Digital Image Processing (CSE/ECE 478)

Lecture # 05: Geometric operations

Avinash Sharma

Center for Visual Information Technology (CVIT),

IIIT Hyderabad

Image Transformations

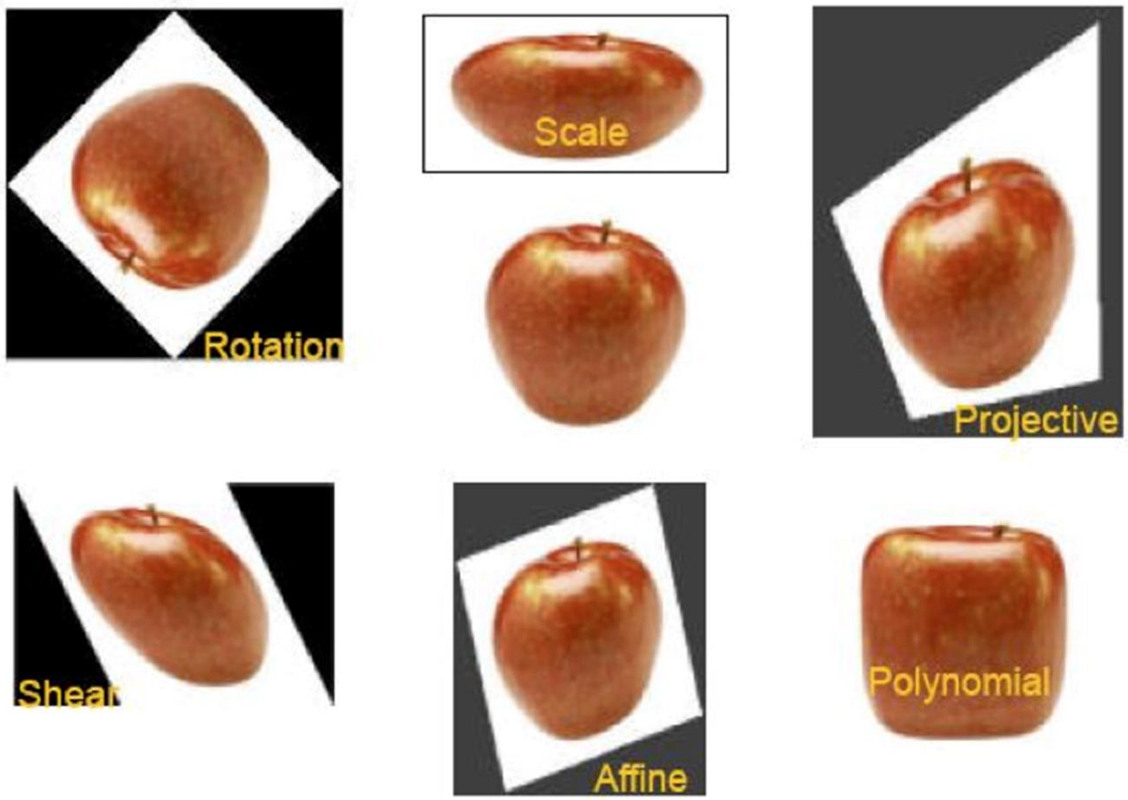
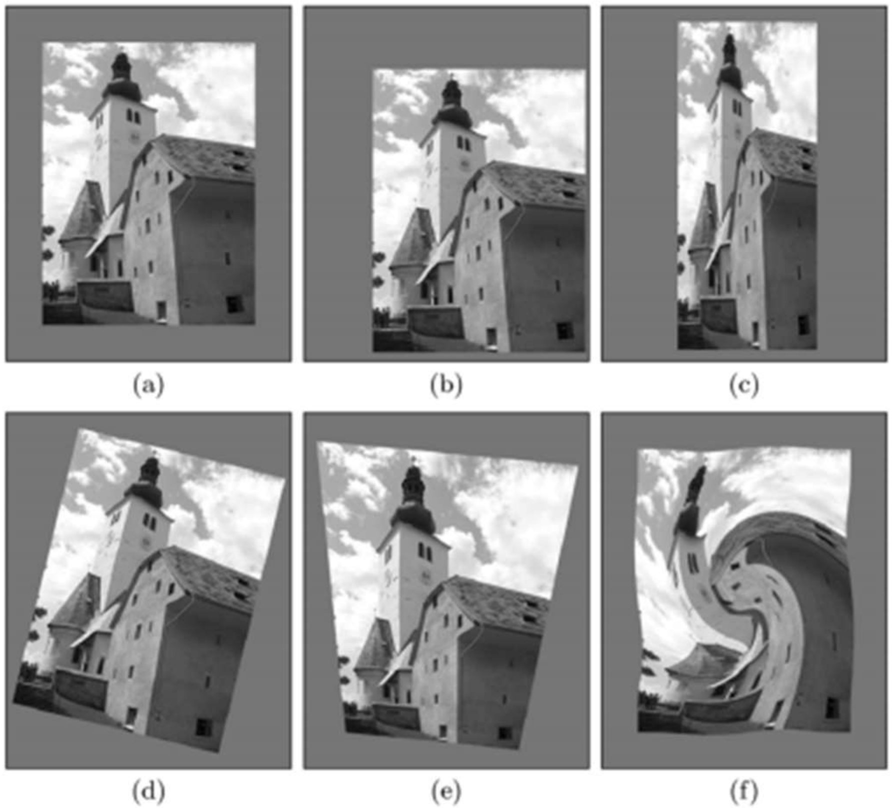


