Digital Image Processing

Abhijit Amrendra Kumar

August 2023

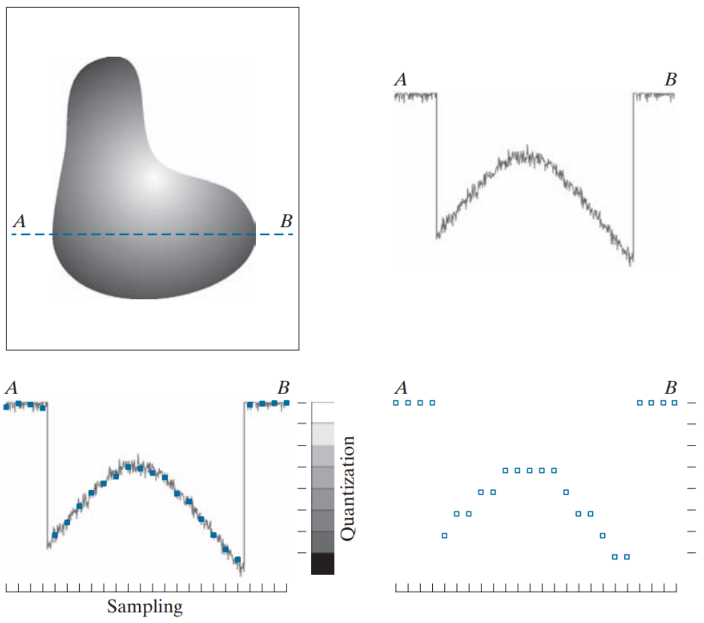
Chapter 1

Image Warping and Alignment

1.1 Basics

A digital image, which is a version of the visual stimulus sampled at discrete locations with discretized values, can be regarded as a function $I = f(x, y)$, where (x, y) are the spatial integer coordinates in typically a rectangular domain $\Omega = [0, W - 1] \times [0, H - 1]$. Each ordered pair (x, y) denote a pixel, which is generally a square. Pixel dimensions relate to the spatial resolution of the light-collecting sensor of a camera.

The actual visual signal is analog, but digital cameras capture a discrete version of it, and also quantize the intensity values. In a typical grayscale image, the intensity values lie from $0 \rightarrow 255$



1.2 Alignment

Consider images I_1, I_2 of a scene acquired through different viewpoints. I_1 and I_2 are said to be aligned iff for every (x, y) in the domain Ω, the pixels at (x, y) in I_1 and I_2 are in physical correspondence.

If not, the images are said to be misalgined with respect to each other, or we say there is relative motion between the images. Image alignment is the process of correcting for the relative motion between I_1 and I_2.

1.3 Motion Models

Let us denote the coordinates in I_1 as (x_1, y_1), and those in I_2 as (x_2, y_2).

Let's first consider **translation**.

$$\forall (x_1, y_1) \in \Omega, \exists t_x, t_y, x_2 = x_1 + t_x, y_2 = y_1 + t_y \tag{1.1}$$

$$\implies \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \tag{1.2}$$

Next, let's consider rotation about a point (x_c, y_c) anti-clockwise through angle θ

$$x_2 = (x_1 - x_c)\cos\theta - (y_1 - y_c)\sin\theta + x_c \tag{1.3}$$

$$y_2 = (x_1 - x_c)\sin\theta + (y_1 - y_c)\cos\theta + y_c \tag{1.4}$$

$$\implies \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & x_c \\ \sin\theta & \cos\theta & y_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 - x_c \\ y_1 - y_c \\ 1 \end{bmatrix} \tag{1.5}$$

