Ακαδημαϊκό Έτος 2016-2017

Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Γεωργιάδης Αχιλλέας: 7415

[ΠΑΡΑΛΛΗΛΑ ΚΑΙ ΔΙΑΝΕΜΗΜΕΝΑ ΣΥΣΤΗΜΑΤΑ – ΕΡΓΑΣΙΑ 3]

Περιγραφή Προβλήματος

Στην εργασία αυτή κληθήκαμε να υλοποιήσουμε τον αλγόριθμο Non Local Means σε CUDA.

Σε αντίθεση με φίλτρα local mean, που υπολογίζουν την μέση τιμή σε μία γειτονιά κάθε pixel για να εξομαλύνουν την εικόνα, ο Non Local Means υπολογίζει τον μέσο όρο όλων των pixels στην εικόνα, σταθμισμένο με το βαθμό ομοιότητας με το pixel αναφοράς. Το αποτέλεσμα είναι καλύτερη ευκρίνεια και διατήρηση των λεπτομερειών της αρχικής εικόνας.

Στο συνοδευτικό αρχείο της εργασίας μας παρέχεται κώδικας σε MATLAB που υλοποιεί το pipeline του αλγορίθμου Non Local Means.

Στόχος μας λοιπόν είναι η βελτίωση της απόδοσής του με τη χρήση CUDA.

Περιγραφή Μεθόδων

Η υλοποίηση χωρίζεται σε 2 τμήματα, ακολουθώντας την δομή του δοθέντος MATLAB κώδικα.

Πρώτο Τμήμα - neighborCubeKernel.cu

Σε πρώτη φάση, υλοποιούμε τον Kernel “neighborCubeKernel.cu”. Ο kernel αυτός είναι υπεύθυνος για το πρώτο βήμα του αλγορίθμου NLM, το οποίο είναι το σκανάρισμα της εικόνας με συγκεκριμένο μέγεθος παραθύρου γειτονιάς, 3x3 – 5x5 – 7x7.

Μπορούμε να φανταστούμε το output του kernel ως έναν κύβο με διαστάσεις M*N*(K*K), για διαστάσεις εικόνας M*N και μέγεθος γειτονιάς K*K. Ο κύβος αυτός αναπαριστάτε ως ένας 2D πίνακας διαστάσεων (M*N) * (K*K), ονόματι B, με αριθμό γραμμών όσα τα pixel της εικόνας, και κάθε γραμμή περιλαμβάνει το κάθε pixel μαζί με την γειτονιά του. Η υλοποίηση σε CUDA έγινε ακλουθώντας το sampleKernel.cu.

Επιπλέον, ο kernel αυτός είναι υπεύθυνος και για το filtering του κάθε pixel με ένα φιλτρο H που υλοποιείται εντός MATLAB.

Δεύτερο Τμήμα - denoisingKernel.cu

Σε δεύτερη φάση, υλοποιούμε τον Kernel “denoisingKernel.cu”. Ο kernel αυτός είναι υπεύθυνος για το δεύτερο βήμα του αλγορίθμου NLM, το οποίο είναι ο υπολογισμός των βαρών και το τελικό Denoising της εικόνας.

Αρχικά υπολογίζονται για κάθε pixel της εικόνας, μέσω του πίνακα – κύβου B, οι ευκλείδειες απόστασης της γειτονιάς του, με όλες τις υπόλοιπες γειτονιές. Η ευκλείδεια απόσταση θεωρείται ως μία μετρική Dissimilarity, πόσο δηλαδή διαφέρει ή όχι η κάθε γειτονιά από τις υπόλοιπες.

