Ονοματεπώνυμο: Θεοδώρου Γεώργιος

ΑΕΜ: 0497

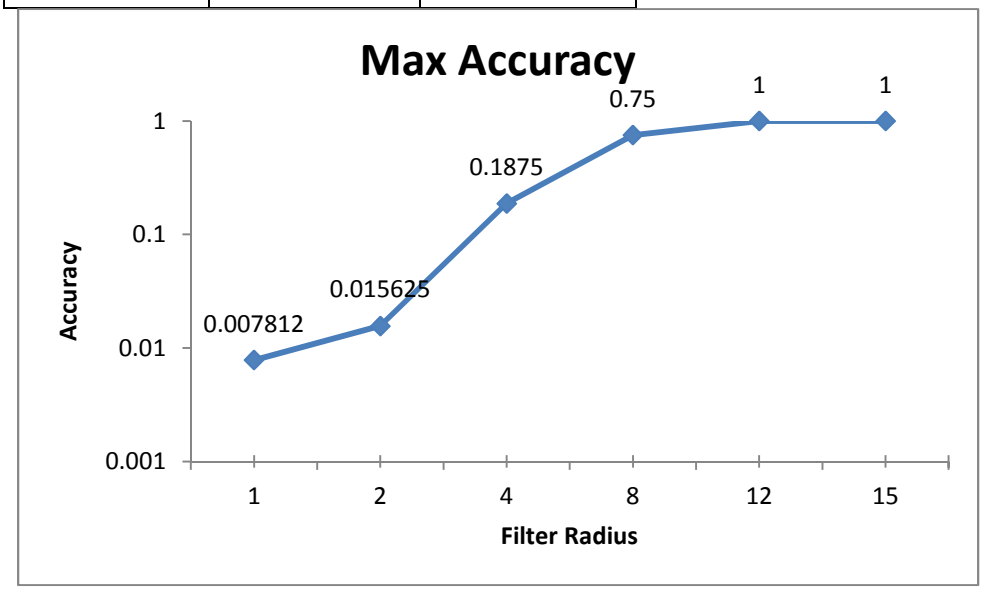
Απαντήσεις:

3$^{\alpha}$) Μπορούμε να υποστηρίξουμε μέχρι 32 x 32 εικόνες γιατί χρησιμοποιούμε μόνο ένα block και κάθε block έχει 1024 threads.

3$^{\beta}$) Ακολουθεί πίνακας και σχετικό διάγραμμα με τις μέγιστες αποκλίσεις σε σχέση με τα διάφορα μεγέθη φίλτρου και με σταθερή εικόνα 32 x 32. (Το μέγεθος της εικόνας δεν επιρεάζει τα αποτελέσματα των αποκλίσεων.)

Παρατηρούμε ότι όσο αυξάνεται η ακτίνα του φίλτρου τόσο αυξάνεται και η μέγιστη ακρίβεια που μπορούμε να πετύχουμε.(Από ένα σημείο και περα βέβαια σταθεροποιήται)

Filter Radius	Image Size	Max Accuracy
1	32 x 32	0.007812
2	33 x 32	0.015625
4	34 x 32	0.1875
8	35 x 32	0.75
12	36 x 32	1
15	37 x 32	1

