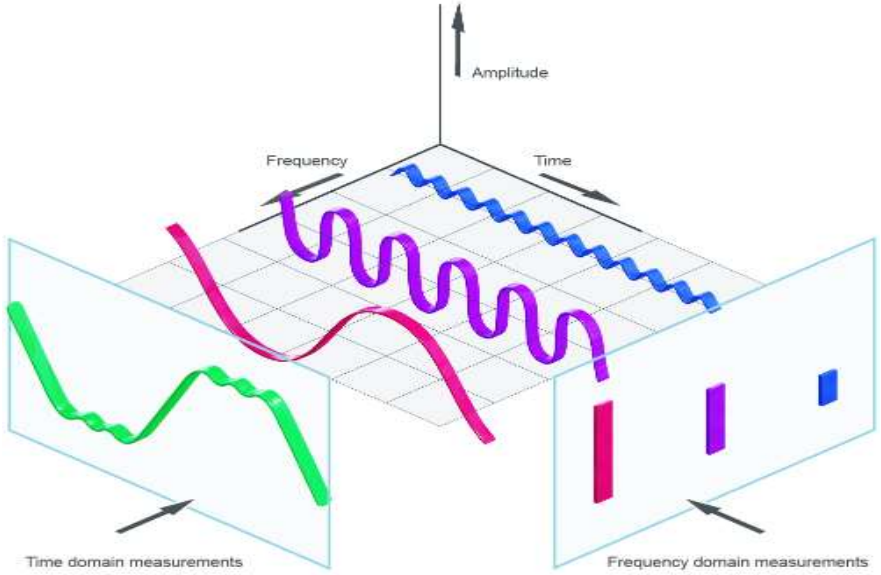
Transform
DCT/DFT/DWT

Yih-Lon Lin (林義隆)
Associate Professor,
Department of Computer Science and Information Engineering,
National Yunlin University of Science and Technology

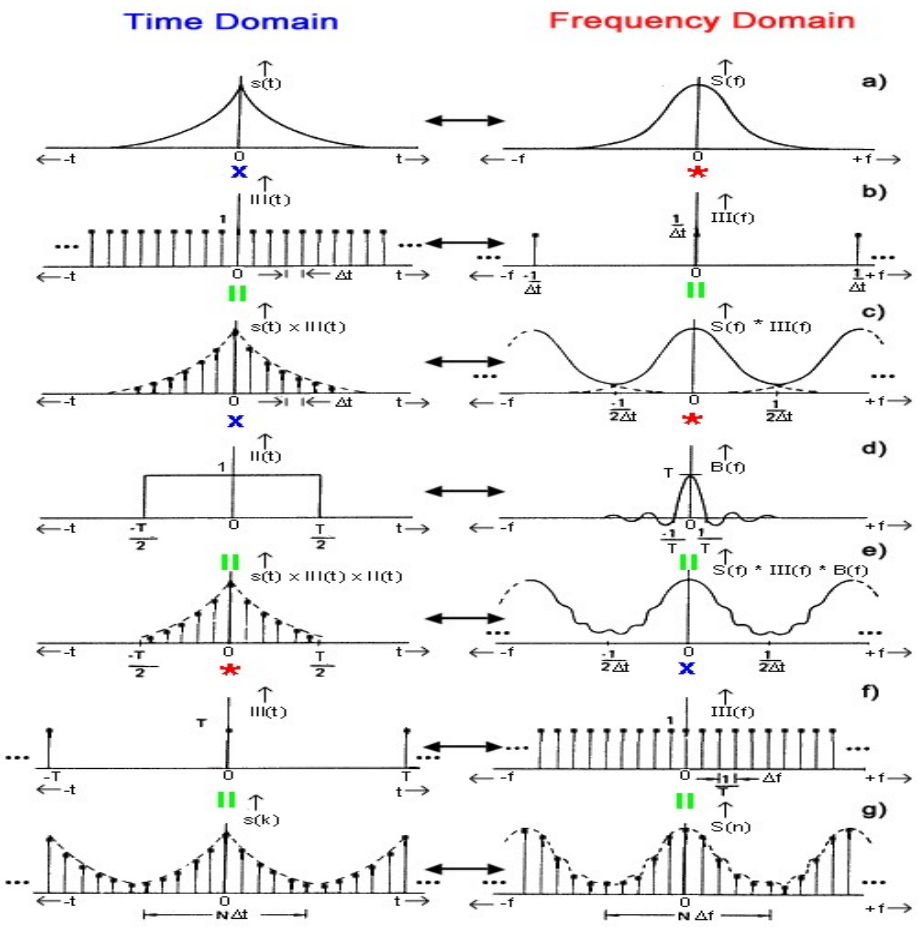
Image Compression

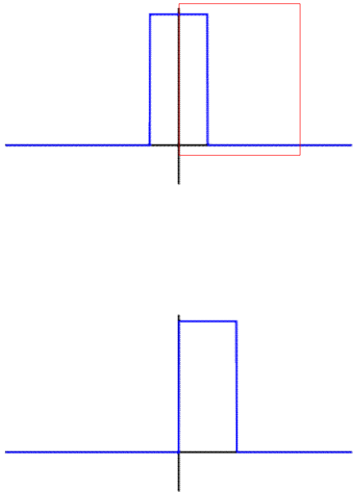
- Discrete Fourier Transform (DFT)
- Discrete Cosine Transform (DCT)
- Discrete Wavelet Transform (DWT)

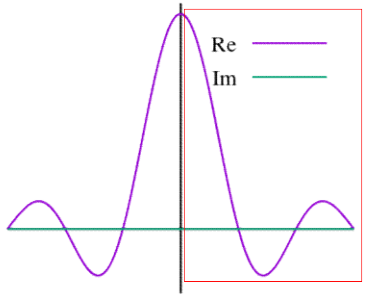


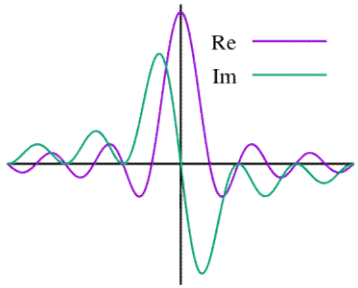
Transform (domain)