Βάση αυτής της απόστασης υπολογίζεται το Z(i) του τύπου $Z(i) = \sum_j e^{-\frac{\|f(\mathcal{N}_i) - f(\mathcal{N}_j)\|^2_{G(a)}}{\sigma^2}}$

Τέλος υπολογίζεται για κάθε pixel η νέα, αποθορυβοποιήμενη τιμή του, βάση του τύπου

$$w(i,j) = \frac{1}{Z(i)} e^{-\frac{\|f(\mathcal{N}_i) - f(\mathcal{N}_j)\|^2_{G(a)}}{\sigma^2}}.$$

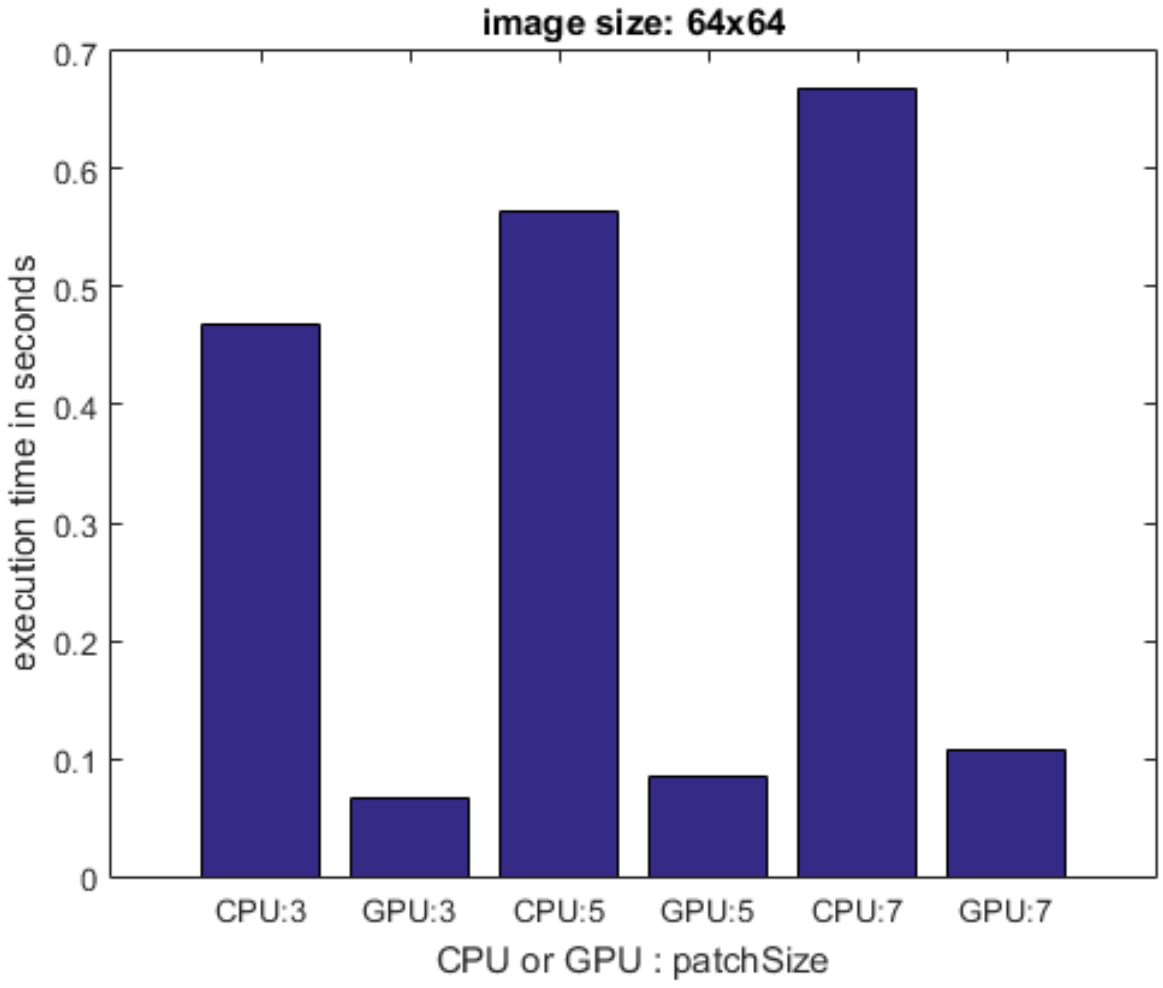
Σύνδεση με MATLAB

Όσον αφορά τη σύνδεση CUDA με MATLAB είναι αρκετά απλή, και γίνεται με την χρήση των εντολών που παρέχονται στο sample που μας δόθηκε.

