Κωνσταντίνος Χατζής

ΑΕΜ: 9256

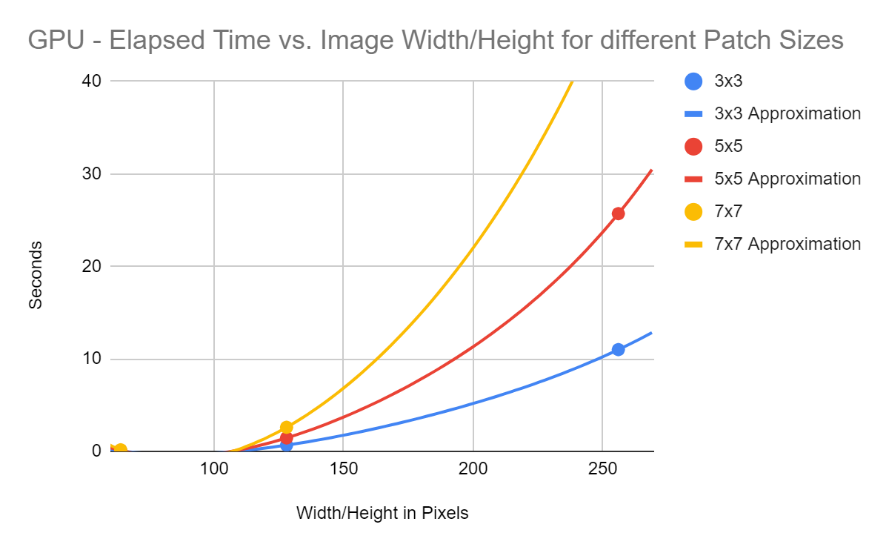
[github.com/kostascc/PDS-Project-3](https://github.com/kostascc/PDS-Project-3)

Στη Τρίτη εργασία ασχολούμαστε με το φίλτρο Non Local Means και την υλοποίησή του σε CUDA. Ακολουθεί η παρουσίαση του κώδικα, ορισμένες μετρήσεις και σχόλια / προτεινόμενες βελτιώσεις για την απόδοση του αλγορίθμου σε CPU και GPU, και εικόνες που παράχθηκαν με το φίλτρο αυτό.

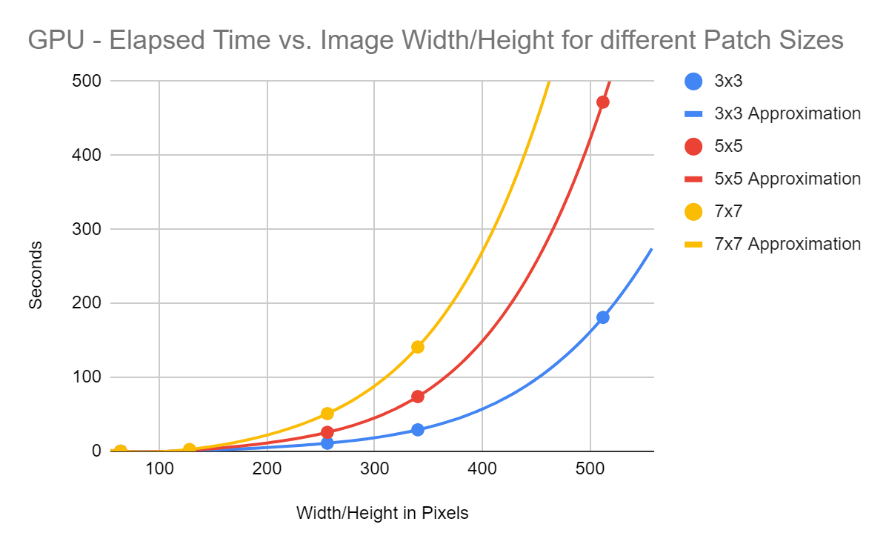
## Κώδικας

Για την παραγωγή του αλγορίθμου χρησιμοποιήθηκε το αρχείο Matlab που περιέχεται στα αρχεία της εργασίας, και με βάση αυτό παράχθηκε ο κώδικας για CPU με χρήση OpenMP (είναι αντίστοιχος με αυτόν της GPU που ακολουθεί) και στη συνέχεια μετατράπηκε σε CUDA.