Fourier Transform



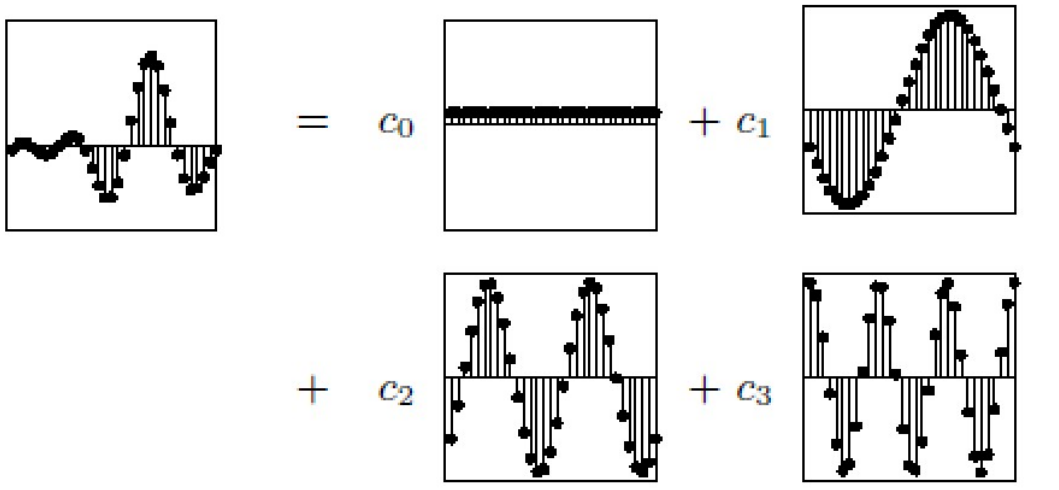


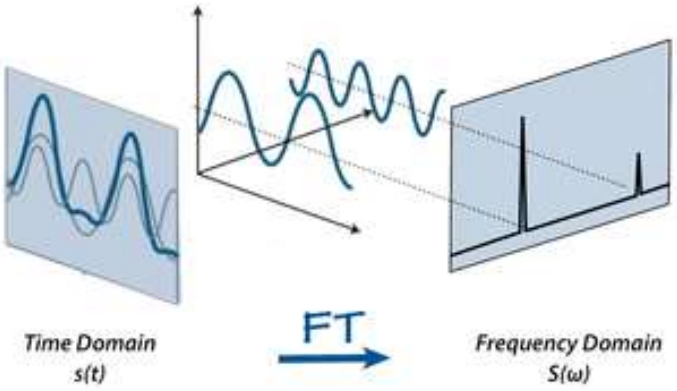


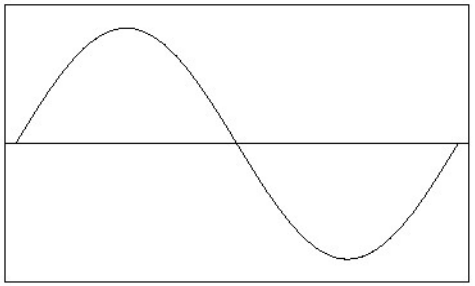


https://www.
rohan.sdsu.edu/~jiracek/DAGSAW/3.5.html

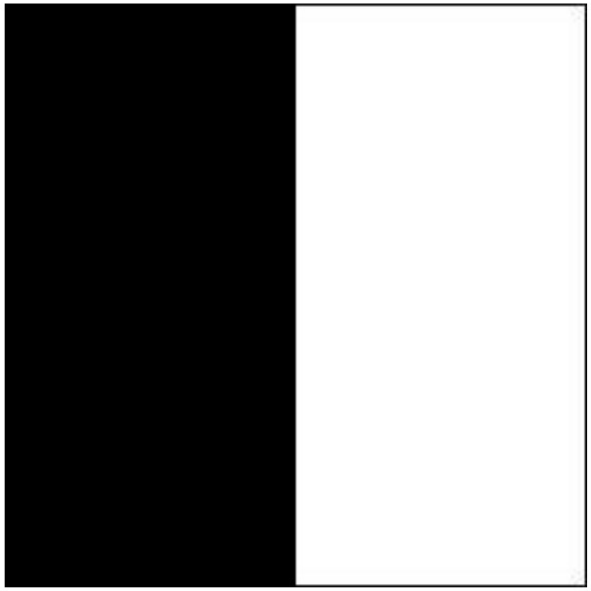
Fourier Transform

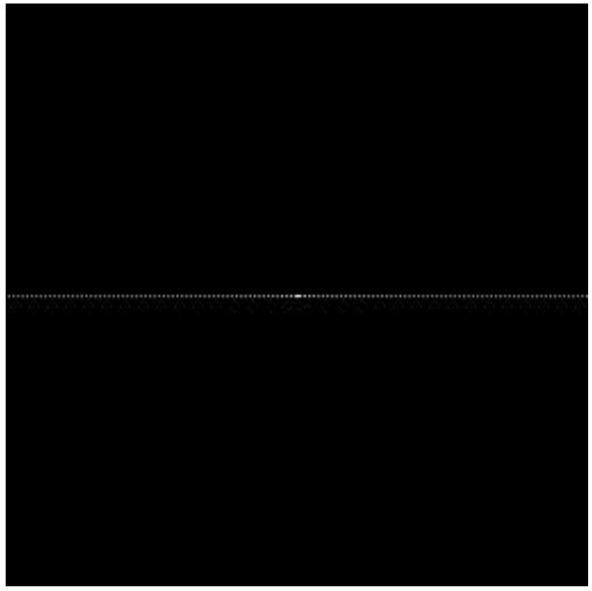




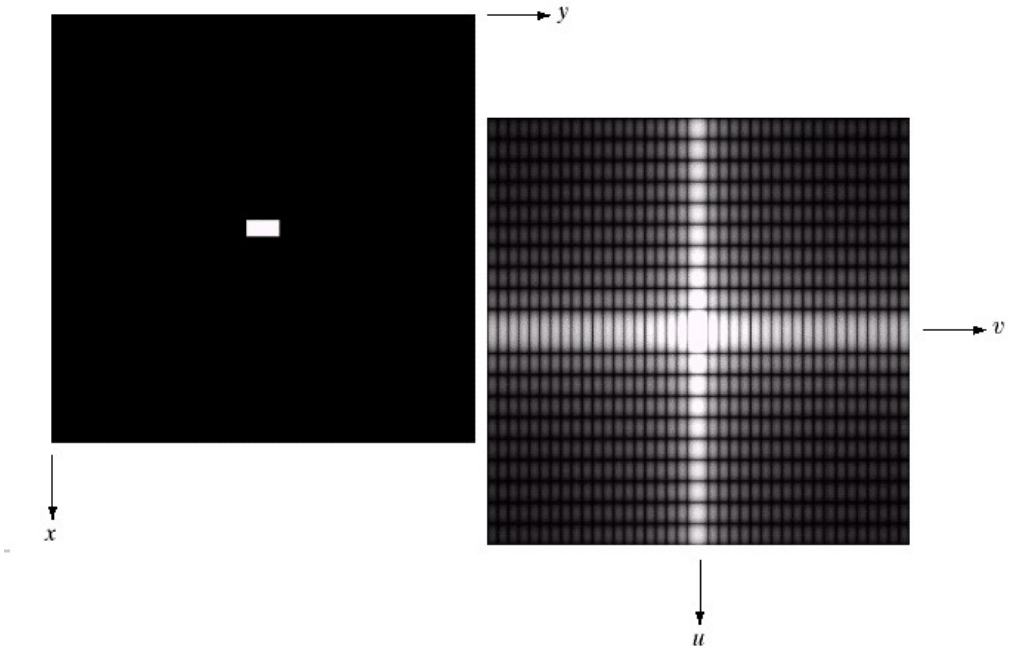


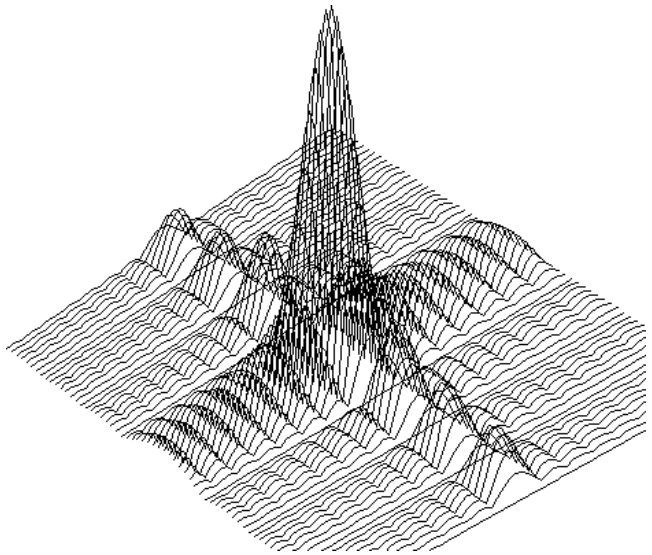
