Towards Parallelization of a Texture Description Algorithm for Breast Lesion Classification using OpenMP and CUDA

Lisa Pal[1]
Amilcar Meneses Viveros[1], Wilfrido Gómez Flores[2]
[1] Computer Science Department, Centro de Investigación y de Estudios Avanzados (CINVESTAV), Zacatenco; [2] CINVESTAV, Unidad Tamaulipas

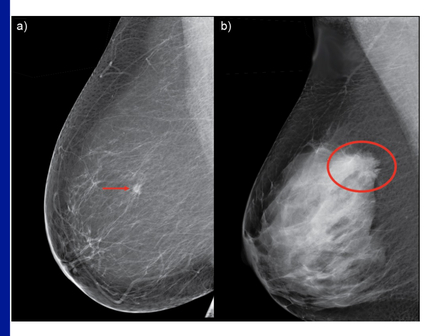
INTERNATIONAL CONFERENCE ON PARALLEL PROCESSING

Objective

- To extract texture features for breast lesion classification.
- To make this algorithm adaptable for large medical images.
- To identify sections in the algorithm that are independent and can be parallelized.
- To compare performance of sequential C++, OpenMP, and CUDA versions.

Motivation

[1]



- Different imaging modalities are required to analyze distinct types of breast tissue.
- Average mammography size is 5262 x 3131 in DDSM dataset
- AMI sequential version would take ~1.5 hr to run

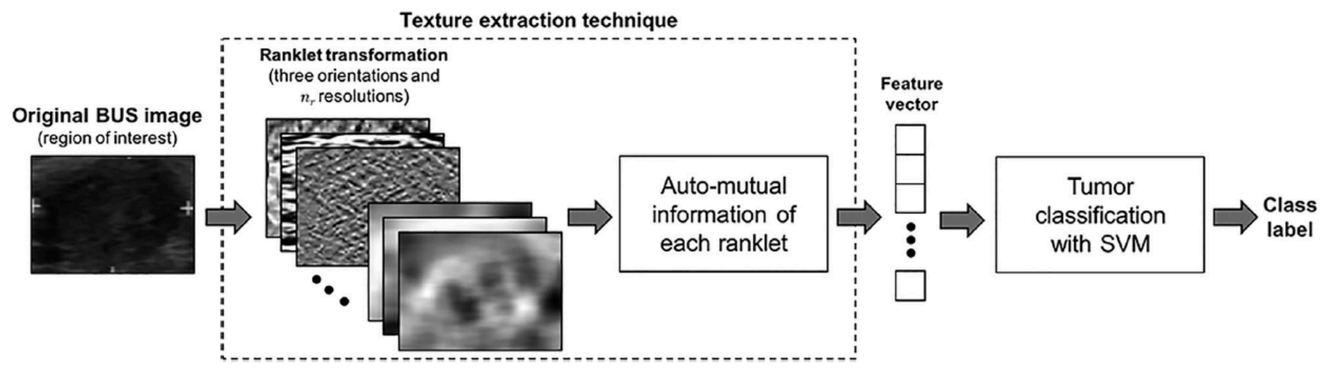
Dataset

Four sets of squared images composed of the regions of interest (ROI) obtained from mammograms. The ROIs are of size 128, 256, 512 and 1024 pixels per side. Each of the four sets contained 100 images, for a total of 400 images.

Equipment

Two Xeon processors with six cores each.
Three different Tesla C2070 NVIDIA graphics cards.

AMI Algorithm



Ranklet Transform

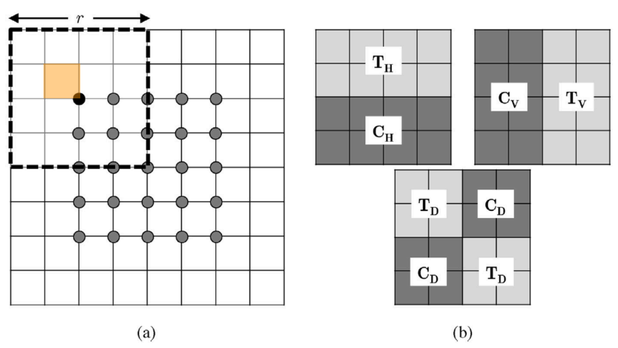


Figure 1: (a) From an image of size 8 x 8, the ranklet mask *(dashed black square)* with resolution r=4 generates 25 overlapping squares centered at gray points, **(b)** Three orientations of the ranklet mask in (a): horizontal (H), vertical (V) and diagonal (D), with their corresponding treatment (T) and control (C) subsets [2].

For each window, we substitute a single pixel in it with its calculated ranklet coefficient.

Mutual Information Features

$$I(X;Y) = \sum_{y\in Y}\sum_{x\in X} p(x,y)\log\frac{p(x,y)}{p(x)p(y)}$$

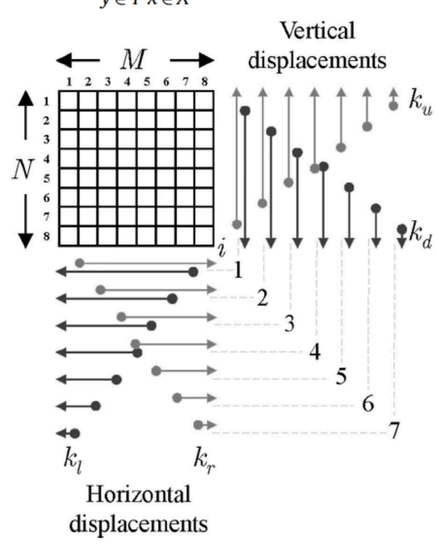


Figure 2: Image displacements for computing the directional auto-mutual information. For an image of size 8 x 8, seven displacements are performed in the vertical and horizontal directions [2],[4].

