coordinates can be obtained from the original image using an interpolation scheme.

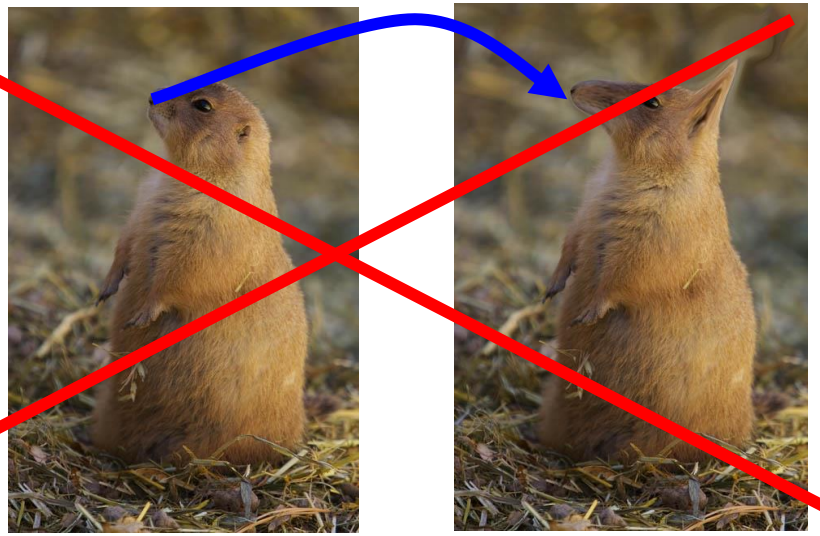
Sumber: Non-rigid Registration Using Free-form Deformations
Authors: Loren Arthur Schwarz

Applying a warp: use inverse

Forward warping:

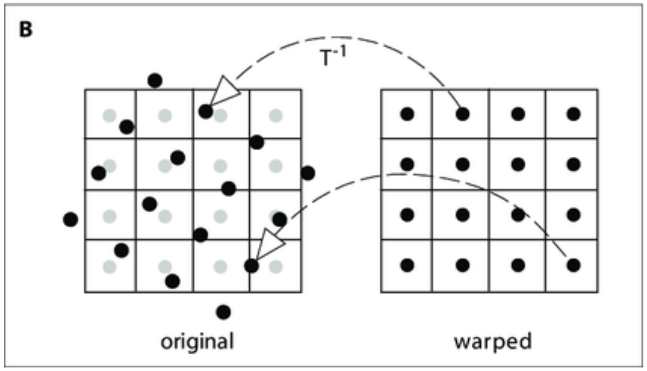
- For each pixel in **input** image
  - Paste color **to warped** location in output
    - Problem: gaps

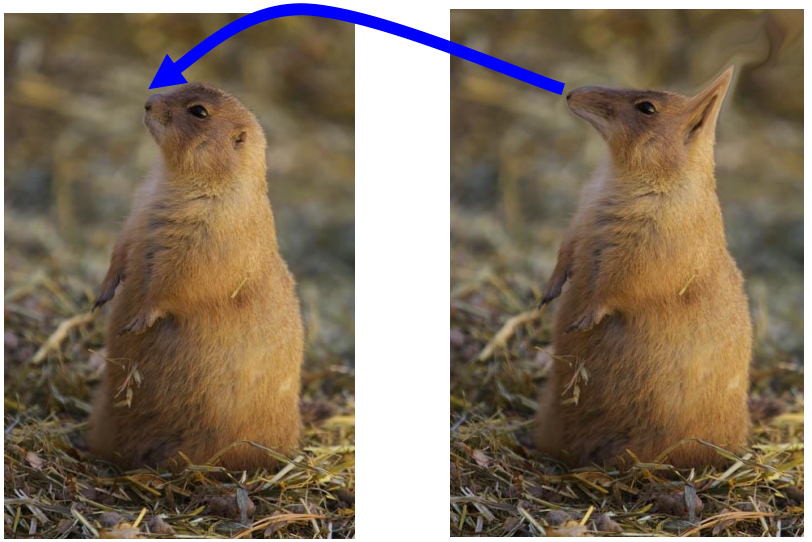




Inverse warping:

- For each pixel in **output** image
  - Lookup color **from inverse-warped** location





Fungsi MATLAB untuk Image Warping

T = maketform('affine',A) builds a TFORM struct T for an N-dimensional affine transformation. A is a nonsingular real (N+1)-by-(N+1) or (N+1)-by-N matrix. If A is (N+1)-by-(N+1), the last column of A must be [zeros(N,1);1]. Otherwise, A is augmented automatically, such that its last column is [zeros(N,1);1]. The matrix A defines a forward transformation such that tformfwd(U,T), where U is a 1-by-N vector, returns a 1-by-N vector X, such that X = U * A(1:N,1:N) + A(N+1,1:N). T has both forward and inverse transformations.

Transformasi affine:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a_1 & a_2 \\ b_1 & b_2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} a_0 \\ b_0 \end{bmatrix}$$

$$\mathbf{x} = \mathbf{Au} + \mathbf{b}$$

In MATLAB notation

$$T = \begin{bmatrix} a_1 & b_1 & 0 \\ a_2 & b_2 & 0 \\ a_0 & b_0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{A}^T & 0 \\ \mathbf{b}^T & 1 \end{bmatrix}$$

In MATLAB, 'affine' transform is defined by:
[a1,b1,0;a2,b2,0;a0,b0,1]

B = imtransform(A,tform) transforms the image A according to the 2-D spatial transformation defined by tform. If ndims(A) > 2, such as for an RGB image, then imtransform applies the same 2-D transformation to all 2-D planes along the higher dimensions.

Contoh 1: Horizontal shear

$$\mathbf{A} = \begin{bmatrix} 1 & 0.5 \\ 0 & 1 \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a_1 & a_2 \\ b_1 & b_2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} a_0 \\ b_0 \end{bmatrix}$$

In MATLAB, 'affine' transform is defined by:
[a1,b1,0;a2,b2,0;a0,b0,1]

```
I = imread('cameraman.bmp');
T = maketform('affine',[1 0 0; 0.5 1 0; 0 0 1]);
I2 = imtransform(I,T);
imshow(I), figure, imshow(I2)
```

Hasil run program:

Example of Image Warping (1)

WAVE1

WAVE2

wave1:x(u,v)=u+20sin(2πv/128);y(u,v)=v;
wave2:x(u,v)=u+20sin(2πu/30);y(u,v)=v.

By Onur Guleyuz

Example of Image Warping (2)

WARP

SWIRL

WARP $x(u,v) = sign(u - x_0) * (u - x_0)^2 / x_0 + x_0; y(u,v) = v$

SWIRL

$$x(u,v) = (u - x_0)\cos(\theta) + (v - y_0)\sin(\theta) + x_0;$$
$$y(u,v) = -(u - x_0)\sin(\theta) + (v - y_0)\cos(\theta) + y_0;$$
$$r = ((u - x_0)^2 + (v - y_0)^2)^{1/2}, \theta = \pi r / 512.$$

By Onur Guleyuz

Image Morphing

- Image morphing has been widely used in movies and commercials to create special visual effects. For example, changing a beauty gradually into a monster.
- The fundamental techniques behind image morphing is image warping.
- Let the original image be f(**u**) and the final image be g(**x**). In image warping, we create g(**x**) from f(**u**) by changing its shape. In image morphing, we use a combination of both f(**u**) and g(**x**) to create a series of intermediate images.
- In image morphing, we create a series of images, starting with f(**u**) at k=0, and ending at g(**x**) at k=K. The intermediate images are a linear combination of the two end images:

f(u)

g(x)

