Embedded computing for scientific and industrial imaging applications

Lecture 10 - Affine transformation

Outline

- Affine transformations
- BLAS and LAPACK

Ref : Computer graphics (CS384G) of University of Texas at Austin
Computer graphics (CSE557) of University of Washington

Geometric transformations

- Geometric transformations will map points in one space to points in another: $(x',y',z') = f(x,y,z)$.
- These transformations can be very simple, such as scaling each coordinate, or complex, such as nonlinear twists and bends.
- We'll focus on transformations that can be represented easily with matrix operations.
- 2D

Representation

- We can represent a **point**, **p** = (x, y), in the plane

  As a column vector $\begin{bmatrix} x \\ y \end{bmatrix}$

  As a row vector $\begin{bmatrix} x & y \end{bmatrix}$

Representation, cont.

- We can represent a 2-D transformation **M** by a matrix

$$M = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

- If p is a column vector, M goes on the left: p'=Mp

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Two-dimensional transformations

- Here's all you get with a 2 x 2 transformation matrix **M**:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

- So $x' = ax + by$

  $y' = cx + dy$

Identity

- Suppose we choose $a=d=1,\ b=c=0$:
- Gives the **identity** matrix:

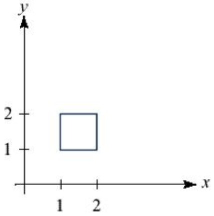
$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

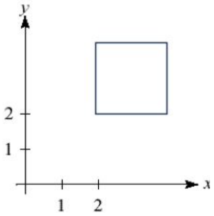
- Doesn’t move the points at all

Scaling

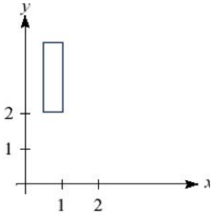
- Suppose b=c=0, but let a and d take on any positive value:
- Gives a scaling matrix: $\begin{bmatrix} a & 0 \\ 0 & d \end{bmatrix}$
- Provides **differential (non-uniform) scaling** in x and y:

$$x' = ax$$
$$y' = dy$$





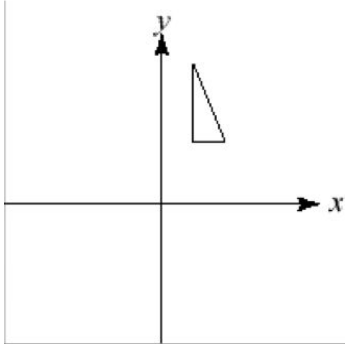
$$\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$



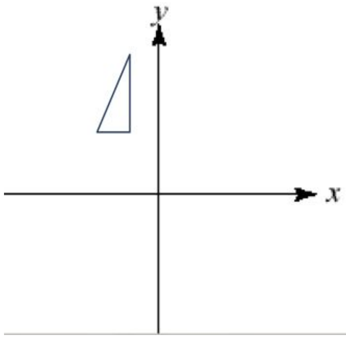
$$\begin{bmatrix} 1/2 & 0 \\ 0 & 2 \end{bmatrix}$$

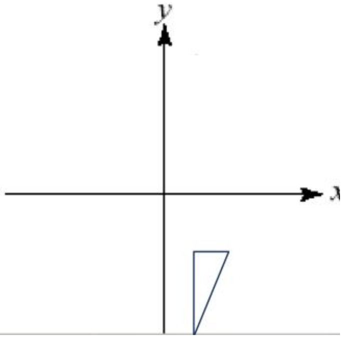
Reflection

- Suppose $b=c=0$, but let either a or d go negative.
- Examples:



$$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$



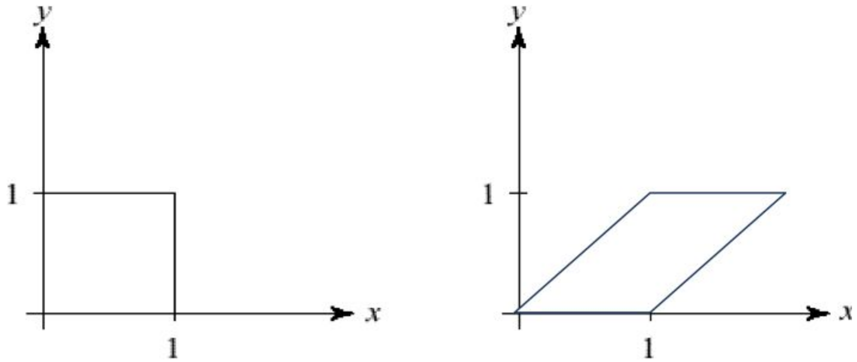


$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Shear

- Now leave a=d=1 and experiment with b
- The matrix $\begin{bmatrix} 1 & b \\ 0 & 1 \end{bmatrix}$

