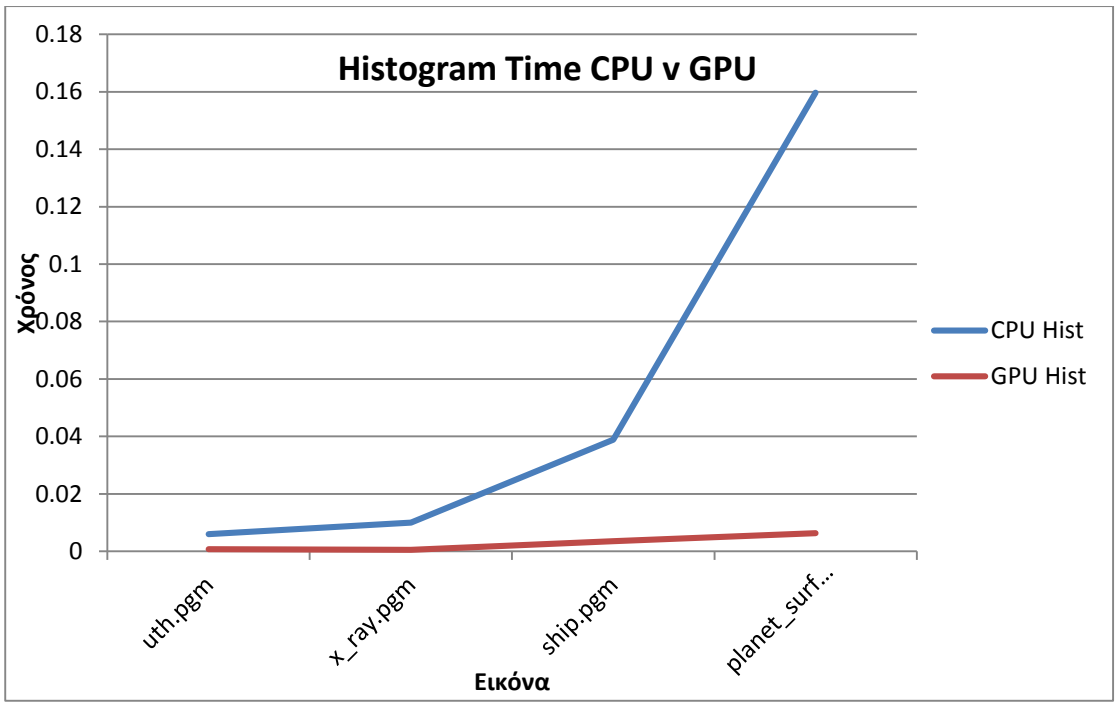
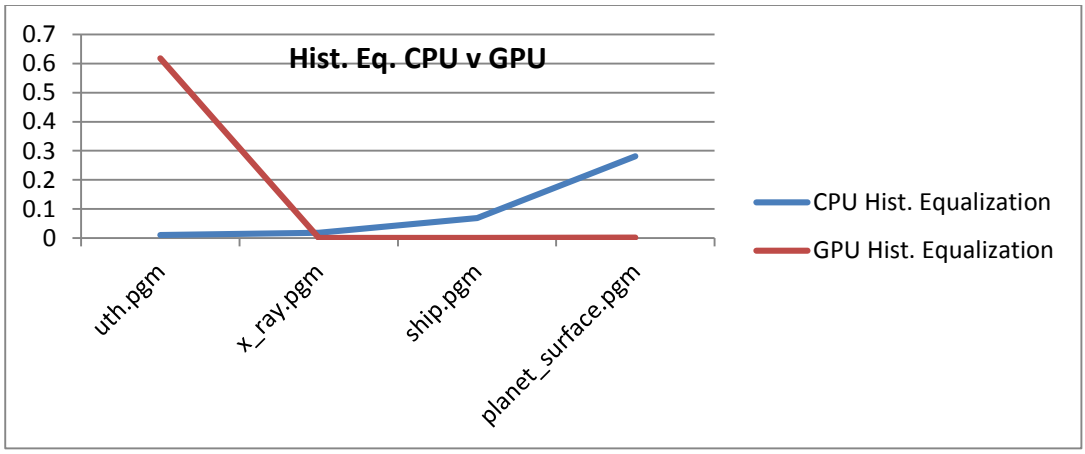
# Συστήματα Υπολογισμού Υψηλών Επιδόσεων
# Καλημέρης Μάρκος 1659
# Lab 4