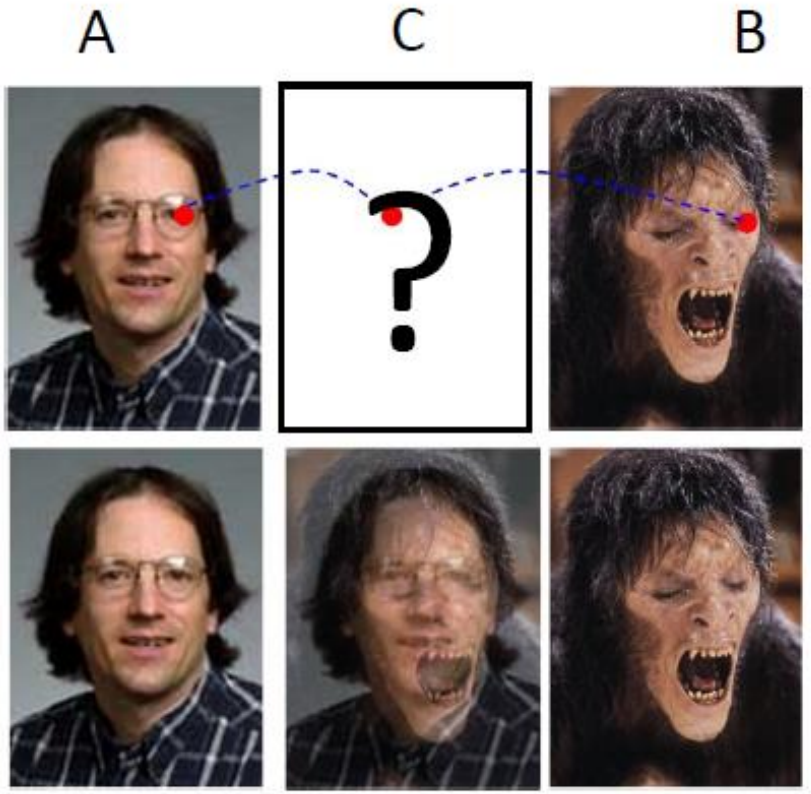
Image morphing

- How to compute intermediate image C
- Naive approach - weighted sum of pixels

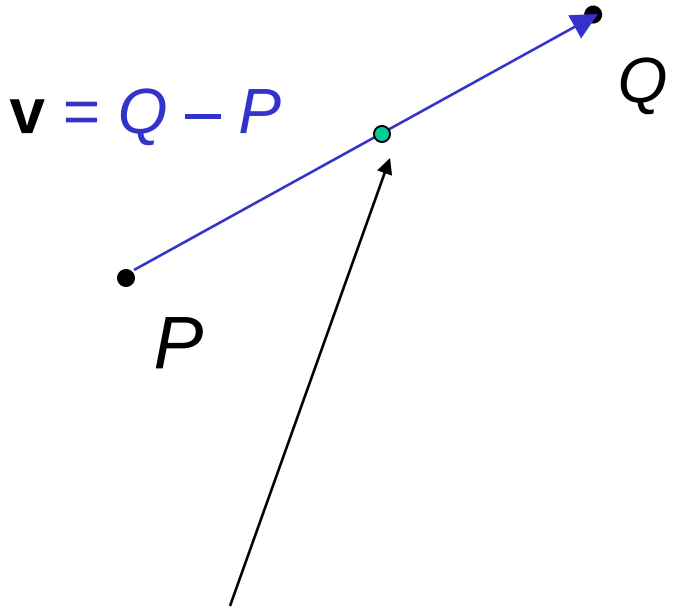
$$C_t = \alpha_t A + (1 - \alpha_t) B$$
$$0 \leq \alpha_t \leq 1$$

- Not realistic - not combining semantic parts

This is called **cross-dissolve** in film industry



Idenya dari linear Interpolation



How can we linearly transition between point *P* and point *Q*?