Combining the two motions, we obtain

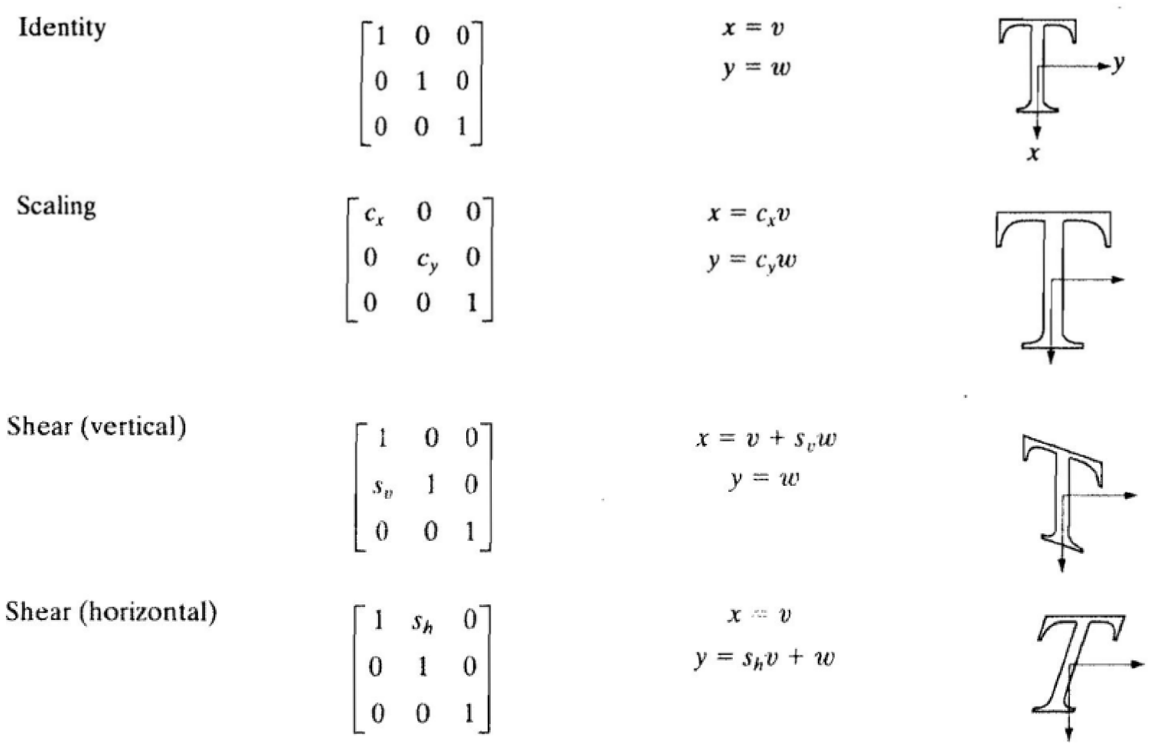
$$x_2 = (x_1 - x_c)\cos\theta - (y_1 - y_c)\sin\theta + t_x \tag{1.6}$$

$$y_2 = (x_1 - x_c)\sin\theta + (y_1 - y_c)\cos\theta + t_y \tag{1.7}$$

$$\implies \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 - x_c \\ y_1 - y_c \\ 1 \end{bmatrix} \tag{1.8}$$

If the 2×2 matrix $\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$ is rank-deficient, we term the transformation as an **affine transformation**, since it will transform 2D figures into a line/point.

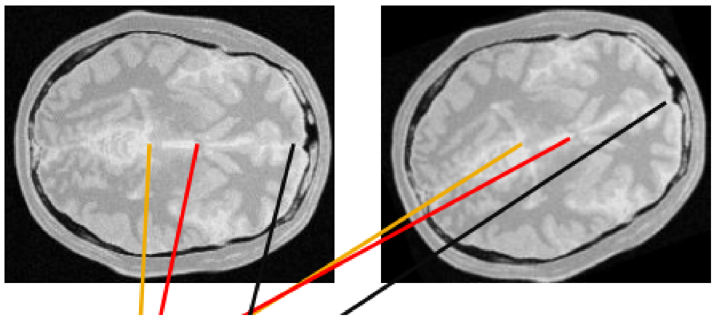
Note: Composition of multiple types of motion is given by the multiplication of their corresponding matrices. However, most motion compositions are not commutative.



