Σχεδιασμός Ενσωματωμένων Συστημάτων

ημμυ |Δ.Π.Θ.

εργασια 1

θωμασ δημακησ 56305 εμμανουηλ καλλιτσουνακης 56373

2016

**ΕΙΣΑΓΩΓΗ**

Το θέμα της εργασίας μας είναι η *περιγραφή ενός αλγορίθμου αύξησης οξύτητας grayscale εικόνων εφαρμόζοντας την τεχνική unsharp masking*. Ο αλγόριθμος θα περιγράφει σε γλώσσα C και θα χρησιμοποιηθεί το περιβάλλον του armulator με σκοπό το τελικό πρόγραμμα να τρέχει σε έναν επεξεργαστή ARM. Κατά το δεύτερο μέρος της εργασίας θα γίνει προσπάθεια βελτιστοποίησης του κώδικα. Επίσης για τον έλεγχο των αποτελεσμάτων θα χρησιμοποιηθεί το περιβάλλον του MATLAB.

**ΠΕΡΙΓΡΑΦΗ ΑΛΓΟΡΙΘΜΟΥ**

Βήμα 1ο: Κατασκευή φίλτρου Gauss με παραμέτρους N και S, όπου το Ν προσδιορίζει τις NxN διαστάσεις του φίλτρου Gauss και το S να προσδιορίζει τη ρίζα της τυπικής απόκλισης σ2.

Βήμα 2ο: Εφαρμογή του φίλτρου μέσω συνέλιξης για την παραγωγή της εξομαλυμένης εικόνας Ιs.

Βήμα 3ο:

Βήμα 4ο:   
όπου k είναι τρίτη παράμετρος του προγράμματος που προσδιορίζει «πόσο» οξύτητα θα έχει η τελική εικόνα.

**ΚΩΔΙΚΑΣ ΣΤΗ C**

#include <stdio.h>

#include <math.h>

#include <string.h>

#include <stdlib.h>

#define IW 352

#define IH 288

#define EIW 356

#define EIH 292

#define N 5

#define S 0.88

#define K 3

#define CG 14.7

#define filename "ak.yuv"

#define finalfilename "akout.yuv"

int i,j;

int image[IH][IW];

double is[IH][IW];

double g[N][N];

double imask[IH][IW];

double iun[IH][IW];

int fim[IH][IW];

int Rep[EIH][EIW];

void crefi(){

for(i=0;i<IH;i++){

for(j=0;j<IW;j++){

if(iun[i][j]>255) fim[i][j] = 255;

else if(iun[i][j]<0) fim[i][j] = 0;

else fim[i][j] = floor(iun[i][j]);

}

}

}

void read()

{

FILE \*frame\_c;

if((frame\_c=fopen(filename,"rb"))==NULL)

{

printf("current frame doesn't exist\n");

exit(-1);

}

for(i=0;i<IH;i++)

{

for(j=0;j<IW;j++)

{

image[i][j]=fgetc(frame\_c);

}

}

fclose(frame\_c);

}

void write()

{

FILE \*frame\_y;

frame\_y=fopen(finalfilename,"wb");

for(i=0;i<IH;i++)

{

for(j=0;j<IW;j++)

{

fputc(fim[i][j],frame\_y);

}

}

fclose(frame\_y);

}

void calfin(){

for(i=0;i<IH;i++){

for(j=0;j<IW;j++){

iun[i][j] = image[i][j] + K\*imask[i][j];

}

}

}

void subim(){

for(i=0;i<IH;i++){

for(j=0;j<IW;j++){

imask[i][j] = image[i][j] - is[i][j];

}

}

}

void gauss(){

double c = exp(pow(N/S,2));

int a = -floor(N/2);

int b;

for(i=0;i<N;i++){

b = -floor(N/2);

for(j=0;j<N;j++){

g[i][j] = c \* exp( -(pow(a,2)+pow(b,2)) / (2\*pow(S,2)) );

g[i][j] = g[i][j]/pow(10,CG);

b++;

}

a++;

}

}

void rot90(){

double tmp;

for(i=0; i<N/2; i++){

for(j=i; j<N-i-1; j++){

tmp=g[i][j];

g[i][j]=g[j][N-i-1];

g[j][N-i-1]=g[N-i-1][N-j-1];

g[N-i-1][N-j-1]=g[N-j-1][i];

g[N-j-1][i]=tmp;

}

}

}

void initialIS(){

for(i=0;i<IH;i++)

for(j=0;j<IW;j++)

is[i][j]=0.0;

}

void initialRep(){

for(i=0;i<EIH;i++)

for(j=0;j<EIW;j++)

Rep[i][j]=0;

}

void conv2(){

int center = floor(N/2);

int left = center-1;

int top = center - 1;

int x,y;

initialRep();

rot90();

rot90();

for(i=top+1;i<=IH+top;i++){

for(j=1+left;j<=IW+left;j++){

Rep[i][j]= image[i-top-1][j-left-1];