$P + t\,\mathbf{v} = P + t(Q - P)$

$= (1- t)P + tQ$, e.g. t = 0.5

P and Q can be anything:

- points on a plane (2D) or in space (3D)
- Colors in RGB or HSV (3D)
- Whole images (m-by-n D)… etc.

Cross-Dissolve

Interpolate whole images:

$Image_{halfway} = (1-t)*Image_1 + t*image_2$

```
A = imread('face2.jpg');
B = imread('face1.jpg');

alpha = 1;
for i=1:10
    C = alpha*A + (1 - alpha)*B;
    figure, imshow(C);
    alpha = alpha - 0.1;
end
```

Hasil running program

Examples of Image Morphing

Cross Dissolve

$I(t) = (1-t)*S+t*T$

Mesh based

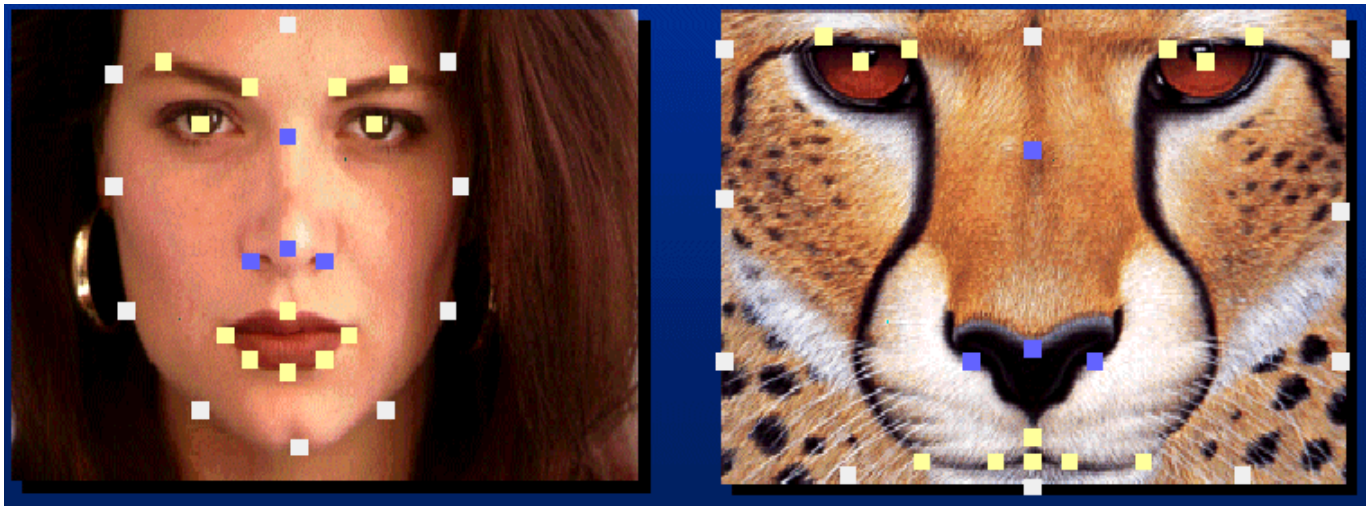
George Wolberg, "Recent Advances in Image Morphing", *Computer Graphics Intl. '96*, Pohang, Korea, June 1996.

But what if the images are not aligned?

