(3.7) Compositing

Digital compositing: Used to blend live actions with computer graphics etc.

Example, dissolving: blend to images, moving from one to the other.

Figure 3.30 Two examples of dissolving one image into another. Source: Screenshots by WETA Digital Ltd. – © 2011 Paramount Pictures. 'The Adventures of Tintin'

$$I'(x,y) = \eta I_A(x,y) + (1-\eta) I_B(x,y)$$

Compositing with binary masks

12 binary comp. op.

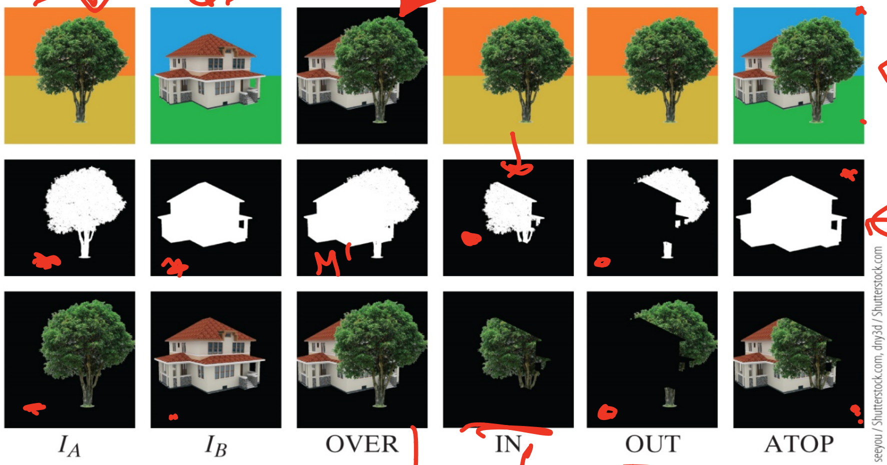


Figure 3.32 Common binary compositing operations applied to a pair of masked images. The top two rows show, from left to right: Original image I_A and mask M_A, original image I_B and mask M_B, and image I' and mask M' resulting from the four operations OVER, IN, OUT, and ATOP, respectively. The bottom row shows the result of ANDing each image with each mask.

Atop:

$I_A \wedge M_A + I_B \wedge \neg M_A = I'$

$M' = M_B$

I' AND M'

NOT

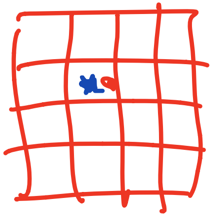
I_A over $I_B = I_A \wedge M_A + I_B \wedge M_B \wedge \neg M_A$

AND OR

I_A in $I_B \Rightarrow I' = I_A \quad M' = M_A \wedge M_B$

I_A out $I_B \Rightarrow I' = I_A \quad M' = M_A \wedge \neg M_B$

(3.8) Interpolation



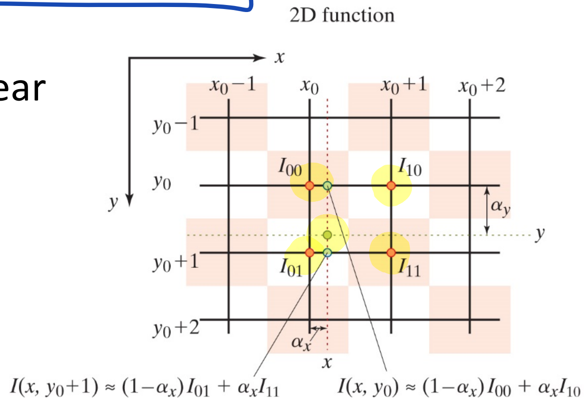
- **Nearest neighbor interpolation:** returns the gray level of the pixel nearest the coordinates:

$$\hat{I}(x, y) \equiv I(\min(\max(\textsc{Round}(x), 0, \mathit{width}-1)), \min(\max(\textsc{Round}(y), 0, \mathit{height}-1)))$$

- if bounds checking can be assumed: $\boxed{\hat{I}(x, y) \equiv I(\textsc{Round}(x), \textsc{Round}(y))}$

- **Bilinear interpolation:** a 2D extension of 1D linear interpolation. The interpolated value is the weighted average of the four nearby pixels:

$$\boxed{\hat{I}(x, y)} = \overline{\alpha}_x \overline{\alpha}_y I_{00} + \alpha_x \overline{\alpha}_y I_{10} + \overline{\alpha}_x \alpha_y I_{01} + \alpha_x \alpha_y I_{11}$$



If the derivatives are known, or can be estimated, **cubic interpolations** can be used. More computationally demanding.

