12.10.2021

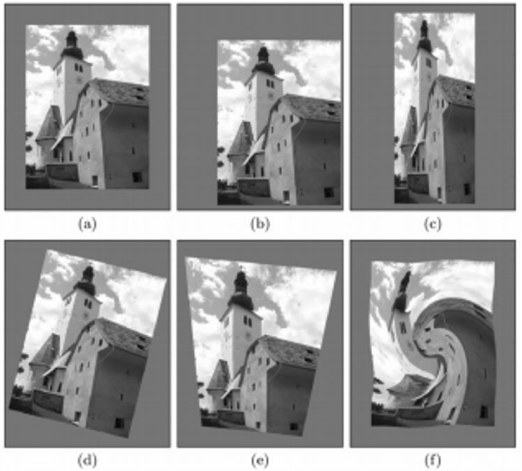
Digital Image Processing (CSE/ECE 478)

Lecture-14: Geometric Operations

Ravi Kiran and Sudipta Banerjee

Center for Visual Information Technology (CVIT), IIIT Hyderabad

Geometric Operations



- Example applications of geometric operations:
  - Zooming images, windows to arbitrary size
  - Computer graphics: deform textures and map to arbitrary surfaces
- **Definition:** Geometric operation transforms image *I* to new image *I'* by modifying **coordinates of image pixels**

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

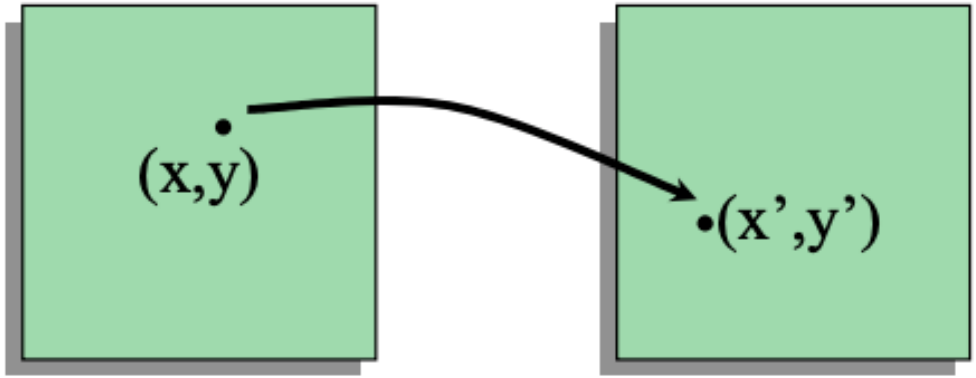
- Intensity value originally at (x,y) moved to new position (x',y')

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

$$x \rightarrow f_x(x,y) = x'$$

$$y \rightarrow f_y(x,y) = y'$$

$$I(x,y) = I'\left(f_x(x,y), f_y(x,y)\right)$$



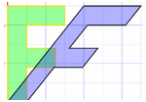
Common Geometric Operations

- **Scale** - change image content size

- **Rotate** - change image content orientation

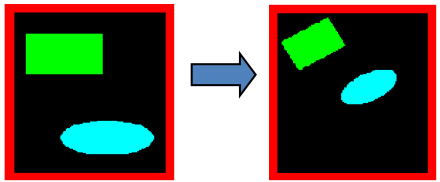
- **Reflect** - flip over image contents

- **Translate** - change image content position



- **Shear** – shift points along a line parallel to fixed line

- **Affine Transformation**
  - general image content linear geometric transformation



Simple Mappings

- **Translation:** (shift) by a vector (d_x, d_y)

$$T_x : x' = x + d_x \quad T_y : y' = y + d_y \quad \text{or} \quad \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} d_x \\ d_y \end{pmatrix}$$

- **Scaling:** (contracting or stretching) along x or y axis by a factor s_x or s_y

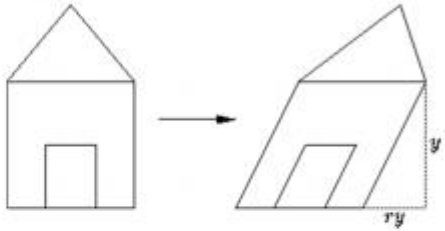
$$T_x : x' = s_x \cdot x \quad T_y : y' = s_y \cdot y \quad \text{or} \quad \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}$$

https://web.cs.wpi.edu/~emmanuel/courses/cs545/S14/slides/lecture11.pdf

Simple Mappings

- **Shearing:** along x and y axis by factor b_x and b_y

$$T_x : x' = x + b_x \cdot y \qquad \text{or} \qquad \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & b_x \\ b_y & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}$$
$$T_y : y' = y + b_y \cdot x$$



