Σχεδιασμός Ενσωματωμένων Συστημάτων Εξαμηνιαία Εργασία
Ομάδα 7

Καμζέλας Γεώργιος 57296

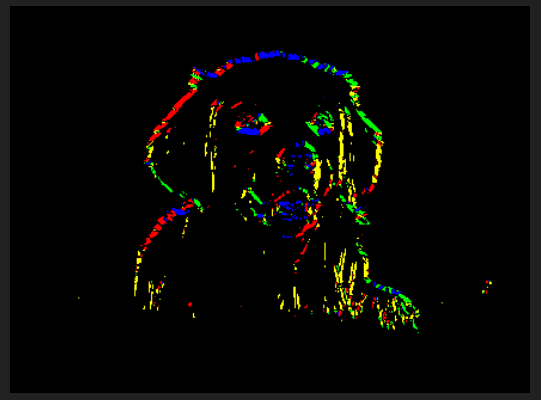
Σαμολαδάς Τριαντάφυλλος 57259

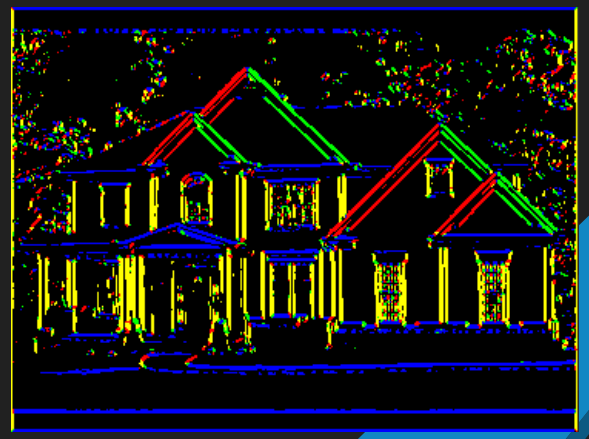
Αλγόριθμος – Edge & Outline Effects

1.1 Edges and Outline Effects

1. Convert the image to grayscale.
2. Apply the filter $\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$ to get the corresponding gaussian image. Don't worry about normalizing the image now; we'll do it later.
3. Compute the gradient image $\nabla I = \begin{bmatrix} I_x & I_y \end{bmatrix}'$ of this gaussian. This gradient image has two components, and these are $I_x = \dfrac{\partial I}{\partial x}$ and $I_y = \dfrac{\partial I}{\partial y}$; one way of computing these is via Sobel filters $\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$ and $\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$ respectively.
4. Compute the θ and $|\nabla I| = \text{magnitude}$ images. The θ image is computed as $\theta = \arctan\left(\dfrac{I_x}{I_y}\right)$ (can also use the reciprocal of our $\arctan$ argument), and $|\nabla I| = \sqrt{I_x^2 + I_y^2}$.
5. The last step is to determine the colour of each pixel, depending upon the angle θ and the intensity (0 – 255) of that pixel depending upon its $|\nabla I|$ value. For this, we need to compute the maximum and minimum magnitude values, and scale that to the range $0 - 255$. Further, colours needs to be assigned such that a vertical edge is yellow; horizontal is blue; and the other two inclined edges are red and green in colour.

Αποτελέσματα Αλγορίθμου





Μετάβαση στον Armulator

Μετά την υλοποίηση του αλγορίθμου, ο κώδικας δοκιμάστηκε στον Armulator και αρχική απόδοση ήτανε η εξής:

Reference...	Instructions	Core_Cycles	S_Cycles	N_Cycles	I_Cycles	C_Cycles	Total
$statistics	492955234	707153655	554357121	115122847	121981991	0	791461959

Για την παρούσα κατάσταση σημειώνεται πως γνωρίζοντας πως η γλώσσα C είναι row-major, όλες οι προσπελάσεις πινάκων γίνονται ανά γραμμή αξιοποιώντας έτσι την διάταξη αποθήκευσης των πινάκων στην μνήμη.

