William Yang
Group 14
Mentor: Dr. Rogerio Richa
Visual Tracking of Surgical Tools in Retinal Surgery using Particle Filtering

Mutual Information Computation and Maximization Using GPU

Yuping Lin and Gérard Medioni
Computer Vision and Pattern Recognition Workshops (CVPR)
Anchorage, AK, pp. 1-6, June 2008

Project Summary and Paper Selection

Our project, which involves developing a software package for the visual tracking of surgical tools during retinal surgery, involves the use of a particle filter for stochastic optimization tracking of mutual information, a visual similarity measure useful under conditions of limited texture information and variable illumination and rotation. A robust optimization procedure would use a higher number of particles in the filter, which results in an increase in computational power to maintain a high refresh rate. Part of our maximum deliverable is to run part of the software on a graphics processing unit (GPU) using CUDA, NVIDIA's GPU computing architecture, which is capable of certain types of massively parallel calculations, and would increase the computational throughput available to us.

This paper, by Lin and Medioni, evaluate a GPU implementation of calculating the mutual information metric. The direct relevance to our project as a result of the use of mutual information with a GPU/CUDA is apparent. In addition, images from retinal surgery were used as testing datasets for this paper, which should give us a realistic idea of what to expect from a GPU in our environment.

Technical Summary

Introduction

Mutual information is an information theory quantity that is useful for registration and comparison of multi-modal images. Aligning these images is especially important in medical imaging, but is a difficult problem to solve especially with direct pixel-to-pixel similarity measures such as sum squared difference (SSD) or normalized cross-correlation (NCC). Maximization of mutual information is a common approach and is the one taken in this paper; however, it is computationally expensive as calculating the marginal and joint probabilities at each pixel requires an exponential number of computations. This paper focuses on adapting

Viola's approach to computing mutual information since it provides a close form solution of the derivatives needed for a gradient descent approach to maximization.

Mutual Information Approximation

The mutual information of two random variables u and v is defined as

$$MI(u,v) = h(u)+h(v)-h(u,v)$$

where the entropy and joint entropy are defined respectively as:

$$h(v) = -sum_v[p(v)\ln(p(v))] \qquad h(u,v) = -sum_{u,v}[p(u,v) * \ln(p(u,v))]$$

Viola's method estimates the probability density p(v) over the samples A by the Parzen Window method:

$$p(v) = (1/N_A) * sum_{(vj\ from\ A)} G_w(v-v_j)$$

where G_w is a Gaussian function with variance w. This technique, robust to noise, uses a Gaussian Mixture model to estimate the density of samples drawn from an unknown distribution. Since the entropy function can also be written as the negative expectation of ln p(v), the entropy of a random variable can then be approximated as

$$h(v) \approx \frac{-1}{N_B} \sum_{v_i \in B} \ln \frac{1}{N_A} \sum_{v_j \in A} G_\psi(v_i - v_j)$$

where B is another sample set. This leads us to arrive at an approximation of mutual information:

$$MI(u,v) \approx \frac{-1}{N_B} \left\{ \sum_{u_i \in B} \ln \frac{1}{N_A} \sum_{u_j \in A} G_{\psi_u}(u_i - u_j) + \sum_{v_i \in B} \ln \frac{1}{N_A} \sum_{v_j \in A} G_{\psi_v}(v_i - v_j) - \sum_{w_i \in B} \ln \frac{1}{N_A} \sum_{w_j \in A} G_{\psi_{uv}}(w_i - w_j) \right\}$$

MI(u,v) is then maximized over a rigid transformation T using an approximated derivative:

$$\frac{d}{dT} MI(I_u(x), I_v(T(x))) = \frac{1}{N_B} \sum_{x_i \in B} \sum_{x_j \in A} (v_i - v_j)^T W(x_i, x_j) \frac{d}{dT}(v_i - v_j)$$

where W is a weighting factor.

CUDA Implementation

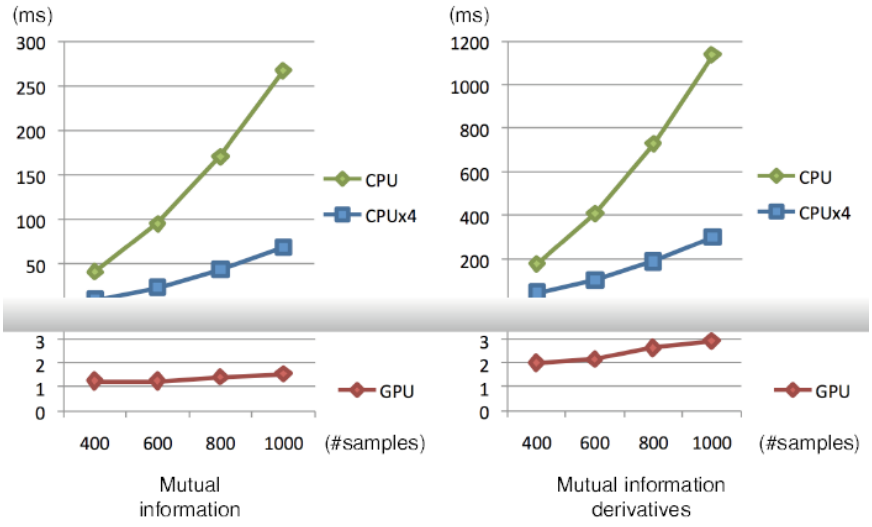
On a fundamental level, CUDA features the Single Instruction, Multiple Data (SIMD) computational architecture, where a single instruction operates on multiple data streams in lock step. Use of the shared memory is key to speeding up computations as it is located on-chip;

however, it is relatively small in size (16K in the GPU tested in the paper). The inner summations of MI(u,v) and (d/dT)MI(u,v) were parallelized as each element in B can be calculated independently of the others. Such an approach also facilitates fast loading of shared memory from global memory since parallel loading is possible when there are no bank conflicts.