}

}

initialIS();

for(x=0;x<IH;x++){

for(y=0;y<IW;y++){

for(i=0;i<N;i++){

for(j=0;j<N;j++){

is[x][y]=is[x][y]+( Rep[i+x][j+y] \* g[i][j] );

}

}

}

}

}

int main(){

read();

gauss();

conv2();

subim();

calfin();

crefi();

write();

return(0);

}

**ΣΤΑΤΙΣΤΙΚΑ ΑΡΧΙΚΟΥ ΚΩΔΙΚΑ**

|  |  |
| --- | --- |
| Instructions | 391152159 |
| Core\_Cycles | 567042251 |
| S\_Cycles | 434982760 |
| N\_Cycles | 99831390 |
| I\_Cycles | 103426633 |
| C\_Cycles | 0 |
| Total | 638240783 |
| Grand Totals | |
| Code | 24200 |
| RO Data | 606 |
| RW Data | 0 |
| ZI Data | 3660348 |
| Debug | 15128 |

**ΒΕΛΤΙΣΤΟΠΟΙΗΣΗ ΚΩΔΙΚΑ**

Αρχικά θα αντικαταστήσουμε όσο είναι αυτό δυνατό τις κλήσεις σε συναρτήσεις με πιο απλές δομές:

* Αντικαθιστούμε τις pow(x,2) με x\*x

|  |  |  |  |
| --- | --- | --- | --- |
|  | Αρχικό | Βελτιστοποιημένο | Απόδοση |
| Instructions | 391152159 | 387356126 | 0.97% |
| Core\_Cycles | 567042251 | 563744463 | 0.58% |
| S\_Cycles | 434982760 | 431443844 | 0.81% |
| N\_Cycles | 99831390 | 100081204 | -0.25% |
| I\_Cycles | 103426633 | 103387396 | 0.04% |
| C\_Cycles | 0 | 0 | 0% |
| Total | 638240783 | 634912444 | 0.52% |
| Grand Totals | | | |
| Code | 24200 | 19292 | 20.28% |
| RO Data | 606 | 470 | 22.44% |
| RW Data | 0 | 0 | 0% |
| ZI Data | 3660348 | 3660348 | 0% |
| Debug | 15128 | 14276 | 5.63% |

* Συγχωνεύσαμε την συνάρτηση calfin και crefi ώστε να συγχωνεύσουμε και τις αντίστοιχες for-loop (loop merging).

|  |  |  |  |
| --- | --- | --- | --- |
|  | Προηγούμενο | Βελτιστοποιημένα | Απόδοση |
| Instructions | 387356126 | 387349431 | 0.002 % |
| Core\_Cycles | 563744463 | 563530956 | 0.038% |
| S\_Cycles | 431443844 | 431336328 | 0.025% |
| N\_Cycles | 100081204 | 99875566 | 0.21% |
| I\_Cycles | 103387396 | 103486723 | -0.096% |
| C\_Cycles | 0 | 0 | 0% |
| Total | 634912444 | 634698617 | 0.034% |
| Grand Totals | | | |
| Code | 19292 | 19200 | 0.48% |
| RO Data | 470 | 470 | 0% |
| RW Data | 0 | 0 | 0% |
| ZI Data | 3660348 | 3660348 | 0% |
| Debug | 14276 | 14064 | 1.49% |

* Συγχωνεύσαμε την συνάρτηση calfin και subim ώστε να συγχωνεύσουμε και τις αντίστοιχες for-loop (loop merging).

|  |  |  |  |
| --- | --- | --- | --- |
|  | Προηγούμενο | Βελτιστοποιημένο | Απόδοση |
| Instructions | 387349431 | 385522532 | 0.47 % |
| Core\_Cycles | 563530956 | 560789335 | 0.49% |
| S\_Cycles | 431336328 | 429104480 | 0.52% |
| N\_Cycles | 99875566 | 99468037 | 0.41% |
| I\_Cycles | 103486723 | 103384159 | 0.099% |
| C\_Cycles | 0 | 0 | 0% |
| Total | 634698617 | 631956676 | 0.43% |
| Grand Totals | | | |
| Code | 19200 | 19136 | 0.33% |
| RO Data | 470 | 470 | 0% |
| RW Data | 0 | 0 | 0% |
| ZI Data | 3660348 | 3660348 | 0% |
| Debug | 14064 | 13860 | 1.45% |

* Ενσωματώσαμε τις initialRep και initialIs στην conv2 και συγχωνεύσαμε τους κοινούς κύκλους των for-loop των initialRep και initialIs.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Προηγούμενο | Βελτιστοποιμένο | Απόδοση |
| Instructions | 385522532 | 385101099 | 0.11 % |
| Core\_Cycles | 560789335 | 559845052 | 0.17% |
| S\_Cycles | 429104480 | 428686169 | 0.097% |
| N\_Cycles | 99468037 | 99046007 | 0.42% |
| I\_Cycles | 103384159 | 103280217 | 0.1% |
| C\_Cycles | 0 | 0 | 0% |
| Total | 631956676 | 631012393 | 0.15% |
| Grand Totals | | | |
| Code | 19136 | 19132 | 0.02% |
| RO Data | 470 | 470 | 0% |
| RW Data | 0 | 0 | 0% |
| ZI Data | 3660348 | 3660348 | 0% |
| Debug | 13860 | 13624 | 1.7% |

