Basic Requirement

- Using geometric algorithms to solve real-world problems in biomedicine
  - Not limited to algorithms in the Modules
  - Not necessary to completely solve the problem, but need to show sufficient progress towards a solution

- Distributable tool
  - Implemented in mainstream languages: C/C++, Java, Python, etc., or as plug-ins to existing software (e.g., ImageJ, Chimera)
    - Prototyping in Mathematica is recommended
  - A graphical user interface (GUI) may be required
Algorithms that will be covered…

- Creating, processing, deforming, and analyzing geometry

Finding a problem

- Pick a problem mentioned by our guest speakers
  - Breast / lesion segmentation from mammograms
  - Measuring thickness of subchondral bone from CT
  - Counting osteoblast cells in histological images
Finding a problem

• Where to look for your own topic:
  – Your friends who work on image-related research
  – Faculty in Med School, BME, Biology, Chemistry, etc.
    • Many faculty working on imaging can be found through Imaging Science Pathway: [http://imagingpathways.wustl.edu/](http://imagingpathways.wustl.edu/)
  – Open problems of current interest
    • Check for research papers in imaging-related conferences (e.g. MICCAI) and journals (e.g., IEEE TMI)

Finding a problem

• Example problems in the past

DNA Gel Electrophoresis Analyzing Tool
(Gabriel Stancu)
Finding a problem

- Example problems in the past

Breast lesion segmentation
(Noa Ben-Zvi)

Finding a problem

- Example problems in the past

Measuring small bowel length
(Billy Bennett)
Finding a problem

- Example problems in the past

Analyzing 3D cell shape in 2-photon microscopy
(Aron Lurie and Daniel Melzer)

Finding a problem

- Example problems in the past

Registering deforming heart surfaces
(Christopher Gloschat)
Finding a problem

- Avoid problems that are too simple
  - E.g., edge detection, image blurring, or anything that can be solved by a few Matlab/Mathematica/Python commands.

- … or too hard
  - E.g., automatic segmentation of X, a complete system for doing Y, etc.

- Talk to Instructor before turning in the proposal!

Guidelines

- Individual or in pairs
  - Requirements will be harder for team projects

- Choose any language you prefer

- Discussion with the instructor or guest speaker is encouraged
Format

• Project proposal
  – What is the problem, what are the required and wish-list features, and timeline of development.

• Final presentation (80%)
  – A 10-minute in-class live demo of what you have accomplished

• Hand-in (20%)
  – Source code, executable tool (or plug-in), and test data
  – Project report
    • Description of the algorithms, features of your tool, and how to use them
    • Any known bugs and future work

Timeline

• Project proposal
  – Due by Thursday, Oct 24 (two weeks away)
  – If you work on a new problem, discussion with Instructor prior to proposal submission is required.

• Final presentations
  – Two dates: Tuesday, Dec 3rd and Thursday, 5th
  – State your preference of these two dates in your proposal

• Hand-ins
  – Due by midnight Tuesday, Dec 10th
Helpful Materials

• 3D programming
  – OpenGL: Industrial standard 3D library for C/C++
    • The “red book”: OpenGL Programming Guide
      • Great tutorial lessons: http://nehe.gamedev.net/lesson.asp?index=01
    – Jogl: Java bindings for OpenGL
      • Docs, samples, forums: http://jogamp.org/jogl/www/

• GUI programming in C++
  – Qt: http://qt.nokia.com/products/
  – FLTK: http://www.fltk.org/