Προγραμματισμός Συστημάτων Υψηλής Επίδοσης

Lab3 Report

Χουλιαράς Ανδρέας ΑΕΜ:2143

Το σύστημα στο οποίο έγινε η εργασία έχει τα εξής χαρακτηριστικά:

CPU: Intel Core i7 6700 RAM: 16GB (2x8) DDR4-2133Mhz GPU: NVidia GTX 1070

OS: Ubuntu 16.4.5 LTS (σε Virtual Machine)

Kernel: 4.15.0-36-generic

Ερώτημα 0: το device Query εμφάνισε τα εξής :

```
 CUDA Device Query (Runtime API) version (CUDART static linking)

Detected 1 CUDA Capable device(s)

Device 0: "GeForce GTX 1070"
  CUDA Driver Version / Runtime Version          10.0 / 10.0
  CUDA Capability Major/Minor version number:    6.1
  Total amount of global memory:                 8118 MBytes (8512602112 bytes)
  (15) Multiprocessors, (128) CUDA Cores/MP:     1920 CUDA Cores
  GPU Max Clock rate:                            1835 MHz (1.84 GHz)
  Memory Clock rate:                             4004 Mhz
  Memory Bus Width:                              256-bit
  L2 Cache Size:                                 2097152 bytes
  Maximum Texture Dimension Size (x,y,z)         1D=(131072), 2D=(131072, 65536), 3D=(16384, 16384, 16384)
  Maximum Layered 1D Texture Size, (num) layers  1D=(32768), 2048 layers
  Maximum Layered 2D Texture Size, (num) layers  2D=(32768, 32768), 2048 layers
  Total amount of constant memory:               65536 bytes
  Total amount of shared memory per block:       49152 bytes
  Total number of registers available per block: 65536
  Warp size:                                     32
  Maximum number of threads per multiprocessor:  2048
  Maximum number of threads per block:           1024
  Max dimension size of a thread block (x,y,z): (1024, 1024, 64)
  Max dimension size of a grid size    (x,y,z): (2147483647, 65535, 65535)
  Maximum memory pitch:                          2147483647 bytes
  Texture alignment:                             512 bytes
  Concurrent copy and kernel execution:          Yes with 2 copy engine(s)
  Run time limit on kernels:                     Yes
  Integrated GPU sharing Host Memory:            No
  Support host page-locked memory mapping:       Yes
  Alignment requirement for Surfaces:            Yes
  Device has ECC support:                        Disabled
  Device supports Unified Addressing (UVA):      Yes
  Device supports Compute Preemption:            Yes
  Supports Cooperative Kernel Launch:            Yes
  Supports MultiDevice Co-op Kernel Launch:      Yes
  Device PCI Domain ID / Bus ID / location ID:   0 / 1 / 0
  Compute Mode:
     < Default (multiple host threads can use ::cudaSetDevice() with device simultaneously) >

deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 10.0, CUDA Runtime Version = 10.0, NumDevs = 1
Result = PASS
```

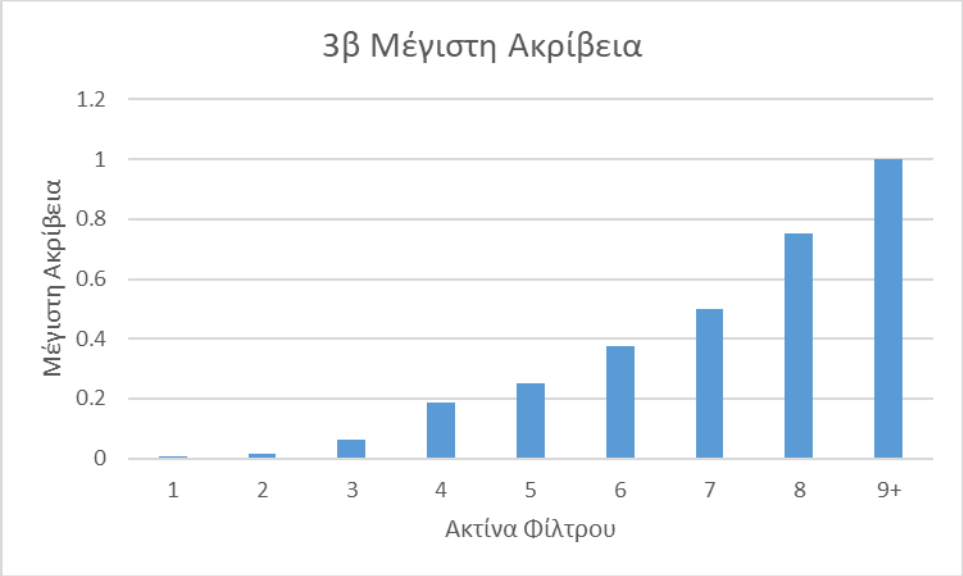
Ερώτημα 3:

α)Για φίλτρο ακτίνας 4, το μέγιστο μέγεθος εικόνας που υποστηρίζει ο κώδικας από το ερώτημα 2 είναι 1024 pixel ή διαστάσεις 32x32.Για μεγαλύτερα μεγέθη συμβαίνει σφάλμα κατά την εκτέλεση που είναι λογικό γιατί ένα block μπορεί να έχει μέχρι 1024 νήματα και εμείς έχουμε αναθέσει κάθε pixel της εικόνας σε ένα νήμα της GPU.

β) Για 32x32 pixel εικόνα, δοκίμασα φίλτρα ακτίνας 1-15 (δεν έχει νόημα να πάμε σε μεγαλύτερα φίλτρα γιατί θα ξεπεράσουμε το μέγεθος της εικόνας).Επίσης ξεκίνησα από ακρίβεια 0.00005 όπως ήταν και άρχισα να βάζω κι άλλα μηδενικά στο δεκάδικό μέρος ουσιαστικά αυξάνοντας την επιθυμητή ακρίβεια κατά περισσότερα δεκαδικά ψηφία. Όμως παρατήρησα ότι για κανένα μέγεθος φίλτρου αυτή η τιμή δεν έφτανε σε κάποιο όριο. Τα αποτελέσματα ανάμεσα σε GPU εικόνα και CPU εικόνα είναι ίδια πάντα.

Υποψιάζομαι ότι φτάιει που χρησημοποιούμε float αντι για doubles ,γιατί έχουν μικρή ακρίβεια και δεν μπορούν να υποστηρίξουν πάρα πολύ μικρές διαφορές σε τιμές. Τα αποτελέσματα στρογγυλοποιούνται λογικά στον κοντινότερο αριθμό που υποστηρίζετε από floats οπότε δεν μπορούμε να εκτιμήσουμε πόσο διαφέρουν τα αποτελέσματα μεταξύ τους.