Single edge and its DFT



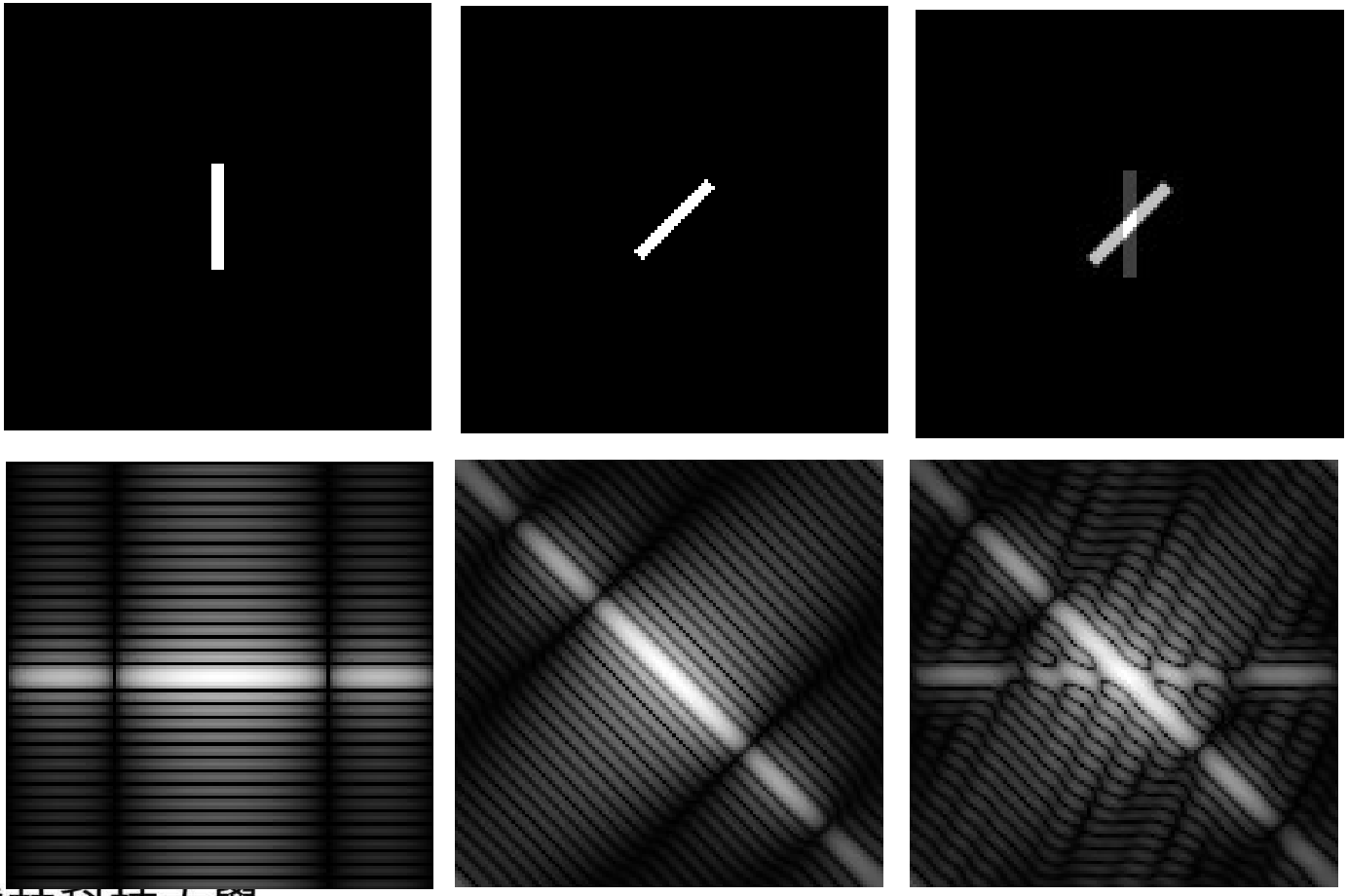


Fourier Transform

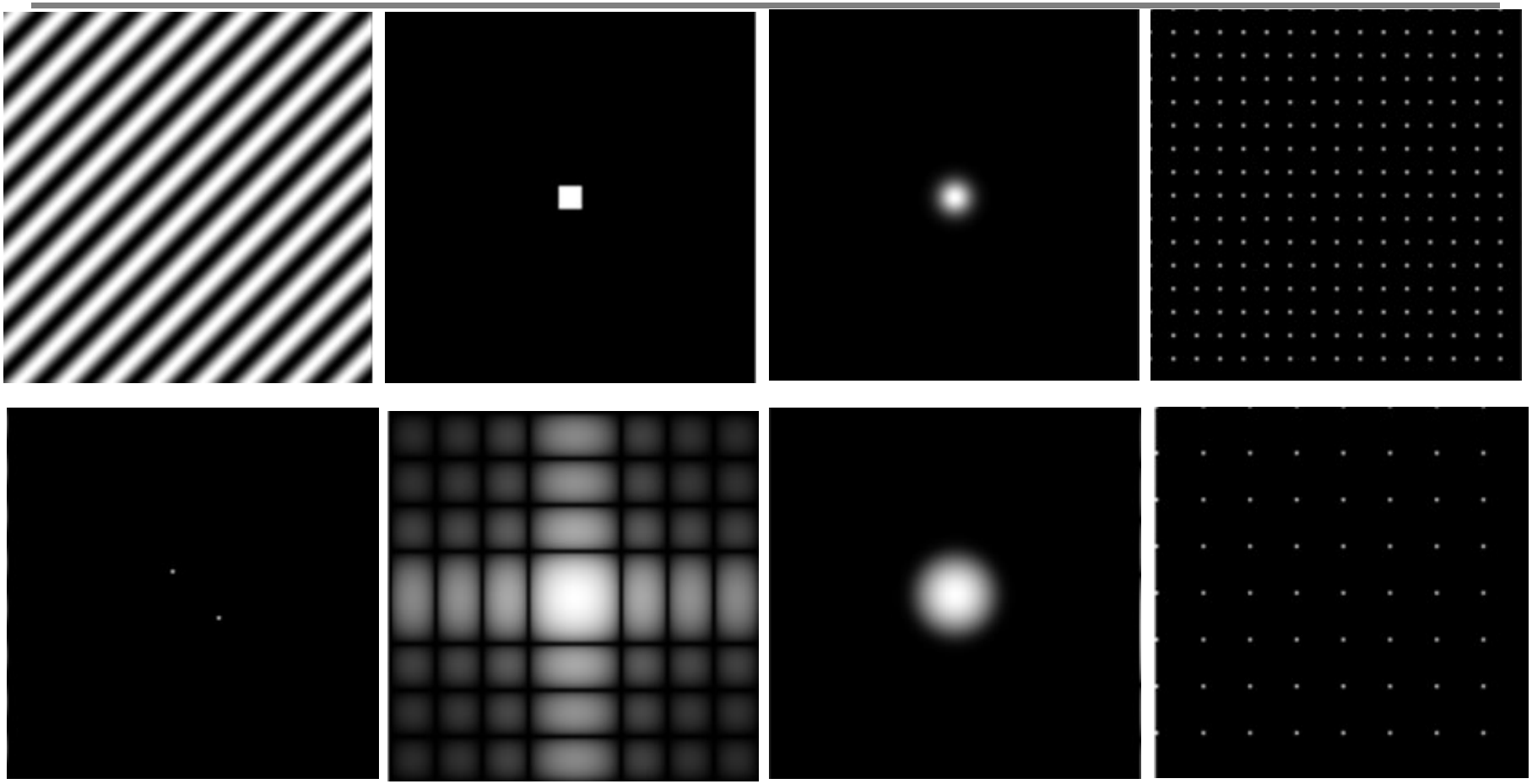




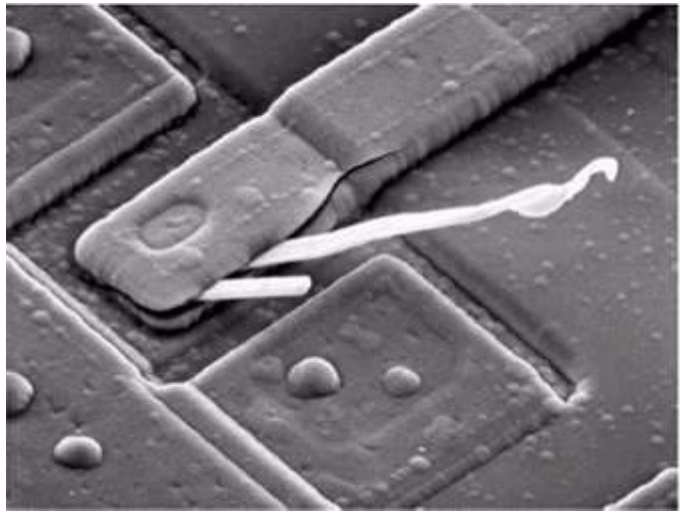
Fourier Transform

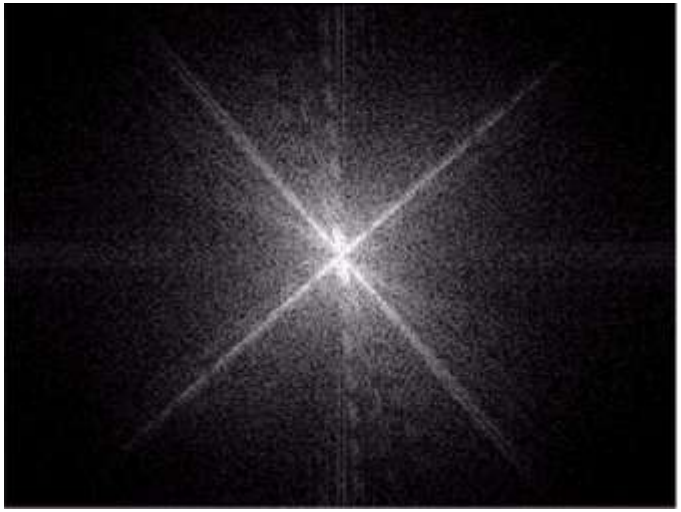


Fourier Transform

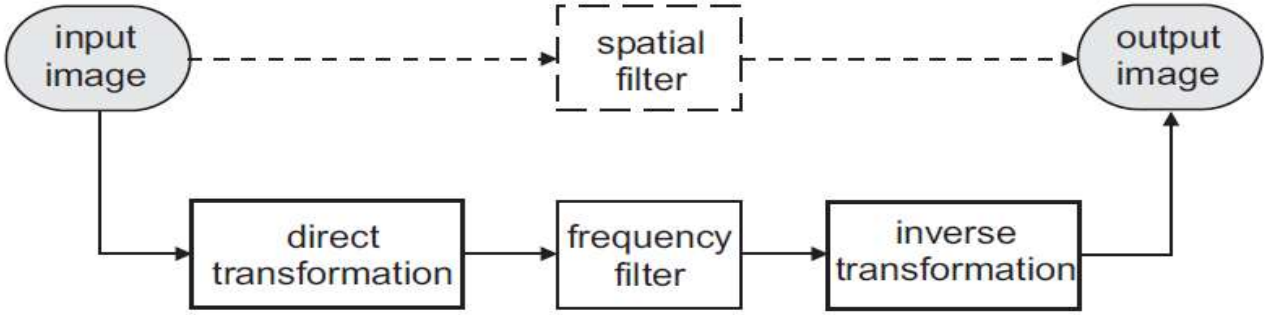


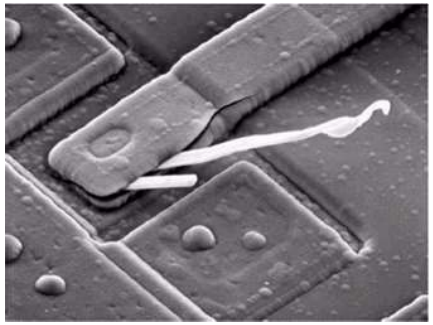
Fourier Transform

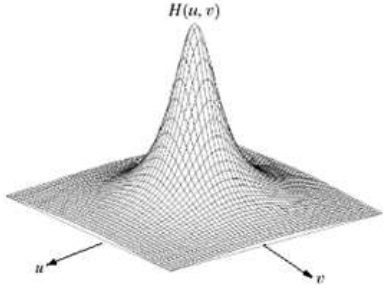