Align first, then cross-dissolve

- Alignment using global warp – picture still valid

Full Morphing

A  B

- What if there is no simple global function that aligns two images?
- User specifies corresponding feature points → **control points**
- Construct warp animations A → B and B → A
- Cross dissolve these

Control points

Control points mark matching pixels in both images

Morphing example

Input images

Control points

Warped images

Blended image

Full Morphing

Image A

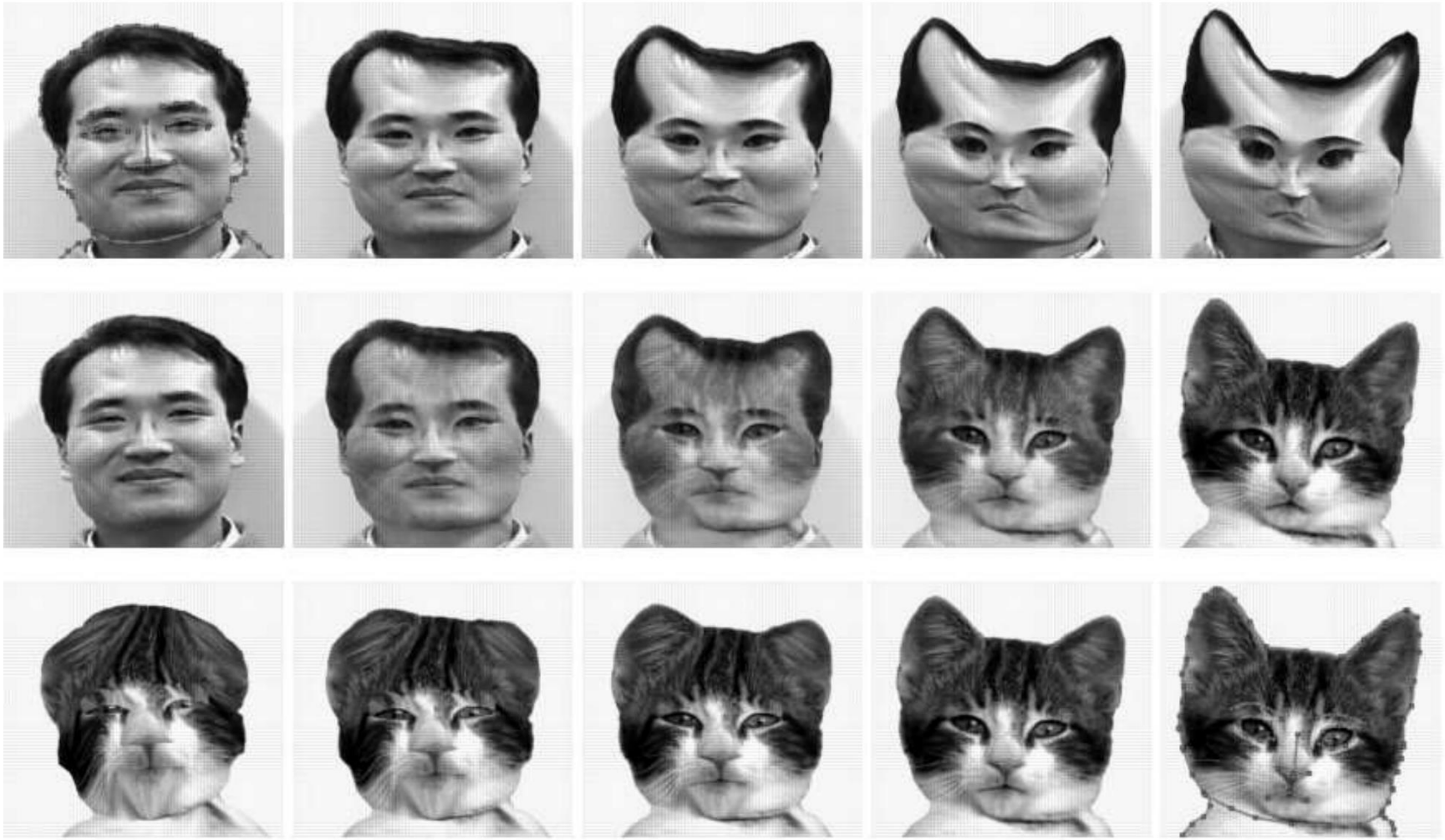
Image B

1. Find warping fields from user constraints (points or lines):
   Warp field T_{AB}(x, y) that maps A pixel to B pixel
   Warp field T_{BA}(x, y) that maps B pixel to A pixel

