CS 179: Lecture 12 Recitation

cuBLAS, cuSolver, and Point Alignment

Announcements, Spring 2021

- See Piazza Post for **small update on Homework 4** about new noisy bunny data
  - Set 4 is set up to be easy (hopefully) to give you time to prepare for midterms in other classes.
- This year, **Project Proposals are now due w/ Lab 5**. See course web page.
- Different from last year, where Projects Proposals were due with Lab 4, since classes last year started 7 days later than normal, due to COVID-19.
- Also for this year, **Lab 6 is no longer optional** since we have one more week over last year. See web page for details on Lab 6 and due dates
- Note: Cuda and cuBLAS were reinstalled on Titan yesterday – check your old HWs to make sure they still compile and work correctly!

Recap

- cuBLAS is CUDA’s linear algebra library!
- Good for operations between vectors, matrices, etc.
  - Feels like Matlab, Numpy, etc.
  - Heavily optimized for us already
  - Very general, can use in many applications. For example, many cublas functions subsume some of the reductions that we have been writing by hand so far such as parallelized max or parallelized sum.
- Why ever write your own kernels?
  - More control, sometimes allows for less data i/o
    - Many calls can increase overhead
    - Bad support / growing environment

Today

- Finish covering cuBlas via example
- What is cuSolver?
    - Matrix factorization
    - Parallel LU solve
- Point alignment
    - Like least squares
    - Will solve for a linear transformation that matches one set of points to another

Cublas Example

What is cuSolver

- There are primitive linear solver capabilities within cuBLAS
  - Under BLAS-like extensions
  - Mostly very primitive, not very well supported.
- cuSOLVER is entirely designed for solving linear systems
- Two big things:
  - Factorizations
  - Backsubstitution / solving factorized system
    - Great for dense linear systems

What is a linear system?

Want to solve this problem:

$$Ax = b$$

Know matrix A, know vector b, want to determine what vector x is.

Naive solution: Invert A if possible!

$$X = A^{-1}b$$

Worst solution! Bad numerical stability (same reason that 1/x generally unstable if x is small).

Naming Convention -- Like cuBLAS

`cusolverDn<t><operation>`

Also Has handle

```
cusolverStatus_t
```

```
cusolverDnCreate(cusolverDnHandle_t *handle);
```

Factorizations?

Fact:

Can factorize any matrix A into the following form:

P A = L U

L = Lower Triangular (1’s on diagonal)

U = Upper triangular

P = Permutation matrix (for numerical stability)

Why useful?

Want to solve

$Ax = b$

$LU x = P b$

=> Solve

$L y = Pb$

$U x = y$

Solving triangular matrices is easy!

Backsubstitution.

$$
\begin{array}{rcl}
a_{1,1}x_1 + a_{1,2}x_2 + a_{1,3}x_3 + \cdots + a_{1,n-1}x_{n-1} + a_{1,n}x_n & = & b_1 \\
a_{2,2}x_2 + a_{2,3}x_3 + \cdots + a_{2,n-1}x_{n-1} + a_{2,n}x_n & = & b_2 \\
a_{3,3}x_3 + \cdots + a_{3,n-1}x_{n-1} + a_{3,n}x_n & = & b_3 \\
\vdots \qquad\qquad \vdots & & \vdots \\
a_{n-1,n-1}x_{n-1} + a_{n-1,n}x_n & = & b_{n-1} \\
a_{n,n}x_n & = & b_n
\end{array}
$$

Can solve multiple RHS simultaneously!

Multiple RHS

Furthermore, multiple RHS increases the parallelism of the application.

Can improve performance, even over fast and optimized CPU code!

http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.2.9349

How to solve on the hw?

cusolverDn<t>getrf()

cusolverDn<t>getrs()

Will do some of the solves simultaneously.

Point Alignment

Everything in homogeneous coordinates (bias term in ML-lingo)

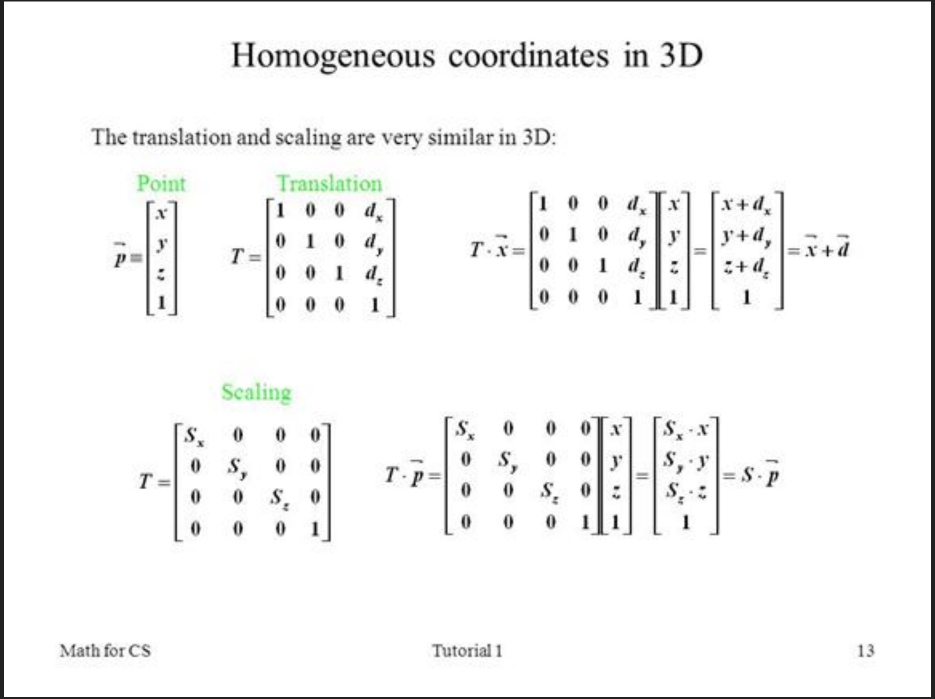
Want to find matrix M of size 4 x 4

Points X1, N x 4 match to X2, N x 4

Looks like least squares, we will solve:

$X1^T X1 . M^T = X1^T X2$

Linear system!