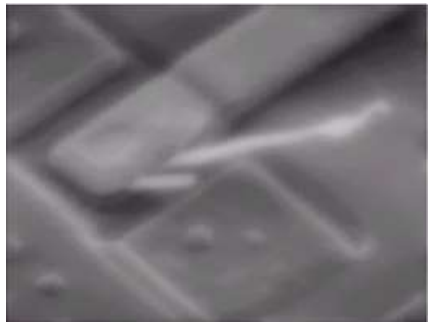


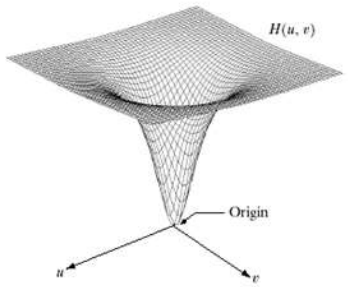
Frequency Filters

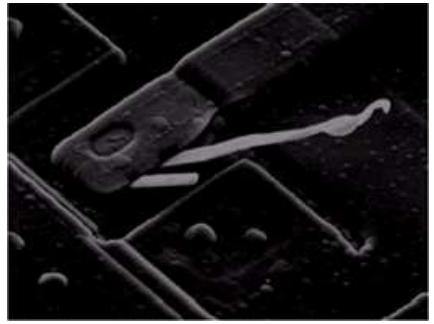




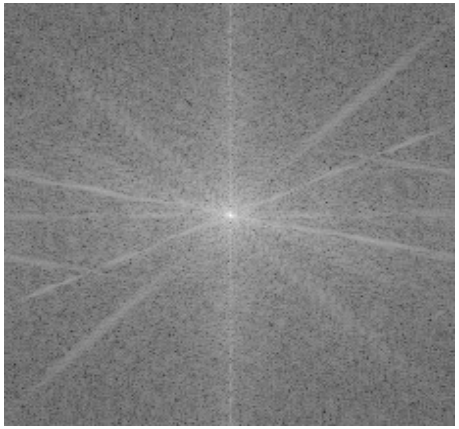


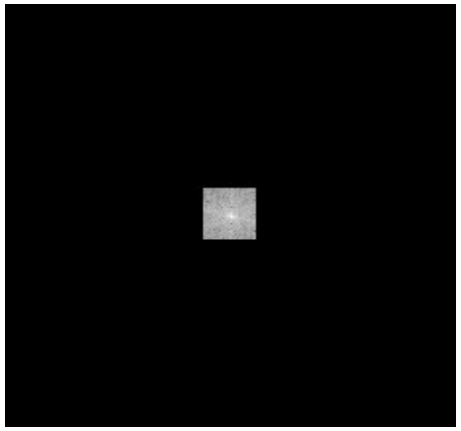




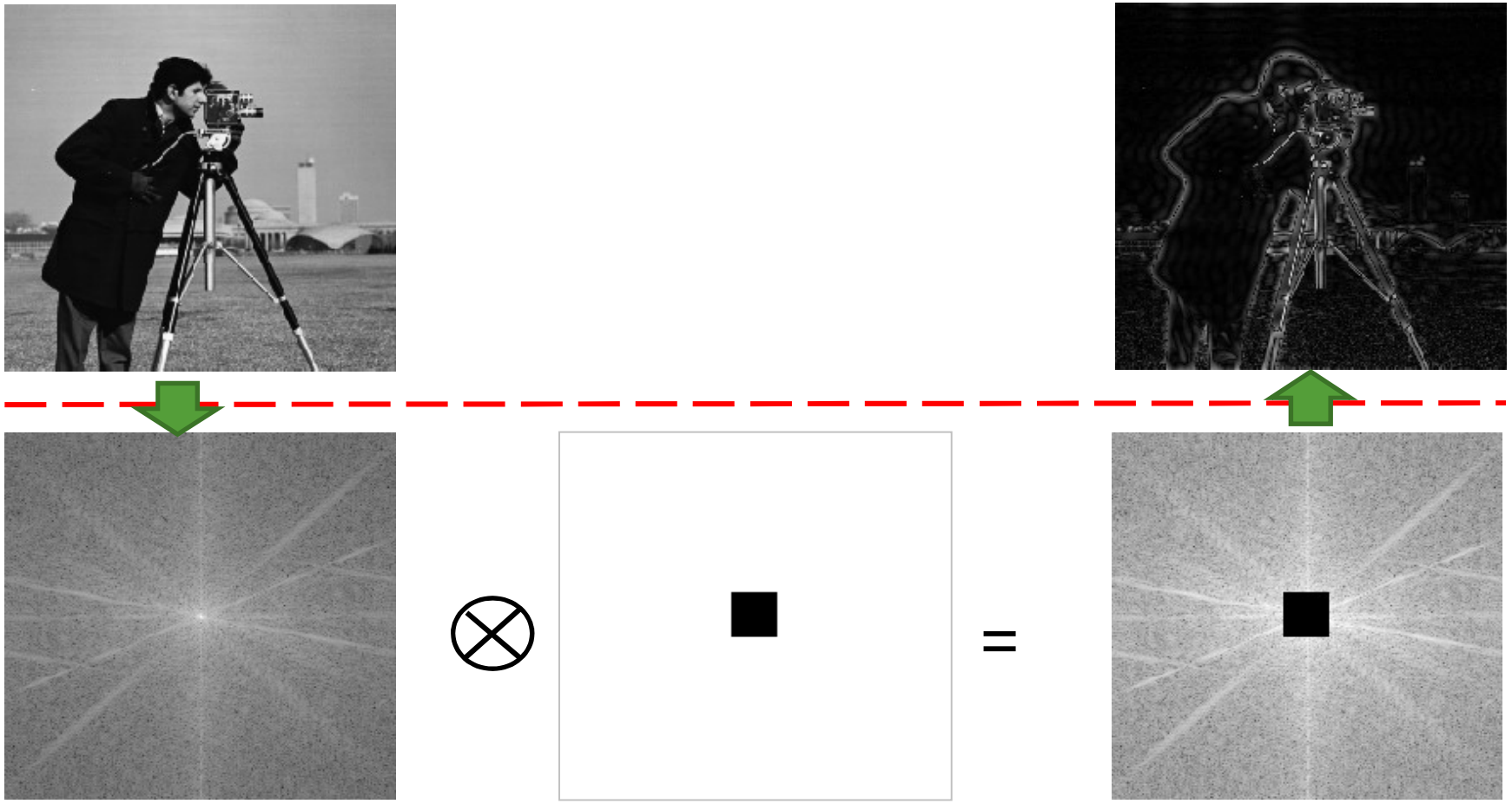


Fourier Transform

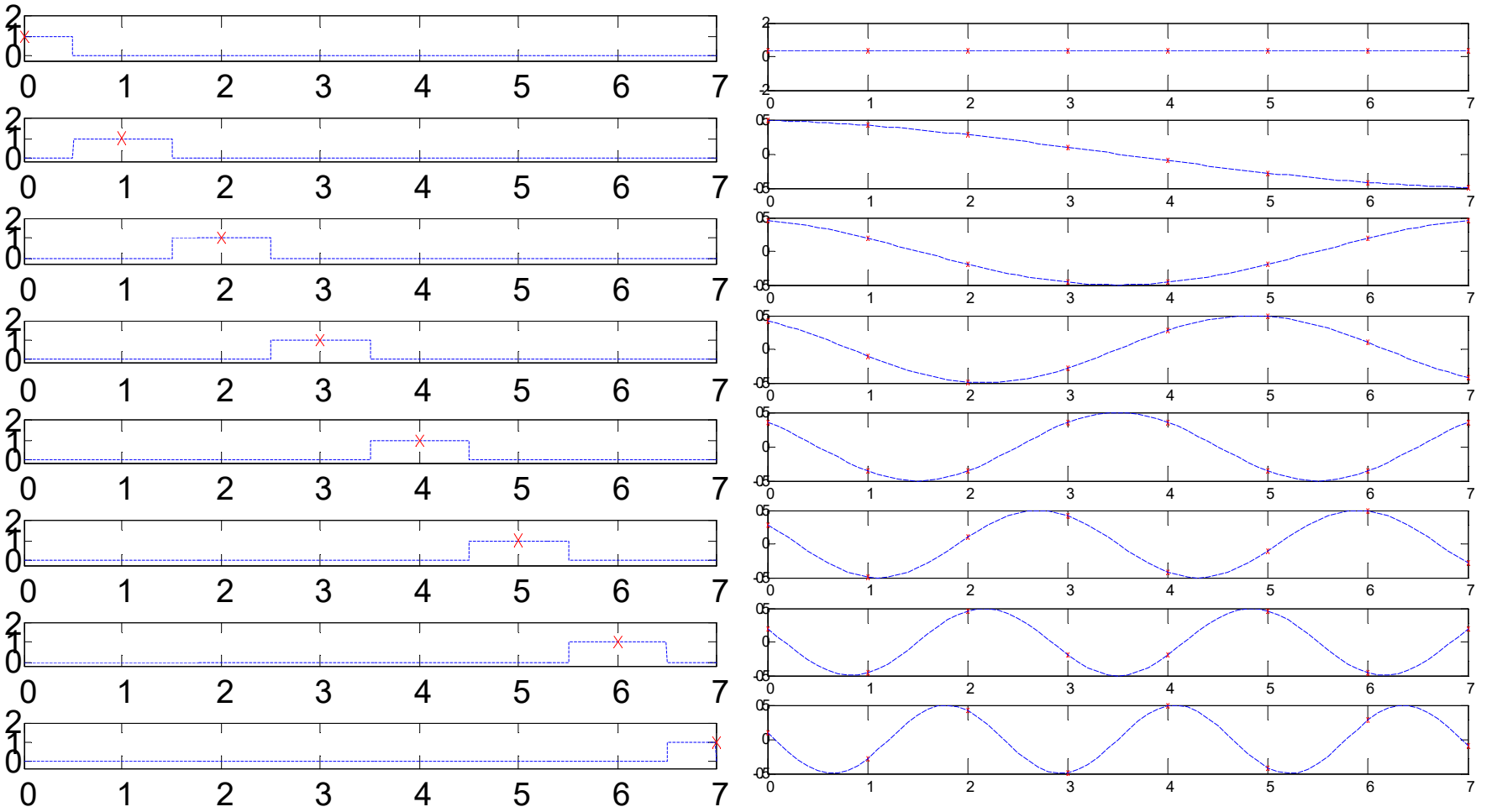




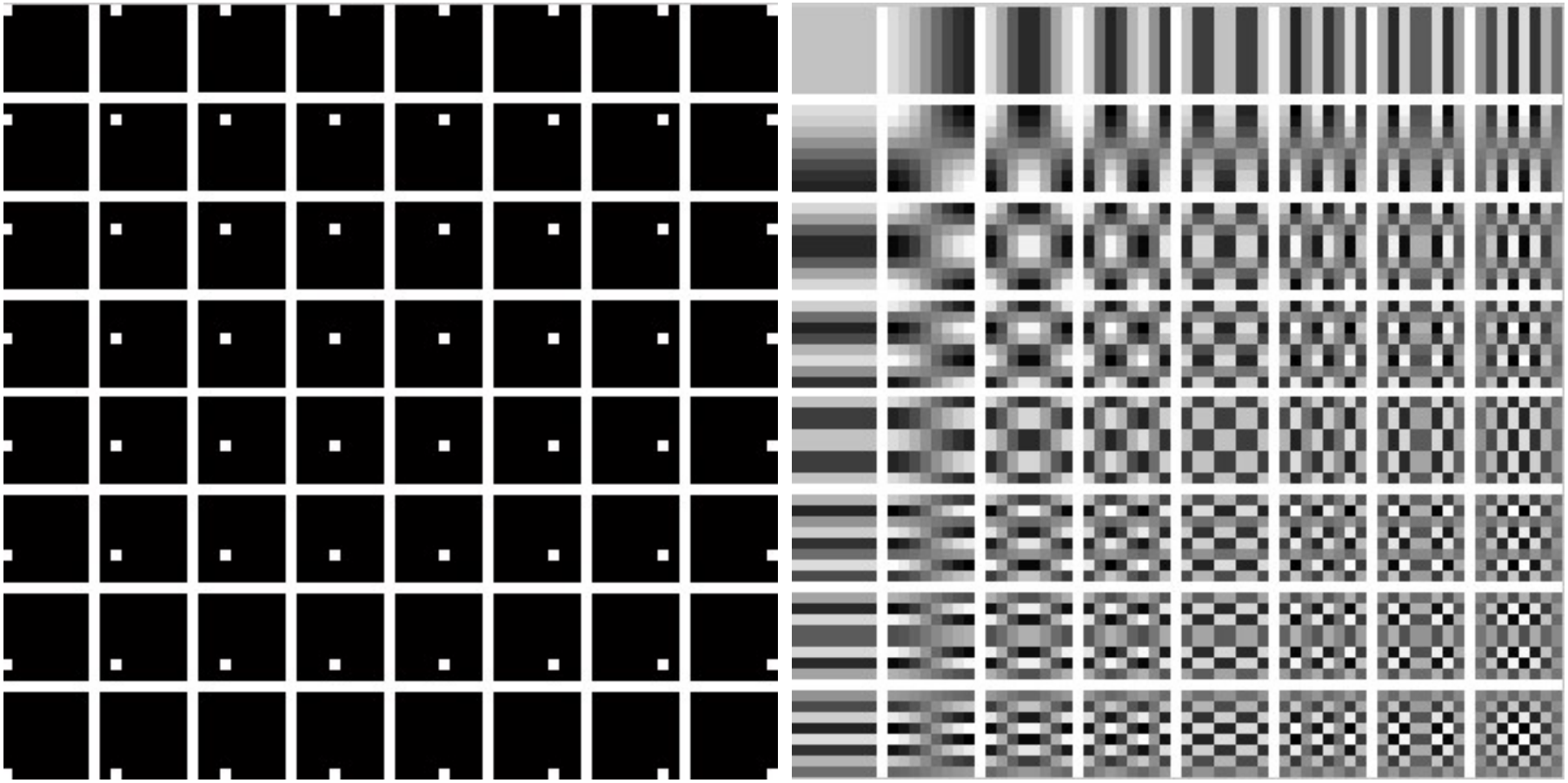