Identity	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v$ $y = w$	
Scaling	$\begin{bmatrix} c_x & 0 & 0 \\ 0 & c_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = c_x v$ $y = c_y w$	
Shear (vertical)	$\begin{bmatrix} 1 & 0 & 0 \\ s_v & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v + s_v w$ $y = w$	
Shear (horizontal)	$\begin{bmatrix} 1 & s_h & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v$ $y = s_h v + w$	

1.4 Image Alignment: Control Points

This method involves marking pairs of physically corresponding points on two images, either manually by an expert or automatically using geometric properties. Then using these points, the transformation matrix can be computed, and the whole image can be transformed to align with the original image.



The matrix equation obtained by this method is as follows:

$$\begin{bmatrix} x_{21} & x_{22} & \dots & x_{2N} \\ y_{21} & y_{22} & \dots & y_{2N} \\ 1 & 1 & \dots & 1 \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} & t_x \\ A_{21} & A_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1N} \\ y_{11} & y_{12} & \dots & y_{1N} \\ 1 & 1 & \dots & 1 \end{bmatrix}$$

In shorthand, we can denote it as

$$P_2 = TP_1$$

Since P_1 can be possibly rank deficient, we can instead minimize $||P_2 - TP_1||_2^2$, the solution to which is

$$T = P_2 \,\text{pinv}(P_1)$$

However, this method is always not feasible, because

- it requires manual intervention
- it is error prone
- there are automatic methods like the SIFT technique to finding matching control points.

1.5 Image Alignment: Mean Squared Error

The mean squared error between 2 images is given by

$$\text{MSSD} = \frac{1}{N} \sum_{x,y \in \Omega} (I_1(x,y) - I_2(x,y))^2, \;\; N = \#\text{pixels in field of view}$$

We can modify the equation in the following manner to find the transform matrix T

$$T^* = \underset{T}{\text{argmin}}\, \text{MSSD}(I_1(v), I_2(Tv)), \quad v = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, T = \begin{bmatrix} A_{11} & A_{12} & t_x \\ A_{21} & A_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

Assuming a simple model involving translation and rotation,

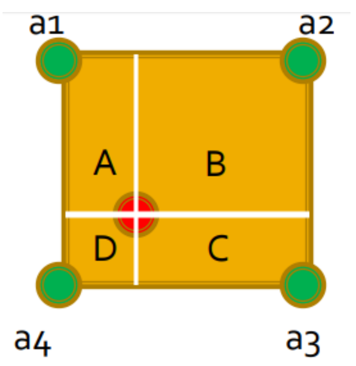
$$T = \begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

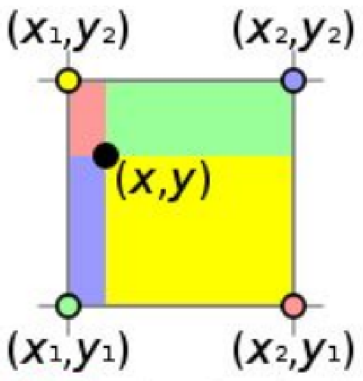
The simplest (although the worst possible way) to solve this would be to iterate over ranges for θ, t_x, t_y.

1.5.1 Image Warping

Applying the transformation T over an image maps each coordinate vector $v = [x, y]$ to the new coordinate Tv in the new image, where $x, y \in \mathbb{Z}$. However, it is very likely that $Tv = [x', y']$ does not consist of integer coordinates. This transformation, also known as **forward warping**, can lead to holes in the new image, and can also lead to multiple intensity values if the image is scaled down.

Instead, we can use **reverse warping**, which involves taking $v' = [x', y'], x', y' \in \mathbb{Z}$ in the new image, and applying the inverse transformation $T^{-1}v$ to get the coordinates in the original image. Here, if $T^{-1}v$ does not consist of integer coordinates, we can use methods like bilinear interpolation or nearest neighbour to calculate the intensity value to be used from the original image.





