

3.1 Geometric Mean Filter

$$\hat{F}(u, v) = \left[\frac{H^*(u, v)}{|H(u, v)|^2} \right]^\alpha \left[\frac{H^*(u, v)}{|H(u, v)|^2 + \beta \left[\frac{S_\eta(u, v)}{S_f(u, v)} \right]} \right]^{1-\alpha} G(u, v)$$

with α and β being positive, real constants

The geometric mean filter consists of the two expressions in brackets raised to the powers α and $1-\alpha$, respectively

$$\hat{F}(u, v) = \left[\frac{H^*(u, v)}{|H(u, v)|^2} \right]^\alpha \left[\frac{H^*(u, v)}{|H(u, v)|^2 + \beta \left[\frac{S_\eta(u, v)}{S_f(u, v)} \right]} \right]^{1-\alpha} G(u, v)$$

- When $\alpha=1$ this filter reduces to the inverse filter
- With $\alpha=0$ the filter becomes the so-called parametric Wiener filter, which reduces to the standard Wiener filter when $\beta=1$
- If $\alpha = 1/2$ the filter becomes a product of the two quantities raised to the same power, which is the definition of the geometric mean, thus giving the filter its name
- With $\beta=1$ as α decreases below $1/2$, the filter performance will tend more toward the inverse filter
- Similarly, when α increases above $1/2$, the filter will behave more like the Wiener filter
- When $\alpha=1/2$ and $\beta=1$ the filter also is commonly referred to as the spectrum equalization filter
- Equation is quite useful when implementing restoration filters because it represents a family of filters combined into a single expression

3.2 Geometric Transformations

- Geometric transformations modify the spatial relationship between pixels in an image
- These transformations often are called rubber-sheet transformations because they may be viewed as analogous to “printing” an image on a sheet of rubber and then stretching the sheet according to a predefined set of rules
- In terms of digital image processing, a geometric transformation consists of two basic operations:
 - a spatial transformation of coordinates and
 - intensity interpolation that assigns intensity values to the spatially transformed pixels
- The transformation of coordinates may be expressed as

$$(x, y) = T\{(v, w)\}$$

- where (v, w) are pixel coordinates in the original image and (x, y) are the corresponding pixel coordinates in the transformed image
- One of the most commonly used spatial coordinate transformations is the affine transform, which has the general form

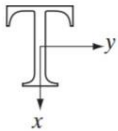
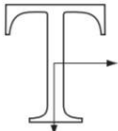
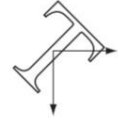
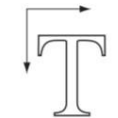

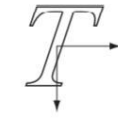
$$[x \ y \ 1] = [v \ w \ 1] \mathbf{T} = [v \ w \ 1] \begin{bmatrix} t_{11} & t_{12} & 0 \\ t_{21} & t_{22} & 0 \\ t_{31} & t_{32} & 1 \end{bmatrix}$$

This transformation can scale, rotate, translate, or shear a set of coordinate points, depending on the value chosen for the elements of matrix T.

The preceding transformations relocate pixels on an image to new locations. To complete the process, we have to assign intensity values to those locations. This task is accomplished using intensity interpolation. For an example of zooming an image and the issue of intensity assignment to new pixel locations:

- Zooming is simply scaling
- the problem of assigning intensity values to the relocated pixels resulting from the other transformations
- we consider nearest neighbor, bilinear, and bicubic interpolation techniques when working with these transformations

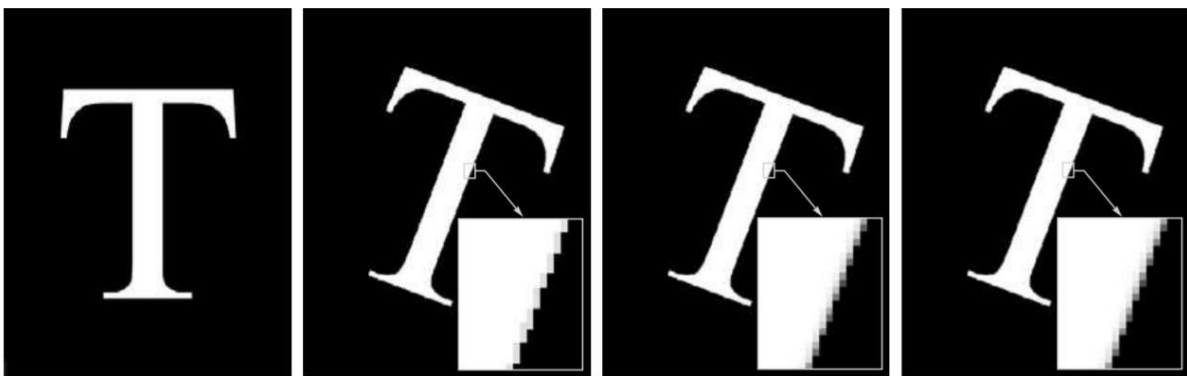
Table 3.1 Affine Transformations

Transformation Name	Affine Matrix, T	Coordinate Equations	Example
Identity	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{aligned} x &= v \\ y &= w \end{aligned}$	
Scaling	$\begin{bmatrix} c_x & 0 & 0 \\ 0 & c_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{aligned} x &= c_x v \\ y &= c_y w \end{aligned}$	
Rotation	$\begin{bmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{aligned} x &= v \cos \theta - w \sin \theta \\ y &= v \sin \theta + w \cos \theta \end{aligned}$	
Translation	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix}$	$\begin{aligned} x &= v + t_x \\ y &= w + t_y \end{aligned}$	
Shear (vertical)	$\begin{bmatrix} 1 & 0 & 0 \\ s_v & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{aligned} x &= v + s_v w \\ y &= w \end{aligned}$	
Shear (horizontal)	$\begin{bmatrix} 1 & s_h & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{aligned} x &= v \\ y &= s_h v + w \end{aligned}$	

In practice, we can use intensity interpolation in two basic ways.

- The first, called a forward mapping, consists of scanning the pixels of the input image and, at each location, , computing the spatial location, (x, y), of the corresponding pixel in the output image directly.

- A problem with the forward mapping approach is that two or more pixels in the input image can be transformed to the same location in the output image, raising the question of how to combine multiple output values into a single output pixel
- In addition, it is possible that some output locations may not be assigned a pixel at all
- The second approach, called inverse mapping, scans the output pixel locations and, at each location, (x, y) , computes the corresponding location in the input image using $(v, w) = T^{-1}(x, y)$
- It then interpolates among the nearest input pixels to determine the intensity of the output pixel value
- Inverse mappings are more efficient to implement than forward mappings and are used in numerous commercial implementations of spatial transformations



a b c d

FIGURE 3.9 (a) A 300 dpi image of the letter T. (b) Image rotated 21° using nearest neighbor interpolation to assign intensity values to the spatially transformed pixels. (c) Image rotated 21° using bilinear interpolation. (d) Image rotated 21° using bicubic interpolation. The enlarged sections show edge detail for the three interpolation approaches.