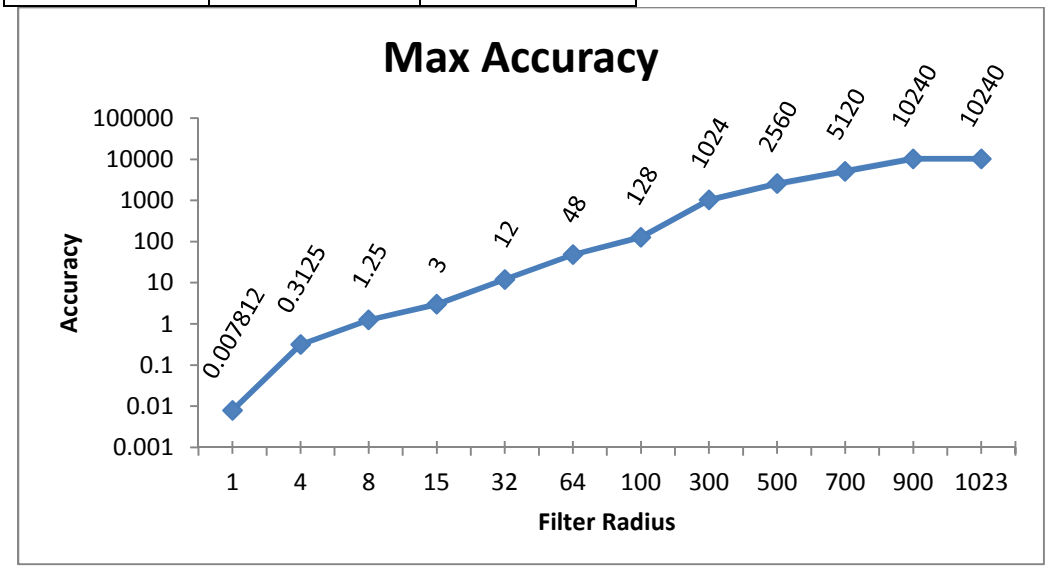


4) Πλέον δεν μας περιορίζει το 1 block (ερώτημα 3), μας περιορίζει όμως η μνήμη της GPU μας. Στον Mars με 2 gb μνήμη μπόρεσα να τρέξω μέχρι και 8192 x 8192 εικόνα.

5$^{\alpha}$) Αρχικά παρατηρούμε ότι όσο αυξάνεται η ακτίνα του φίλτρου, αυξάνεται και η μέγιστη ακριβεια. Αυτό συμβαίνει διότι αυξάνεται το μέγεθος του πίνακα του φίλτρου ,επομένως γίνονται πιο πολλές πράξεις για να υπολογιστεί το τελικό αποτέλεσμα, και στην διαδικασία χάνονται πολλά δεκαδικά ψηφία ,αφού οι float αριθμοί προκύπτουν από στρογγυλοποιήσεις.

Ακολουθεί πίνακας με τις μέγιστες αποκλίσεις σε σχέση με τα διάφορα μεγέθη φίλτρου και με σταθερή εικόνα 2048 x 2048. (Το μέγεθος της εικόνας δεν επιρεάζει τα αποτελέσματα των αποκλίσεων.)

Filter Radius	Image Size	Max Accuracy
1	2048 x 2048	0.007812
4	2048 x 2048	0.3125
8	2048 x 2048	1.25
15	2048 x 2048	3
32	2048 x 2048	12
64	2048 x 2048	48
100	2048 x 2048	128
300	2048 x 2048	1024
500	2048 x 2048	2560
700	2048 x 2048	5120
900	2048 x 2048	10240
1023	2048 x 2048	10240