Nearest Neighbour: Pick a_4

Bilinear interpolation: Use the following value (weights being the areas of the 4 regions)

$$\frac{Ba_4 + Aa_3 + Ca_1 + Da_2}{A + B + C + D} = Ba_4 + Aa_3 + Ca_1 + Da_2 \qquad (A + B + C + D = 1 \text{ for unit pixel})$$

Essentially, we have to find the following function

$$f(x, y) = a_0 + a_1 x + a_2 y + a_3 xy$$

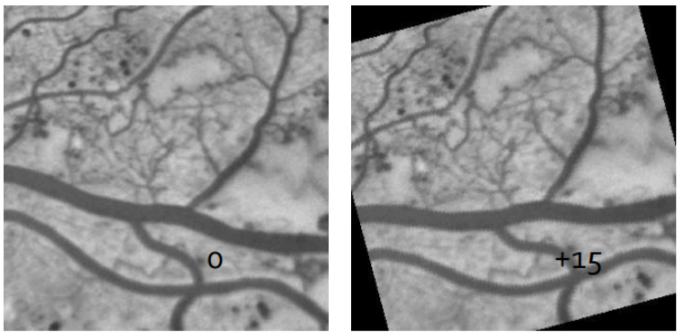
To determine the coefficients, we can solve the following equation

$$\begin{bmatrix} 1 & x_1 & y_1 & x_1y_1 \\ 1 & x_1 & y_2 & x_1y_2 \\ 1 & x_2 & y_1 & x_2y_1 \\ 1 & x_2 & y_2 & x_2y_2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} f(x_1, y_1) \\ f(x_1, y_2) \\ f(x_2, y_1) \\ f(x_2, y_2) \end{bmatrix}$$

$$\implies \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 1 & x_1 & y_1 & x_1y_1 \\ 1 & x_1 & y_2 & x_1y_2 \\ 1 & x_2 & y_1 & x_2y_1 \\ 1 & x_2 & y_2 & x_2y_2 \end{bmatrix}^{-1} \begin{bmatrix} f(x_1, y_1) \\ f(x_1, y_2) \\ f(x_2, y_1) \\ f(x_2, y_2) \end{bmatrix}$$

Note: Alignment with MSSD relies heavily on the fact that the intensity for physically corresponding points is the same. However, we can have images of the same physical entity from different sensors or lighting conditions.

Caution: When an image is rotated, because of the rectangular shape of an image, the rotated image will have some black regions, which denote the area which is outside the field of view.



We need to compute the MSSD only over the overlapping region between the two images.

1.6 Alignment with Differing Intensities

Chapter 2

Edge and Corner Detection

Edge pixels are those at which image intensity changes abruptly.
Edge segments or edges are sets of connected edge pixels

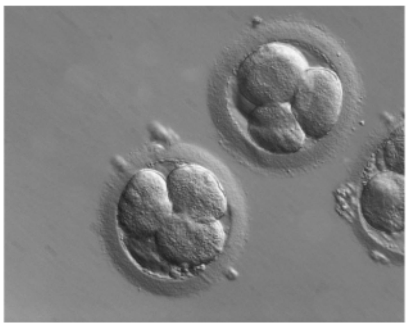
2.1 Digital Derivatives

Along a ramp, first derivative is a non-zero constant, while the second derivative is zero along the ramp (except the start and end).

Chapter 3

Hough Transform

A hough transform is a method used to detect the presence of different types of shapes in an image. This can be extended to detect important features in an image, like roads and buildings in satellite imagery, biological cells in microscope images, etc.