- **Rotation:** the image by an angle α

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

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

Original image | After rotation of 45°

https://web.cs.wpi.edu/~emmanuel/courses/cs545/S14/slides/lecture11.pdf

Homogeneous Coordinates

- Notation useful for converting scaling, translation, rotating into point-matrix multiplication
- To convert ordinary coordinates into homogeneous coordinates

$$\boldsymbol{x} = \begin{pmatrix} x \\ y \end{pmatrix} \quad \text{converts to} \quad \hat{\boldsymbol{x}} = \begin{pmatrix} \hat{x} \\ \hat{y} \\ h \end{pmatrix} = \begin{pmatrix} h\,x \\ h\,y \\ h \end{pmatrix}$$

https://web.cs.wpi.edu/~emmanuel/courses/cs545/S14/slides/lecture11.pdf

Homogeneous coordinates

- Homogeneous coordinates transforms a point from Euclidean plane to projective plane by adding a dummy variable (x,y,1)
- Overall scaling is unimportant so (x,y,1) ≅ (ax,ay,a), for any non-zero ‘a’
- Because scaling does not play role (aX,aY,aW) are called homogeneous coordinates of the point
- Linear transformation equations specify **affine** transformation
  - $x' = a_0x + a_1y + a_2;\ \ y' = b_0x + b_1y + b_2$
  - Two successive affine transformations $T = T_2T_1$
  - Inverse: $T^{-1} = T_1^{-1}T_2^{-1}$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a_0 & a_1 \\ b_0 & b_1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} a_2 \\ b_2 \end{bmatrix} \Leftrightarrow \underbrace{\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_0 & a_1 & a_2 \\ b_0 & b_1 & b_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}}_{\text{affine transformation in homogeneous coordinates}}$$

All linear transformations are affine but not all affine transformations are linear

http://robotics.stanford.edu/~birch/projective/node4.html
https://www.graphicsmill.com/docs/gm5/Transformations.htm
http://www.songho.ca/math/homogeneous/homogeneous.html

Affine (3-Point) Mapping

1. Translation
2. Scaling
3. Rotation

- Can use homogeneous coordinates to rewrite translation, rotation, scaling, etc as vector-matrix multiplication

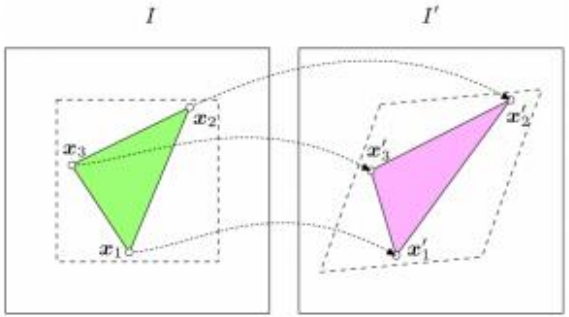
$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

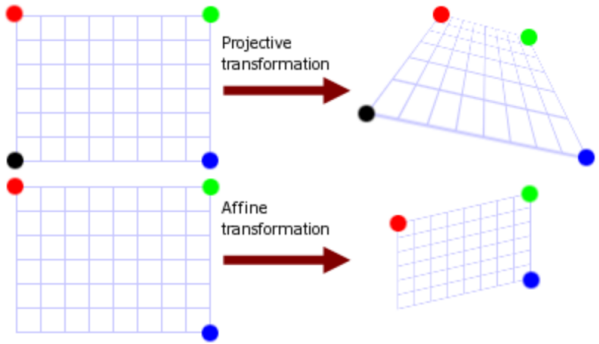
$$H = RST = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_0 & 0 & 0 \\ 0 & s_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & x_0 \\ 0 & 1 & x_1 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} s_0\cos\theta & s_1\sin\theta & s_0x_0\cos\theta + s_1x_1\sin\theta \\ -s_0\sin\theta & s_1\cos\theta & s_1x_1\cos\theta - s_0x_0\sin\theta \\ 0 & 0 & 1 \end{bmatrix}$$

- **Affine mapping:** Can then derive values of matrix that achieve desired transformation (or combination of transformations)

$$\begin{pmatrix} a_1 & a_2 & b_1 \\ a_3 & a_4 & b_2 \\ c_1 & c_2 & 1 \end{pmatrix} \times \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix}$$



