CSE 473 / 573 prerequisites lecture

Kevin R. Keane, PhD

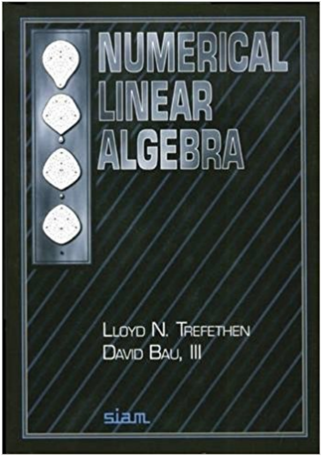
Overview

1. Revised schedule / do-over
2. Linear Algebra
   - Definitions
   - Matrix Times a Vector
   - Matrix Times a Matrix
   - Matrix Inverse
3. Homework Zero
   - Color Channel alignment
   - Image warping

Today's lecture

- HW0 deadline slides two days: September 8, 2017
- Course is very dependent on linear algebra and MATLAB
- Today's (revised) plan:
  - Introduction to (or review of) crucial linear algebra
  - How to learn more linear algebra (at least SVD)
  - Introduction to (or review of) MATLAB hacking skills
  - How to learn more MATLAB

Numerical Linear Algebra



- First section, *Fundamentals*, available on UB*learns*.
- Read it! Learn it! Live it!

Column vector

Let x be an n-dimensional column vector:

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad .$$

Matrix

Let A be an $m \times n$ matrix (m rows, n columns):

$$A = \begin{bmatrix} a_{1,1} & \dots & a_{1,n} \\ \vdots & & \vdots \\ a_{m,1} & \dots & a_{m,n} \end{bmatrix} \quad .$$

Matrix-vector product

Matrix A is $m \times n$ and vector x is $n \times 1$, so resulting vector b is $m \times 1$.

Dimensions: $(m \times n)(n \times 1) \rightarrow (m \times 1) \quad .$

Matrix-vector product $b = Ax$ is an m-dimensional column vector with elements

$$b_i = \sum_{j=1}^{n} a_{ij} x_j, \quad i = 1, \ldots, m. \tag{1}$$

b_i is the ith entry of b, a_{ij} denotes the i, j entry of A (ith row, jth column), and x_j denotes the jth entry of x.

Linear map

The map $x \to Ax$ is *linear*, which means that

$$\begin{aligned} A(x+y) &= Ax + Ay, \\ A(\alpha x) &= \alpha A x. \end{aligned}$$

Conversely, every linear map from R^n to R^m can be expressed as multiplication by an $m \times n$ matrix.

*** Preferred interpretation: x acts on A to produce b ***

Let a_j denote the jth column of A, an m-vector. Equation 1 can be rewritten to emphasize x_j's action on column vector a_j:

$$b = Ax = \sum_{j=1}^{n} x_j a_j \quad . \tag{2}$$

This equation can be displayed schematically as follows:

$$b = Ax = \left[\begin{array}{c|c|c|c} & & & \\ a_1 & a_2 & \dots & a_n \\ & & & \end{array}\right] \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

$$b = x_1 \begin{bmatrix} \\ a_1 \\ \\ \end{bmatrix} + x_2 \begin{bmatrix} \\ a_2 \\ \\ \end{bmatrix} + \dots + x_n \begin{bmatrix} \\ a_n \\ \\ \end{bmatrix}$$

Matrix - matrix multiplication

For the matrix-matrix product $B = AC$,
each column of B is a linear combination of the columns of A.

Matrix - matrix multiplication

If A is $l \times m$ and C is $m \times n$, then B is $l \times n$, with entries

$$b_{ij} = \sum_{k=1}^{m} a_{ik} c_{kj} \quad . \tag{3}$$

Here b_{ij}, a_{ik}, and c_{kj} are entries of B, A and C respectively.

Matrix - matrix multiplication

$$B = AC$$

$$\left[\begin{array}{c|c|c|c} & & & \\ b_1 & b_2 & \ldots & b_n \\ & & & \end{array}\right] = A \left[\begin{array}{c|c|c|c} c_1 & c_2 & \ldots & c_n \end{array}\right] \tag{4}$$

$$b_j = Ac_j$$

You know how to do matrix-vector multiplication from Equation 2.

Matrix - matrix multiplication

Equation 3 becomes

$$b_j = Ac_j = \sum_{k=1}^{m} c_{kj} a_k \quad . \tag{5}$$

Thus column b_j in matrix B is a linear combination of the columns a_k with coefficients c_{kj} in vector c_j.

Homework / home school

Homework: read the parts we are skipping in class!