Image Transformations



Applications

- Align images
- Correct images for lens distortion
- Correct effects for camera orientation
- Image morphing
- Create interesting image effects

Geometric operations

- Geometric operation transforms image I to new image I' by modifying coordinates of image pixels:

$$I(x, y) \rightarrow I'(x', y')$$

- Intensity value (x, y) moved to a new position (x', y')

Example: Translation geometric operation moves value at (x,y) to $(x + d_x, y + d_y)$

Simple mappings (Translation)

- Shift by a vector (d_x, d_y)

$$T_x : x' = x + d_x$$
$$T_y : y' = y + d_y$$

or

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} d_x \\ d_y \end{pmatrix}$$

Simple mappings (Scaling)

- Contracting or Stretching along x or y axis by a factor of s_x or s_y

$$T_x : x' = s_x \cdot x$$
$$T_y : y' = s_y \cdot y$$

or

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}$$

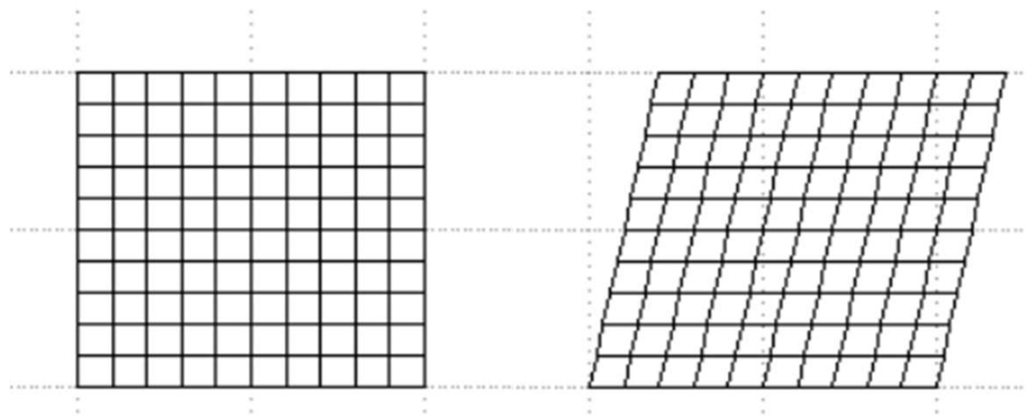
Simple mappings (Shearing)

- Along x and y axis by factor b_x and b_y

$$T_x : x' = x + b_x \cdot y$$
$$T_y : y' = y + b_y \cdot x$$

or

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & b_x \\ b_y & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}$$



$$x' = x + 0.2y$$

Simple mappings (Rotation)

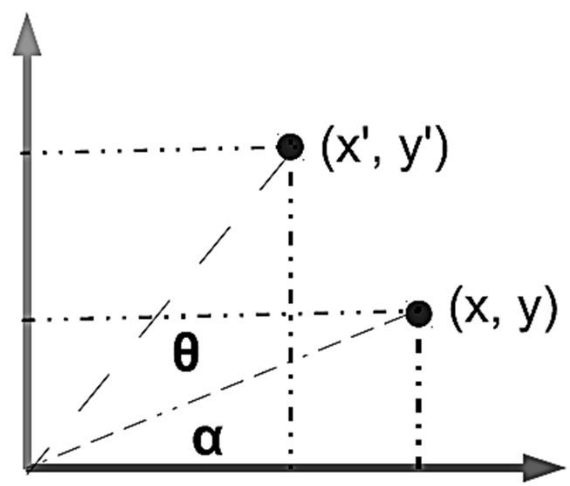
- 2D Rotation

Rotate about origin CCW by θ.

$$x' = x\cos\theta - y\sin\theta,$$
$$y' = x\sin\theta + y\cos\theta.$$

Matrix notation: $P' = R\,P$

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



Simple mappings (Rotation)

$$T_x : x' = x \cdot \cos\alpha - y \cdot \sin\alpha$$
$$T_y : y' = x \cdot \sin\alpha + y \cdot \cos\alpha$$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}$$

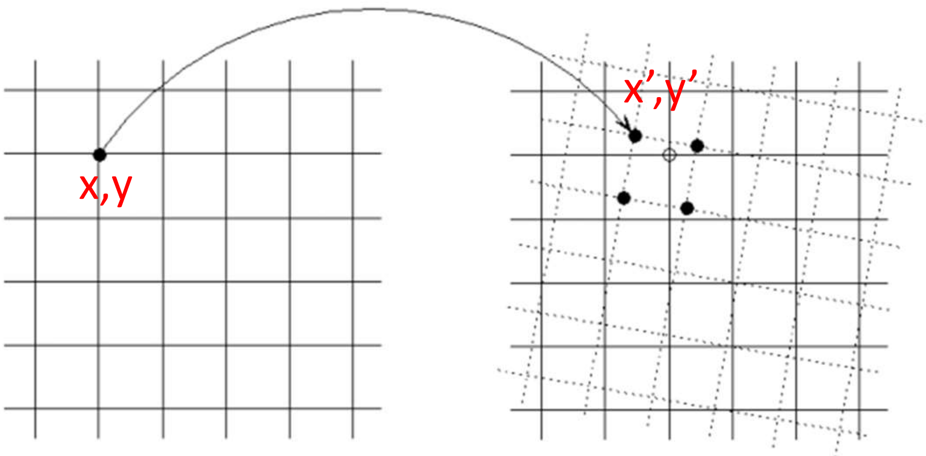
Image flipping and rotation by 90 degrees