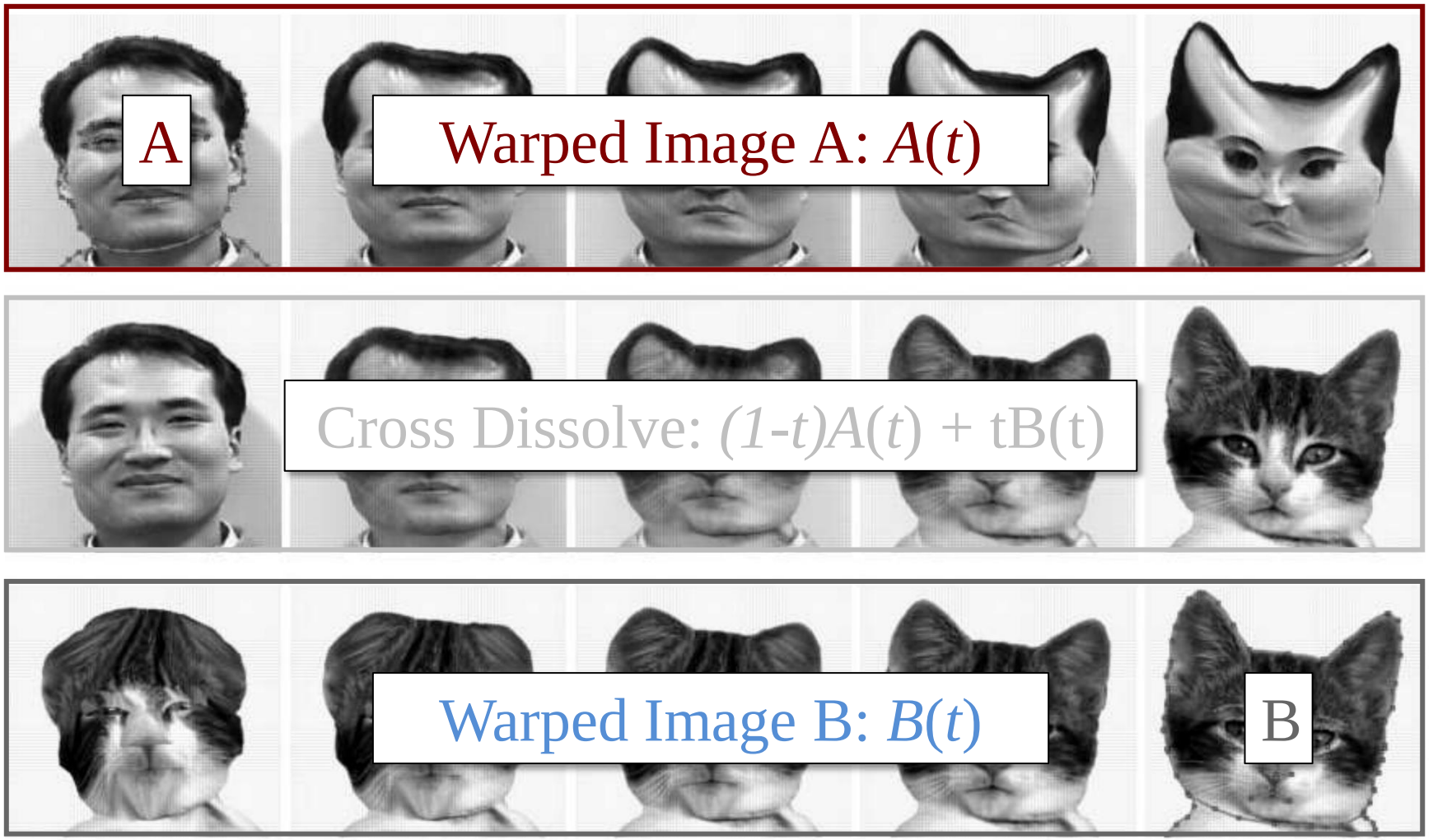
2. Make video $A(t)$ that warps A over time to the shape of B
   Start warp field at identity and linearly interpolate to T_{BA}
   Construct video $B(t)$ that warps B over time to shape of A