Πειράματα – Σχόλια

Όλα τα πειράματα και οι μετρήσεις έγιναν στον προσωπικό μου υπολογιστή σε κάρτα γραφικών NVIDIA GTX 970 με CUDA v.8.0, CPU Intel i7 4790k, 16GB RAM και OS Ubuntu 16.04 LTS.

Στο παρακάτω διάγραμμα φαίνονται οι χρόνοι εκτέλεσης του Non Local Means, για μεγέθη εικόνας 64x64, 128x128, 256x256, και γειτονιές 3x3, 5x5, 7x7.



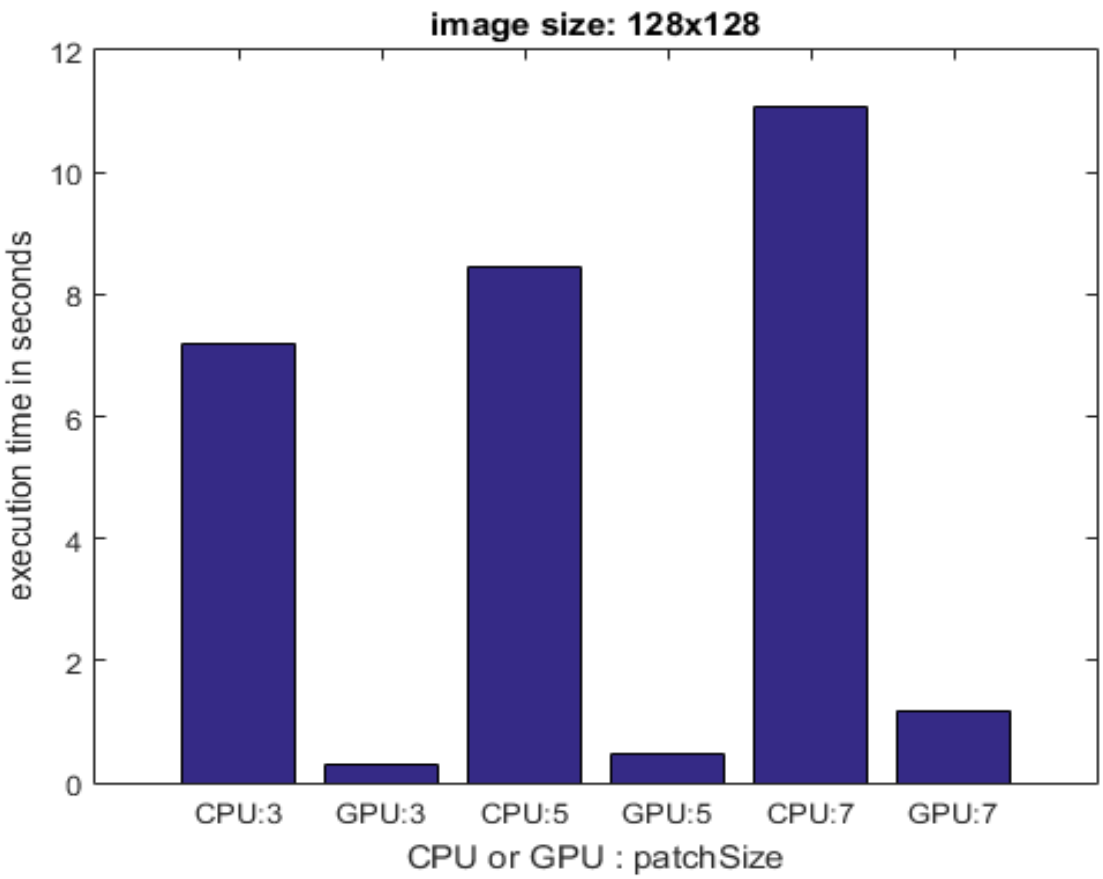


image size: 128x128
execution time in seconds
0
2
4
6
8
10
12
CPU:3
GPU:3
CPU:5
GPU:5
CPU:7
GPU:7
CPU or GPU : patchSize