Fourier Transform



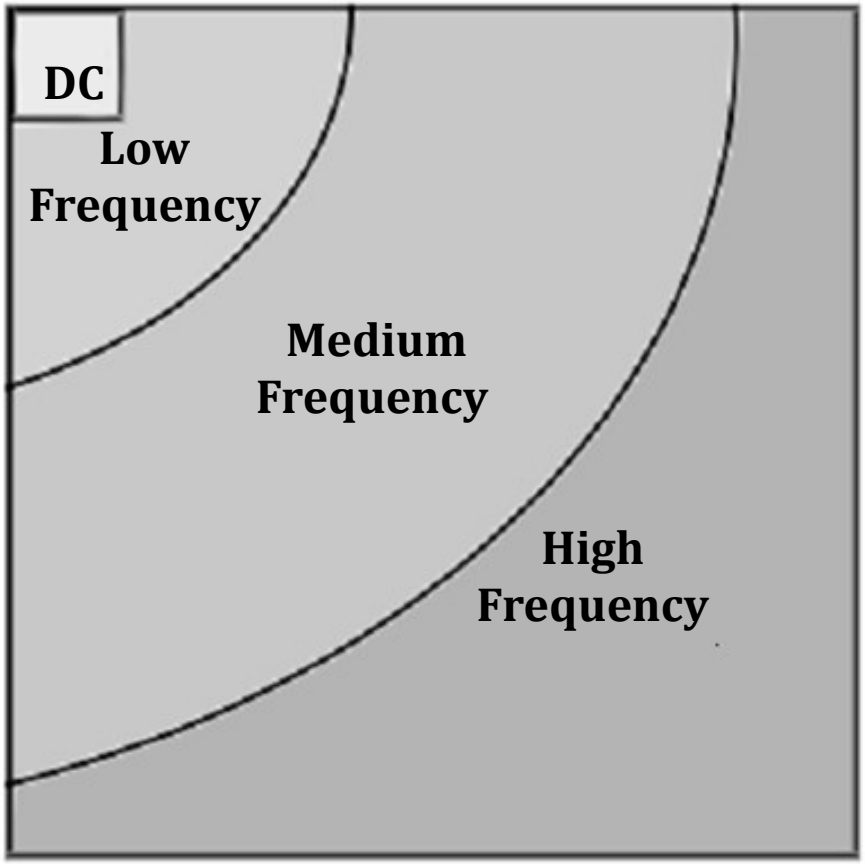
Discrete Cosine Transform



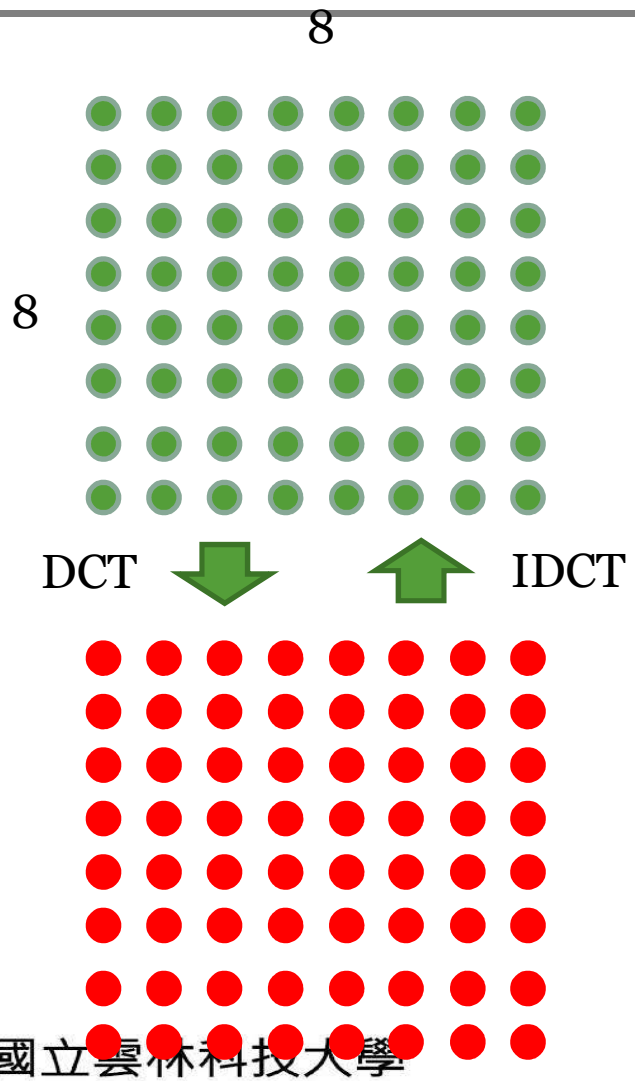
Discrete Cosine Transform

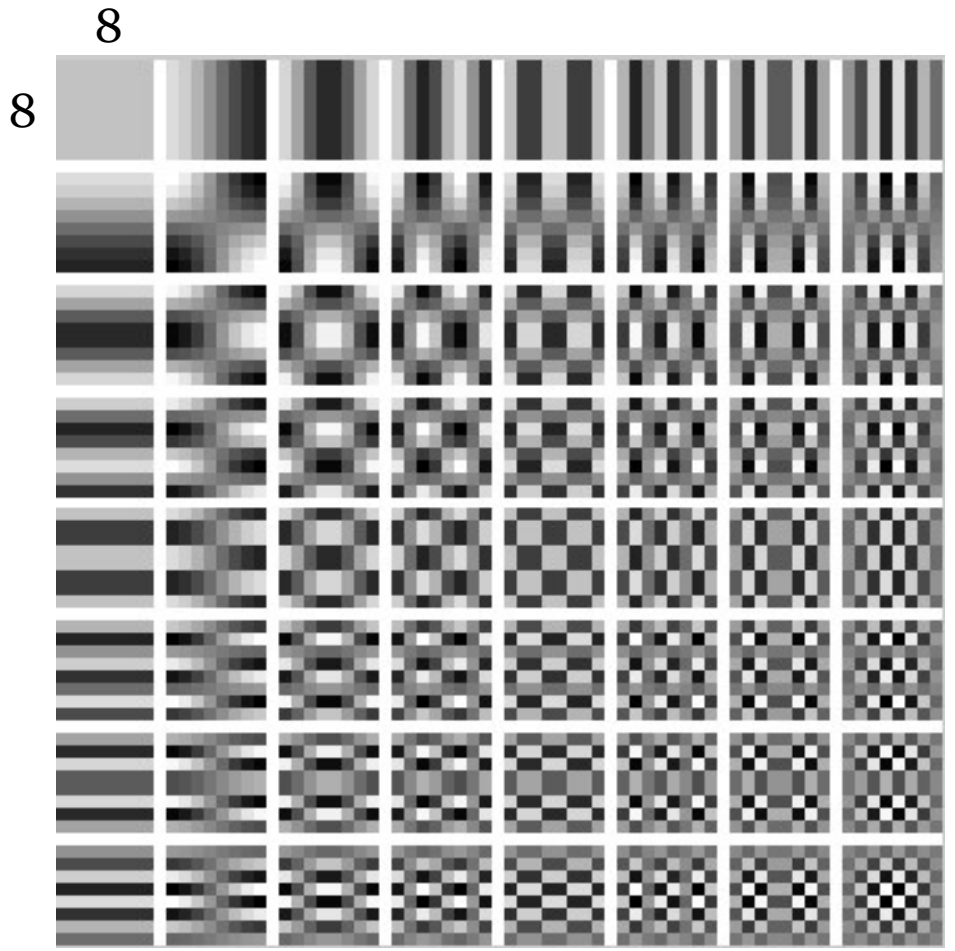


Discrete Cosine Transform

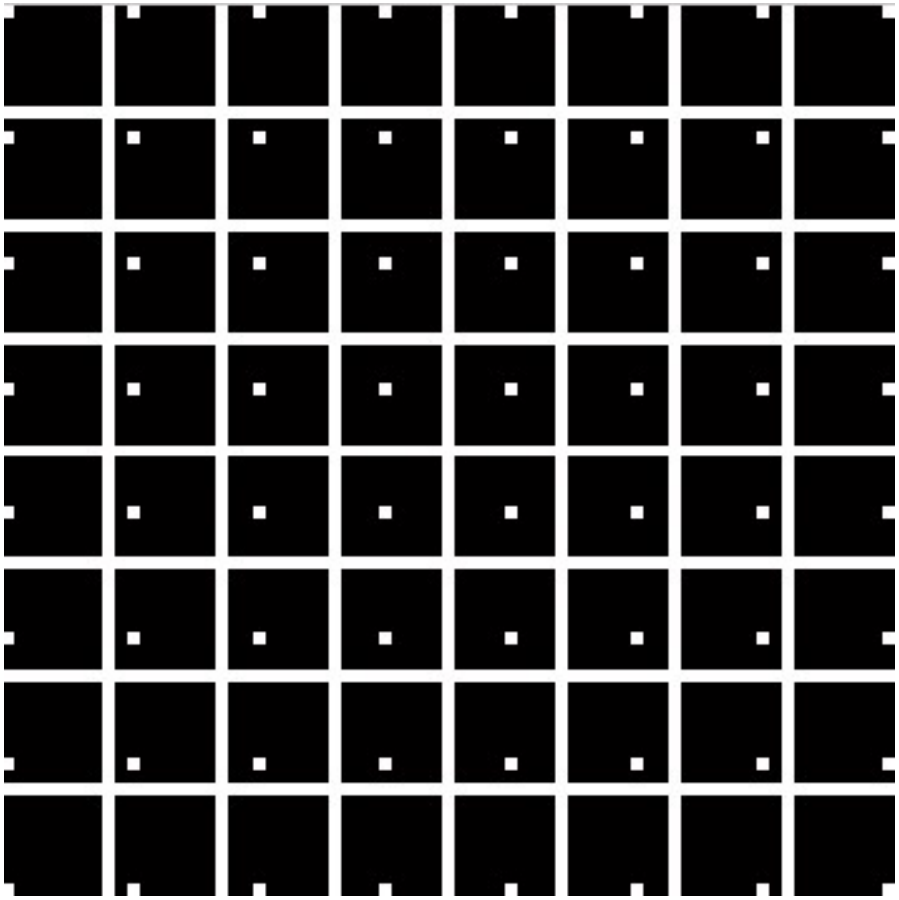


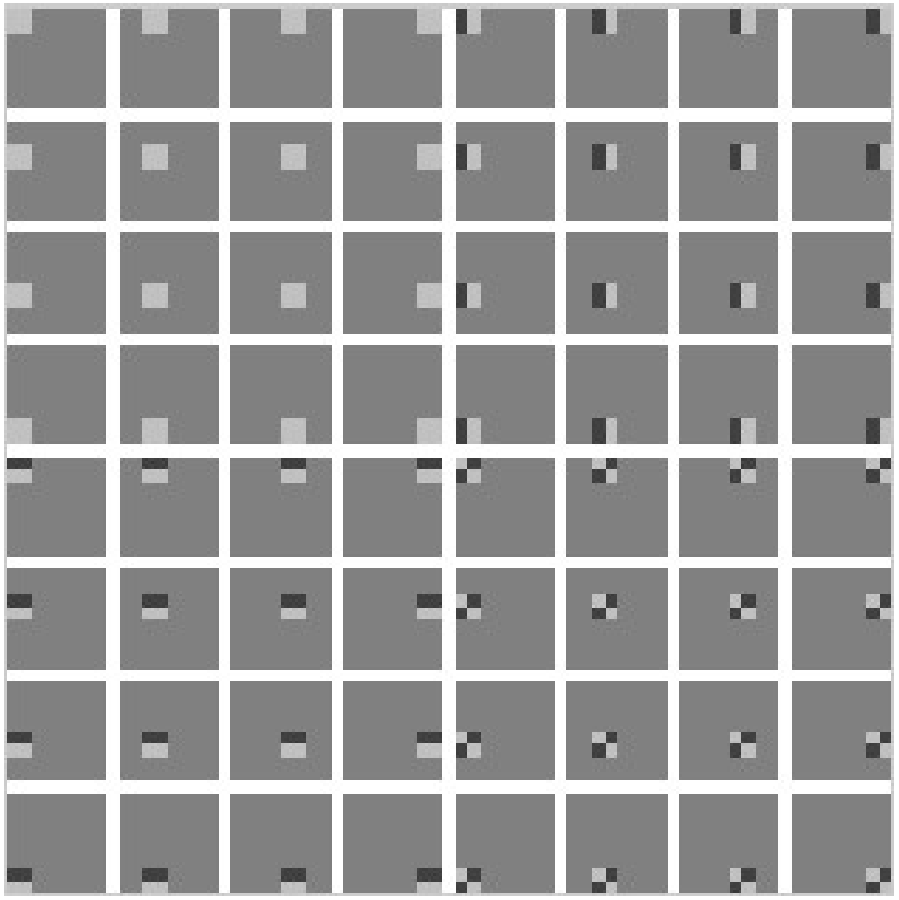
Discrete Cosine Transform





Discrete Wavelet Transform



