Report

Ονοματεπώνυμο: Κομνάς Καφάσης

ΑΜ: 3190081

Το πρόγραμμα εκτελέστηκε σε Desktop με επεξεργαστή Ryzen 7 3700x (8 cores, 16 threads) στο Release Configuration μέσω Visual Studio.

Αποτελέσματα Gausian Blur Seperate Serial

	Χρόνος
1ο run	1954 ms
2ο run	1930 ms
3ο run	1902 ms
4ο run	1951 ms
Μέσος Όρος	1934 ms

Gausian Blur Seperate Parallel

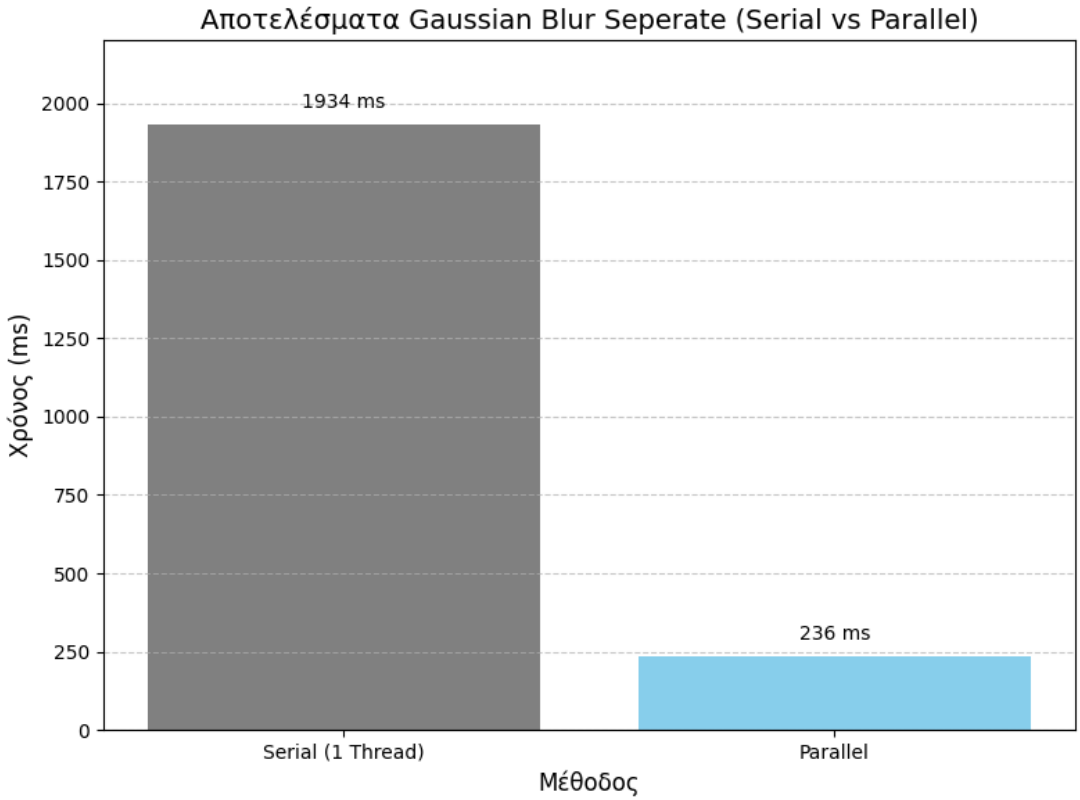
Η Gausian Blur Seperate Parallel έχει σχεδιαστεί έτσι ώστε για τα horizontal blur και vertical blur να δημιουργούνται νήματα όπου κάθε ένα να κάνει θόλωση κατά τον αντίστοιχο άξονα σε όλα τα pixel σε μία σειρά. Αφού κάποιο νήμα τελειώσει την δουλειά που του αντιστοιχεί αναλαμβάνει την θόλωση για την επόμενη σειρά δυναμικά. Αυτό επιτυγχάνεται με την παρακάτω εντολή:

```
#pragma omp parallel for schedule(dynamic, 1) private(x, pixel, channel)
```

Επίσης δηλώνονται οι μεταβλητές x, pixel, channel ως ιδιωτικές σε κάθε νήμα.

Αποτελέσματα Gausian Blur Separate Parallel

	Χρόνος
1ο run	236 ms
2ο run	232 ms
3ο run	240 ms
4ο run	237 ms
Μέσος Όρος	236 ms



Παρατηρούμε ότι η μέθοδος gaussian_blur_separate_parallel είναι σημαντικά πιο γρήγορη από την μέθοδο gaussian_blur_separate_serial αφού τρέχει παράλληλα σε πολλά νήματα, ανάλογα τον μέγιστο αριθμό υποστηριζόμενων νημάτων κάθε CPU, δηλαδή 16 στην περίπτωσή μας . Επίσης η gaussian_blur_separate_serial είναι πιο γρήγορη ακόμα και από την gaussian_blur_parallel που τρέχει σε 8 νήματα διότι η μέθοδος από μόνη της έχει πολύ μικρότερη πολυπλοκότητα.

Bloom Parallel

Στην συνάρτηση bloom_parallel φτιάχνουμε την εικόνα που μας δόθηκε προσθέτοντας το φίλτρο bloom το οποίο τρέχει παράλληλα. Αρχικά μέσα σε ένα παράλληλο block της OpenMP (#pragma omp parallel) εκτελούνται οι εργασίες που πρέπει να εκτελεστούν σύμφωνα με την εκφώνηση. Κάθε εργασία εκτελείται σε ένα for loop το οποίο εκτελείται παράλληλα με δυναμικό schedule και κάθε νήμα είναι υπεύθυνο για μια σειρά από pixels. Αν κάποιο νήμα τελειώσει με την εργασία που του ανατέθηκε, αναλαμβάνει δυναμικά την επόμενη σειρά από pixels. Ως προς τον υπολογισμό της μέγιστης τιμής φωτεινότητας πρέπει να σημειωθεί ότι χρησιμοποιούμε μια τοπική ιδιωτική μεταβλητή local_max_luminance για την αποθήκευσή της και μετά το τέλος του for loop εκχωρούμε την τιμή της στην κοινή μεταβλητή max_luminance μέσα σε ένα block `#pragma omp critical` , ώστε να αποδοθεί η σωστή τιμή συγκρίνοντας την μέγιστη τιμή που υπολόγισε κάθε νήμα. Επίσης πρέπει να σημειωθεί ότι μετά από κάθε for loop που παραλληλοποιείται υπονοείται barrier.

Αποτελέσματα Bloom Parallel

	Χρόνος
1ο run	239 ms
2ο run	240 ms
3ο run	249 ms
4ο run	245 ms
Μέσος Όρος	243 ms