Επίσης παρατήρησα το εξής: Τα παραπάνω ήταν με την παράμετρο -G στον nvcc. Αφαιρώντας τη, βλέπω ότι τα αποτελέσματα αρχίζουν να διαφέρουν για διαφορετικές τιμές ανάλογα με το μέγεθος του φίλτρου. Ακολουθεί το διάγραμμα.



Όσο για τον λόγο που συμβαίνει κάτι τέτοιο σε σχέση με την παράμετρο -G ,δεν έχω ιδέα...

Ερώτημα 4:

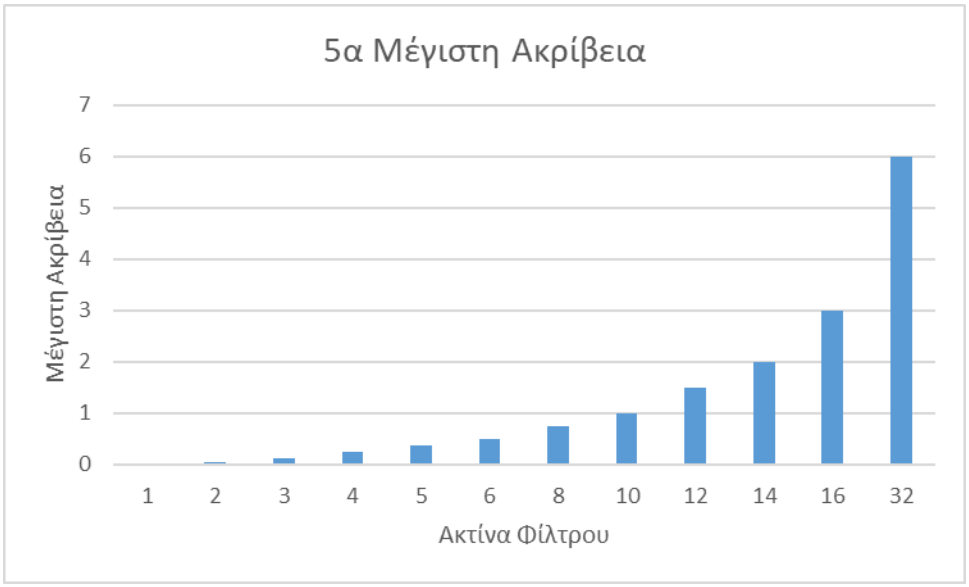
Από 32x32 φτάνουμε να υπόστηρίζουμε πλέον μέχρι 16384x16384 πρίν να αρχίσει να σκάει το πρόγραμμα λόγω ανεπαρκούς μνήμης.

Ερώτημα 5:

α) Όμοια με πριν, μόνο χωρίς την παράμετρο -G φαίνονται διαφορές στις εικόνες. Μάλιστα η συμπεριφορά φαίνετε να είναι παρόμοια με πριν. Όσο μεγαλώνει το φίλτρο αυξάνετε η μέγιστη ακρίβεια που οδηγεί σε επιτυχείς συγκρίσεις, και αυξάνετε πολύ. Η ακρίβεια από φίλτρο ακτινας 1 σε ακτίνα 32 αυξάνεται κατά 6 περίπου μονάδες! Και ξεκινάει περίπου από το 0.0008 που σημαίνει ότι 6 μοναδες είναι τεράστια μεταβολή!!!

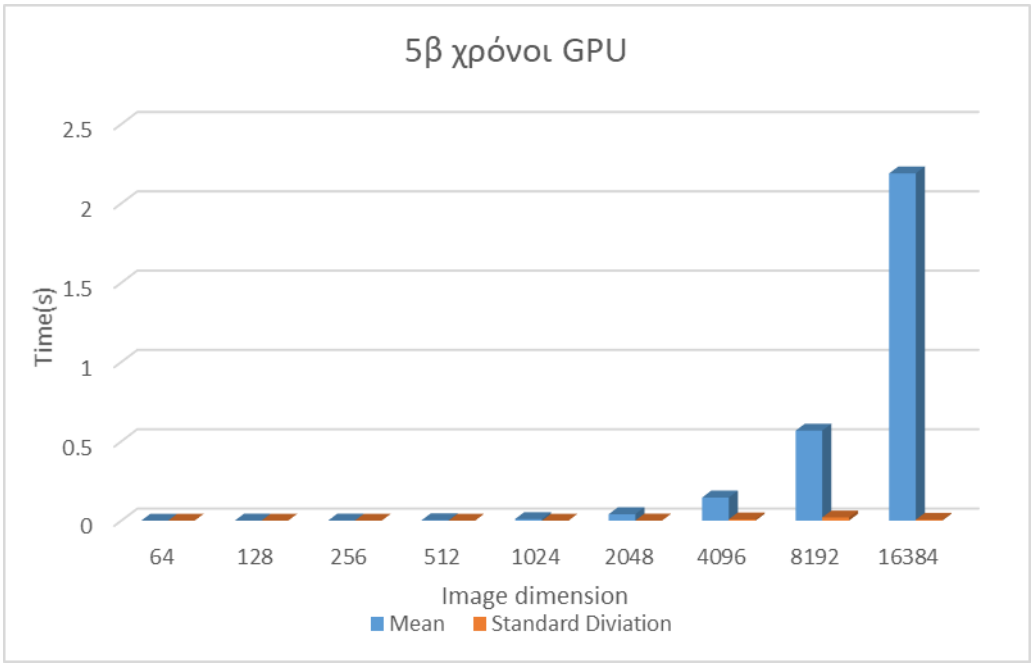
Η μόνη αιτία για αυτό είναι όπως ανέφερα πριν, η χρήση τύπου δεδομένων float για τα pixel της εικόνας.