Homogeneous coordinates in 3D

The translation and scaling are very similar in 3D:

Point

$$\vec{p} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Translation

$$T = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad T \cdot \vec{x} = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x + d_x \\ y + d_y \\ z + d_z \\ 1 \end{bmatrix} = \vec{x} + \vec{d}$$

Scaling

$$T = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad T \cdot \vec{p} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} S_x \cdot x \\ S_y \cdot y \\ S_z \cdot z \\ 1 \end{bmatrix} = S \cdot \vec{p}$$

Math for CS Tutorial 1 13

Multiple RHS!

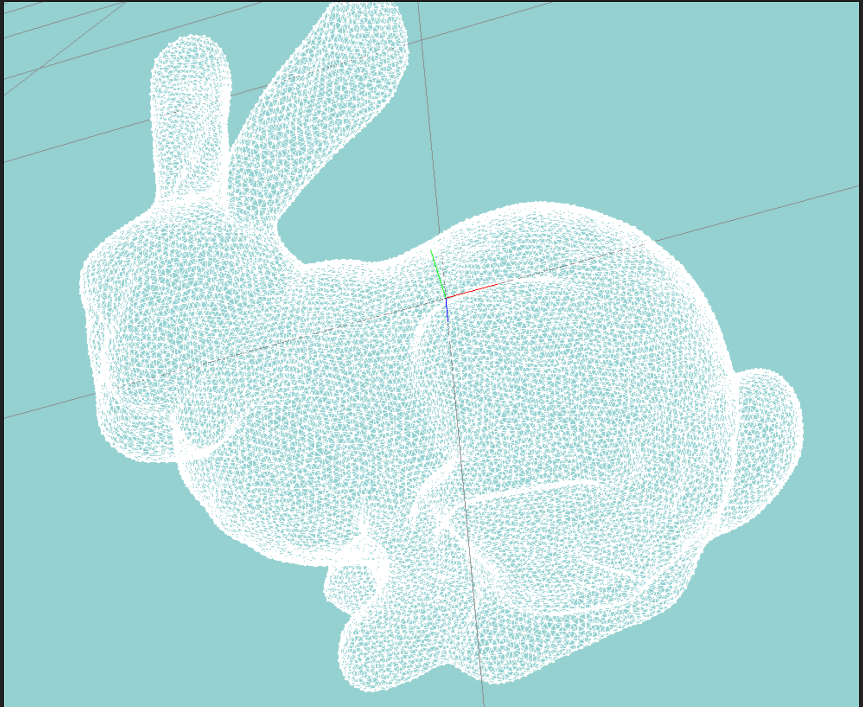
$$X1^T X1 \,.\, M^T = X1^T X2$$

Minimizes distance from X1 points transformed by M to points X2.

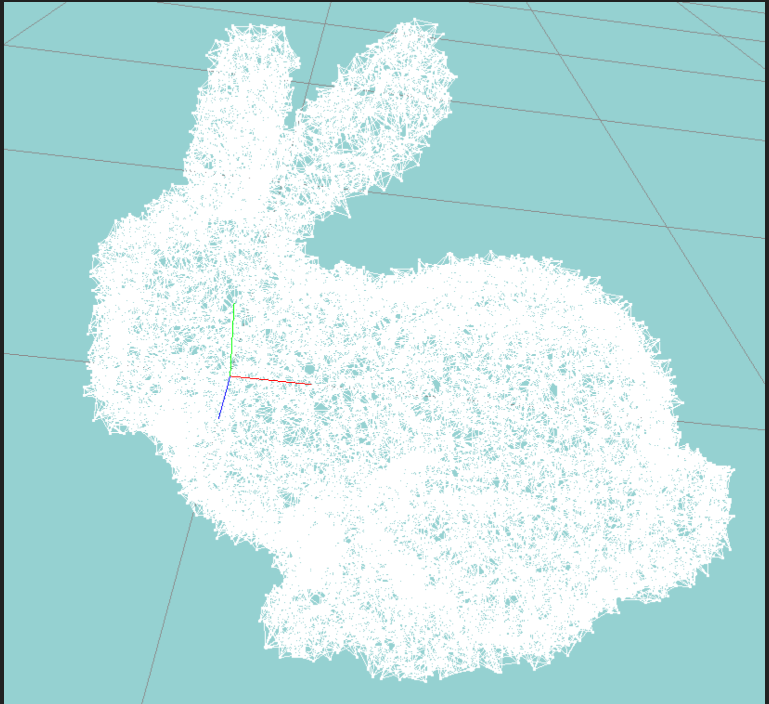
Set bottom row of M to zero with a 1 at the end.

Datasets for point alignment

Original & rotated bunny



Noisy rotated bunny



Questions?