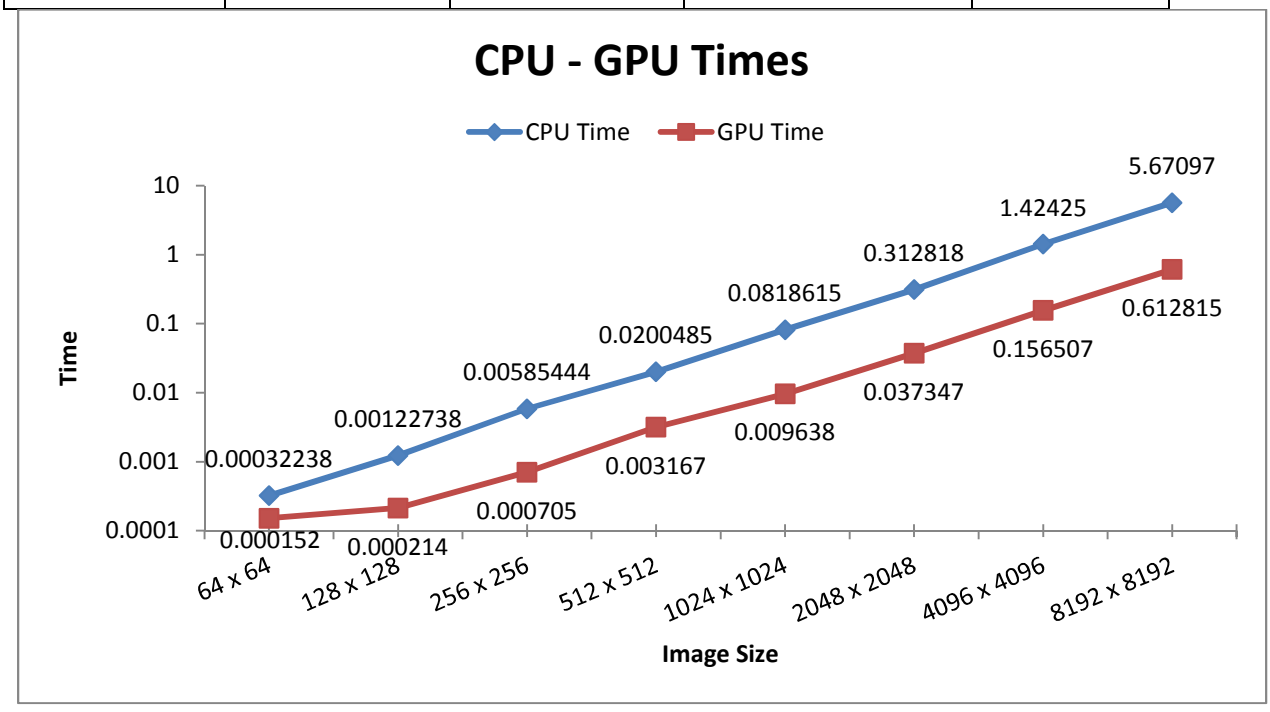


5^{β}) Παρατηρούμε ότι στην πλειοψηφεία των πειραμάτων ο υπολογισμός στην GPU είναι κατα 9 φορές ταχύτερος απο ότι στην CPU.

Ακολουθεί πίνακας και σχετικά διαγράμματα με τους χρόνους εκτέλεσης στην CPU και στην GPU, καθώς και η διαφορά τους. (Στα πειράματα είχαμε σταθερή ακτίνα φίλτρου ίση με 16.)

(* Όλα τα πειράματα με μετρήσεις χρόνων ,εκτελέστηκαν 12 φορές και βρέθηκε ο μέσος όρος τους.)

Filter Radius	Image Size	CPU Time	GPU Time	Difference
16	64 x 64	0.00032238	0.000152	0.000171
16	128 x 128	0.00122738	0.000214	0.001013
16	256 x 256	0.00585444	0.000705	0.005149
16	512 x 512	0.0200485	0.003167	0.016882
16	1024 x 1024	0.0818615	0.009638	0.072223
16	2048 x 2048	0.312818	0.037347	0.275471
16	4096 x 4096	1.42425	0.156507	1.267746
16	8192 x 8192	5.67097	0.612815	5.058156