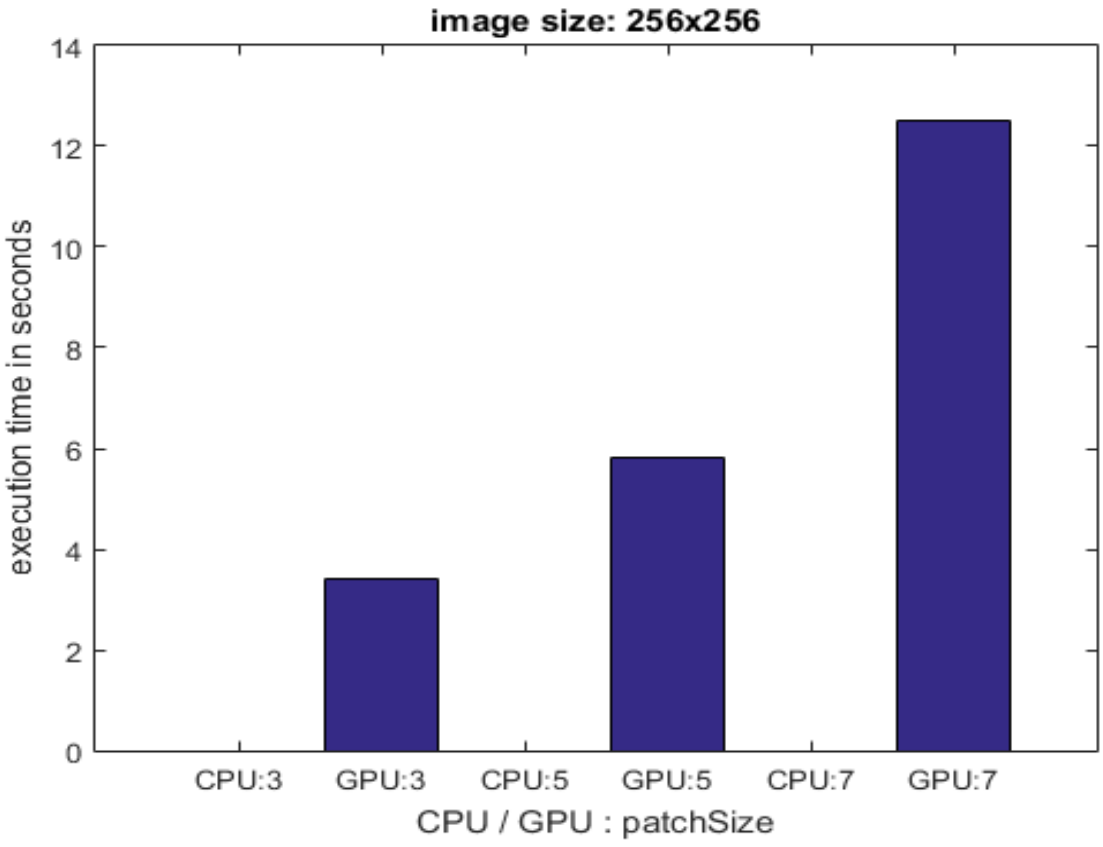


image size: 256x256
execution time in seconds
0
2
4
6
8
10
12
14
CPU:3
GPU:3
CPU:5
GPU:5
CPU:7
GPU:7
CPU / GPU : patchSize

Παρατηρούμε πολύ μεγάλη βελτίωση στους χρόνους εκτέλεσης πράγμα αναμενόμενο, καθώς η υπολογιστικής ισχύς μια σύγχρονης κάρτας γραφικών ξεπερνά κατά πολύ αυτή του επεξεργαστή.

Θα μπορούσαμε να έχουμε ακόμα πιο μεγάλη βελτίωση με χρήση shared memory όπως απαιτούνταν στην εργασία, όμως δυστυχώς δεν υλοποιήθηκε λόγο έλλειψης χρόνου.

Τέλος, εντός του φακέλου images, περιλαμβάνονται εικόνες πριν και μετά την αποθορυβοποίηση τους. Έγιναν πειράματα με την εικόνα house που δόθηκε στην εργασία αλλά και με την εικόνα της Lena.