- Basic idea: look up a transformed pixel address instead of the current one
- To flip an image upside down (vertical flip)
  - At pixel location (x, y), look up the color at location $(m - x, y)$
- For horizontal flip
  - At pixel location (x, y), look up the color at location $(x, n - y)$

Rotation by
90 degrees!

Image Warping

- Forward mapping (iterate over source image)

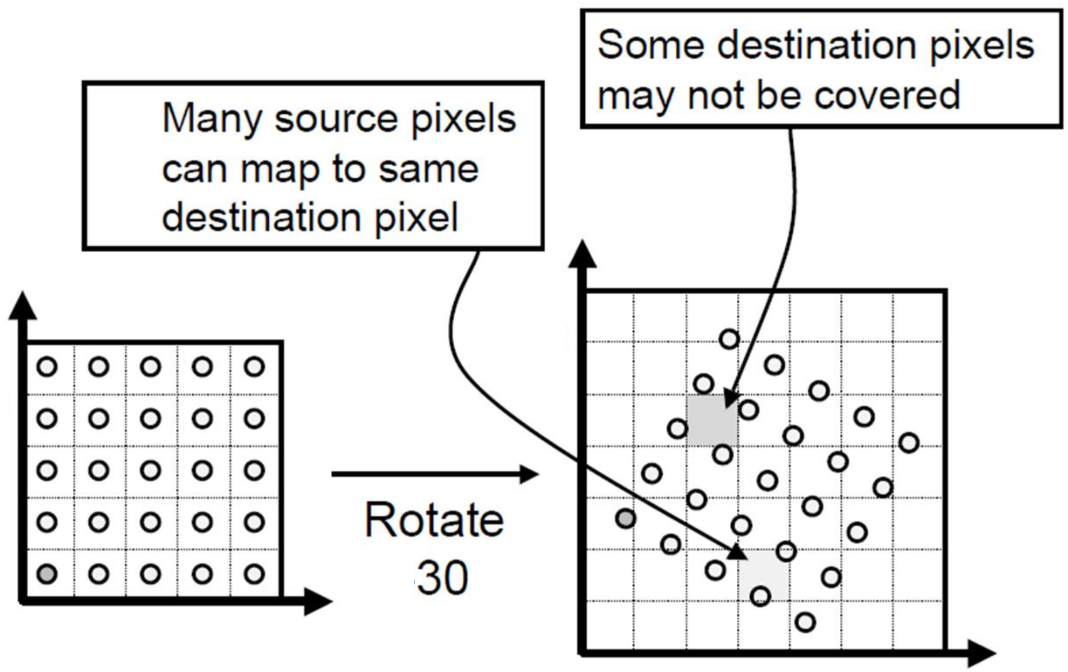



```
for (int x=0; x<W; x++)
    for(int y=0; y<H; y++)
        float x' = f_x(x,y);
        float y' = f_y(x,y);
        dst(x',y') = source(x,y);
    }
}
```

Transformed points may not fall on exact grid!!

Image Warping

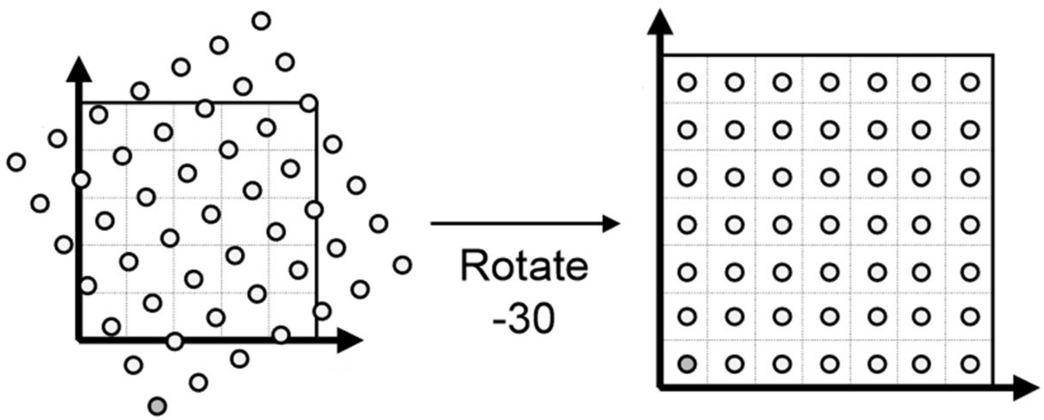
- Forward mapping (iterate over source image)



Courtesy: Thomas Funkhouser

Image Warping

- Reverse mapping (iterate over destination image)