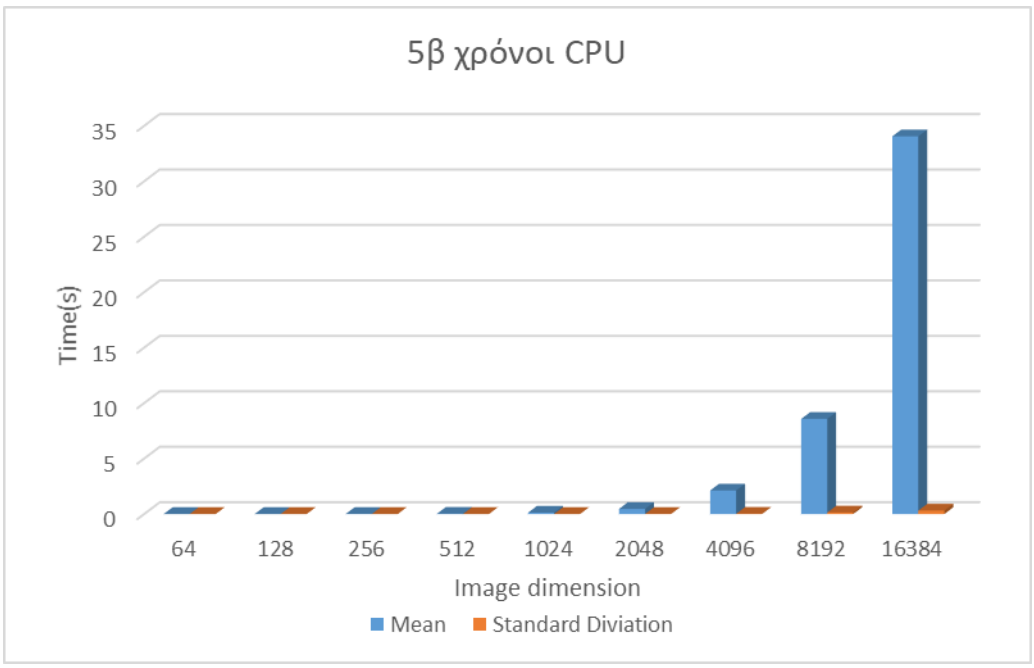
Ακολουθεί το διάγραμμα

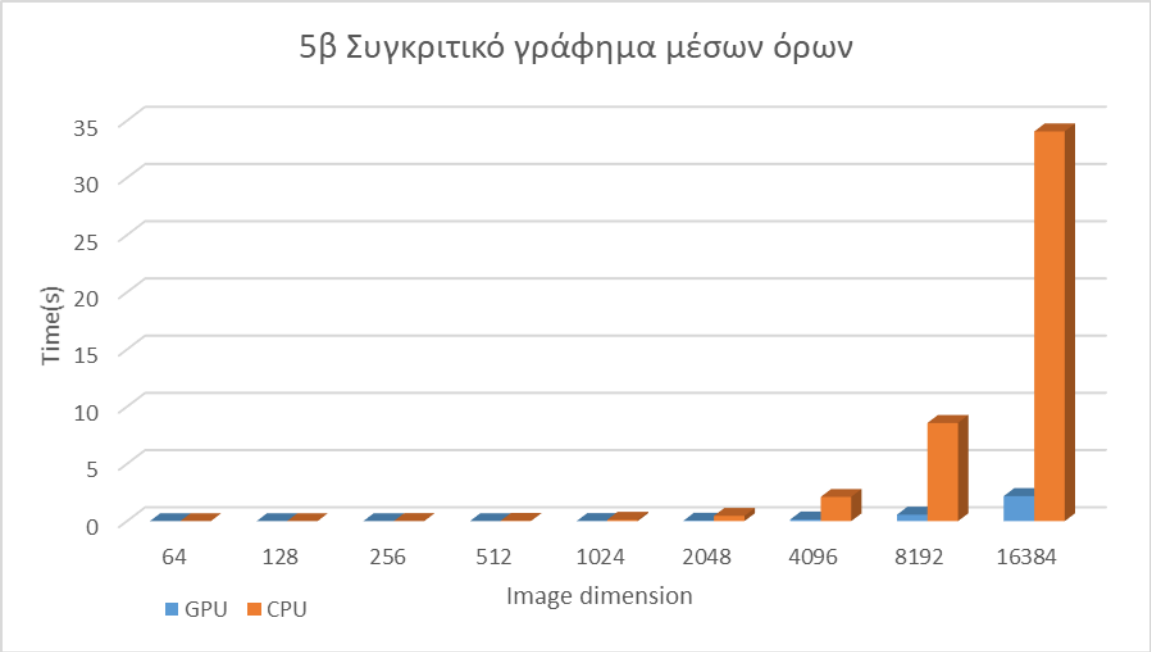


β)

Ακολουθούν τα σχετικά διαγράμματα χρόνων :