Interpolation can be vuisualized by using a kernel function:

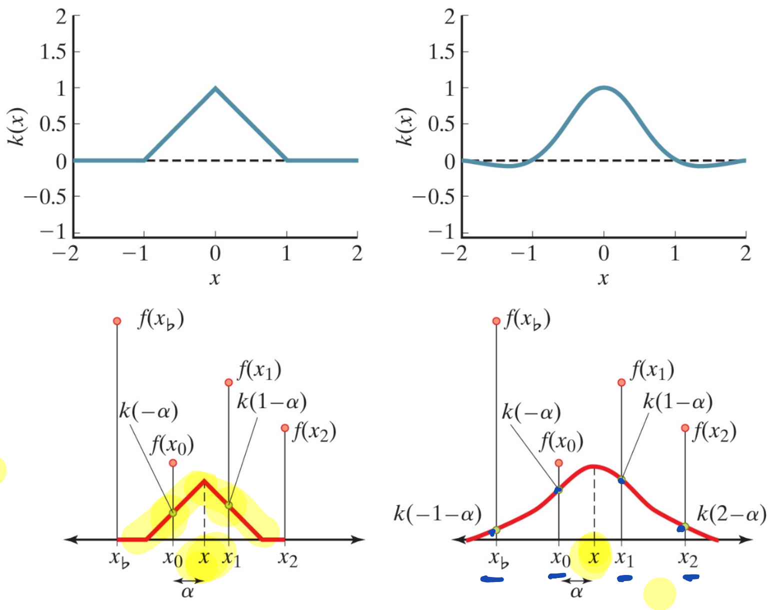
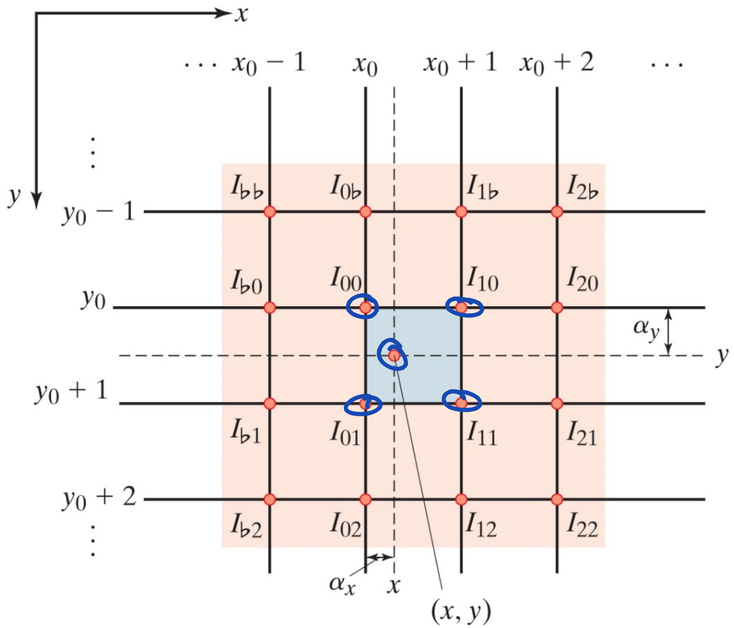


Figure 3.35 Top: Linear (left) and cubic (right) 1D interpolation kernels. The dashed line indicates $k(x) = 0$ to emphasize that the cubic interpolation kernel contains negative values. Bottom: Interpolation involves shifting the kernel so that it is centered at the desired position x, then the neighboring samples are combined using the weights from the kernel.

Figure 3.36 Bicubic interpolation at a point (x, y) is a weighted average of the 16 nearby gray levels.

$I'(x,y)$



Less computationally demanding, and better cubic filters can be found: Key filters, Mitchell filter.
Other interpolation kernels: Lancroz, based on sinc function.