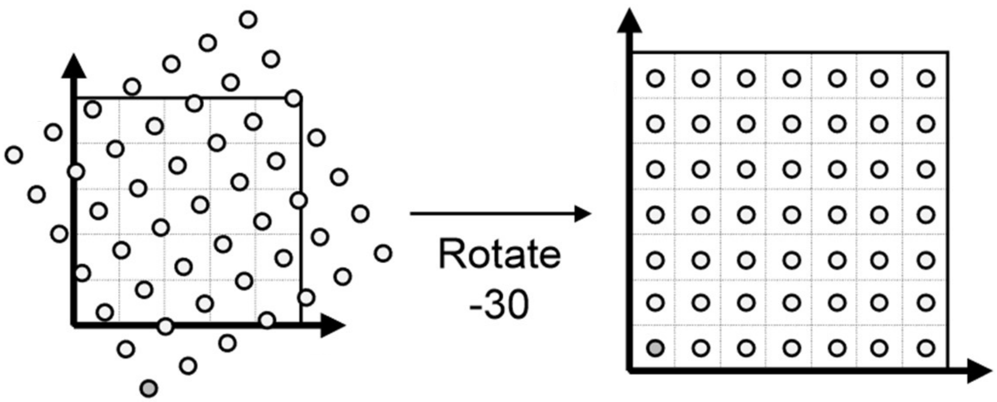

```
for (int x'=0; x'<W; x'++)
        for(int y'=0; y'<H; y'++)
                float x = f_x^{-1}(x', y');
                float y = f_y^{-1}(x', y');
                dst(x',y') = source(x,y);
        }
}
```

$$x = f_x^{-1}(x', y'), \quad y = f_y^{-1}(x', y')$$

Courtesy: Thomas Funkhouser

Image Warping

- Reverse mapping (iterate over destination image)



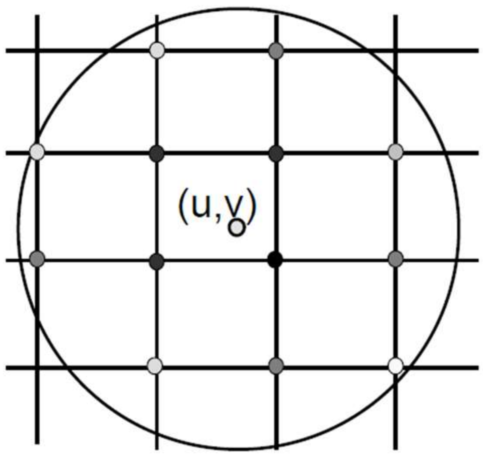

```
for (int x'=0; x'<W; x'++)
    for(int y'=0; y'<H; y'++)
        float x = f_x^{-1}(x', y');
        float y = f_y^{-1}(x', y');
        dst(x',y') = resample_src(x,y,w);
    }
}
```

Where float $x = f_x^{-1}(x', y')$; float $y = f_y^{-1}(x', y')$;

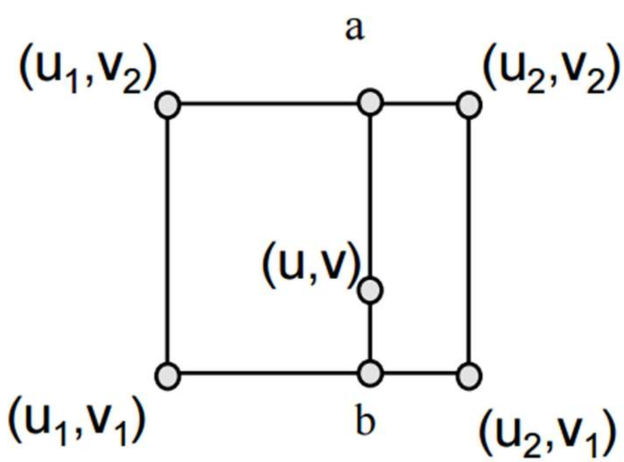
Courtesy: Thomas Funkhouser

Interpolation

- Reverse mapping (may not be exact integer)



Gaussian Interpolation



Bilinear Interpolation

Courtesy: Thomas Funkhouser

Image Warping (Example Scaling)

$$T_x : x' = s_x \cdot x$$
$$T_y : y' = s_y \cdot y$$

```
for (int x'=0; x'<W; x'++)
    for(int y'=0; y'<H; y'++)
        float x = x'/s_x;
        float y = y'/s_y;
        dst(x',y') = resample_src(x,y,w);
    }
}
```

Image Warping (Example Rotation)

```
for (int x'=0; x'<W; x'++)
        for(int y'=0; y'<H; y'++)
                float x = x'cos(α) + y'sin(α);
                float y = −x'sin(α) + y'cos(α);
                dst(x',y') = resample_src(x,y,w);
        }
}
```