OpenMP Performance - speedup of 8.37x - 10.27x

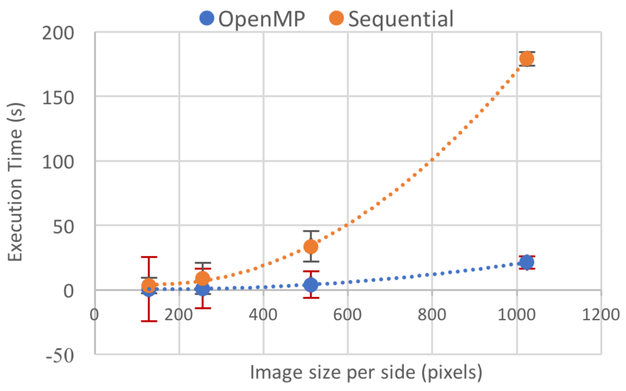


Figure 3 : Execution times for AMI for different image sizes. Relative standard deviation has been magnified by 5.

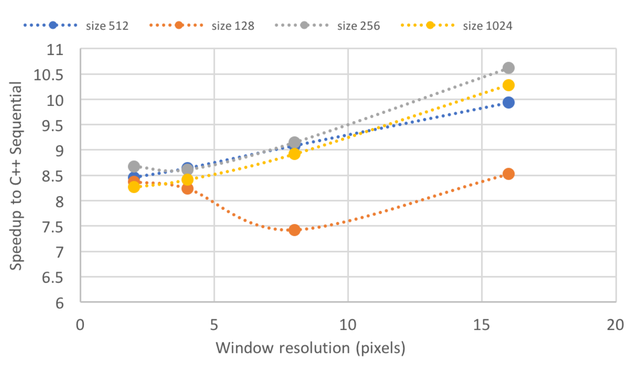


Figure 4 : Speedup to C++ sequential code for all window and image sizes in the dataset.

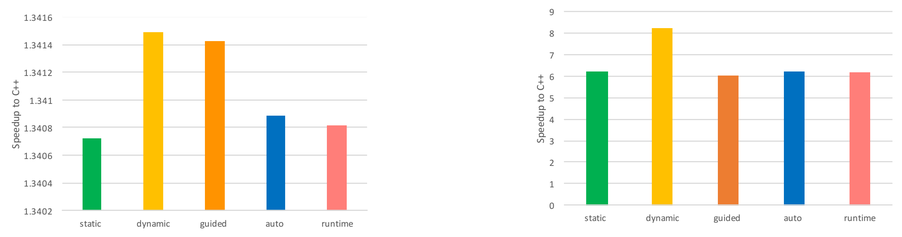
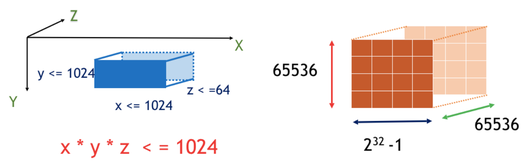


Figure 5: Speedup to C++ sequential code for different thread scheduling types for ranklet (left) and AMI (right).

CUDA Strategy

Maximum overhead expected comes from image transfer between host and device.



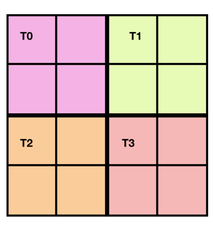


Figure 6: Each ranklet coefficient calculated by a single thread.

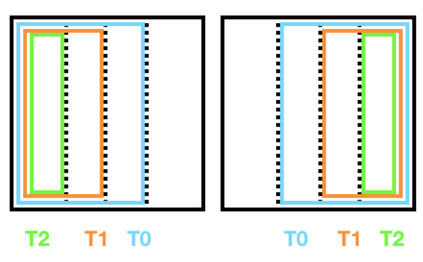


Figure 7: MI features for every pair of left and right displacements will be calculated by a single thread. Similarly for every pair of up and down displaced images.

Hybrid Strategy

Task + Instruction parallelism
Context aware implementation

```
cudaGetNumDevices(&d);

if (d>0) {
   #pragma omp setNumThreads(3)
   {
     i =  omp_get_thread_num(void);
     cudaSetDevice(i%d);
     doTaskinDevice(i);
   }
} else {
   #pragma omp setNumThreads(3)
   {
     i =  omp_get_thread_num(void);
     doTask(i)
    }
}
```

Results and Future Prospects

- OpenMP gave a 8.37-10.27 speedup to the C++ sequential algorithm.
- Performance improvement was 1.34x for ranklet transform and 8x for AMI.
- The best thread scheduling type was dynamic.
- Hybrid implementation will parallelize tasks in OpenMP and instructions in CUDA, using all resources available in the system it is running.
- We expect out future implementation to improve ranklet transform from $O(M^2r^2logr^2)$ to $O(r^2logr^2)$ and to improve AMI from $3n_r2(M-1)(M^2logM^2)$ to (M^2logM^2).

[1] Disha et. al., 2009, *Bosnian journal of basic medical sciences*, 9(2), 131–136. ***[2]*** *Masotti, Campanini,2008, Pattern Recognition Letters 29(14):1980-1986.* ***[3]*** *Gómez et al., 2019, Ultrasound in Medicine & Biology, 45(8):2213-2225.* ***[4]*** Peng et al.,2005, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8): 1226-1238.