- Range and Nullspace
- Rank

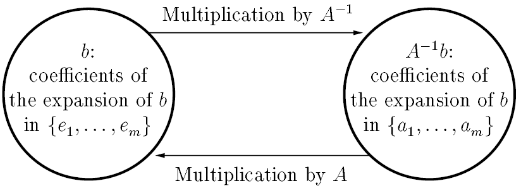
Inverse

e_j is the canonical unit vector with 1 in the jth entry and zeros elsewhere. e_j can be *expanded*

$$\left[\begin{array}{c|c|c} e_1 & \ldots & e_m \end{array}\right] = I = AZ \quad . \tag{6}$$

- The matrix Z is the *inverse* of A.
- Any square nonsingular matrix A has a unique inverse, written A^{-1}, that satisfies $AA^{-1} = A^{-1}A = I$.
- Read about *** *Matrix Inverse Times a Vector* ***.
  - Interpretation: multiplication by A^{-1} is a change of basis operation.

Matrix Inverse Times a Vector



$$A^{-1}b \to c$$
$$Ac \to b$$

More home school

For subduing *BIG DATA*, singular value decomposition (SVD) is very useful and very important. On UB*learns*, read the Wall et.al. *SVD & PCA* article. Its well written and very important for your toolbox.

hw0 clues

A few alignment metrics are proposed.

- Sum of Squared Differences (SSD)
- Normalized Cross Correlation (NCC)

hw0 / script1.m

Example (starter code)

```
% Problem 1: Image Alignment

%% 1. Load images (all 3 channels)
red = [];   % Red channel
green = [];   % Green channel
blue = [];   % Blue channel

%% 2. Find best alignment
% Hint: Lookup the 'circshift' function
rgbResult = alignChannels(red, green, blue);

%% 3. Save result to rgb_output.jpg
```

hw0 / alignChannels.m

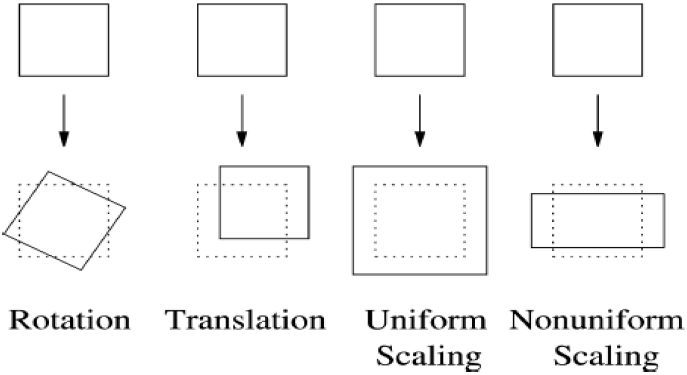
Example (starter code)

```
function [rgbResult] = alignChannels(red, green, blue)
% alignChannels - Given 3 images corresponding to different
%       color image, compute the best aligned result with m
%       aberrations
% Args:
%   red, green, blue - each is a matrix with H rows x W col
%       corresponding to an H x W image
% Returns:
%   rgb_output - H x W x 3 color image output, aligned as d

%% Write code here

end
```

Hacking strategy

- Hit a search engine for more info on SSD and NCC
- Find out how to load MATLAB data
- Find out how to . . .
- Read good MATLAB / GNU Octave code
    - Most functions *blah* have readily available open source *blah*.m for Octave implementation !!!
    - Read it
    - Learn from it
    - DONT CUT & PASTE - we are usually asking for a simpler case

Computer vision matrix operations



Thanks to Dustin Bielecki for sharing Professor Esfahani's slides.
See UB*learns* / Course Documents / Linear Algebra / Geometric Transforms.

hw0 / script2.m

The three important matrices for computer vision in the HW code.

- Scale
- Transform
- Rotate

Example (cool snippets)

```
% Define inline function to create an
% affine scaling matrix:
Scalef = @(s)([ s 0 0; 0 s 0; 0 0 1]);
% Same for translation
Transf = @(tx,ty)([1 0 tx; 0 1 ty; 0 0 1]);
% Same for rotation
Rotf=@(t)([cos(t) -sin(t) 0; sin(t) cos(t) 0; 0 0 1]);
```

Scale

$$\begin{aligned} S &= \begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ Sp &= \quad ? \end{aligned} \tag{7}$$

Translation

$$T = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \tag{8}$$

$$Tp = \quad ?$$

Rotation

$$
\begin{aligned}
R &= \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \\
Rp &= \quad ?
\end{aligned} \tag{9}
$$

hw0 / script2.m

How are these matrices used?

Example (cool snippets)

```
% Center around cx,cy, rotate it a bit and scale.
A = Transf(out_size(2) / 2, out_size(1) / 2) ...
    * Scalef(0.8) ...
    * Rotf(-30 * pi / 180) ...
    * Transf(-cx, -cy);
warp_im = warpA_check( im_gray, A, out_size );
% warp_im2 = warpA( im_gray, A, out_size );
```

The End