GPU Accelerated Hausdorff Distance Computation Using Mathematical Morphology

Érick Oliveira Rodrigues, Esteban Clua and Aura Conci
Institute of Computer Science, Universidade Federal Fluminense, Niterói – Rio de Janeiro, Brazil (contact: erickr@id.uff.br)

ABSTRACT

Hausdorff distance (HD) is a vastly used metric in visual computing for comparing images, finding patterns, performing registrations, etc. However, its exhaustive implementation can be seen as a potentially large combinatorial problem, which consequently consumes a great amount of processing time. That is, in a two binary images comparison, every pixel of one image must be evaluated with respect to every pixel of the other, and this step must be performed twice, switching the images. In this work, we propose two parallel algorithms for computing the Hausdorff distance using morphological dilations on the GPU. The algorithms require block synchronization and both the CPU-based and GPU-based block synchronizations were evaluated. Furthermore, we compare the efficiency of the proposed algorithms to implementations on the CPU and GPU regarding distinct programming languages including C++, Java and also the Aparapi library. Experimental results have shown that the CUDA CPU-based synchronized algorithm provided the best results and was approximately 26 times faster than the CPU for large binary images. As to the extent of grey images (one binary computation for each grey level), the obtained speed can be up to 6630 times faster with CUDA for images of sizes 4096x4096 and 8196x8196.

HAUSDORFF DISTANCE

Given two finite non-empty sets $A = \{a_1, a_2, \dots, a_n\}$ and $B = \{b_1, b_2, \dots, b_m\}$, the Hausdorff distance between both is defined as the maximum distance an element contained by one of the sets has to travel until arriving at the other set. In other words, the Hausdorff distance is equal to the maximum distance from a point of a set you may reside to the nearest element or point of the other set. The formulation is given by $H(A, B)$:

$$H(A,B) = \max(h(A,B), h(B,A))$$

where,

$$h(A,B) = \max_{a \in A}(\min_{b \in B} ||a - b||)$$
$$h(B,A) = \max_{b \in B}(\min_{a \in A} ||b - a||)$$

and $||.||$ stands for the distance norm between the points of the sets (e.g., Euclidean norm). The functions $h(A, B)$ and $h(B, A)$ are respectively called forward and backward Hausdorff distance. When $h(A, B) = d$, the distance from all points in A to the nearest point in B does not exceed d. In other words, all points in A are at distance d or less from set B [1].

METHODOLOGY

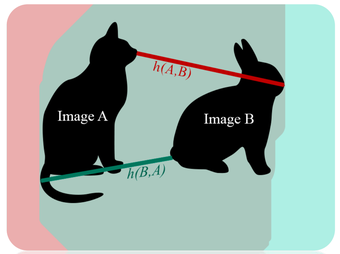
Mathematical Morphology (MM) [2] studies the geometrical structures present in images using mathematics. Its main principle is extracting information related to the geometry and topology of subsets of an image using structuring elements [3]. The structuring element is often defined by a set of elements representing pixels. Dilations and erosions are the principles and the two main operations in MM. A binary dilation of an image I by a structuring element S is given by the equation below:

$$dil^S(I) = \bigcup_{s \in S} I_{-s}$$

Where I_{-s} represents the elements of I translated by $-s$. In other words, the dilation is given by the union of $I - s$ for every s in S. The following figure illustrates a dilation.

Binary image dilation

Assuming we have two binary images A and B as input, it is possible to compute HD by dilating image A until it covers image B and dilating image B until it covers image A while computing the amount of dilations. The greater between the two amounts is the Hausdorff distance. This process is illustrated by the figure below:



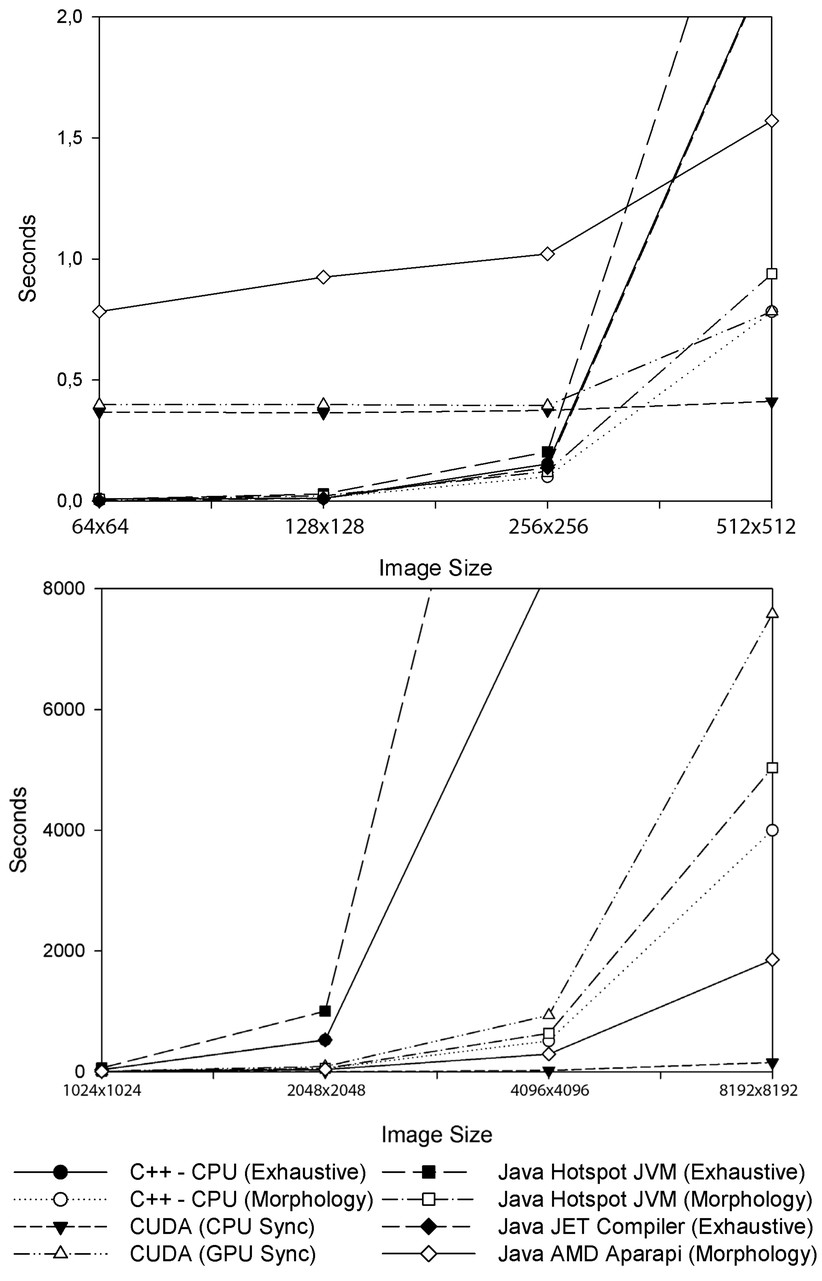
The proposed CPU-synchronized kernel is depicted in the pseudo code bellow, where image1P and image1PAux are copies of the input image img1, as well as image2P and image2PAux are copies of img2 at iteration 0. The copies imageP and imagePAux are used to process the dilations, and remain in the GPU memory throughout the iterations. For each input binary image there is therefore a usage of 3 times the memory each binary image requires. In conclusion, the algorithm requires $6wh$ bits of memory for comparing two binary images of width w and height h. The number of iterations (also the Hausdorff distance) is incremented in the CPU. If the *grownEnough* variable remains true after a kernel call, the iterations halt and the Hausdorff distance is obtained.

```
//grownEnough is reset to true in the CPU before each kernel call, if it remains true
after the kernel call then the algorithm is finished
1. int id = blockDim.x * blockIdx.x + threadIdx.x;
2. if (id < w*h){
    1. if (img1PAux[id]) /*dilate img1P at pixel id*/;
    2. if (img2PAux[id]) /*dilate img2P at pixel id*/;
3. }
4. atomicAnd(grownEnough, (img2P[id] || !img1[id]) && (img1P[id] || !img2[id]));
```

In summary, the approaches using mathematical morphology are faster but require more memory ($6wh$) while the brute force or exhaustive approaches require only $2wh$ but are much slower as it can be seen in the following sections.

EXPERIMENTAL RESULTS

The results were obtained using a Nvidia Geforce GTX 960M and an Intel Core i7-4720HQ clocked at 2.60GHz. The Java code was run in a Hotspot JVM and also compiled with Excelsior JET. The CPU-oriented algorithms were implemented sequentially (run in a single core). However, even if they were run in a multi-core paradigm, the GPU-oriented algorithm would still produce significant speed ups if large images are regarded. In a no overhead scenario, a CPU-oriented algorithm would have its computation time divided by the number of cores in a CPU. Therefore, if a quadcore CPU runs the mathematical morphology approach (which requires dependence of data at iteration level and is not trivial to parallelize), the GPU-version would still be at least 6 times faster for binary images of sizes 4096x4096 and 8196x8196 and up to 1530 times faster for grey images of the same size. The following charts compare the processing time (including memory copies and kernel execution) of each addressed implementation for the binary case.



CONCLUSION / REFERENCES

For images of size greater than 256x256, the computation of the Hausdorff distance on the GPU starts to be beneficial, with 2x speed ups for 512x512 instances. We have evaluated the algorithm for images up to 4096x4096 and 8192x8192, which provided speed ups of 26 and 6630 times for the binary and grey image cases, respectively.

[1] A. Conci, S.L. Galvão, G.O. Sequeiros, D.M. Saade, and T. MacHenry, A new measure for comparing biomedical regions of interest in segmentation of digital images, Discrete Appl. Math., 197 (2015), pp. 103–113
[2] G. Matheron and J. Serra. The birth of mathematical morphology, ISMM 2000 , Proceedings of the International Symposium on Mathematical Morphology.
[3] E. P. Calixto, and A. Conci, Chromaticity constant: Introducing a new ordination for automated extraction of grain–size data from true colour images, ISMM (2007), Proceedings of the 8th International Symposium on Mathematical Morphology, Vol. 2 pp.63-64