Τεχνικές Βελτιστοποίησης

- Αφαίρεση της συνάρτησης pow(x) → Αντικατάσταση με "x * x" -8.7 %

Reference...	Instructions	Core_Cycles	S_Cycles	N_Cycles	I_Cycles	C_Cycles	Total
$statistics	452103338	643475994	503759302	104636113	114028747	0	722424162

- Loop Merging (Υπολογισμός theta, magnitude, max, min) -0.3%

Reference...	Instructions	Core_Cycles	S_Cycles	N_Cycles	I_Cycles	C_Cycles	Total
$statistics	451083303	641329343	502740265	103799065	113560421	0	720099751

- Loop Unroll (max U = 4) όπου επιτρέπεται από τις αλληλοεξαρτήσεις -1.4%

Reference...	Instructions	Core_Cycles	S_Cycles	N_Cycles	I_Cycles	C_Cycles	Total
$statistics	444700262	628967617	496226654	98900288	112538022	0	707664964

Συνολική εξοικονόμηση : 83 εκατομμύρια κύκλοι

Σύνδεση με «πραγματική» ιεραρχία μνήμης

Μετάβαση σε μια RAM με αργές ταχύτητες ανάγνωσης/εγγραφής (250/50)

Reference...	Instructions	Core_Cycles	S_Cycles	N_Cycles	I_Cycles	C_Cycles	Wait_States	Total	True_Idle...
$statistics	443543412	624260764	494790183	95926950	112240978	0	501413002	1204371113	43218030

- Ιδανική Μνήμη → Πραγματική Μνήμη + 70.19 %

Οπότε οι επιπλέον βελτιώσεις θα επιτευχθούν αξιοποιώντας την επαναχρησιμοποίηση των δεδομένων σε συνδυασμό με μια καλύτερη ιεραρχία μνήμης