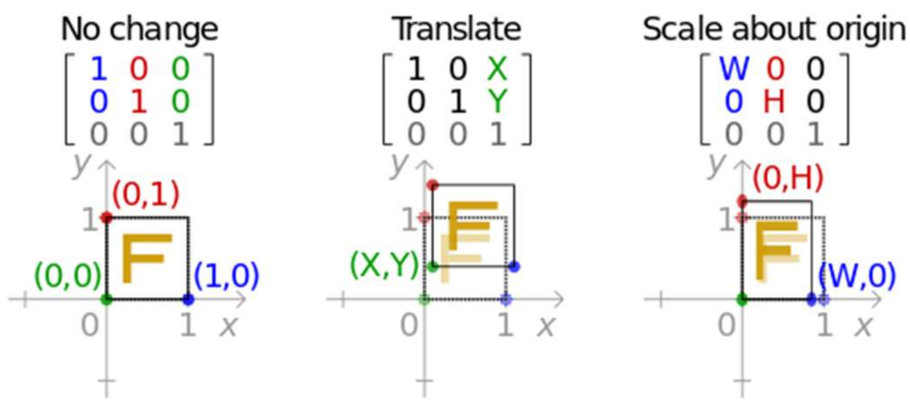
$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}$$

Homogeneous coordinates and Affine Transformation

- Using homogenous coordinates. We can write translation, scaling, rotation etc. as a vector matrix multiplication

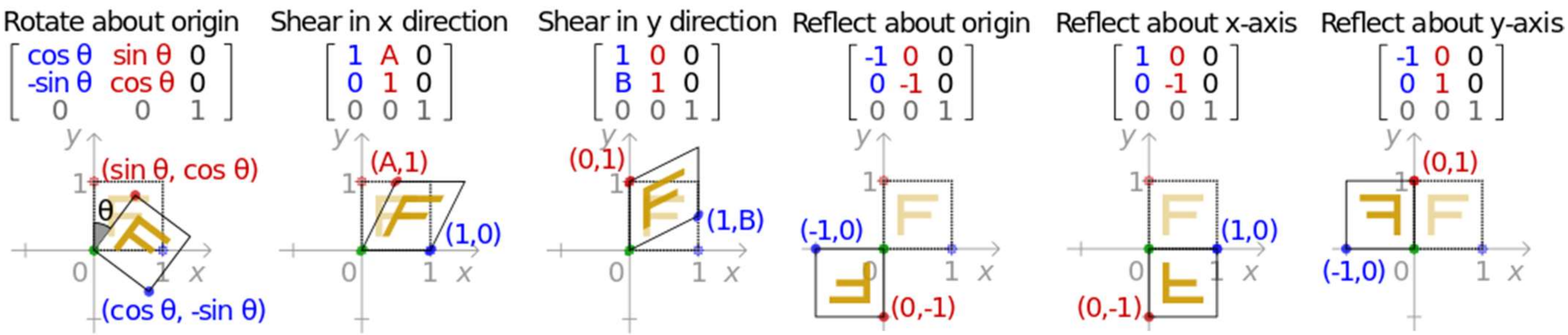
$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

- Example



Courtesy: wikipedia

Affine transformation



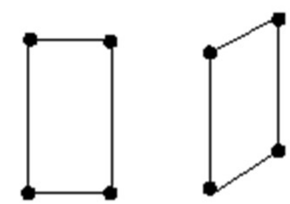
Courtesy: wikipedia

Affine transformation

- Preserves collinearity (all points on a line, remain on a line)
  - Parallel lines remain parallel
  - Does not necessarily preserve angles between lines or distances between points (any triangle can be transformed into another using affine transformation)
  - Preserve ratios of distances between points lying on a straight line

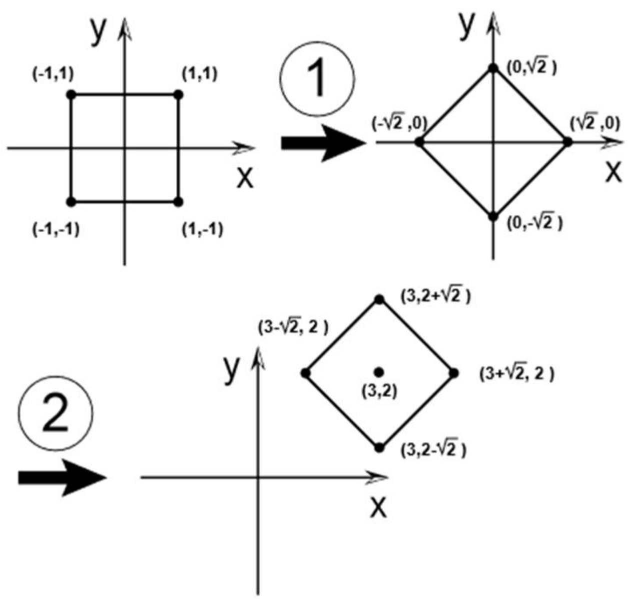
- Desired transformation as combination of simpler ones

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



Affine transformation

- Desired transformation as combination of simpler ones



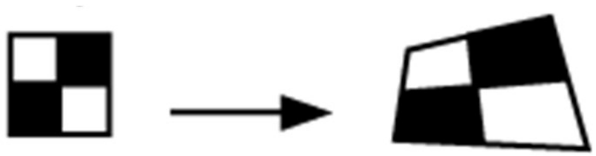
$$M = T_{(3,2)} R_{45^\circ} = \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos 45^\circ & -\sin 45^\circ & 0 \\ \sin 45^\circ & \cos 45^\circ & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos 45^\circ & -\sin 45^\circ & 3 \\ \sin 45^\circ & \cos 45^\circ & 2 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \sqrt{2}/2 & -\sqrt{2}/2 & 3 \\ \sqrt{2}/2 & \sqrt{2}/2 & 2 \\ 0 & 0 & 1 \end{bmatrix}.$$