(more details in the book)

(3.9) Warping

- Consider arbitrary geometric transformations from real-valued coordinates (*x, y*) to real-valued coordinates (*x'*, *y'*):

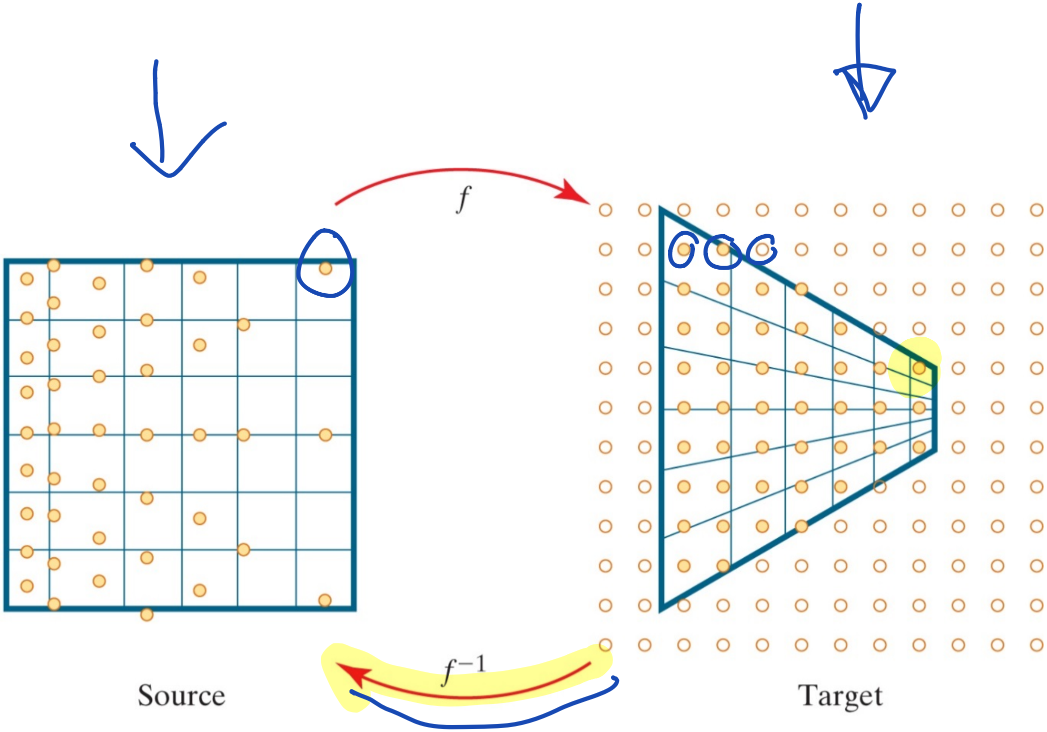
$$\boxed{I'(x', y') = I(x, y)}$$

- The **mapping function** $f : \mathbb{R}^2 \to \mathbb{R}^2$ specifies the transformation, or **warping**, from the input coordinates to the output coordinates:

$$\boxed{(x', y') = f(x, y)} \qquad \boxed{(x, y) = f^{-1}(x', y')}$$

- Use the invers transformation to be sure to give each output pixel one and only one value.
- Interpolation is used to find I(x,y) for real valued (x,y).

Figure 3.41
A frontoparallel plane in the input is warped to a slanted plane in the output. The inverse transformation guarantees that every pixel in the output receives a value, whereas the forward transformation leads to some pixels not receiving values while others receive multiple values. Based on Burger and Burge: W. Burger and M. J. Burge. *Digital Image Processing: An Algorithmic Introduction Using Java*. Springer, 2008.



$$f^{-1}(x', y') = \boxed{(x, y)}$$

Warping - Transformations

(x´,y´) is output image coordinates and (x,y) input image coordinates: $\boxed{I'(x',y')=I(x,y)}$

Translation:

$$x' = x + t_x$$
$$y' = y + t_y$$

inverse

$$\begin{bmatrix} x \\ y \end{bmatrix} = f^{-1}(x', y') = \begin{bmatrix} x' - t_x \\ y' - t_y \end{bmatrix}$$

edges need a strategy

Warping - Transformations

(x′,y′) is output image coordinates and (x,y) input image coordinates: I′(x′,y′)=I(x,y)

Rotation: clockwise by angle θ (origin be upper left corner)

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\boxed{R^{-1} = R^T}$$

rows & colums

$\|\cdot\| = 1$

$\underline{x}' = \underset{\sim}{R} \cdot \underline{x}$ rotate around $\underline{c}$:

$$\underline{x}' = \underset{\sim}{R}(\underline{x} - \underline{c}) + \underline{c} \Rightarrow \begin{cases} \underline{x} = \underset{\sim}{R}^{-1}(\underline{x}' - \underline{c}) + \underline{c} \\ \underline{x} = \underset{\sim}{R}^{T}(\underline{x}' - \underline{c}) + \underline{c} \end{cases}$$