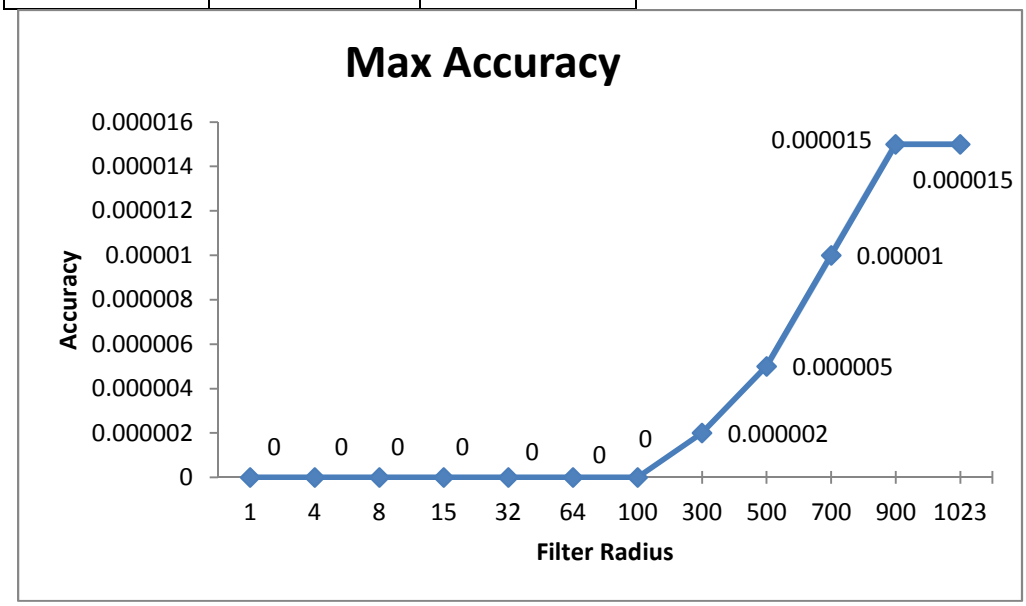


6) Στα πειράματα της άσκησης 6 αλλάξαμε όλους τους τύπους των στοιχείων από floats σε doubles. Δεν παρατηρήσαμε τα αναμενόμενα μηνύματα διότι στην 7.5 έκδοση της cuda αναγνορίζονται οι double.

6$^{\alpha}$) Στο συγκεκριμένο πείραμα εφόσον χρησιμοποιήσαμε doubles είδαμε σηματική βελτίωση στην μέγιστη ακρίβεια. Αυτό συναίβει διότι οι doubles έχουν πολύ καλύτερη ακρίβεια δεκαδικών από τους float , επομένως “χάνονται” πολύ λιγότερα δεκαδικά στις πράξεις που εκτελούνται κατά την εφαρμογή του φίλτρου.

Ακολουθεί πίνακας με τις μέγιστες αποκλίσεις σε σχέση με τα διάφορα μεγέθη φίλτρου και με σταθερή εικόνα 2048 x 2048. (Το μέγεθος της εικόνας δεν επιρεάζει τα αποτελέσματα των αποκλίσεων.)

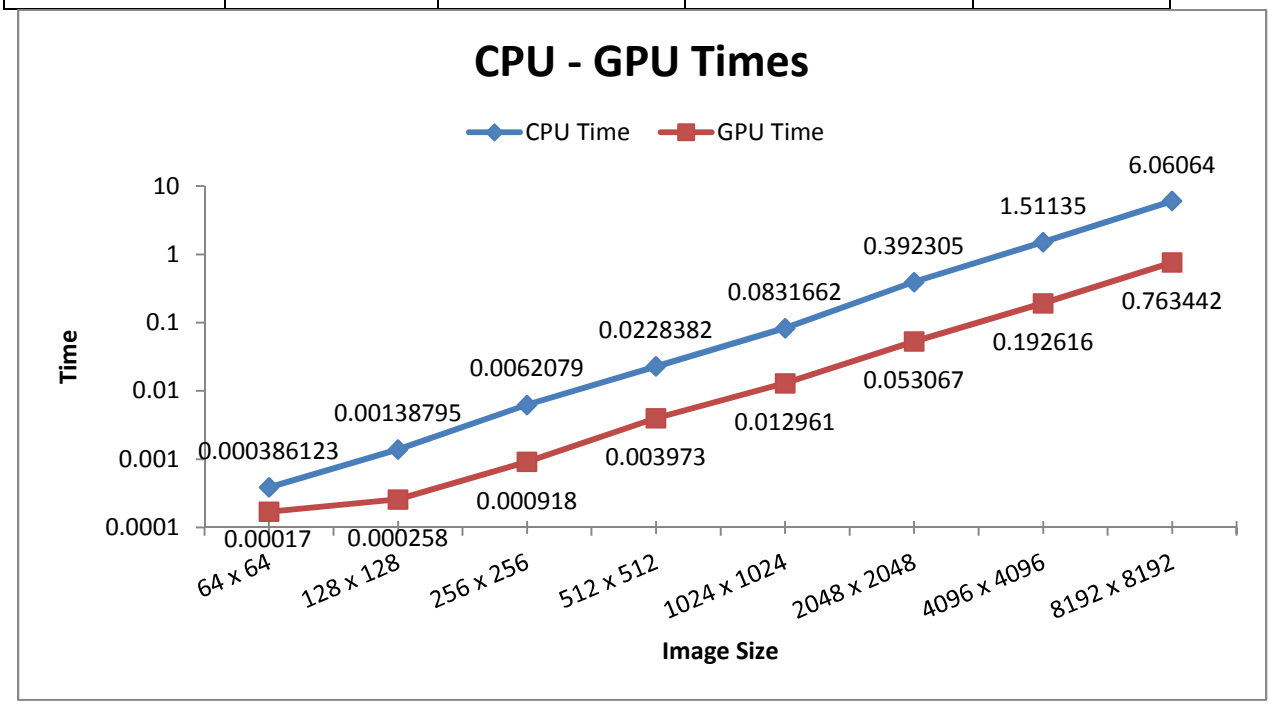
Filter Radius	Image Size	Max Accuracy
1	2058 x 2048	0
4	2048 x 2048	0
8	2048 x 2048	0
15	2048 x 2048	0
32	2048 x 2048	0
64	2048 x 2048	0
100	2048 x 2048	0
300	2048 x 2048	0.000002
500	2048 x 2048	0.000005
700	2048 x 2048	0.00001
900	2048 x 2048	0.000015
1023	2048 x 2048	0.000015