Συγκεκριμένα, γίνεται κλήση του Kernel για δισδιάστατο πλήθος Block, όσα και τα Patch που περιέχει η εικόνα (πχ [256-5]x[256-5] ). Κάθε ένα από τα Βlock αυτά περιέχει επίσης δισδιάστατο πλήθος Τhread 8x8, τα οποία μοιράζονται όλα τα Patch της εικόνας στα οποία πρέπει να γίνει αναζήτηση για βάρη (δηλαδή ξανά [256-5]x[256-5] Patch, μείον ενός, αυτού από το οποίο γίνεται η αναζήτηση).

 Ένα Patch σε κάθε Block - το Patch για οποίο γίνεται η αναζήτηση - βρίσκεται πάντα στην Shared Memory απαράλλαχτο, και ένα ακόμα Patch άδειο. Στο δεύτερο γίνονται διαδοχικές προσθέσεις Pixel, ανάλογα της απόστασης που έχει το εκάστοτε Patch με το οποίο γίνεται σύγκριση. Στο τέλος, το σύνολο των βαρών που υπολογίστηκαν, διαιρείται από όλα τα Pixel του τελικού Patch, και αυτό εκχωρείται πίσω στην Global Memory.

## Απόδοση

Από τα επόμενα δυο διαγράμματα φαίνεται ότι ο αλγόριθμος κοστίζει αρκετά (σε αντίθεση με άλλα φίλτρα, όπως αυτό του kNN, ή σε σχέση με τις βελτιωμένες εκδόσεις του NLM), καθώς για μια μικρή εικόνα, στα

|  |
| --- |
|  |
| Πίνακας Pixel, όπου κόκκινο είναι ένα Patch. |
|  |
| Ένας από τους πολλαπλούς πίνακες Pixel, όπου κοκκινό είναι το Patch που μεταλλάχθηκε σε διάνυσμα. |

256x256 pixel, απαιτούνται περίπου 25 δευτερόλεπτα για την αποθορυβοποίησή της.

Οι δοκιμές που παρουσιάζονται, καθώς και η παραγωγή των εικόνων, έγιναν σε κάρτα γραφικών GTX 1650, η οποία παρέχει 4 GB μνήμης (Global Memory), και 1 MB κοινής μνήμης (Shared Memory), και ο κώδικας γράφτηκε με σκοπό την καλύτερη απόδοση στα χαρακτηριστικά της συγκεκριμένης GPU.

Παρατηρείται, από τη διάρκεια της αποθορυβοποίησης για Patches μεγέθους 7x7, έναντι των μικρότερων, ότι η μεγαλύτερη καθυστέρηση που επιβαρύνει τον αλγόριθμο είναι η χρήση της Global Memory χωρίς κάποια βελτίωση της τοπολογίας της. Αυτό έχει ως αποτέλεσμα, το Cache Miss Rate να είναι πολύ μεγάλο και ο αλγόριθμος να είναι αργός.

Μια πιθανή βελτίωση που θα μπορούσε να προταθεί για το πρόβλημα αυτό, είναι η δημιουργία αντιγράφων της εικόνας στη μνήμη, όσων και το πλάτος των Patch, και η μεταφορά των διαφορετικών σειρών ενός Patch στην ίδια σειρά, μετατρέποντάς τα έτσι σε 1D διανύσματα.

## Αποτελέσματα

Στις παρακάτω εικόνες φαίνεται η

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  | 40 |
| Patch Size: 5x5, Image Size: 340x340 | | |





|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
| Patch Size: 5x5, Image Size: 340x340 | | |