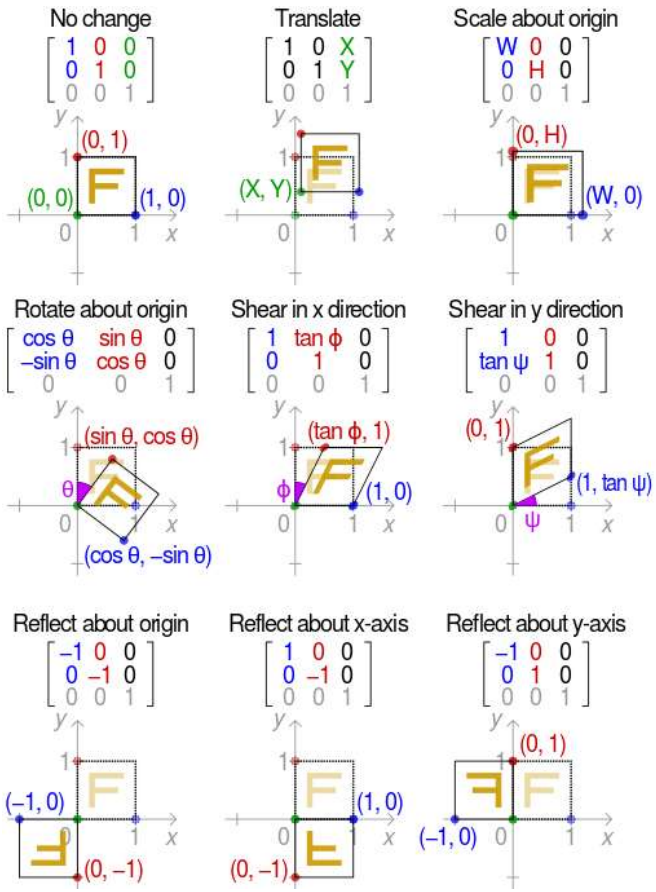


Discrete Wavelet Transform

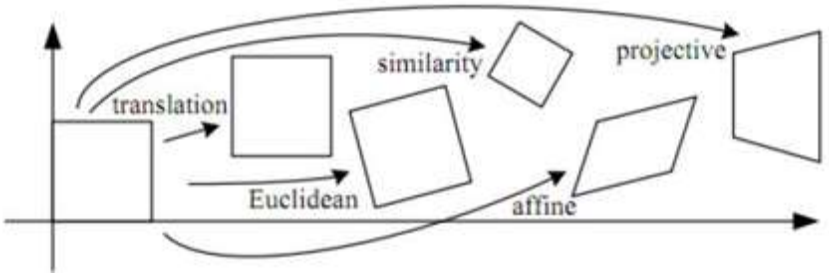
Transforms

- Geometric Transformation
  - Scaling (Resize)
  - Shifting (Translation)
  - Skew
  - Rotation
  - Affine Transformation
  - Project Transformation
  - Perspective Transformation

Transforms



https://en.wikipedia.org/wiki/Affine_transformation