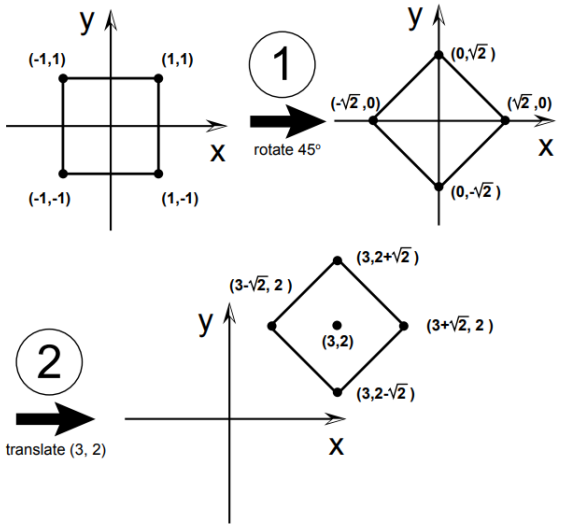


- Inverse of transform matrix is **inverse mapping**

https://web.cs.wpi.edu/~emmanuel/courses/cs545/S14/slides/lecture11.pdf
https://www.graphicsmill.com/docs/gm5/Transformations.htm

Example

Now, suppose we have a 2×2 square centered at the origin and we want to first rotate the square by $45°$ about its center and then move the square so its center is at $(3, 2)$. We can do this in two steps, as shown in the diagram to the right.



In matrix form:

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

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

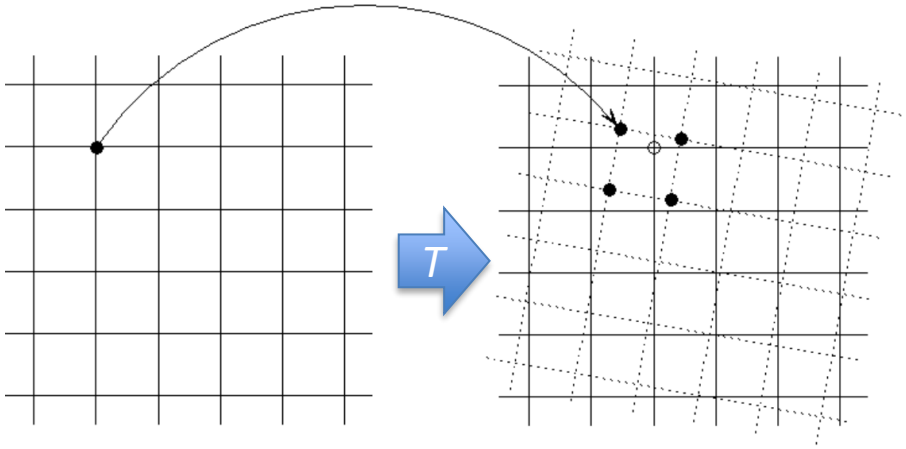
$$= \begin{bmatrix} \sqrt{2}/2 & -\sqrt{2}/2 & 3 \\ \sqrt{2}/2 & \sqrt{2}/2 & 2 \\ 0 & 0 & 1 \end{bmatrix}.$$

Note that

$$M \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 + \sqrt{2} \\ 1 \end{bmatrix}, \text{and } M \begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 - \sqrt{2} \\ 2 \\ 1 \end{bmatrix},$$

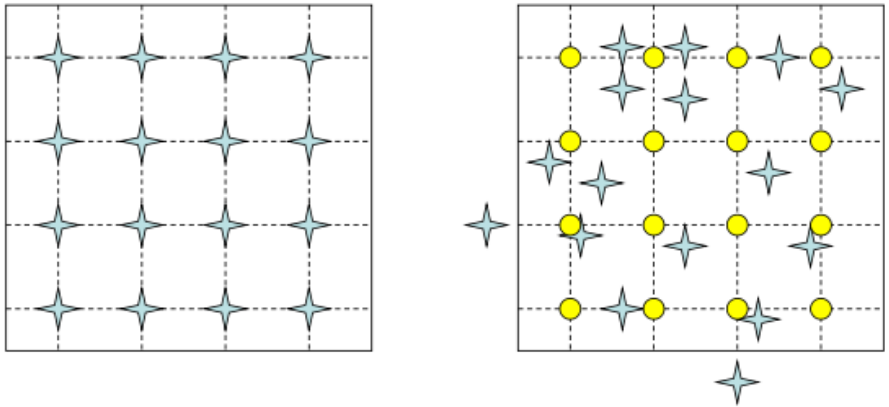
https://people.cs.clemson.edu/~dhouse/courses/401/notes/affines-matrices.pdf

Forward Mapping



Original Image

Transformed Image



- Destination image may not accommodate all transformed pixels (expand view)
- Transformed coordinates may not be integers (assign to nearest integer)

https://www.cis.rit.edu/class/simg782/lectures/lecture_02/lec782_05_02.pdf

Two issues

- Expanded view increases the overall image dimensions
- Each output pixel may be addressed several times while other pixels are not at all, resulting in holes (no value is assigned to a particular pixel in transformed image)

(a) Source image. (b) Rotation. (c) Expanded view of rotated image.

Figure 7.2. Destination dimensionality under rotation.