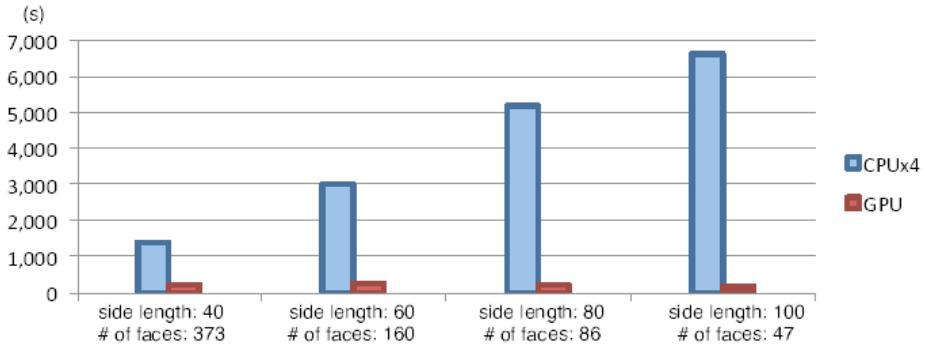
Validation and Results

Registration of two retinal images using a rigid transformation was performed by maximization of mutual information. A triangular mesh was used, where the mutual information derivative is computed over the displacements of the vertices of each triangle. Gradient descent was then performed iteratively until convergence.

The paper found that, as the number of samples increased, the time for the GPU to compute MI derivatives remained relatively constant while CPU time increased quadratically. At 1000 samples, a CPU running four parallel threads was still two orders of magnitude slower than the GPU.



In another experiment, a 50 iteration registration between two retinal images was performed. Samples on fractional positions used bilinear interpolation on the CPU; since each iteration required a new interpolation, additional overhead of copying the interpolated data to the GPU memory for every iteration was incurred. Nevertheless, the CPU implementation took hours while the GPU implementation completed in minutes.



Conclusions and Future Work

Using the approximation methods described here as well as effectively using shared memory enables using mutual information with a high enough number of samples and iterations for use in image registration within a reasonable amount of time as compared to a CPU-based approach. Future work would entail integration into applications and conducting additional experiments. The authors also expressed interest in comparing the performance with other approximation methods.

Critical Analysis

This paper is interesting from both a technical standpoint and in terms of presenting a novel approach using current technology. The mathematical details of the algorithm were presented adequately, and a serial pseudocode algorithm was also included in the paper.

On the other hand, while there is some description of the experiments that the authors performed to evaluate performance, I found that more detail about the test datasets would have been helpful. In addition, it would be ideal if the retinal images used could be released or be from a standardized research dataset. The paper could have also discussed the pros and cons of choosing Viola's algorithm as compared to other options, although the authors do mention exploring other algorithms as part of future work. Variance between different testing datasets was also absent, which limited my ability to assess how data-specific their results were.

From a technical standpoint, I felt that some consideration should have been given to the memory bandwidth of the CPU implementation, especially as it was considered an important factor in GPU performance. Bilinear interpolation is expensive on the CPU and can be done on the GPU using texture memory[1], and is future work that I would like to see. I will grant the

benefit of the doubt that the older CUDA architecture being used in the paper may not yet have had this feature, which is present in the CUDA architecture today.

Relevance to Project and Conclusions

We currently use a joint histogram method to calculate mutual approximation, which the authors of the paper identified as being more difficult to extend to a GPU for computation of MI derivatives. We may proceed with our current histogram method as a paper is cited as having successfully implemented mutual information with histograms on a GPU [2]. In addition, we do not need to calculate the derivatives of MI for gradient descent maximization; we identified early on that, for our purposes, gradient descent was inappropriate as an optimization method due to the problem of local minima in tool tracking.

The factor not addressed is the performance of calculating many mutual information measures in succession over a smaller number of samples, as would effectively be our case with the particle filter. Preliminary testing with source code released by the authors of this paper as well as by Shams has not yielded promising results; however, the current method employed is admittedly inefficient where the high number of memory transfers between the host and GPU device is too high to realize any benefits from the parallel processing on the GPU. In addition, we hope to run multiple mutual information calculations on the GPU in parallel, rather than accelerate the speed of an individual MI calculation. Nevertheless, this paper effectively demonstrates that a GPU implementation of mutual information possesses the capability to significantly increase computational throughput.

Reading List

1. NVIDIA CUDA C Programming Guide 4.1. 2011

2. R. Shams and N. Barnes. Speeding up mutual information computation using NVIDIA CUDA hardware. *Digital Image Computing Techniques and Applications*, pages 555–560, 2007.