2D Projective transformation (homography)

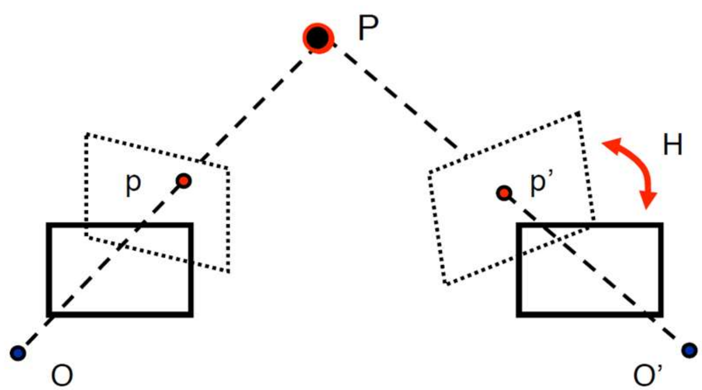
- Preserves collinearity (parallel lines not parallel)

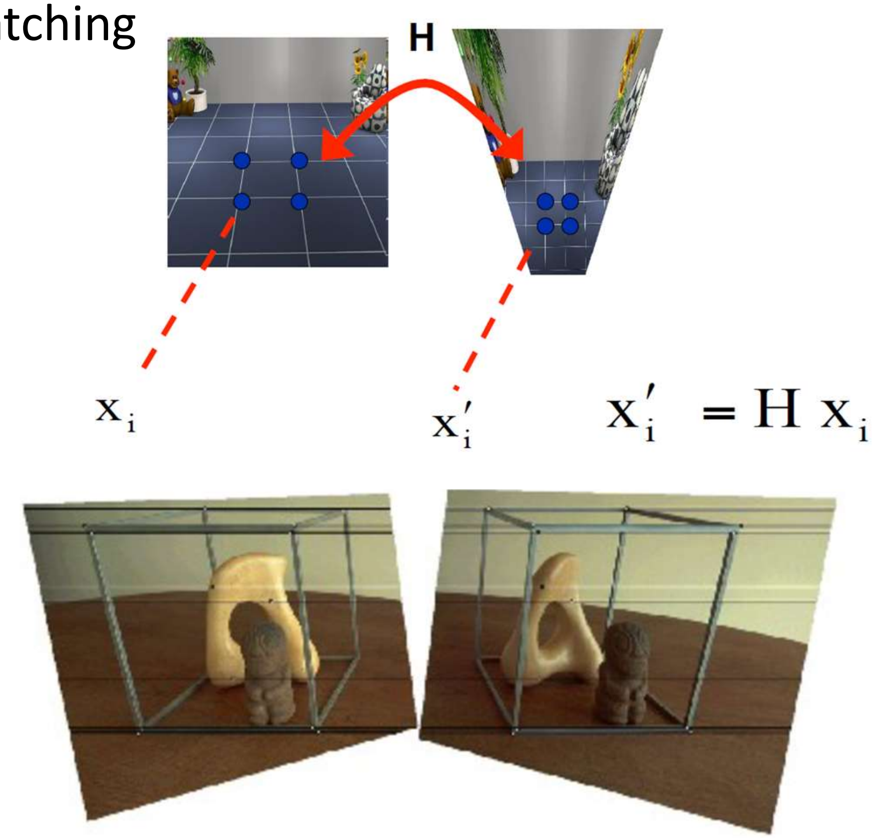


$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

2D Projective transformation (homography)

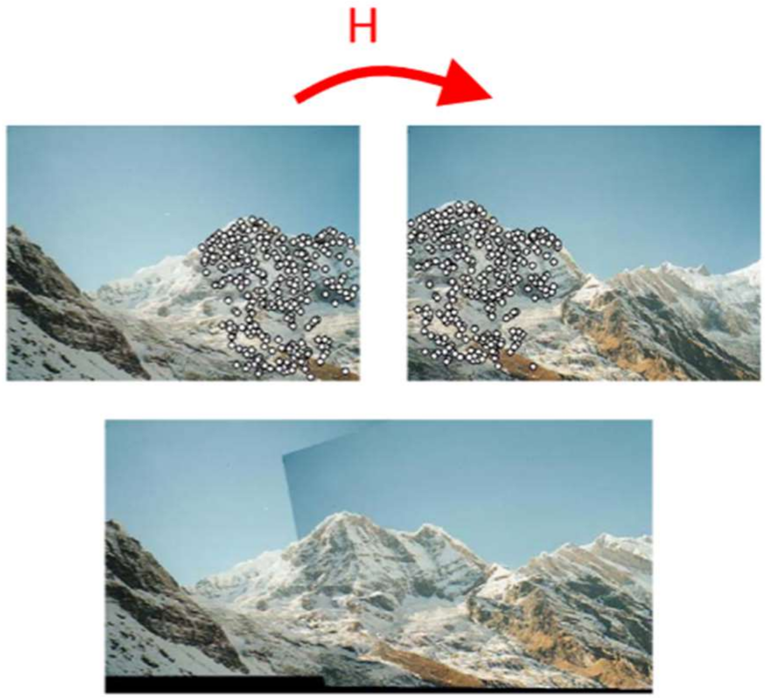
- Homography estimation for stereo matching