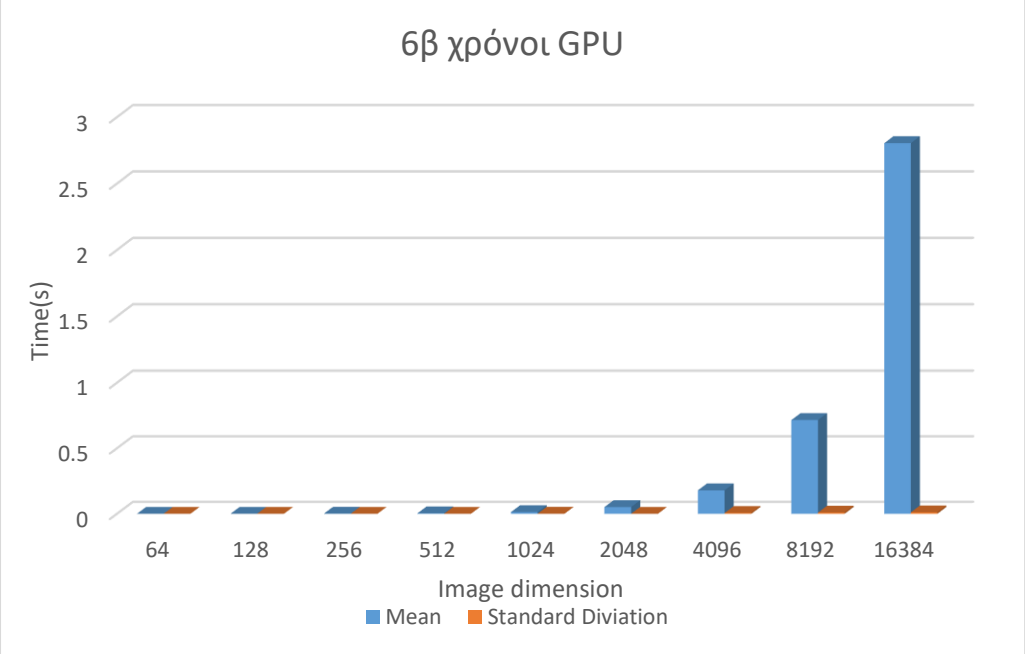


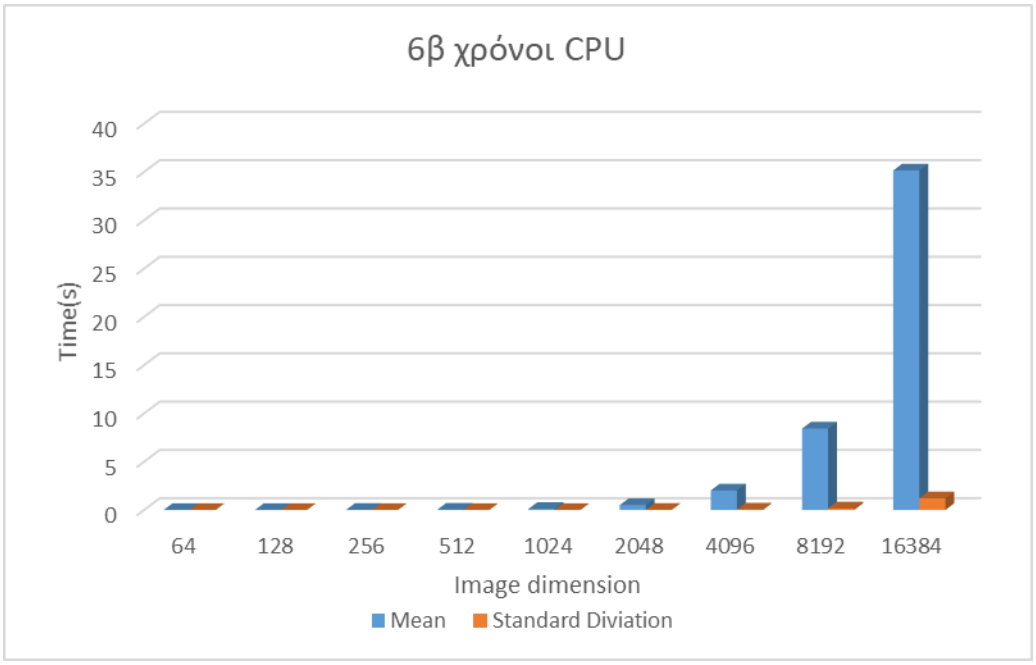
Ερώτημα 6:

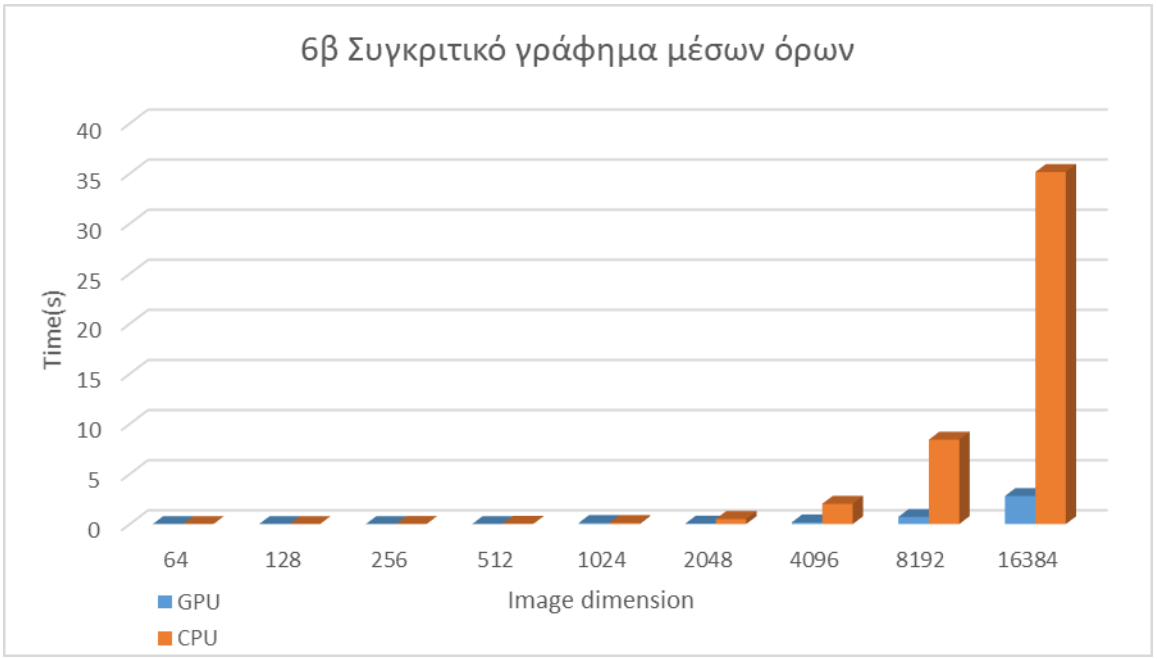
Κατ' αρχάς, η μέγιστη ακρίβεια που οδηγεί σε επιτυχείς συγκρίσεις μπορεί να πέσει στο 0.000001 και παραμένει ίδια ανεξάρτητα από το μέγεθος του φίλτρου,αντιθέτως με πρίν. Επίσης οι χρόνοι εκτέλεσης στην κάρτα γραφικών έχουν πέσει λίγο,όπως φαινετε και στα παρακάτω γραφήματα. Άρα βλέπουμε ότι υπάρχει ένα trade-off ανάμεσα σε ακρίβεια και απόδοση. Οι doubles προσφέρουν μεγαλύτερη ακρίβεια σε κόστος απόδοσης και οι floats το αντίστροφο.

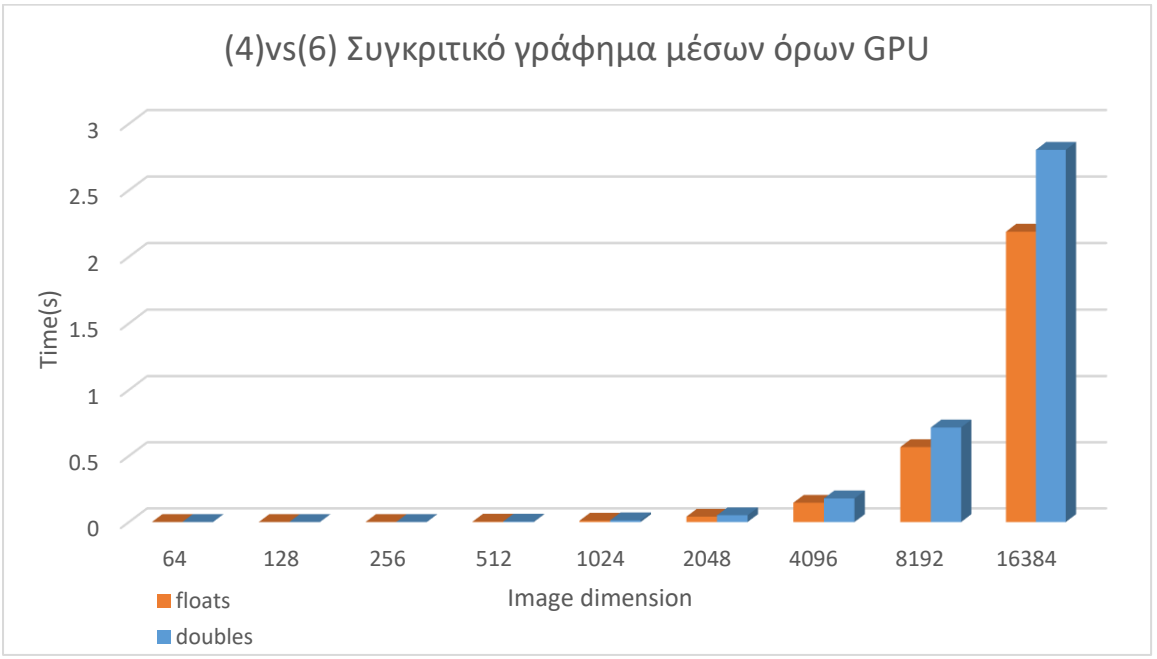
Παρά το γεγονός ότι το trade-off ισχύει και για GPU και για CPU, είναι πολύ πιο έντονο για την κάρτα γραφικών.