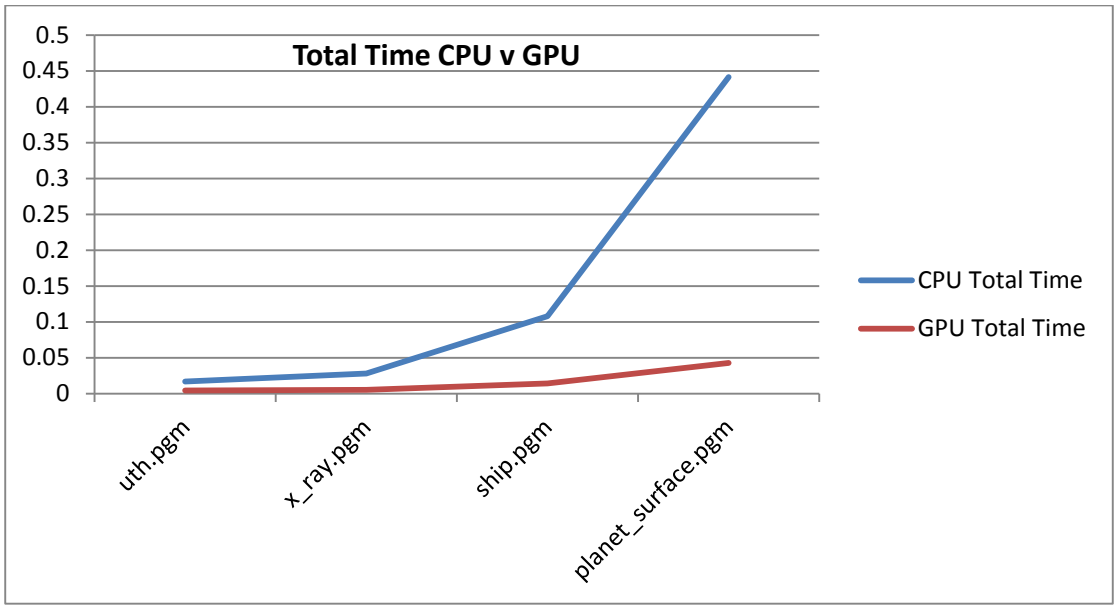
Από το παρακάτω διάγραμμα που δείχνει τη δημιουργία του ιστογράμματος η επίδοση της GPU σε σχέση με την CPU είναι εμφανής ιδιαίτερα σε εικόνες μεγαλύτερου μεγέθους.




Όπως και προηγουμένως στο υπολογισμό της εξίσωσης του ιστογράμματος η επίδοση της GPU είναι εμφανής απέναντι στη CPU ιδιαίτερα για μεγάλες εικόνες. Μόνο στην 1η εικόνα που χρησιμοποιήθηκε υπάρχει αισθητή διαφορά της CPU με την GPU λόγω του θορύβου που δημιουργεί η GPU.



Τέλος όσον αφορά τον συνολικό χρόνο η GPU έχει καλύτερη επίδοση από την CPU σε όλα τα μεγέθη των εικόνων συνεπώς η χρήση της GPU είναι εμφανώς λογικότερη, καθώς μπορεί στα μικρότερα μεγέθη η διαφορά να μην είναι ιδιαίτερα μεγάλη καθώς όμως αυξάνεται το μέγεθος της εικόνας αυξάνεται και η διαφορά του χρόνου.




## Υλοποίηση παραλληλοποίησης

Για την υλοποίηση της τελικής λύσης προστέθηκαν 2 kernel και μία συνάρτηση υπολογισμού του lookup table.

Kernel για τον υπολογισμό του Histogram: Ολόκληρη η συνάρτηση υπολογισμού του histogram αντικαταστάθηκε από μία cudaMemset με 0 και έναν kernel όπου κάθε block του έχει 256 threads και ισχύει image_size <= blocks * threads και κάθε thread υπολογίζει τις τιμές του τοπικού histogram με τη χρήση της atomicAdd για να μην υπάρχουν conficts μέσα στο warp και το συγχωνεύει στον τελικό kernel.

Kernel για τον υπολογισμό του Histogram Equalization: Η συνάρτηση υπολογισμού του histogram equalization αντικαταστάθηκε από έναν kernel όπου κάθε block του έχει 256 threads και ισχύει πάλι image_size <= blocks * threads. Κάθε thread υπολογίζει το αντίστοιχο pixel με τον lookup table να φορτώνεται στη shared memory για απευθείας χρήση του.

Υπολογισμός Lookup Table: Η όλη διαδικασία απλώς υλοποιήθηκε σε μία συνάρτηση.

Προβλήματα: Έγινε χρήση της constant memory όπου προστέθηκε ο lut όπως η λύση εγκαταλείφτηκε διότι δε μπορούμε να προβλέψουμε την ταυτόχρονη χρήση της ίδιας θέσης για να εκμεταλλευτούμε την constant memory.

Έτσι η τελική λύση υλοποιήθηκε στα παρακάτω στάδια:

1. Δημιουργία των 2 kernel και της συνάρτησης υπολογισμού του lut όπως περιγράφηκε πιο πάνω. Εδώ παρόλο που παίρνουμε καλύτερο χρόνο κάθε thread έχει πολύ λίγη δουλειά να κάνει.
2. Προστέθηκε ο υπολογισμός σε tiles ώστε κάθε thread να έχει περισσότερη δουλειά να κάνει.
3. Χρησιμοποιήθηκε η constant memory αλλά προέκυψε το πρόβλημα που αναφέρθηκε πιο πάνω και έτσι εγκαταλήφθηκε.
4. Προστέθηκε ο lut στην shared memory και έτσι γίνονται περισσότερα read.
5. Χρήση της shared memory στο kernel ιστογράμματος ώστε κάθε block να υπολογίζει ένα τοπικό ιστόγραμμα και να ενώνεται στο τέλος με την global. Επίσης κάθε kernel έχει 256 threads για να μην κάθετε κανένα.