2D Projective transformation (homography)

- Important for image stitching application



Transformation categories

- Planar
  - Rigid
  - Similarity
  - Affine
  - Projective

	Rotate	Translate	Scale	Skew/shear
Rigid	√	√	-	-
Similarity	√	√	√	-
Affine	√	√	√	√
Projective	√	√	√	√

Transformation categories

- Rigid (Isometric) → preserves length, angle, area

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} \epsilon \cos\theta & -\sin\theta & t_x \\ \epsilon \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

3 DOF (θ, t_x, t_y)

	Rotate	Translate	Scale	Skew/shear
Rigid	√	√	-	-
Similarity	√	√	√	-
Affine	√	√	√	√
Projective	√	√	√	√

Transformation categories

- Similarity → parallel lines, angle, ratio between any two points

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} s\cos\theta & -s\sin\theta & t_x \\ s\sin\theta & s\cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

4 DOF (θ, t_x, t_y,s)

	Rotate	Translate	Scale	Skew/shear
Rigid	√	√	-	-
Similarity	√	√	√	-
Affine	√	√	√	√
Projective	√	√	√	√

Transformation categories

- Affine → collinear, parallel lines, ratio between any two points on a line

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

6 DOF ($a_{11}, a_{12}, a_{21}, a_{22}, t_x, t_y$)

	Rotate	Translate	Scale	Skew/shear
Rigid	√	√	-	-
Similarity	√	√	√	-
Affine	√	√	√	√
Projective	√	√	√	√

Transformation categories

- Projective → collinear, parallel lines not parallel

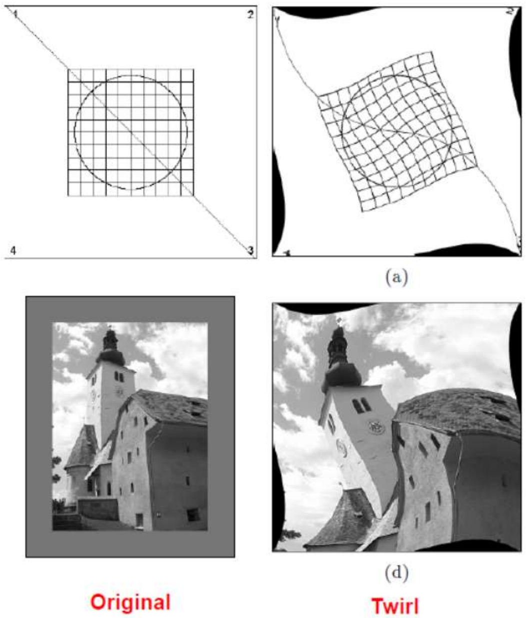
$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

8 DOF

	Rotate	Translate	Scale	Skew/shear
Rigid	√	√	-	-
Similarity	√	√	√	-
Affine	√	√	√	√
Projective	√	√	√	√