6$^{\beta}$) Φυσικά σε αυτό το πείραμα παρατηρήσαμε την αναμενόμενη αύξηση του χρόνου εκτέλεσης του προγράματος στην CPU και στην GPU, αφού χρησιμοποιούμε double αριθμούς, οι οποίοι είναι διπλάσιοι σε μέγεθος σε σχέση με τους float.

Ακολουθεί πίνακας και σχετικά διαγράμματα με τους χρόνους εκτέλεσης στην CPU και στην GPU, καθώς και η διαφορά τους. (Στα πειράματα είχαμε σταθερή ακτίνα φίλτρου ίση με 16.)

Filter Radius	Image Size	CPU Time	GPU Time	Difference
16	64 x 64	0.000386123	0.00017	0.000217
16	128 x 128	0.00138795	0.000258	0.00113
16	256 x 256	0.0062079	0.000918	0.00529
16	512 x 512	0.0228382	0.003973	0.018865
16	1024 x 1024	0.0831662	0.012961	0.070206
16	2048 x 2048	0.392305	0.053067	0.339238
16	4096 x 4096	1.51135	0.192616	1.318729
16	8192 x 8192	6.06064	0.763442	5.297203



7$^{\alpha}$) Για κάθε κλήση του kernel διαβάζουμε ασυμπτωτικά FILTER LENGTH φορές καθένα στοιχείο της εικόνας εισόδου. (FILTER LENGTH = 2 * filter_radius + 1). Λέμε ασυπτωτικά διότι ορισμένες φορές δεν διαβάζουμε στοιχείο της εικόνας, διότι μπορεί να βρισκόμαστε εκτός των ορίων της.

Ενώ για κάθε κλήση του kernel διαβάζουμε image_Width * image_Height φορές κάθε στοιχείο του φίλτρου, δηλαδή όσα και τα στοιχεία της εικόνας που έχουμε.