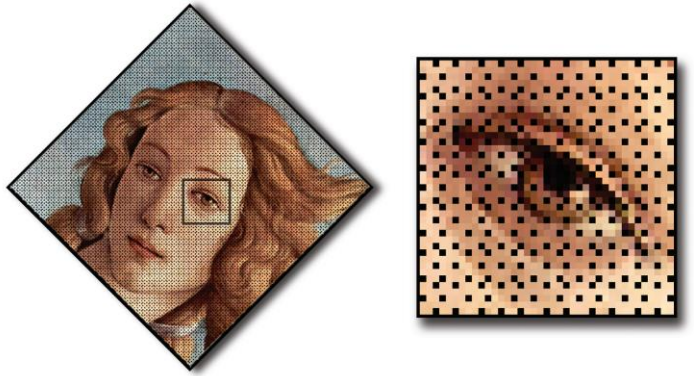
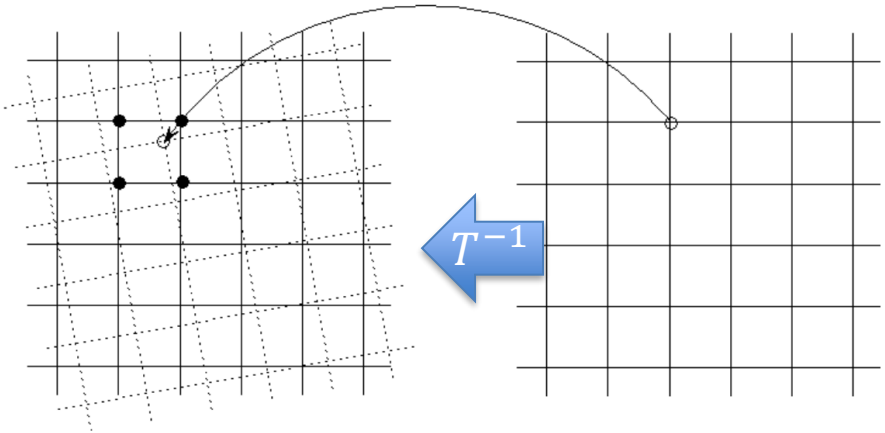


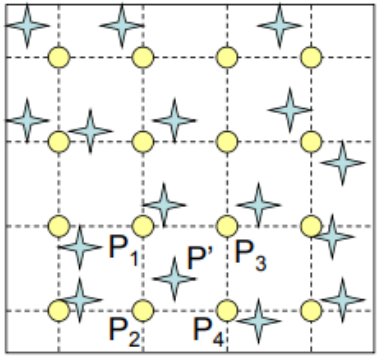
Figure 7.3. Rotation by forward mapping.