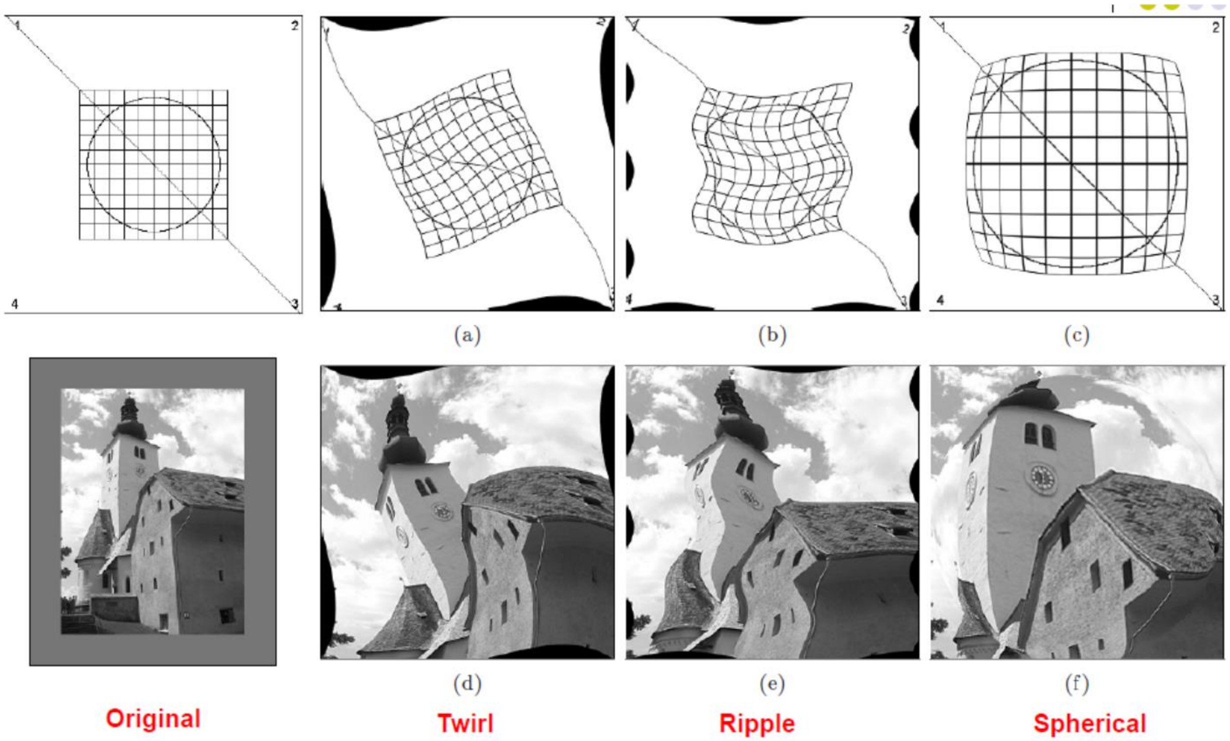
Transformation categories

- Non Planar
  - Curved → lines do not map to lines, shapes deform (expressed as polynomials)



Courtesy: Emmanuel Agu

Non Linear image warps



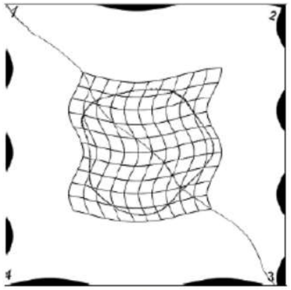
Courtesy: Emmanuel Agu

Ripple

- Wavelike displace along both x and y directions

$$T_x^{-1}: \quad x = x' + a_x \cdot \sin\left(\frac{2\pi \cdot y'}{\tau_x}\right),$$
$$T_y^{-1}: \quad y = y' + a_y \cdot \sin\left(\frac{2\pi \cdot x'}{\tau_y}\right).$$

Sample values of parameter: $a_x = 10, a_y = 15,$
$\tau_x = 120, \tau_y = 150$



(b)


(e)

Courtesy: Emmanuel Agu

Twirl

- Rotation about centre or some anchor point
  - Increasingly rotate as radial distance r from centre increases
  - Image unchanged after r_{max}

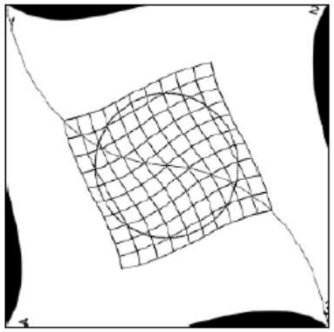
$$T_x^{-1}: \quad x = \begin{cases} x_c + r \cdot \cos(\beta) & \text{for } r \le r_{\max} \\ x' & \text{for } r > r_{\max}, \end{cases}$$

$$T_y^{-1}: \quad y = \begin{cases} y_c + r \cdot \sin(\beta) & \text{for } r \le r_{\max} \\ y' & \text{for } r > r_{\max}, \end{cases}$$

with

$$d_x = x' - x_c, \qquad r = \sqrt{d_x^2 + d_y^2},$$

$$d_y = y' - y_c, \qquad \beta = \operatorname{Arctan}(d_y, d_x) + \alpha \cdot \left(\tfrac{r_{\max} - r}{r_{\max}}\right).$$

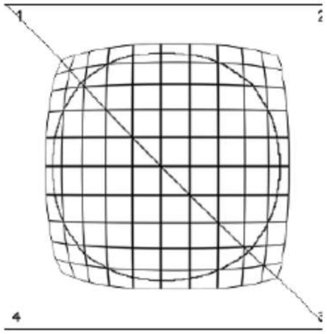


(a)

Courtesy: Emmanuel Agu

Spherical Transformation

- Imitates viewing image through a lens placed over image
  - Increasingly rotate as radial distance r from center increases
  - Lens center (x_c, y_c), radius (r_{max}), refractive index (ρ)



(c)

Skew

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} ntan\theta \\ 0 \end{bmatrix}$$

$$T_x^{-1}: \quad x = x' - \begin{cases} z \cdot \tan(\beta_x) & \text{for } r \leq r_{\max} \\ 0 & \text{for } r > r_{\max}, \end{cases}$$

$$T_y^{-1}: \quad y = y' - \begin{cases} z \cdot \tan(\beta_y) & \text{for } r \leq r_{\max} \\ 0 & \text{for } r > r_{\max}, \end{cases}$$

$$d_x = x' - x_c, \qquad r = \sqrt{d_x^2 + d_y^2}\,,$$

$$d_y = y' - y_c, \qquad z = \sqrt{r_{\max}^2 - r^2}\,,$$

$$\beta_x = \left(1 - \tfrac{1}{\rho}\right) \cdot \sin^{-1}\left(\frac{d_x}{\sqrt{(d_x^2 + z^2)}}\right),$$

$$\beta_y = \left(1 - \tfrac{1}{\rho}\right) \cdot \sin^{-1}\left(\frac{d_y}{\sqrt{(d_y^2 + z^2)}}\right).$$

Courtesy: Emmanuel Agu

More specific grid based warping