Large Histogram Computation for Normalized Mutual Information on GPU

Sophie Voisin Devin A. White
Oak Ridge National Laboratory - GIST
Oak Ridge, TN 37831-6017
voisins@ornl.gov whiteda1@ornl.gov

Jeremy S. Archuleta
Oak Ridge National Laboratory - CDA
Oak Ridge, TN 37831-6085
archuletajs@ornl.gov

I. Introduction

This poster presents two GPU implementations of normalized mutual information (NMI) for a two-step image registration used in version 1.0 of the Photogrammetric Registration of Imagery from Manned and Unmanned Systems (PRIMUS) pipeline [1]. This fully-automated, sensor-agnostic application enables the precise geolocation of aerial and orbital imagery, a fundamental building block for accurate geographic applications and data analysis. In PRIMUS, NMI is used for coarse and refined alignment of multi-spectral imagery, referred to as *global localization* and *registration*, respectively. However, different algorithms have been implemented in CUDA to fit the different characteristics of each alignment.

Although NMI is commonly used to register images from different modalities [2], GPU implementations usually focus on single coefficient computation leveraging entropy approximation calculus or generating histograms with a limited number of bins [3][4][5]. However, for our use case, we need to generate the full joint-histogram of 65,536 bins to mitigate the loss of accuracy due to compression that occurs when reducing the image-intensity range from 16-bit to 8-bit, needed to leverage most of the functions in OpenCV library. It is also important to note that our use case involves a mask that indicates valid pixels that are considered for the NMI coefficient computation this adds to the computational complexity and uniqueness of our solution but can significantly improve its overall accuracy. Our preliminary approaches target the NVIDIA Tesla K80, not only to leverage the global memory available on the GPU but also its queue manager, which lends themselves well to our specific problem.

II. Normalized Mutual Information

In image processing, the NMI coefficient is used as a similarity measurement between two images with different modalities. It is derived from the entropies of the two images $H(I_1)$ and $H(I_2)$ and their joint entropy $H(I_1, I_2)$.

$$NMI(I_1, I_2) = (H(I_1) + H(I_2))/H(I_1, I_2)$$
$$H(I) = -\sum_i^n P(i) \log_2 P(i)$$

The entropy $H(I)$ is derived from the probability distribution P of the intensity values i for each valid pixel. Here it corresponds to the normalized histograms for the valid pixels in both images and to the normalized joint-histogram.

III. Coarse Alignment - Global Localization

A. Dense Joint-Histogram for Full Image Matching

Our coarse alignment is based on image template matching performed on reduced resolution images. In the PRIMUS pipeline, the original images are resized by a factor $\sim 20\times$. Since resizing affects accuracy via loss of resolution and aggregation of intensity values, global localization has been designed as an exhaustive search of the solution space. As a proof of concept, the problem has been simplified to a rigid deformation: translation with no scaling. This assumption can be made since a sensor model contained in the metadata of the images and corresponding terrain data are used to initialize the relative position and resolution of the images. The solution space is therefore all possible translations of the source image over the control image. As a concrete example of figure 1, for a resized source image of 500x100 pixels and a control image of 1,200x300 pixels, there exist 700×200 = 140,000 NMI coefficients to be calculated.

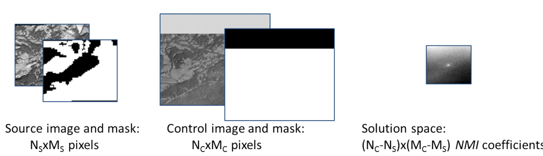

Fig. 1. Example of the inputs and output for the coarse alignment.

B. Approach

Our preliminary kernel only currently leverages the global memory of one GPU of the NVIDIA Tesla K80. To avoid thread synchronization and atomic operations we compute one NMI coefficient per thread. Through testing we have chosen blocks with 128 threads to optimize memory storage, reduce memory access, and minimize divergence from conditional statements. Contrary to a kernel that would compute one NMI coefficient at a time with the CPU launching as many kernels as the number of coefficients, this implementation saves time by launching only one kernel encompassing all the computations using the internal GPU queue manager to iterate over the number of blocks.

C. Results

The following experiment has been designed to evaluate the performance of the kernel with respect to the number of coefficients to be computed. A source image of 512×256 pixels and control images ranging from 527×383 to 991×383 pixels

This manuscript has been authored by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725 with the U.S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes. The Department of Energy will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan(http://energy.gov/downloads/doe-public-access-plan).

were used to generate from 2048 to 61,440 NMI coefficients. While the first variant of the kernel K1 simply uses the mask values (0 or 1) to increment the joint-histogram considering the entire source image, the second implementation K2 uses a conditional statement to access only the valid pixels to increment the joint-histogram.

Although similar timings are reported for K1 and K2 with all valid data (131,072 pixels), fig. 2, the timing when the mask contains 50% of valid data (65,536 pixels), K2*, is almost cut by half. We believe this to be due to a decrease in 1) memory accesses to the pixel values of both images, 2) computations of the corresponding index to the histogram, and 3) increment operations of the histogram entries. In addition, the way our problem is set up reduces the likelihood of diverging threads within a block since it is likely that neighbors of a masked pixel are likely masked themselves.

