CIS 510 Final Project

Predicting Atom Positions Using Machine Learning Algorithms

Project Team:
Adam Martini
Wes Erickson
Ran Tian

April 25, 2014

Executive Summary

The Steck Lab is investigating the motion of a single atom in a Magneto Optical Trap (MOT). The atom is imaged using a pair of aspheric lenses and a CCD camera. Given an 2D image from the CCD, the lab wants to predict the probability distribution of the atom's location within the MOT. This project intends to both simulate the images generated by the camera, and create a classification algorithm that will estimate the atom probability distribution.

1 Project Description

To tackle this problem we will run simulations of the CCD output for a given atom location by solving the linear wave equation using a Partial Differential Equation (PDE) Solver and/or a raytracing simulation. The output images will be labeled by atom location and used as training set for machine learning algorithms. The algorithms can then be used to generate a probabilistic distribution for the atom's location for a new CCD image.

2 Highlevel Architecture

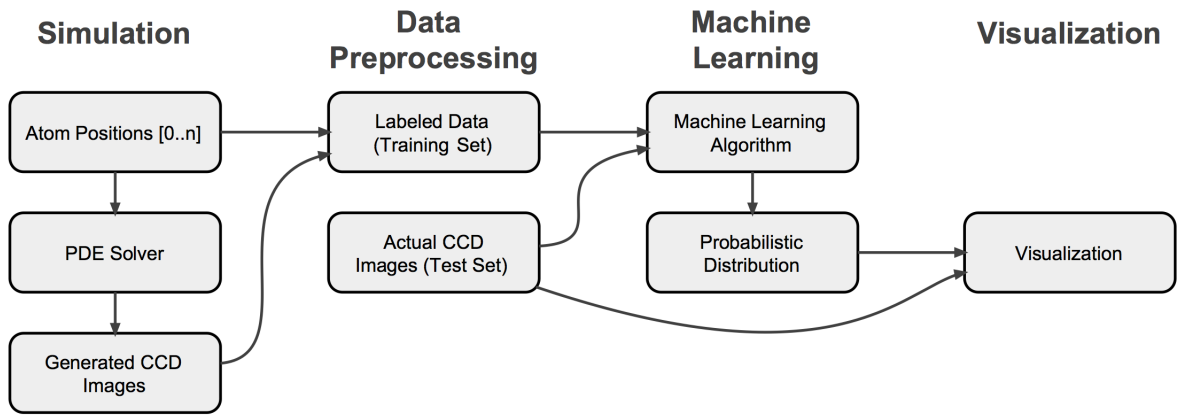

Figure 1: High Level Data Flow Architecture Diagram

Figure 1 shows a block level diagram of the project data flow. There are three main components to the solution.

Simulation

Machine Learning Test multiple machine learning algorithms to find and optimize a strong predictive model.

Visualization

2.1 Simulation

The essence of the simulation is starting with a flourescent atom at a chosen location, and then simulating how light emitted from the atom appears on the CCD. There are two approaches to simulating the generation of CCD images from a given atom location.

The first is a classical raytracing approach. Rays are emitted in a random direction from the atom, and obey refraction via Snell's Law as they pass interfaces between different materials. The rays also contain phase information for the purposes of determining interference. When the rays intersect with the CCD plane, we increment an array value cooresponding to which CCD pixel it hit.

The second approach is a full simulation of the linear wave equation. This is a full 3D simulation, and is much more computationally intensive. A field discretized into $200 \times 200 \times 1000$ is given an initial value of zero everywhere except for a peak at our atom location, and then is stepped forward in time according to the linear wave equation. The essence of this algorithm is basic matrix multiplication, but with very large sparse matrices.

2.2 Machine Learning

We will train and test multiple machine learning algorithms on the simulated images by splitting the training data into a training and validation set. The validation set will be used to determine the best algorithm for this application and then to tune its parameters. The final classifier will be used to generate probabilistic predictions for the test set of actual CCD images.

2.3 Visualization

Visualization is particularly important for visualizing three dimensional volumes, and it may become essential for feedback as to whether our simulations are functioning properly, and for our presenting our project results. However, our project focus is not on programming the visualization methods ourselves, but using existing tools such as Visit or ParaView.

The main items to be visualized are the field generated from the differential equation solvers, and the CCD images (which are a cross section of the field). We can also use these tools for post processing on the image data to highlight features for the benefit of the machine learning algorithm.

2.4 Data Format

The data will be formatted in the standard Visualization Tool Kit (VTK) format. We will use a regular regtagonal structured grid.

3 Parallel Plan

The each stage of the project data flow includes several places to introduce and explore parallelism.

Simulation The two simulation approaches outlined our architecture section both are highly parallelizable problems.

The first approach involving raytracing can be parallelized over each simulated ray while accumulating the results of the CCD image in a single 2D matrix. The quality of the output is determined by the number of rays emitted (and the quality of random numbers generated). The ray simulations are completely independant as well, so the problem can easily be parallelized by launching multiple instances with different random number generator seeds.

The second approach involves numerically solving differential equations. Existing differential equation libraries, such as PETSc, are capable of running in parallel. We will use one of these for simulating how light propegates from a point source, through lenses, to a CCD camera. Additionally, we can explore creating a more basic differential equation solver using MPI and basic linear algebra packages and compare performance to the more sophisticated PETSc library. Both of these approaches require more finnesse when adapting a serial program to many cores, since we cannot segment the data and simulate independantly. Rather, updates along the borders must be shared between cores.

Machine Learning There is at least one parallel machine learning library written in c++ that we can use as a starting point. To get specialized functionality or algorithms that perform well for image recognition, we may need to extend existing libraries or code our own algorithms. It is unclear whether a distributed program will be necessary to train the models given the number of features and size of the dataset.

Visualization Applications such as Visit or ParaView are powerful visualization tools that can be run in a distributed enviornment. We can use these tools to generate animations and imagery of the solutions in parallel. It is possible that parallelization may prove unnessesary depending on how long it takes our data to render, but the feature is avaliable if we need it.

4 Project Schedule

Week	Deliverable
5	PETSc PDE solver installed and running data runs OR raytracing implementation generating data runs, identify parallel classifiers and get them running on ACISS.
6	Get classifiers working on ACISS and extend/create code based on prediction goals.
7	Process initial data ouput to identify best classifiers based on performance and accuracy.
8	Finalize and optimize simulations and classifier code.
9	Create visualizations, do performance analysis, begin writup and presentation.
10	Finalize writeup and presentation.