  gives : $$x' = x + by$$
  $$y' = y$$



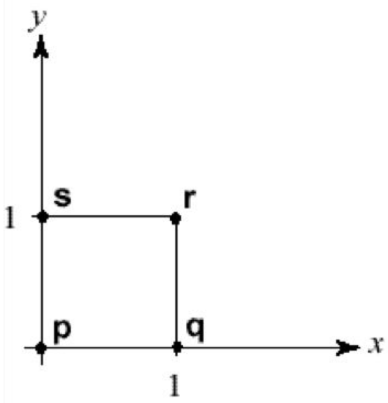
$$\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$$

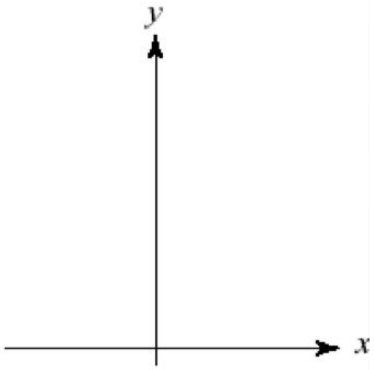
Effect on unit square

- Let's see how a general 2 x 2 transformation **M** affects the unit square:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} \mathbf{p} & \mathbf{q} & \mathbf{r} & \mathbf{s} \end{bmatrix} = \begin{bmatrix} \mathbf{p}' & \mathbf{q}' & \mathbf{r}' & \mathbf{s}' \end{bmatrix}$$

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & a & a+b & b \\ 0 & c & c+d & d \end{bmatrix}$$



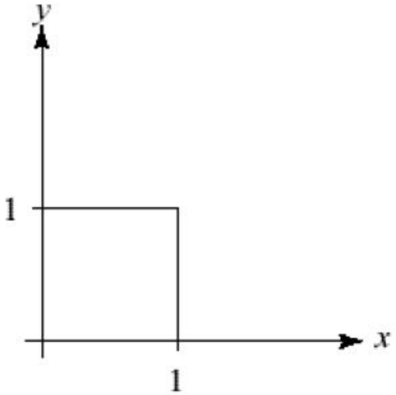


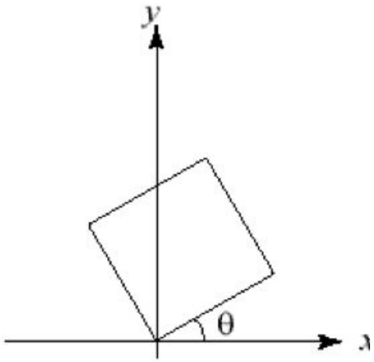
Effect on unit square, cont.

- Observe:
- Origin invariant under M
- M can be determined just by knowing how the corners (1,0) and (0,1) are mapped
- a and d give x- and y-scaling
- b and c give x- and y-shearing

Rotation

- From our observations of the effect on the unit square, it should be easy to write down a matrix for "rotation about the origin":





$$\begin{bmatrix} 1 \\ 0 \end{bmatrix} \rightarrow \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix}$$

$$\begin{bmatrix} 0 \\ 1 \end{bmatrix} \rightarrow \begin{bmatrix} -\sin(\theta) \\ \cos(\theta) \end{bmatrix}$$

Thus

$$M_R = R(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

Linear transformations

The unit square observations also tell us the 2x2 matrix transformation implies that we are representing a point in a new coordinate system:

$$
\begin{aligned}
\mathbf{p}' &= \mathbf{M}\mathbf{p} \\
&= \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{u} & \mathbf{v} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \\
&= x \cdot \mathbf{u} + y \cdot \mathbf{v}
\end{aligned}
$$

Where $\mathbf{u} = [a\ c]^{\mathrm{T}}$ and $\mathbf{v} = [b\ d]^{\mathrm{T}}$ are vectors that define a new basis for a **linear space**.
The transformation to this new basis (a.k.a., change of basis) is (a.k.a., change of basis) is a
linear transformation.

Limitations of the 2 x 2 matrix

A 2 x 2 linear transformation matrix allows

- Scaling
- Rotation
- Reflection
- Shearing

Q: What important operation does that leave out?

Affine transformations

- In order to incorporate the idea that both the basis and the origin can change, we augment the linear space $\mathbf{u}$, $\mathbf{v}$ with an origin $\mathbf{t}$.
- Note that while $\mathbf{u}$ and $\mathbf{v}$ are **basis vectors**, the origin $\mathbf{t}$ is a **point**.
- We call $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{t}$ (basis and origin) a **frame** for an **affine space**.
- Then, we can represent a change of frame as:

$$\mathbf{p}' = x \cdot \mathbf{u} + y \cdot \mathbf{v} + \mathbf{t}$$

- This change of frame is also known as an **affine transformation**.