https://publiclab.org/notes/anishshah101/06-02-2014/gsoc-update-leaflet-draw-and-non-affine-transformations

Affine Transformation

- Wrap Parameters

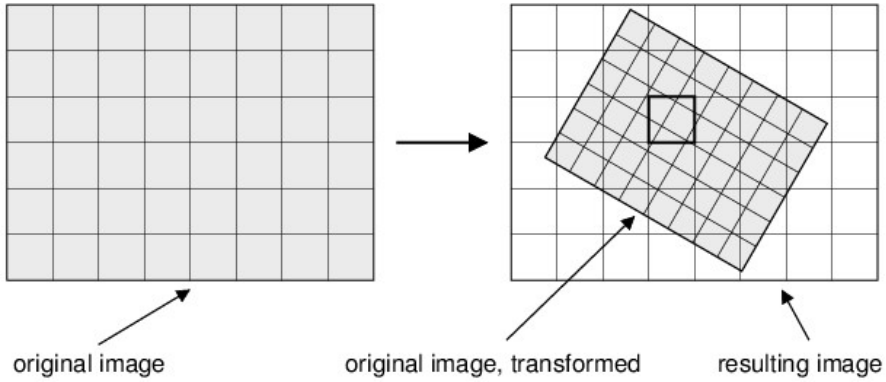
$$\mathbf{W}(\mathbf{x};\mathbf{p}) = \begin{bmatrix} \mathbf{W}_1 \\ \mathbf{W}_2 \end{bmatrix} = \begin{bmatrix} 1+p_1 & p_3 \\ p_2 & 1+p_4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} p_5 \\ p_6 \end{bmatrix}$$