Warping - Transformations

- Translation and rotation can be combined into a single **Euclidean transformation**:

$$\mathbf{x}' = \mathbf{R}(\mathbf{x} - \mathbf{c}) + \mathbf{c} + \mathbf{t} = \mathbf{R}\mathbf{x} + \tilde{\mathbf{t}}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}}_{\mathbf{R}} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} \tilde{t}_x \\ \tilde{t}_y \end{bmatrix} \qquad \boxed{\tilde{\mathbf{t}}} \equiv [\tilde{t}_x \quad \tilde{t}_y]^{\mathsf{T}} = \boxed{-\mathbf{R}\mathbf{c} + \mathbf{c} + \mathbf{t}}$$

Euclidean transformations preserves shape and scale of an object

- Can be written compactly and conveniently using homogeneous coordinates:

$$\boxed{\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta & \tilde{t}_x \\ \sin\theta & \cos\theta & \tilde{t}_y \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}}$$

Warping - Transformations

- **Similarity transformations:** a superset of Euclidean transformations including translations, rotations, AND uniform scaling:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} k\cos\theta & -k\sin\theta & k\tilde{t}_x \\ k\sin\theta & k\cos\theta & k\tilde{t}_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Uniform scaling: $x'=kx$, $y'=ky$

Similarity transformations preserves shape of an object

Affine transformations

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} a_{13} \\ a_{23} \end{bmatrix}$$

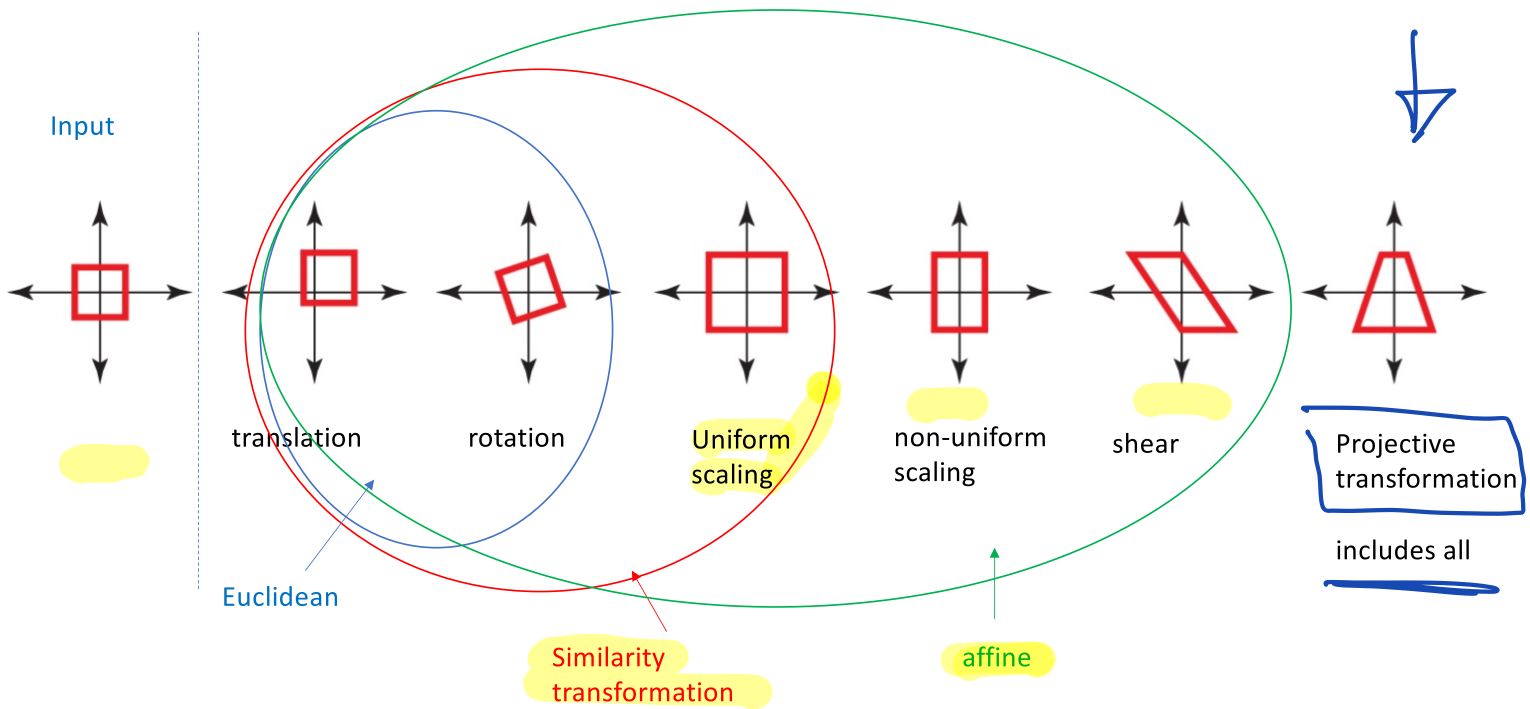
- **Affine transformations**. Any 2x2 invertable matrix. In homogeneous coordinates:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \boxed{\frac{1}{a_{11}a_{22} - a_{12}a_{21}} \begin{bmatrix} a_{22} & -a_{12} & a_{23}a_{12} - a_{22}a_{13} \\ -a_{21} & a_{11} & -a_{23}a_{11} + a_{21}a_{13} \\ 0 & 0 & a_{11}a_{22} - a_{12}a_{21} \end{bmatrix}} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