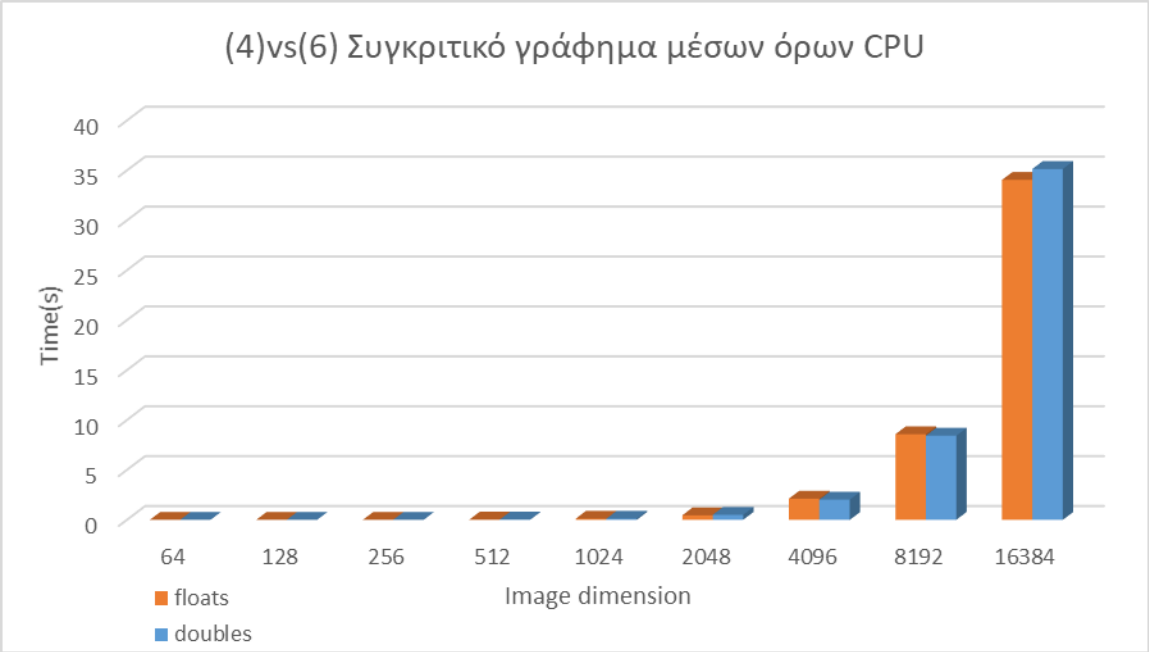
Ακολουθούν τα σχετικά διαγράμματα χρόνων :




6β χρόνοι CPU
40
35
30
25
20
15
10
5
0
Time(s)
64
128
256
512
1024
2048
4096
8192
16384
Image dimension
Mean
Standard Diviation


6β Συγκριτικό γράφημα μέσων όρων
40
35
30
25
20
15
10
5
0
Time(s)
64
128
256
512
1024
2048
4096
8192
16384
Image dimension
GPU
CPU


(4)vs(6) Συγκριτικό γράφημα μέσων όρων GPU
3
2.5
2
1.5
1
0.5
0
Time(s)
64
128
256
512
1024
2048
4096
8192
16384
Image dimension
floats
doubles



Ερώτημα 7:

α) Κάθε στοιχείο της εικόνας διαβάζετε 2(filter_length) φορές. Κάθε στοιχείο του φίλτρου διαβάζετε $2N^2$ φορές. Όπου N το ύψος-πλάτος της εικόνας και filter_length το μήκος του φίλτρου.

β) Για τον υπόλογισμο κάθε pixel της τελικής εικόνας γίνονται $filter\_length$ προσπελάσεις του φίλτρου, $filter\_length$ προσπελάσεις στοιχείων που πέφτουν κάτω από το φίλτρο, επί 2 γιατί γίνετε μια φορά κατά γραμμές και μια κατά στήλες.Άρα $4 * filter\_length$ προσπελάσεις ανά στοιχείο.

Για τον υπόλογισμο κάθε pixel της τελικής εικόνας γίνονται 2 εσωτερικά γινόμενα,ένα κατά γραμμές, ένα κατά στήλες. Κάθε εσωτερικό γινόμενο έχει 1 πρόσθεση και έναν πολλαπλασιασμό για κάθε στοιχείο του φίλτρου. Άρα $4 * filter\_length$ πράξεις ανά στοιχείο.

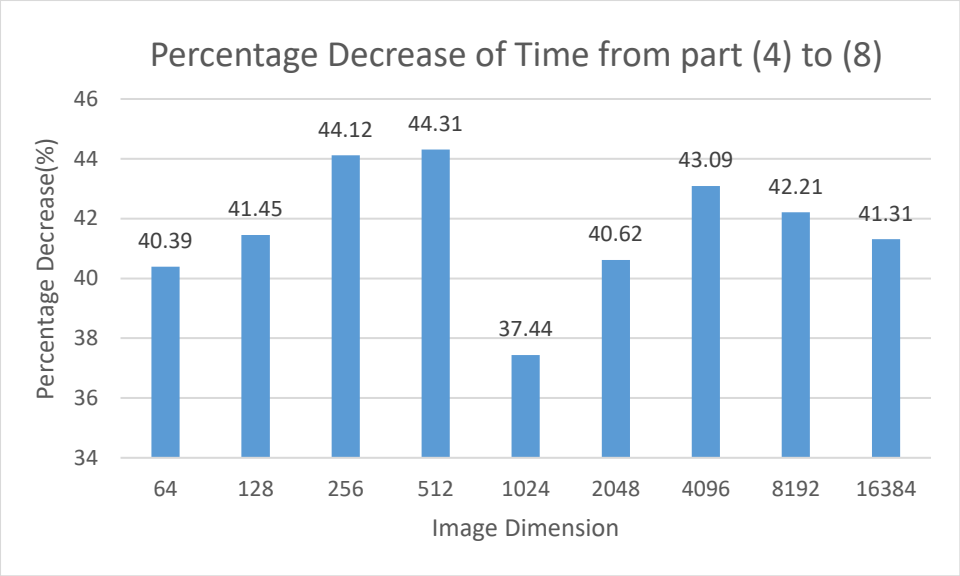
Λόγος προσπελάσεων μνήμης προς πράξεις κινητής υποδιαστολής για NxN στοιχεία:

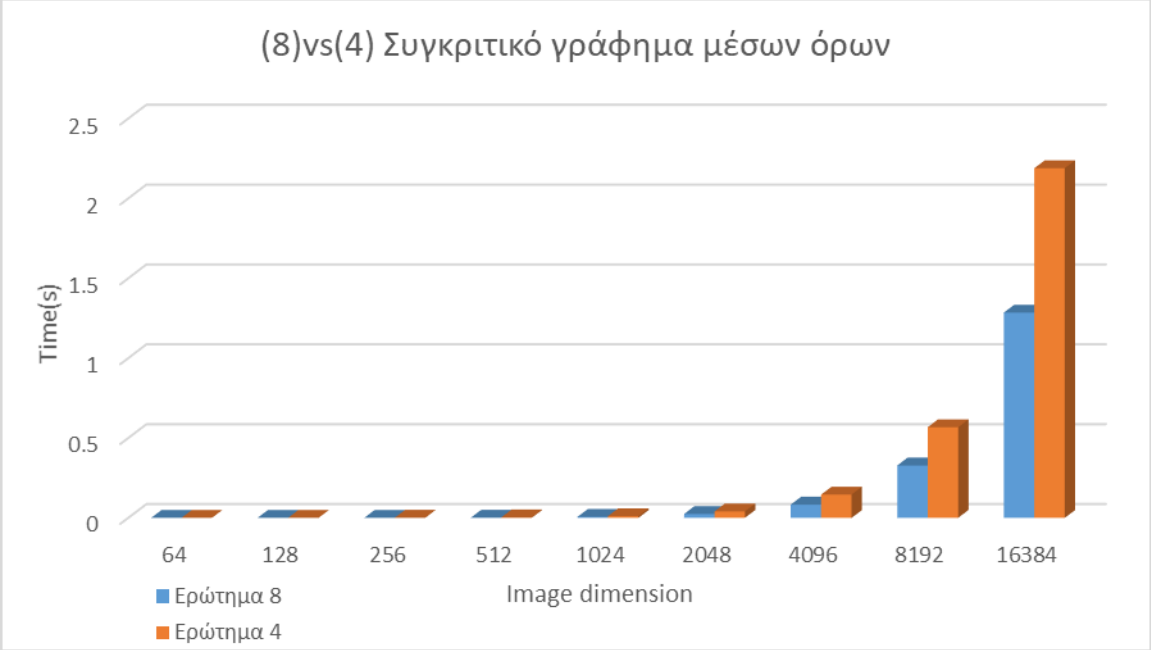
$$\lambda = \frac{4 * filter\_length * N^2}{4 * filter\_length * N^2} = 1$$

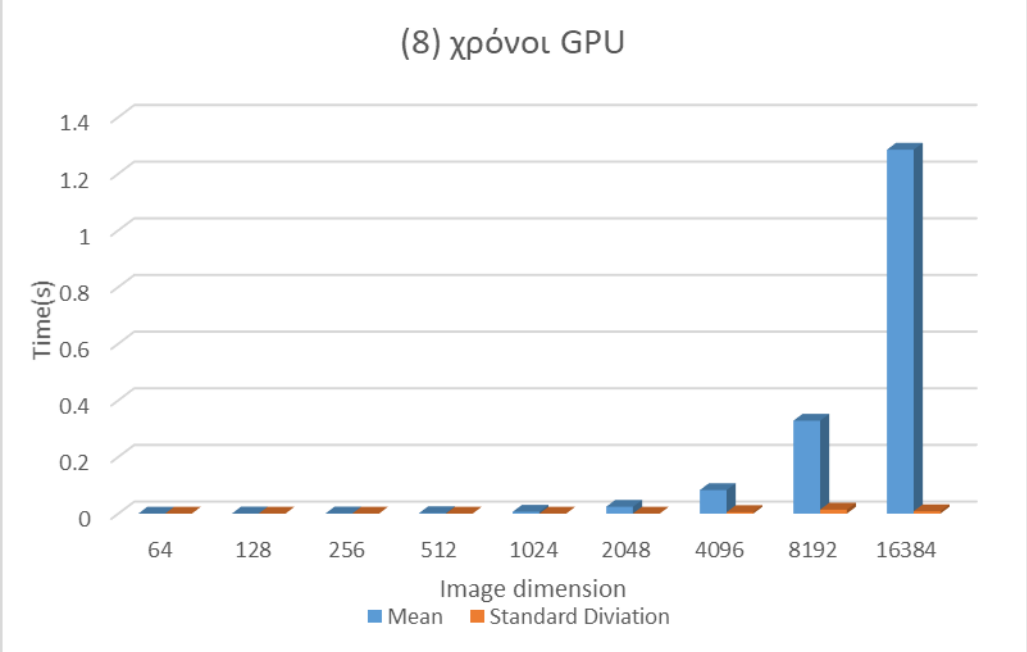
Αυτό σημαίνει ότι για κάθε πράξη χρειάζετε να γίνει και μία προσπέλαση στην μνήμη. Αυτό κανεί το πρόγραμμα να θεωρείτε memory bound που σημαίνει ότι δεν μπορεί να αξιοποιήσει την GPU στο έπακρο.