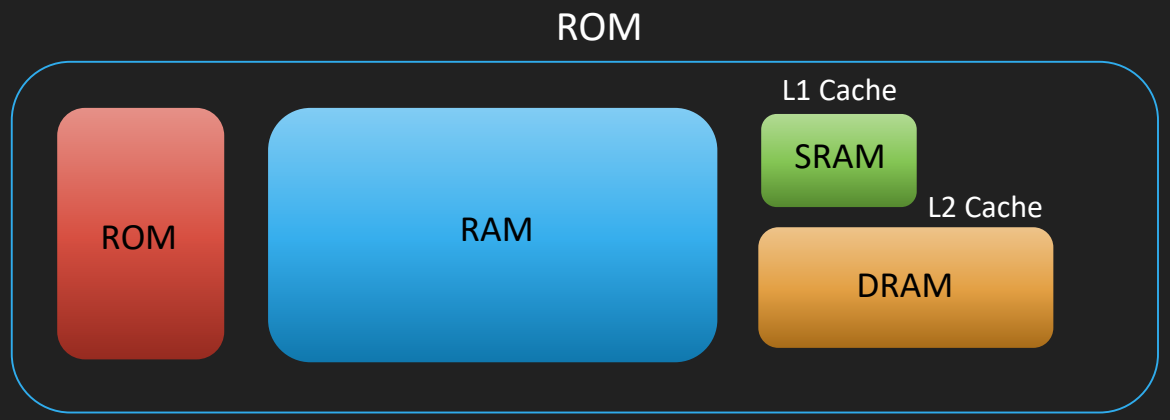
Επαναχρησιμοποίηση Δεδομένων

Στην convolution(), αποθηκεύεται ολόκληρη η γραμμή του πίνακα/εικόνας σε μια μεταβλητή buffer. Χρησιμοποιούνται τρεις buffer (εξαιτίας του 3x3 φίλτρου), όπου στο τέλος κάθε γραμμής χρειάζεται να φορτωθεί μόνο η επόμενη νέα γραμμή και να διαγράφει η παλαιότερη.

```
for(i=1;i<(N+1);++i) {
    //ASSIGN VALUES TO BUFFERS
    if (i == 1) {
        for (j = 0; j < (M + 2); j++) {
            buffer1[j] = temp[0][j]; //zero row
            buffer2[j] = temp[1][j];
            buffer3[j] = temp[2][j];
        }
    } else {
        for (j = 0; j < (M + 2); j++) {
            buffer1[j] = buffer2[j];   //shift
            buffer2[j] = buffer3[j];
            buffer3[j] = temp[i + 1][j];
        }
    }
    for (j = 1; j < (M + 1); ++j) {
        out[i - 1][j - 1] =
                buffer1[j - 1] * kernel[0][0] + buffer1[j] * kernel[0][1] + buffer1[j + 1] * kernel[0][2] +
                buffer2[j - 1] * kernel[1][0] + buffer2[j] * kernel[1][1] + buffer2[j + 1] * kernel[1][2] +
                buffer3[j - 1] * kernel[2][0] + buffer3[j] * kernel[2][1] + buffer3[j + 1] * kernel[2][2];
    }
}
```

Προτεινόμενη Ιεραρχία Μνήμης

Για την αξιοποίηση της προηγούμενης αλλαγής, χρειάζεται μια ταχύτερη μνήμη.



memory.map

c: > Users > trias > Documents > Emb2 > memory.map

```
00000000 00080000 ROM 4 R 1/1 1/1
00080000 08000000 RAM 4 RW 250/50 250/50
08080000 00090000 DRAM 4 RW 50/1 50/1
08110000 00001000 SRAM 4 RW 10/1 10/1
```

scatter.txt

c: > Users > trias > Documents > Emb2 > scatter.txt

```
ROM 0x0 0x8111000
{
ROM 0x0 0x80000
{
*.o ( +RO )
}
RAM 0x80000 0x8000000
{
* ( +ZI )
}
DRAM 0x8080000 0x90000
{
* (cacheL2)
}
SRAM 0x8110000 0x1000
{
* (cacheL1)
}
}
```

Προτεινόμενη Ιεραρχία Μνήμης

SRAM – L1 Cache

Περιλαμβάνει μεταβλητές που επαναχρησιμοποιούνται συχνά κατά την συνέλιξη της εικόνας.

DRAM – L2 Cache

Περιλαμβάνει τον πίνακα theta για γρηγορότερες προσπελάσεις μνήμης κατά τον χρωματισμό των edges.

```
12  /* code for armulator*/
13  short temp[N+2][M+2];
14  short I_x[N][M];
15  short I_y[N][M];
16  short gray_image[N][M];
17  short gaussian_image[N][M];
18  short norm_image[N][M];
19  float magnitude[N][M];
20  short norm_Y[N][M];
21  short norm_U[N][M];
22  short norm_V[N][M];
23  short i,j;
24
25  #pragma arm section zidata="cacheL1"
26  short buffer1[M+2];
27  short buffer2[M+2];
28  short buffer3[M+2];
29  short Gauss[3][3];
30  short Sobel_x[3][3];
31  short Sobel_y[3][3];
32  #pragma arm section
33
34  #pragma arm section zidata="cacheL2"
35  float theta[N][M];
36  #pragma arm section
```

Τελικό Αποτέλεσμα

- Επαναχρησιμοποίηση Δεδομένων και Προτεινόμενη Ιεραρχία -5.6%

Συνολική εξοικονόμηση : 65 εκατομμύρια κύκλοι

Referenc...	Instruct...	Core_Cycles	S_Cycles	N_Cycles	I_Cycles	C_Cycles	Wait_States	Total	True_Idl...
Statistics	451003395	637865667	500938414	101626375	113998222	0	419360374	1135923385	43218030

```
Image component sizes

      Code    RO Data    RW Data    ZI Data      Debug

      3916         80          0    3780998       8604   Object Totals
     21056        474          0        300       8480   Library Totals

      Code    RO Data    RW Data    ZI Data      Debug

     24972        554          0    3781298      17084   Grand Totals

    Total RO  Size(Code + RO Data)                25526 (  24.93kB)
    Total RW  Size(RW Data + ZI Data)           3781298 (3692.67kB)
    Total ROM Size(Code + RO Data + RW Data)      25526 (  24.93kB)
```