Solution: Backward/Inverse mapping

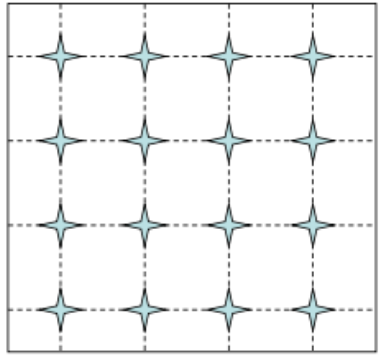


Original Image

Transformed Image



P' will be interpolated from P_1, P_2, P_3, and P_4



T : Affine transform matrix

v : location in the transformed image

$v' = [x', y', 1]^T$

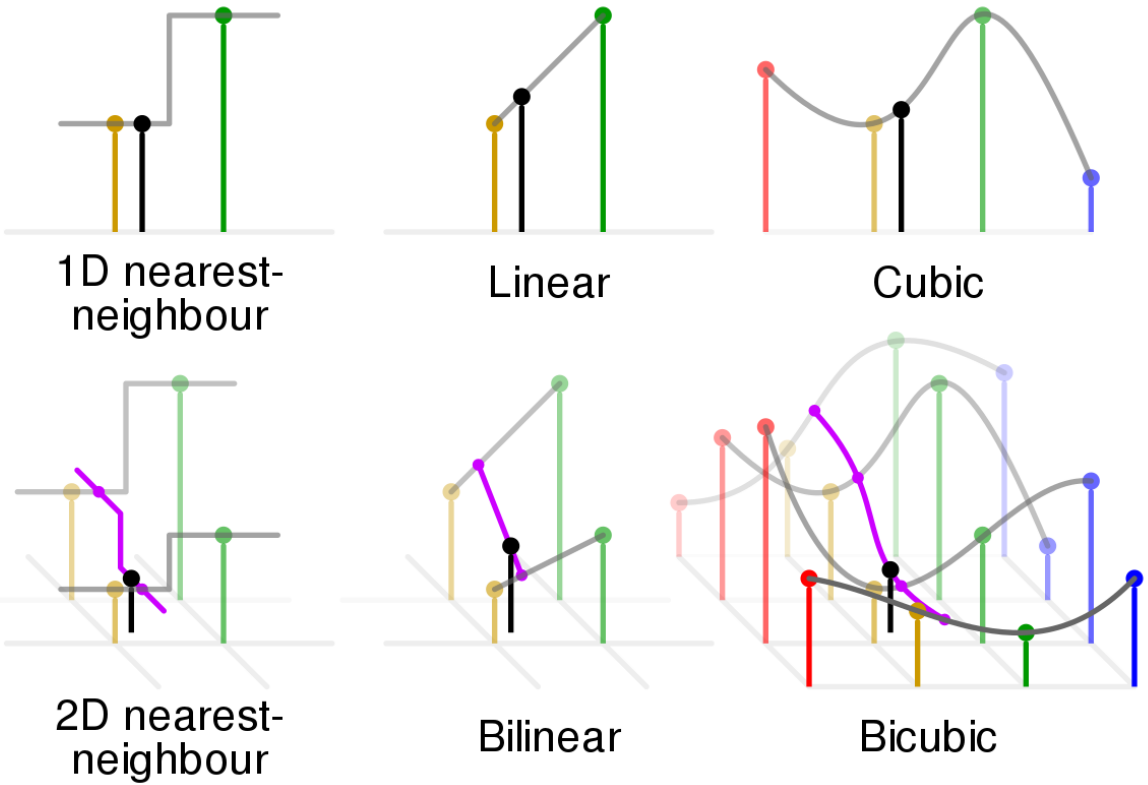
$$v' = Tv$$

$$v = T^{-1}v'$$

Backward mapping iterates over each pixel in output (transformed) image and uses inverse transformation to determine the position in input (original) image from which the pixel intensity value must be sampled (interpolated). Determined input points may not be integers, but no holes in output image.

https://www.cs.auckland.ac.nz/courses/compsci773s1c/lectures/ImageProcessing-html/topic2.htm