Ερώτημα 8:

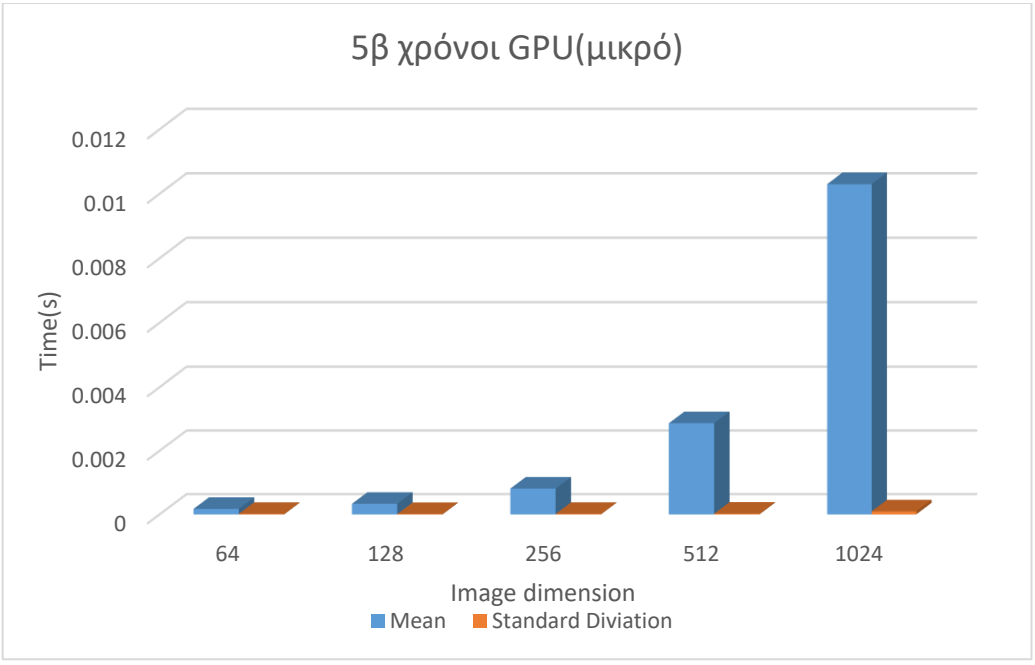
Εφαρμόζοντας την τεχνική padding για να λύσουμε το το πρόβλημα του divergence εντός των warps αύξησε την επίδοση του κώδικα κατά μέσο ότο 41.66% με τυπική απόκλιση 2.0025%.

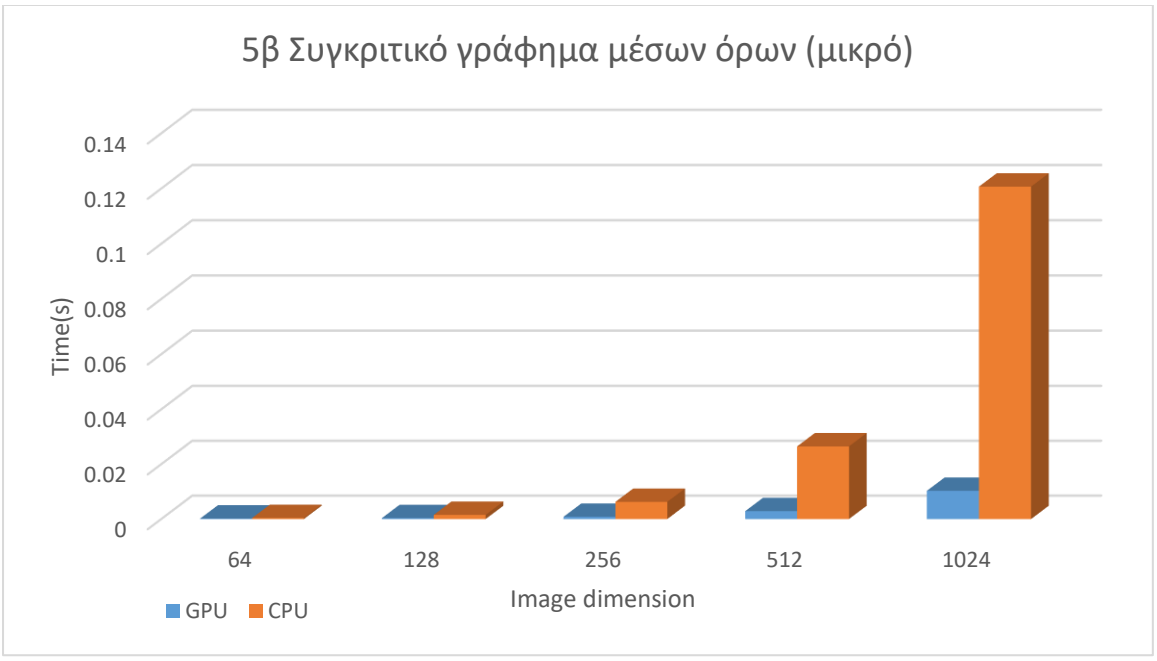

Percentage Decrease of Time from part (4) to (8)
Percentage Decrease(%)
46
44
42
40
38
36
34
40.39
41.45
44.12
44.31
37.44
40.62
43.09
42.21
41.31
64
128
256
512
1024
2048
4096
8192
16384
Image Dimension

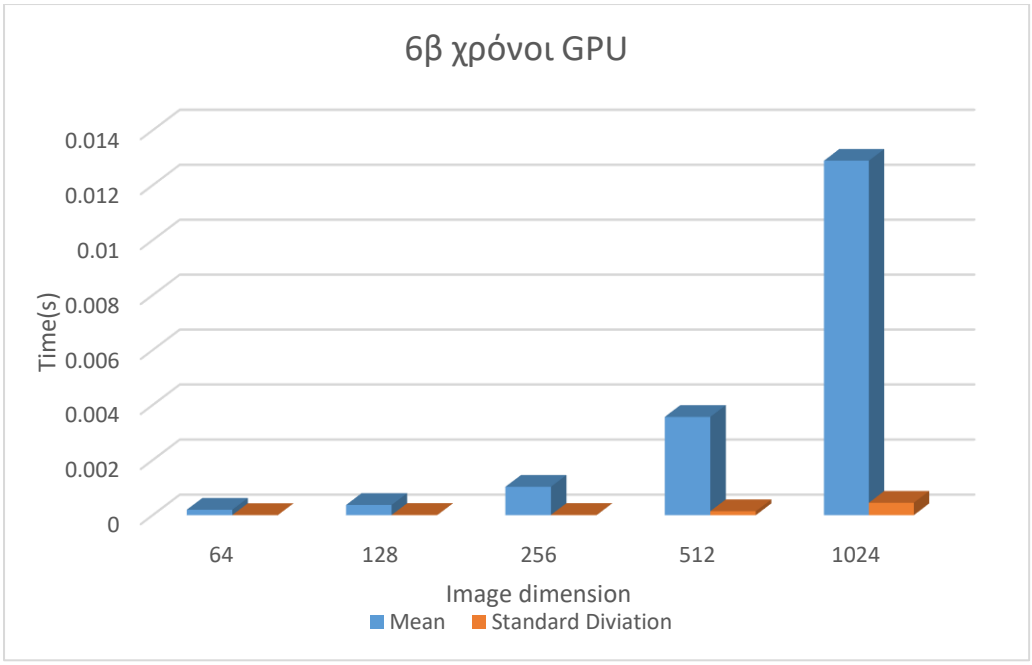

(8)vs(4) Συγκριτικό γράφημα μέσων όρων
Time(s)
2.5
2
1.5
1
0.5
0
64
128
256
512
1024
2048
4096
8192
16384
Image dimension
Ερώτημα 8
Ερώτημα 4