Affine transformations, cont.

- An affine transformation is any transformation that preserves collinearity (i.e., all points lying on a line initially still lie on a line after transformation) and ratios of distances (e.g., the midpoint of a line segment remains the midpoint after transformation).
- In general, an affine transformation is a composition of rotations, translations, magnifications, and shears.

$$u = c_{11}x + c_{12}y + c_{13}$$

$$v = c_{21}x + c_{22}y + c_{23}$$

- c_{13} and c_{23} affect translations, c_{11} and c_{22} affect magnifications, and the combination affects rotations and shears.

Combinations of Transforms

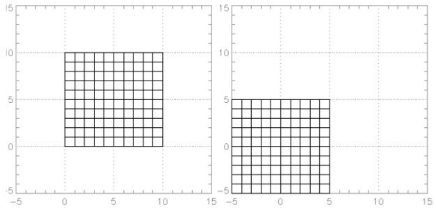
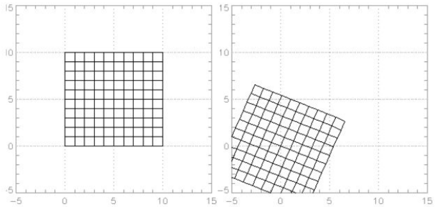
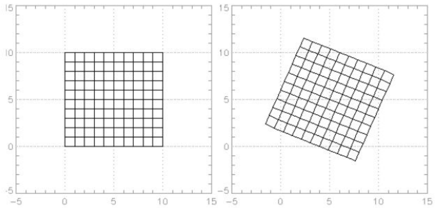
- Complex affine transforms can be constructed by a sequence of basic affine transforms.
- Transform combinations are most easily described in terms of matrix operations. To use matrix operations we introduce homogeneous coordinates. These enable all affine operations to be expressed as a matrix multiplication. Otherwise, translation is an exception.
- The affine equations are expressed as

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- An equivalent expression using matrix notation is

$$\mathbf{q} = \mathbf{T}\mathbf{p}$$

Combined Transform Operations

Operation	Expression	Result
Translate to Origin	$\mathbf{T}_1 = \begin{bmatrix} 1.00 & 0.00 & -5.00 \\ 0.00 & 1.00 & -5.00 \\ 0.00 & 0.00 & 1.00 \end{bmatrix}$	
Rotate by 23 degrees	$\mathbf{T}_2 = \begin{bmatrix} 0.92 & 0.39 & 0.00 \\ -0.39 & 0.92 & 0.00 \\ 0.00 & 0.00 & 1.00 \end{bmatrix}$	
Translate to original location	$\mathbf{T}_3 = \begin{bmatrix} 1.00 & 0.00 & 5.00 \\ 0.00 & 1.00 & 5.00 \\ 0.00 & 0.00 & 1.00 \end{bmatrix}$	

Composite Affine Transformation

- The transformation matrix of a sequence of affine transformations, say T1 then T2 then T3 is

$$\mathbf{T} = \mathbf{T}_3 \mathbf{T}_2 \mathbf{T}_1$$

- The composite transformation for the example above is

$$\mathbf{T} = \mathbf{T}_3 \mathbf{T}_2 \mathbf{T}_1 = \begin{bmatrix} 0.92 & 0.39 & -1.56 \\ -0.39 & 0.92 & 2.35 \\ 0.00 & 0.00 & 1.00 \end{bmatrix}$$

- Any combination of affine transformations formed in this way is an affine transformation.
- The inverse transform is

$$\mathbf{T}^{-1} = \mathbf{T}_1^{-1} \mathbf{T}_2^{-1} \mathbf{T}_3^{-1}$$

How to Find the Transformation

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

Point Matching

Find a number of points $\{p_0, p_1, \dots, p_{n-1}\}$ in image A that match points $\{q_0, q_1, \dots, q_{n-1}\}$ in image B. Use the homogeneous coordinate representation of each point as a column in matrices $\mathbf{P}$ and $\mathbf{Q}$:

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

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

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

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

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

Image Transformation

The transformed A image is on the right and the original B image is on the left. The dark area on is the region of B that is not contained in A. The gray image values were computed by triangular interpolation of the gray values of A.

Mapped A

Original B

Affine Transformations in OpenCV

- [OpenCV Tutorial link]

- Homework3
  a. Follow OpenCV tutorial link above.
  b. Compare two affine warp
     - Using OpenCV API `getAffineTransform()`
     - Using LAPACK (MKL) solver