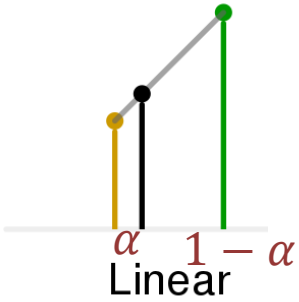
Interpolation Function

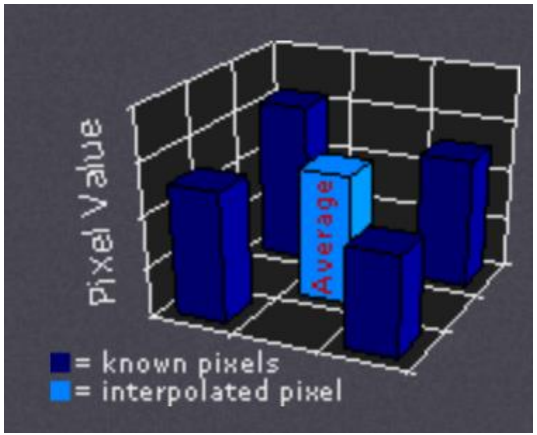


Zero-order

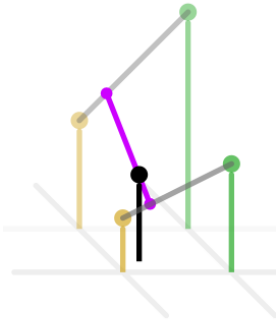
Interpolation Function

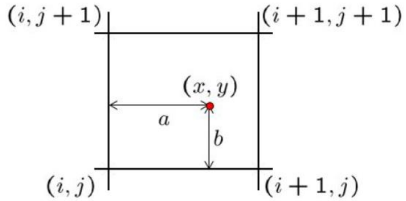
$$p = p_1(1-\alpha) + p_2\alpha$$



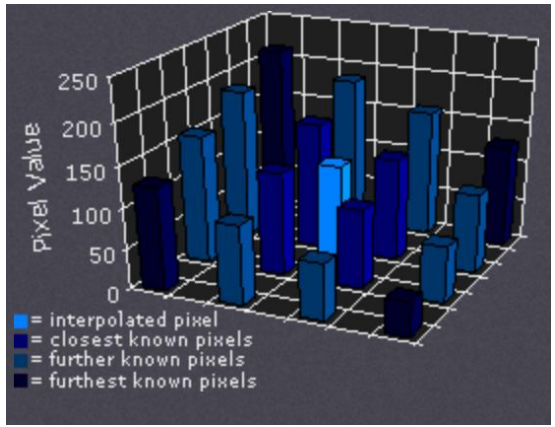


Bilinear interpolation is weighted average of (2x2) 4 neighboring pixels





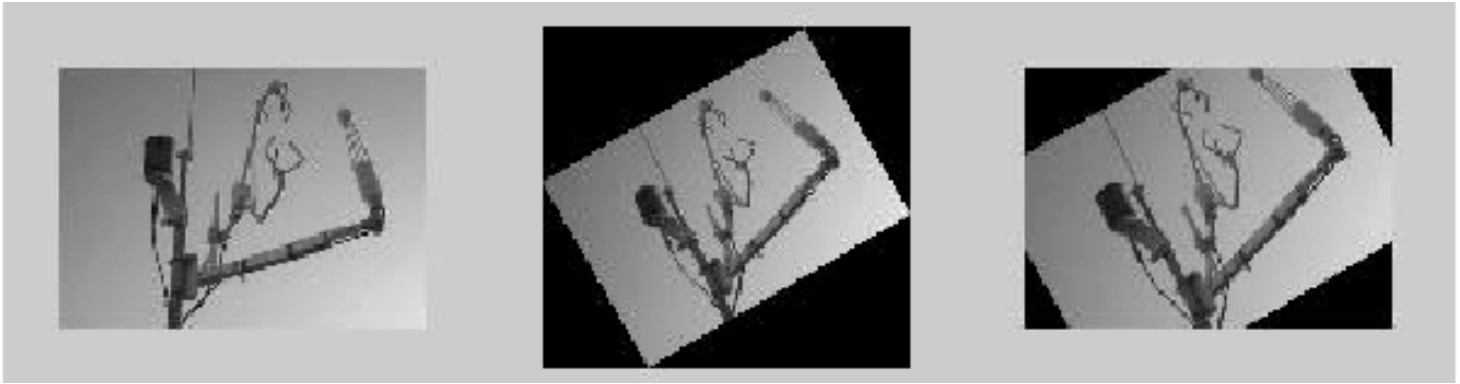
$$\begin{aligned} f(x,y) = \; & (1-a)(1-b) \; f[i,j] \\ & +a(1-b) \; f[i+1,j] \\ & +ab \; f[i+1,j+1] \\ & +(1-a)b \; f[i,j+1] \end{aligned}$$



Bicubic interpolation is weighted average of (4x4) 16 neighboring pixels. Closer pixels are weighed more

Example: imrotate

J=imrotate(I, angle in degrees (CCW), interpolation method, size of output image)



I

J = imrotate(I,30,`loose’)

J = imrotate(I,30,`crop’)

Geometric Transformation: Perspective

- Two aspects
  - Mapping (Type of transform)
  - Interpolation (Quality of transform)

Geometric Transforms and Registration

- Given: T (transformation)
- Determine: Effect of T on source image I, get target image O
- Variant:
  - Given : Source image I, Target image O
  - Determine : Transformation T

Point-to-point mapping

Suppose that you are given a pair of images to align. You want to try an affine transform to register one to the coordinate system of the other. How do you find the transform parameters?

Image A

Image B

https://www.cis.rit.edu/class/simg782/lectures/lecture_02/lec782_05_02.pdf

Point-to-point mapping

Find a number of points $\{\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_{n-1}\}$ in image A that match points $\{\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_{n-1}\}$ in image B. Use the homogeneous coordinate representation of each point as a column in matrices $\mathbf{P}$ and $\mathbf{Q}$:

$$\mathbf{P} = \begin{bmatrix} x_0 & x_1 & \ldots & x_{n-1} \\ y_0 & y_1 & \ldots & y_{n-1} \\ 1 & 1 & \ldots & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{p}_0 & \mathbf{p}_1 & \ldots & \mathbf{p}_{n-1} \end{bmatrix}$$

$$\mathbf{Q} = \begin{bmatrix} u_0 & u_1 & \ldots & u_{n-1} \\ v_0 & v_1 & \ldots & v_{n-1} \\ 1 & 1 & \ldots & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{q}_0 & \mathbf{q}_1 & \ldots & \mathbf{q}_{n-1} \end{bmatrix}$$

Then

$$\mathbf{q} = \mathbf{H}\mathbf{p} \quad \text{becomes} \quad \mathbf{Q} = \mathbf{H}\mathbf{P}$$

The solution for $\mathbf{H}$ that provides the minimum mean-squared error is

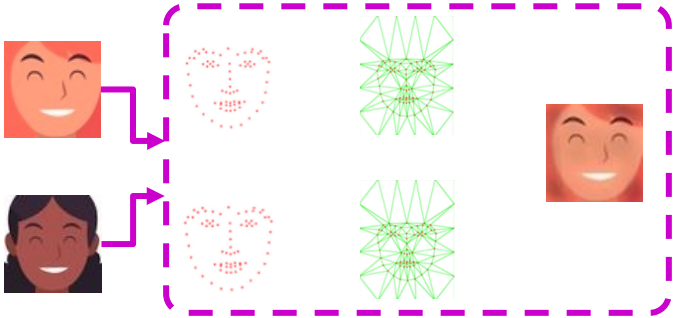
$$\mathbf{H} = \mathbf{Q}\mathbf{P}^T(\mathbf{P}\mathbf{P}^T)^{-1} = \mathbf{Q}\mathbf{P}^\dagger$$

where $\mathbf{P}^\dagger = \mathbf{P}^T(\mathbf{P}\mathbf{P}^T)^{-1}$ is the (right) *pseudo-inverse* of $\mathbf{P}$.