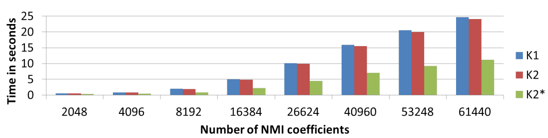

Fig. 2. Kernel performance with respect to the number of solutions.

IV. Refined Alignment - Registration

A. Sparse Joint-Histogram for Keypoint Descriptor Matching

Our refined alignment is based on keypoint matching between images at higher resolution, once they are coarsely aligned. Keypoints are detected in the source image and corresponding keypoints are searched within a given window in the control image using spatial knowledge, fig. 3. This transforms the problem from section III-A into matching an 11×11 pixel template keypoint descriptor, extracted from the source image with a 73×73 pixel sub-image from the control image creating the need for 3,969 NMI coefficients to be computed per keypoint. This requires a different approach to implement the NMI computation since an individual keypoint matching problem is at a different scale than the approach described in section III-B. Furthermore, masking is no longer necessary since keypoints and descriptors are only found and defined within the valid pixel areas.

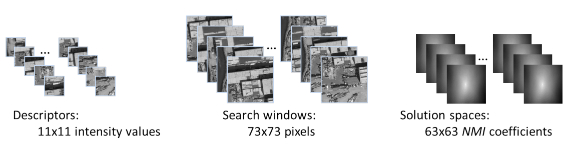

Fig. 3. Example of the inputs and outputs for the refine alignment.

B. Approach

Similar to the first approach this kernel also leverages global memory to store and access joint-histogram data. However the algorithm was designed such that one block is computing the 63x63 NMI coefficients associated with one keypoint. In other words, the kernel has as many blocks as keypoints in the source image. Another difference is that each block has 64 threads (1 idle) with each thread computing 63 NMI coefficients. Shared memory is used to store all the coefficients for subsequent search of the best solution. An important optimization for this particular algorithm is to leverage the size of the descriptor. The fact that the joint-histogram of 65536 bins only contains at most 121 distinct values allows us to use a list to store the histogram entries. The list is kept locally for each thread during the histogram generation for easy access during entropy computation, fig. 4.

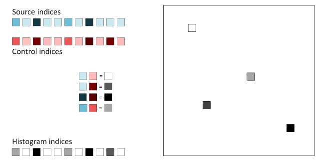

Fig. 4. The joint-histogram indices are computed from the intensity values of the images and stored for future use.

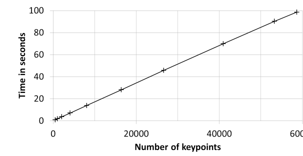

Fig. 5. Kernel performance with respect to the number of keypoints

C. Results

A straightforward experiment was designed to time the kernel with respect to the number of keypoints. The performance shows a linear increase of the time with respect to the number of keypoints. As a result one can see that only 100 seconds is required to process more than 58K keypoints and to compute more than 230M NMI coefficients, fig. 5.

This tailored implementation of NMI coefficient computation may be of interest for similarity measurements of small size descriptors where the scale dimension resides in the number of measurements to be computed and not in the size of the data. If PRIMUS is generalized so it can run on other hardware, that implementation should consider less global memory usage.

V. Discussion

This preliminary work highlights the need for multiple NMI algorithms depending on the specifications of the imagery and particular use cases. We presented two unique GPU implementations to compute NMI coefficients without compromising on the 65,536-bin joint-histogram, even for the global localization where a mask increases the computational complexity. For this work, our target GPU was only one of the two GPUs present on the NVIDIA Tesla K80 but we plan to expand our research to leverage the additional computational power of the full device. A significant amount of research and development is still needed to optimize NMI algorithms for imagery applications where either the images themselves and/or the solution space are larger than the memory and computation resources on the GPU.

References

[1] Devin A. White and Chris R. Davis, *A fully-automated high performance image registration workflow to support precision geolocation for imagery collected by airborne and spaceborne sensors*, GeoComputation, Dallas, TX, USA, 2015.

[2] F. Maes, D. Loeckx, D. Vandermeulen, and P. Suetens, *Image registration using mutual information*, N. Paragios et al. (eds.), Handbook of Biomedical Imaging: Methodologies and Clinical Research,© Springer Science+Business Media New York 2015

[3] Cedric Nugteren, Gert-Jan van den Braak, Henk Corporaal and Bart Mesman, *High Performance Predictable Histogramming on GPUs: Exploring and Evaluating Algorithm Trade-offs*, GPGPU-4, Newport Beach, CA, USA, March 2011.

[4] R. Shams andR. A. Kennedy, *Efficient histogram algorithms for NVIDIA CUDA compatible devices*, International Conference on Signal Processing and Communications Systems, ICSPCS, 17-19 December 2007.

[5] Yuping Lin and Gerard Medioni, *Mutual information computation and maximization using GPU*, IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, 2008. CVPRW '08, Anchorage, Alaska, USA, 23-28 June 2008.