- Include: rotation, translation
  - Uniform scaling
  - Non-uniform scaling   x´=ax   y´=by
  - Shear   x´= x+ ay, y´=y

All affine transformations: Parallel lines in input -> parallel lines in output


Input
translation
rotation
Uniform scaling
non-uniform scaling
shear
Projective transformation
includes all
Euclidean
Similarity transformation
affine

Projective transformations

- **Projective Transformations**. Relax the constraint of the bottom row of the transformation matrix (3x3 invertable matrix, homography)

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \propto \underbrace{\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}}_{\mathbf{H}} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\cancel{\underset{\sim}{H} = \begin{bmatrix} \underset{\sim}{R} & t_x \\ & t_y \\ 0 \quad 0 & 1 \end{bmatrix}}$$

Includes all affine transformations

+ parallel lines in input -> intersecting lines in output

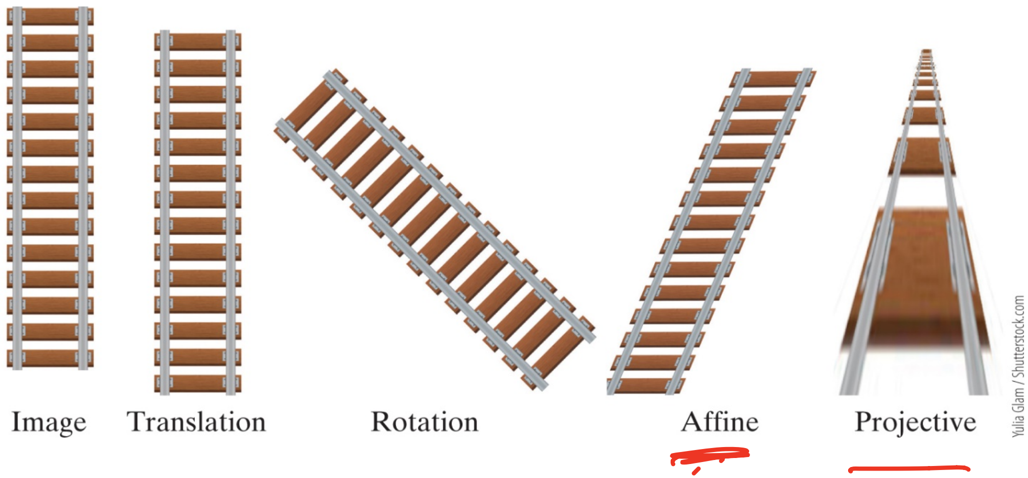


Image
Translation
Rotation
Affine
Projective
Yulia Glam / Shutterstock.com

Other Warping related operations

- **Image registration** – Sometimes it is desirable to align two input images. Simple to complex solutions depending on application
  - Medical images
  - Images of same scene, different locations

- **Morphing** – If two images are both registered and also dissolved into each other, we say that they are morphed.
  - morphing of a persons face onto another, snapchat filters etc
  - Continous scene change in video