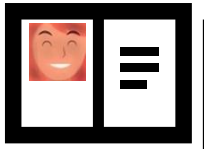
https://www.cis.rit.edu/class/simg782/lectures/lecture_02/lec782_05_02.pdf

What is a face morph?

- Morphing combines face images from multiple identities strategically such that the morphed face image matches to all constituents



Alignment and Blending



Document with Morphed Image

Both Alice and Jenny can pass through e-gate using same document

How easy is to create face morphs?

- Several free software are available online, commercial software exist, smartphone-based applications also readily available
- Open Source: UNIBO tool, OpenCV, GIMP+GAP
- Commercial: Fanta Morph, Adobe After Effects + Re:Flex
- Apps: Face Morpher, Face Story

Image Courtesy: https://3dthis.com/morph.htm

Any real-world example of face morph attack?

- Real-world case of face morphing: (September 2018)
  - An activist morphed her face image with that of Federica Mogherini (Italian High Representative of the European Union) and successfully obtained German passport

- Has piqued the interest of government agencies and research community:
  - EU iMARS project
  - US NIST FRVT MORPH-Performance of Automated Face Morph Detection

Image registration is often used in medical and satellite imagery to align images from different camera sources. Digital cameras use image registration to align and connect adjacent images into a single panoramic image.

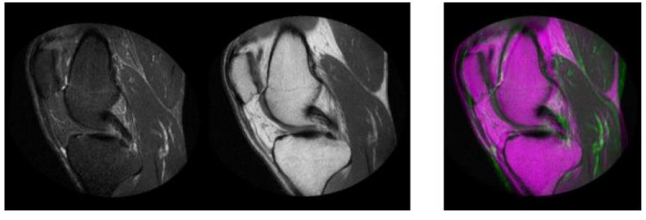
Registering aerial photos using point mapping.

```
[mp,fp] = cpselect(unregistered,ortho,'Wait',true);
```

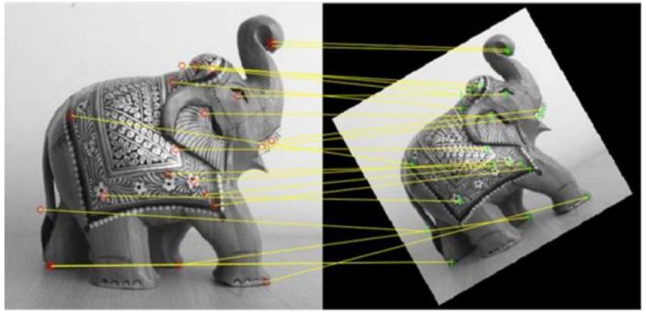
```
t = fitgeotrans(mp,fp,'projective')
```

```
Rfixed = imref2d(size(ortho));
registered = imwarp(unregistered,t,'OutputView',Rfixed);
```

```
imshowpair(ortho,registered,'blend')
```



Automatic registration on multimodal medical images.



Automatic registration using feature matching

https://in.mathworks.com/discovery/image-registration.html

- B = IMTRANSFORM(A,TFORM, INTERP) transforms the image A according to the 2-D spatial transformation defined by TFORMB; INTERP specifies the interpolation filter
- Example 1
- ---
- Apply a horizontal shear to an intensity image.
-

```
I = imread('cameraman.tif');
tform = maketform('affine',[1 0 0; .5 1 0; 0 0 1]);
J = imtransform(I,tform);
figure, imshow(I), figure, imshow(J)
```

tform = maketform('affine',[1 0 0; .5 1 0; 0 0 1]);
In MATLAB, 'affine' transform is defined by:
[a1,b1,0;a2,b2,0;a0,b0,1]

With notation used in this lecture note

$$\mathbf{A} = \begin{bmatrix} 1 & 0.5 \\ 0 & 1 \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

MATLAB function: cp2tform()

TFORM=CP2TFORM(INPUT_POINTS,BASE_POINTS,TRANSFORM TYPE)

