A2.md 9/24/2018

Assignment 2: Optimization

Due: 11:59pm

Introduction

This assignment will introduce you to the basics of gradient-based continuous optimization. The optimization we will solve leverages the fact that our parametric shapes are represented implicit surfaces, functions that are < 0 inside the object surface and > 0 outside. We can formulate the problem of deforming our object such that the nearest surface point ends up at a particular point in space as an optimization. In this assignment you will:

- 1. Formulate the point targetting problem as a continuous optimization.
- 2. Implement a gradient descent optimization to solve this problem.
- 3. Implement backtracking line search to automatically choose the gradient descent step length.
- 4. Implement finite differences to compute the second derivatives, the Hessian, of the objective function.
- 5. Implement Newton's method using your backtracking linesearch
- 6. Produce convergence plots for all implemented algorithms.

Part A: Setup Assignment Code Base

Once you have completed the setup for A1, setting up A2 is substantially easier.

- 1. Run 'cd {SRC_DIRECTORY}/A2'
- 2. Run 'mkdir ./build'
- 3. Run 'cd ./build'
- 4. Run 'cmake .. -DCMAKE_BUILD_TYPE=RELEASE' to build the project Makefile.
- 5. Run 'make -j8'
- 6. This should build an execubtable named ./bin/CompFabA2.

Part B: Formulate the cost function and gradient [4 marks]

- 1. Let \$p\$ and \$y\$ be points in \$\mathcal{R}^3\$ and let \$f\left(y\right)\$ be the value of the implicit function representing your shape, evaluated at \$y\$. Keeping in mind that \$f=0\$ on the surface, write a least squares cost function that is minimized when \$p\$ lies on the surface. **Include your derivation in the assignment writeup**. [2 marks]
- 2. Derive the gradient for your cost function. **Include your derivation in the assignment writeup**. [2 marks]

Part C: Implement Gradient Descent [7 marks]

A2.md 9/24/2018

 Implement basic gradient descent as described in lecture. Function protoypes have been provided in {SRC_DIRECTORY}/A2/include/Optimization.hand {SRC_DIRECTORY}/A2/src/Optimization.inl. [2 marks]

- 2. Modify the file {SRC_DIRECTORY}/A2/scripts/robotArm.cpp to use your gradient descent method to move the robot arm so it touches the specified target point, \$p\$. You can use the member functions **value** and **jacobian** of the **Shape** class to access \$f\left(y\right)\$ and \$\frac{\partial f}{\partial z}}\$ respectively. Here \$z\$ are the parameters of a parameterized shape, in this case the angles of the robot arm. [2 marks]
- 3. Run gradient descent with the step sizes \$1\$, \$0.5\$ and \$0.1\$. Make a single plot showing cost (y-axis) vs number of iterations (x-axis) for all step sizes. [3 mark]. ** Include this plot in the assignment writeup**.

Part D: Backtracking Linesearch [7 marks]

- 1. Implement backtracking linesearch as described in lecture [5 marks].
- 2. Add a version of gradient descent to <code>Optimization.hpp</code> that uses linesearch rather than a step-length parameter. [1 mark].
- 3. Include the convergence of this new method to your plot from part B. [1 mark].

Part E: Finite Differences and Newton's Method [5 marks]

- 1. Implement a centered finite difference approach to compute the Hessian matrix of your cost function. This hessian will be a dense, 2 by 2 matrix. Store it in an Eigen::MatrixXd variable. [2 marks]
- 2. Use your finite difference hessian to implement Newton's method with backtracking linesearch in Optimization.hpp and Optimization.cpp [2 marks].
- 3. Include the convergence of this new method to your plot from part B. [1 mark].

Hand In [1 mark]

Collect all required images into a PDF report which must include your full name and student number. Submit this PDF and a zip file containt your A2 source code via **email** to **diwlevin@cs.toronto.edu**. The subject of the email must be **CompFabA2_LASTNAME_STUDENTNUMBER**.