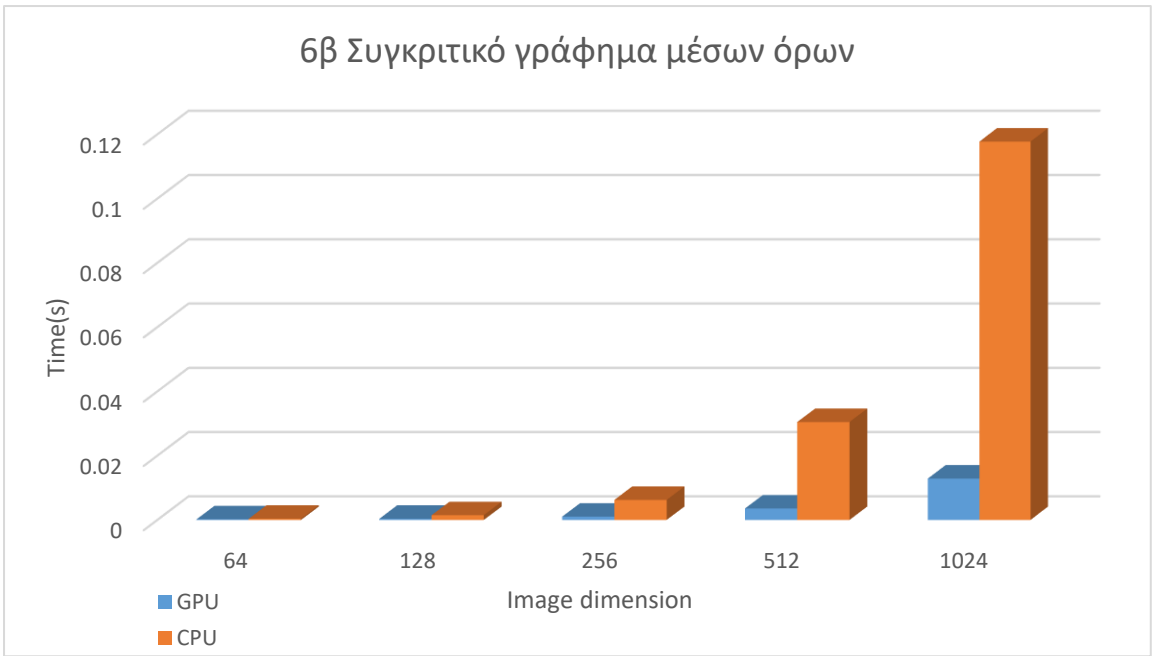

(8) χρόνοι GPU
Time(s)
1.4
1.2
1
0.8
0.6
0.4
0.2
0
64
128
256
512
1024
2048
4096
8192
16384
Image dimension
Mean
Standard Diviation

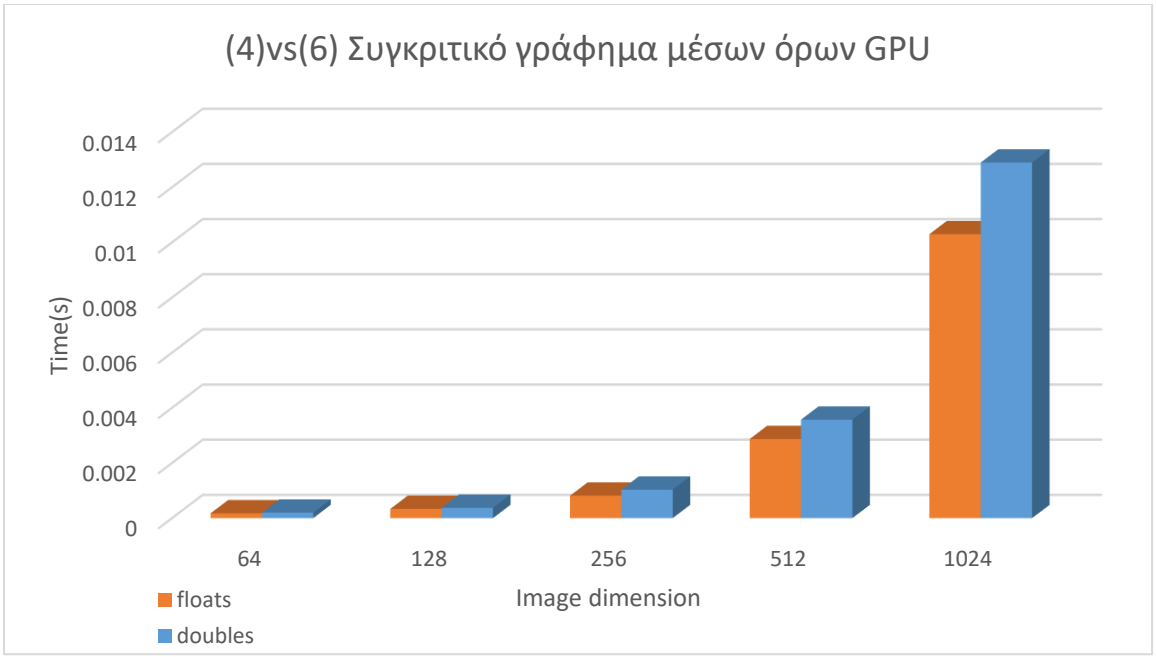
Ακολουθούν μερικά ακόμα γραφίματα χωρίς κάποιες από τις μεγάλες διαστάσεις εικόνας, για να γίνουν λίγο πιο ευδιάκριτες οι τιμές στα μικρότερα μεγέθη.Για τις αναλυτικές τιμές των πειραμάτων και διαγράμματα με δυνατότητα φιλτραρίσματος ανατρέξτε στο αρχείο excel Lab3_Statistics:






6β χρόνοι GPU
0.014
0.012
0.01
0.008
0.006
0.004
0.002
0
Time(s)
64
128
256
512
1024
Image dimension
Mean
Standard Diviation


6β Συγκριτικό γράφημα μέσων όρων
0.12
0.1
0.08
0.06
0.04
0.02
0
Time(s)
64
128
256
512
1024
Image dimension
GPU
CPU


(4)vs(6) Συγκριτικό γράφημα μέσων όρων GPU
0.014
0.012
0.01
0.008
0.006
0.004
0.002
0
Time(s)
64
128
256
512
1024
Image dimension
floats
doubles