- returns a TFORM structure containing a spatial transformation.
- INPUT_POINTS is an M-by-2 double matrix containing the X and Y coordinates of control points in the image you want to transform.
- BASE_POINTS is an M-by-2 double matrix containing the X and Y coordinates of control points in the base image.
- TRANSFORMTYPE can be 'nonreflective similarity', 'similarity', 'affine', 'projective', 'polynomial', 'piecewise linear' or 'lwm'.

$$\mathbf{A} = \begin{bmatrix} 1 & 0.5 \\ 0 & 1 \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Geometric Operations

- Some uses
  - Correct distortions introduced during imaging
  - Transformation: To create special effects (e.g. morphing)
  - Registration: Register two images taken of the same scene at different times/conditions

Homography

- Consider a point x = (u,v,1) in one image and x'=(u',v',1) in another image
- A homography is a 3 by 3 matrix M
- $$M = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix}$$
- The homography relates the pixel co-ordinates in the two images if x' = M x
- When applied to every pixel the new image is a warped version of the original image

http://people.scs.carleton.ca/~c_shu/Courses/comp4900d/notes/homography.pdf

Homography conditions

- Two images are related by a homography if and only if
- Both images are viewing the same plane from a different angle
- Both images are taken from the same camera but from a different angle
  - Camera is rotated about its center of projection without any translation
- Note that the homography relationship is independent of the scene structure
  - It does not depend on what the cameras are looking at
  - Relationship holds regardless of what is seen in the images

http://people.scs.carleton.ca/~c_shu/Courses/comp4900d/notes/homography.pdf

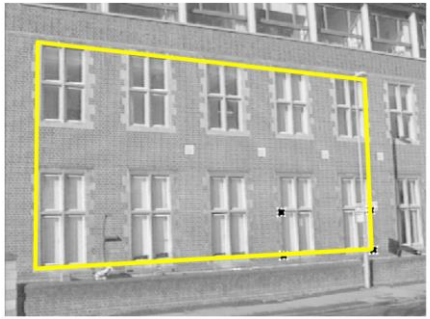
•Computing homography

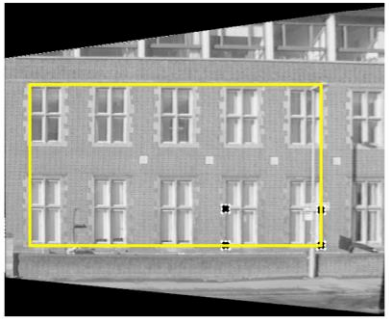
•If we know rotation R and calibration K, then homography M can be computed directly $x' = KRK^{-1}x$

- Applying this homography to one image gives image that we would get if the camera was rotated by R
- Inverting M, to get M^{-1} is same as applying inverse rotation R^{-1}

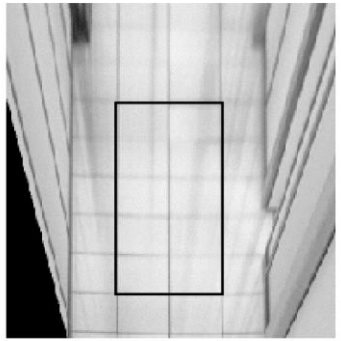
http://people.scs.carleton.ca/~c_shu/Courses/comp4900d/notes/homography.pdf
https://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/OWENS/LECT9/node2.html

Applying Homographies to Remove Perspective Distortion



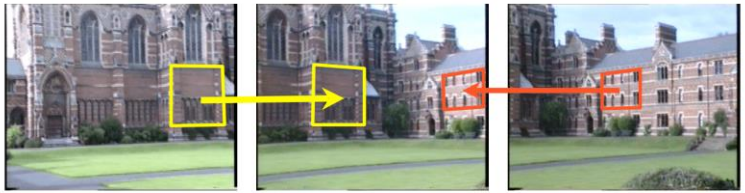


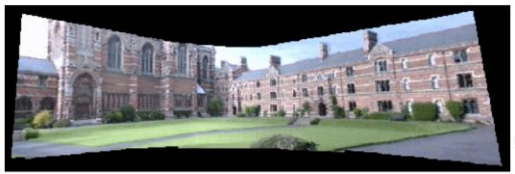
Homographies for Bird's-eye Views



from Hartley & Zisserman

Homographies for Mosaicing





from Hartley & Zisserman

http://www.cs.columbia.edu/~allen/F17/NOTES/homography_pka.pdf

References

- https://in.mathworks.com/help/images/ref/fitgeotrans.html
- https://in.mathworks.com/discovery/image-registration.html
- https://www.cs.princeton.edu/courses/archive/fall00/cs426/lectures/warp/warp.pdf
- http://eeweb.poly.edu/~yao/EL5123/lecture12_ImageWarping.pdf

References

- G&W textbook
  - 2.4.4. Image Interpolation
  - 2.6.5. Geometrical spatial transforms and image registration