Translation

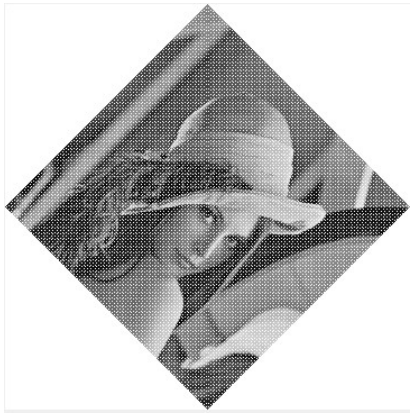
$$M = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix}$$

dst = cv2.warpAffine(img, M, (cols, rows))

Rotation

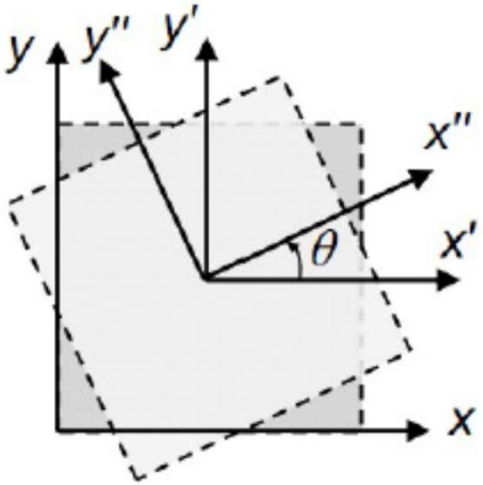

original image
original image, transformed
resulting image


a)
AH 775324 V
original
Detail of a)
b)
AH 775324 V
'nearest_neighbor'
Detail of b)
c)
AH 775324 V
'bilinear'
Detail of c)



Rotation

- cv2. getRotationMatrix2D (center, angle, scale)
  - Translation
  - Rotation
  - Translation



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

$$\begin{bmatrix} \alpha & \beta & (1-\alpha)\cdot center.x - \beta \cdot center.y \\ -\beta & \alpha & \beta \cdot center.x + (1-\alpha)\cdot center.y \end{bmatrix}$$

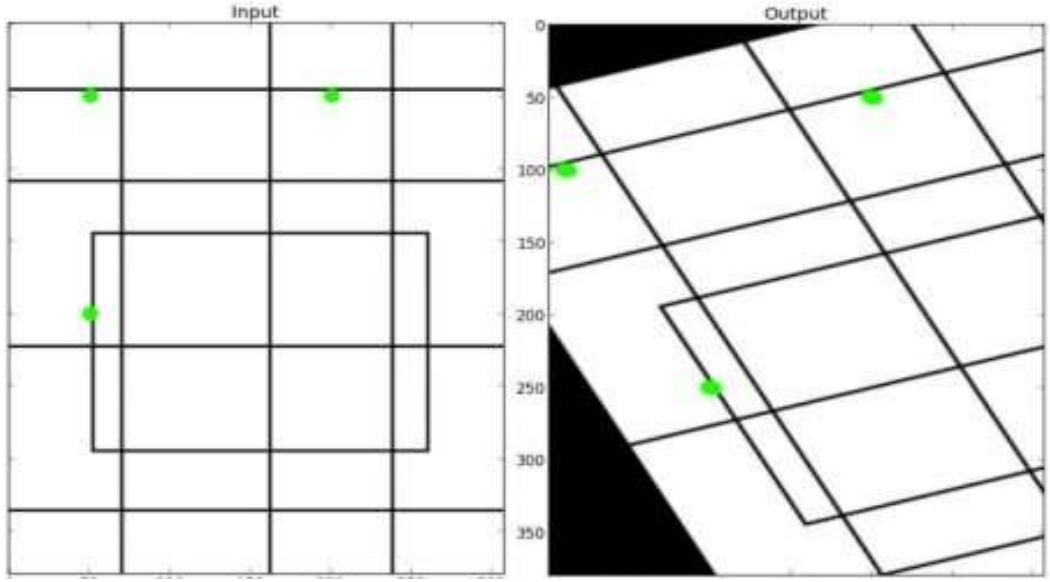
$$\alpha = scale \cdot \cos\theta,$$
$$\beta = scale \cdot \sin\theta$$

Rotation

```
M= cv2.getRotationMatrix2D((cols/2, rows/2), 45, 1)
rotation = cv2.warpAffine(img, M, (cols, rows))
```

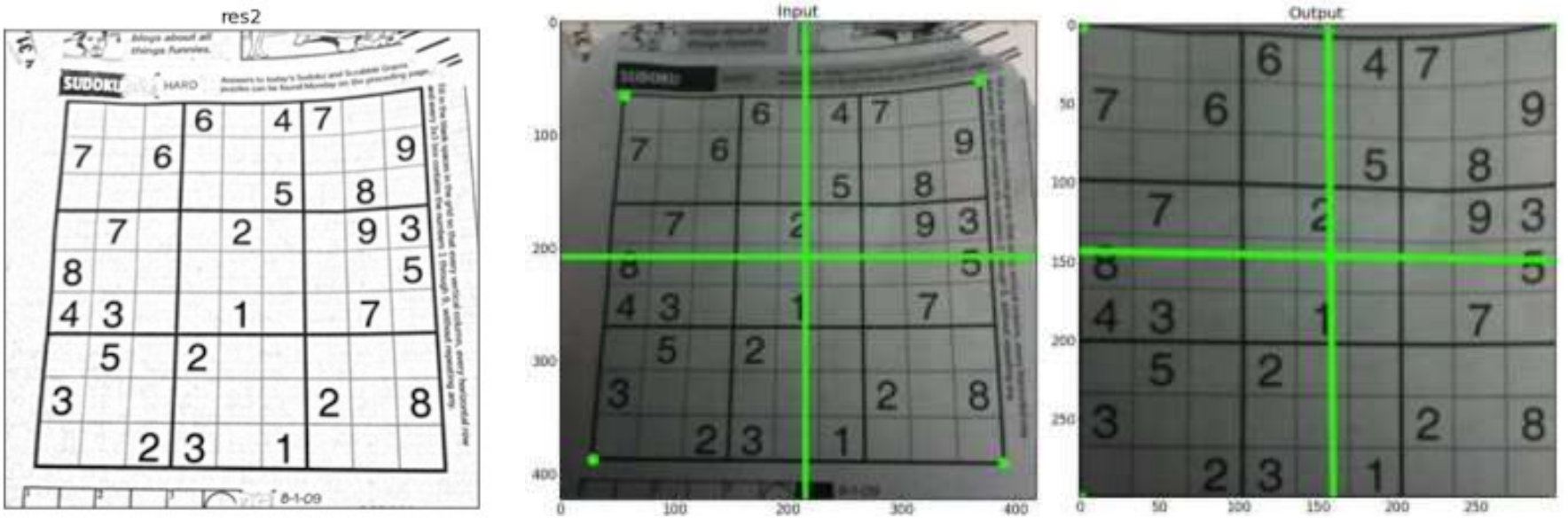
Affine Transformation

- pts1 = np.float32([[50,50],[200,50],[50,200]])
- pts2 = np.float32([[10,100],[200,50],[100,250]])
- M = cv2.getAffineTransform(pts1,pts2) #size: 2x3



Perspective Transformation

Perspective Transformation



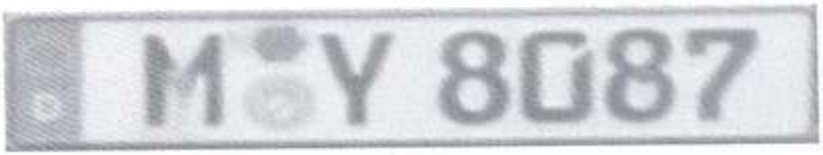

```
pts1 = np.float32([[56,65],[368,52],[28,387],[389,390]])
pts2 = np.float32([[0,0],[300,0],[0,300],[300,300]])
M = cv2.getPerspectiveTransform(pts1,pts2) #size: 3x3
dst = cv2.warpPerspective(img, M, (300,300))
```

12_PersTrans.ipynb

Project Transformation

(a)

(b)



(c)

(d)