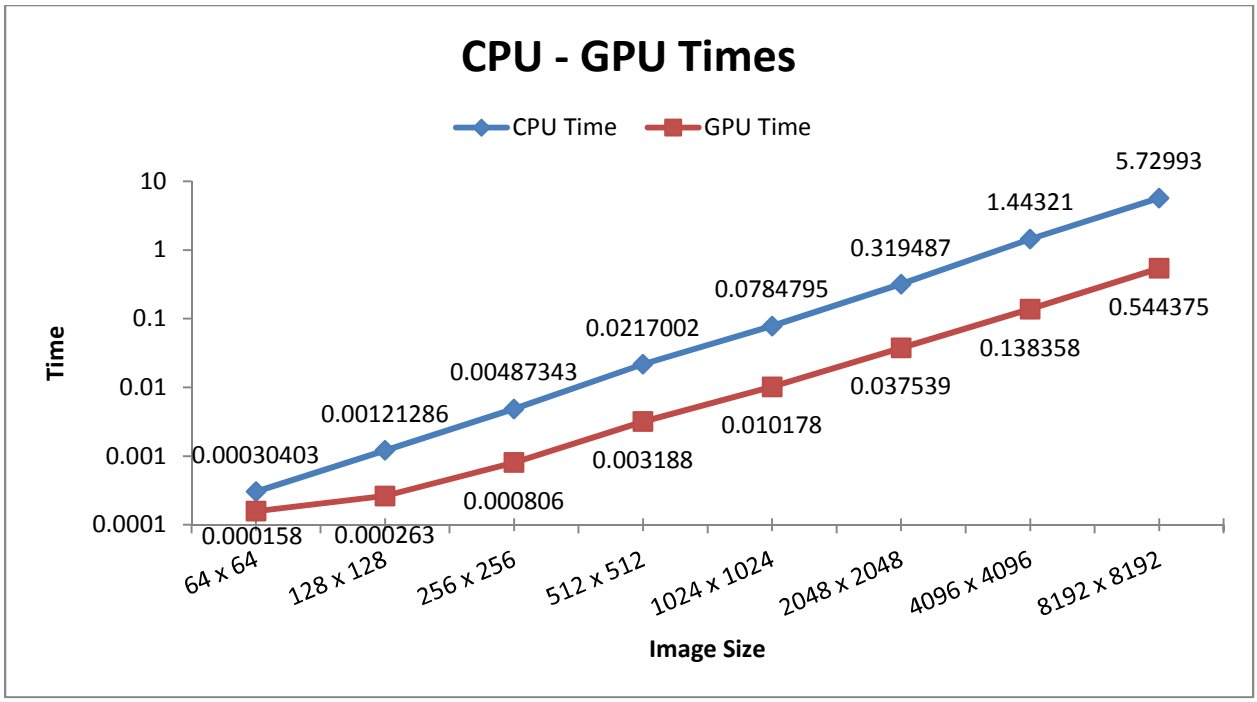
7$^{\beta}$) Ο λόγος προσπελάσεων μνήμης προς πράξεις κινητής υποδιαστολής είναι 1. Ουσιαστικά για κάθε υπολογισμό ενός στοιχείου του αθροίσματος γίνονται 2 αναγνώσεις (του στοιχείου της εικόνας και του στοιχείου του φίλτρου) από την κύρια μνήμη και 2 πράξεις (ένας πολλαπλασιασμός και μία πρόσθεση).

8) Στο συγκεκριμένο παράδειγμα λύσαμε το πρόβλημα του divergence προσθέτοντας ένα padding γυρω από την εικόνα μεγέθους όσο και η ακτίνα του φίλτρου.

Στους χρόνους εκτέλεσης του προγράμματος στην GPU παρατηρήσαμε μία μικρή βελτίωση της τάξεως περίπου του 10%. Αυτό συναίβει διότι πλέον δεν υπάρχει το πρόβλημα του divergence ανάμεσα στα warp.

Ακολουθεί πίνακας και σχετικά διαγράμματα με τους χρόνους εκτέλεσης στην CPU και στην GPU, καθώς και η διαφορά τους. (Στα πειράματα είχαμε σταθερή ακτίνα φίλτρου ίση με 16.)

Filter Radius	Image Size	CPU Time	GPU Time	Difference
16	64 x 64	0.00030403	0.000158	0.000146
16	128 x 128	0.00121286	0.000263	0.00095
16	256 x 256	0.00487343	0.000806	0.004068
16	512 x 512	0.0217002	0.003188	0.018512
16	1024 x 1024	0.0784795	0.010178	0.068301
16	2048 x 2048	0.319487	0.037539	0.281948
16	4096 x 4096	1.44321	0.138358	1.304853
16	8192 x 8192	5.72993	0.544375	5.185555


CPU - GPU Times
CPU Time
GPU Time
Time
10
1
0.1
0.01
0.001
0.0001
0.00030403
0.00121286
0.00487343
0.0217002
0.0784795
0.319487
1.44321
5.72993
0.000158
0.000263
0.000806
0.003188
0.010178
0.037539
0.138358
0.544375
64 x 64
128 x 128
256 x 256
512 x 512
1024 x 1024
2048 x 2048
4096 x 4096
8192 x 8192
Image Size