3. Cross dissolve these two videos.

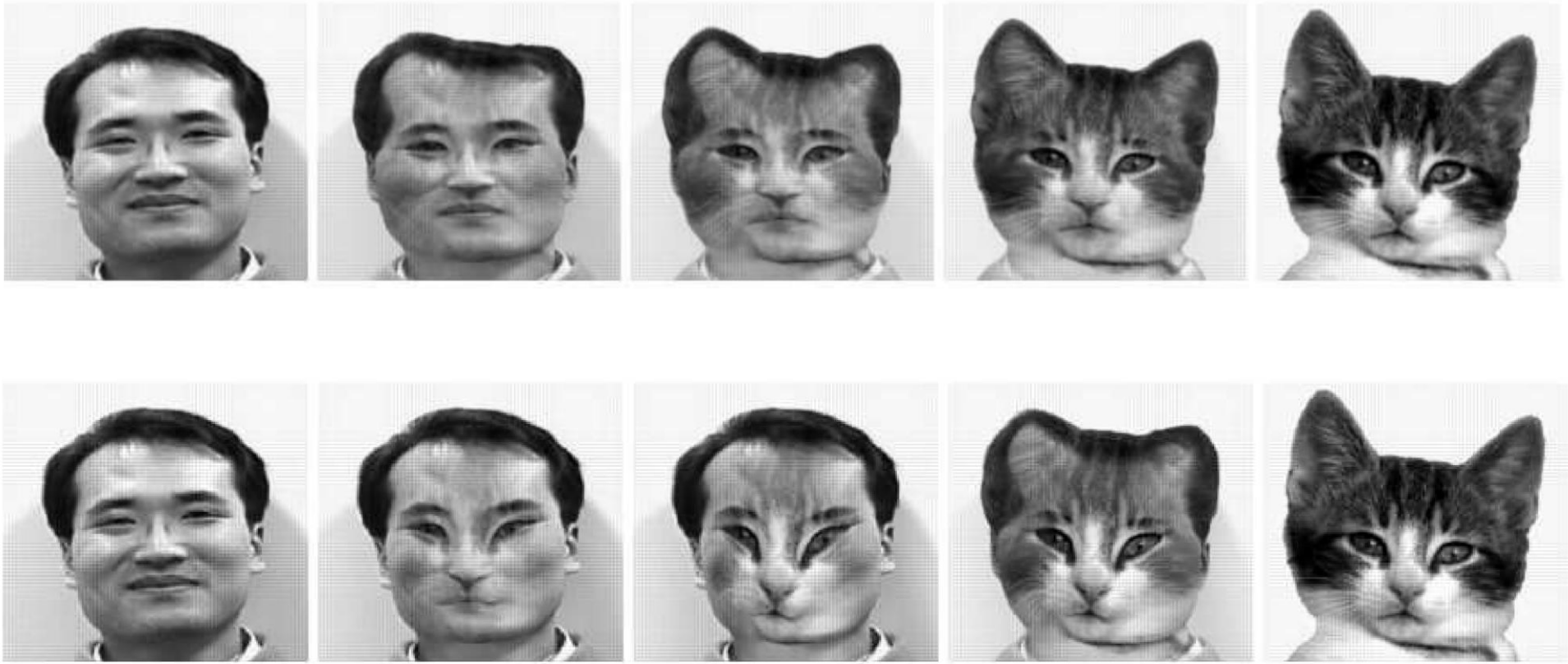
Full Morphing



Full Morphing



Catman!



One more morphing example

Conclusion

Michael Jackson - Black or White