3.1 Line Detection

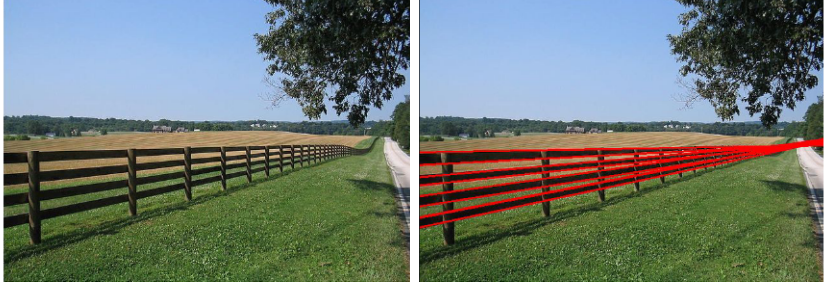
Consider a point (x_i, y_i) in the image. A line passing through it can be represented by $y_i = ax_i + b$, which is parametrized by (a, b). Thus, each line becomes a point (a, b) in the **parameter space**. The set of all lines passing through (x_i, y_i) will constitute a single line in the parameter space.
However, this slope intercept form for the parameter space can prove challenging to execute, since slopes can be unbounded. Instead, we can use the normal representation

$$x \cos\theta + y \sin\theta = p$$

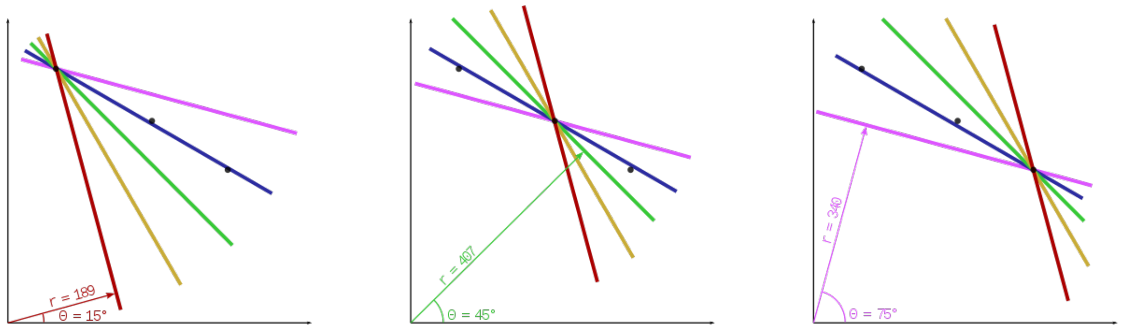
Thus, a single point in the XY space corresponds to a **sinusoid** in this parameter space. The line joining two points in the XY space is represented by a point obtained from the intersection of their corresponding two sinusoids in the parameter space.

3.1.1 Algorithm

- Divide the entire (p, θ) space into a 2D array of bins.
- For every point (x_k, y_k) in the XY space, and for every value of θ from -90 to 90, compute the corresponding value of p using $p = x_k \cos\theta + y_k \sin\theta$.
- Approximate this value to the closest allowed cell value in the (p, θ) space.
- Increment the frequency count of that cell by 1 (voting).
- Finally, pick those lines having maximum votes



Here's an example from Wikipedia



θ	r
15	189.0
30	282.0
45	355.7
60	407.3
75	429.4

θ	r
15	318.5
30	376.8
45	407.3
60	409.8
75	385.3

θ	r
15	419.0
30	443.6
45	438.4
60	402.9
75	340.1

3.1.2 Speed-up Heuristic

Computing the gradient for each point, we know that the edge direction should be approximately perpendicular to the gradient.

Therefore, given the edge direction θ', we can iterate over θ from $\theta' - K$ to $\theta' + K$, K being around 20 degrees.

3.2 Circle and Ellipse Detection

A circle is represented by the equation $(x-a)^2 + (y-b)^2 = r^2$, and so is characterized by 3 parameters (a, b, r). Thus, the parameter space is 3D, and we can use the same iterative algorithm. Similarly, the parameter space for an ellipse whose axes are parallel to the XY axes is 4D, while it is 5D for a general ellipse.

3.3 Pros

- It can even handle cases for a broken shape (gaps in the shape perimeter) caused by occlusions or noise.

3.4 Cons

- Exponential computational complexity in the number of parameters
- It considers a line to be of infinite extent, hence relatively smaller line segments can get missed out.
- Low image resolution can lower the accuracy of this algorithm, as potentially different curves could fall into the same bins. Even very high resolution images can be problematic, as votes of different points lying close to the same curve will get bifurcated. Also, noise worsens the accuracy in the latter case.