* Μειώνουμε τους κύκλους επανάληψης στην conv2 για την αρχικοποίηση των πινάκων κάνοντας τα άλματα κατά 2, κατά 4 και κατά 8, κατά 16 (Loop unrolling).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Κατά 1 | Κατά 2 | | Κατά 4 | |
| Total | 631012393 | 629666251 | 0.21% | 629227683 | 0.28% |
| Code | 19132 | 19156 | -0.13% | 19015 | -0.61% |
|  |  | Κατά 8 | | Κατά 16 | |
| Total |  | 629036841 | 0.31% | 628953299 | 0.32% |
| Code |  | 19476 | -1.8% | 19725 | -3.1% |

Με βάση αυτά τα νούμερα, επιλέγουμε το "Κατά 8" άλματα γιατί επιτυγχάνουμε 0.31% λιγότερους κύκλους το οποίο ισοδυναμεί σε 1,975,552 λιγότερο κύκλους, ενώ η αύξηση στη μνήμη είναι 0.88% που ισοδυναμεί σε 168 bytes περισσότερα. Ωστόσο δεν θα προχωρήσουμε στο κατά 16 ή 32 άλματα γιατί το ποσοστό αύξησης στη μνήμη είναι περίπου 0.20% σε κάθε βήμα: 0.13% -> 0.39% -> 1.8% -> 3.1%

|  |  |  |  |
| --- | --- | --- | --- |
|  | Προηγούμενο | Βελτιστοποιημένο | Απόδοση |
| Instructions | 385101099 | 384159229 | 0.24 % |
| Core\_Cycles | 559845052 | 557870912 | 0.35% |
| S\_Cycles | 428686169 | 427744791 | 0.22% |
| N\_Cycles | 99046007 | 98202141 | 0.85% |
| I\_Cycles | 103280217 | 103090761 | 0.18% |
| C\_Cycles | 0 | 0 | 0% |
| Total | 631012393 | 629037693 | 0.31% |
| Grand Totals | | | |
| Code | 19132 | 19476 | -1.8% |
| RO Data | 470 | 470 | 0% |
| RW Data | 0 | 0 | 0% |
| ZI Data | 3660348 | 3660348 | 0% |
| Debug | 13624 | 13708 | -0.62% |

* Αφαιρούμε την imask και την ενσωματώνουμε μέσα στην iun δηλαδή:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Προηγούμενο | Βελτιστοποιημένο | Απόδοση |
| Instructions | 384159229 | 382434490 | 0.45 % |
| Core\_Cycles | 557870912 | 555233217 | 0.47% |
| S\_Cycles | 427744791 | 426121428 | 0.38% |
| N\_Cycles | 98202141 | 97289471 | 0.93% |
| I\_Cycles | 103090761 | 102989099 | 0.1% |
| C\_Cycles | 0 | 0 | 0% |
| Total | 629037693 | 626399998 | 0.42% |
| Grand Totals | | | |
| Code | 19476 | 19428 | 0.25% |
| RO Data | 470 | 470 | 0% |
| RW Data | 0 | 0 | 0% |
| ZI Data | 3660348 | 2849340 | 22.16% |
| Debug | 13708 | 13664 | 0.32% |

* Αφαιρέσαμε την floor γιατί η floor πηγαίνει στον μικρότερο πλησιέστερο ακέραιο, δηλαδή κόβει το δεκαδικό κομμάτι στους θετικούς. Επειδή έχουμε πάντα θετικούς στις περιπτώσεις που την καλέσαμε, μπορούμε απλά να την αφαιρέσουμε, μιας και πάλι θα κόβεται έτσι το δεκαδικό κομμάτι.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Προηγούμενο | Βελτιστοποιημένο | Απόδοση |
| Instructions | 382434490 | 374045573 | 2.19 % |
| Core\_Cycles | 555233217 | 540952035 | 2.57% |
| S\_Cycles | 426121428 | 415335665 | 2.53% |
| N\_Cycles | 97289471 | 94593002 | 2.77% |
| I\_Cycles | 102989099 | 101990451 | 0.97% |
| C\_Cycles | 0 | 0 | 0% |
| Total | 626399998 | 611919118 | 2.31% |
| Grand Totals | | | |
| Code | 19428 | 18968 | 2.37% |
| RO Data | 470 | 470 | 0% |
| RW Data | 0 | 0 | 0% |
| ZI Data | 2849340 | 2849340 | 0% |
| Debug | 13664 | 13340 | 2.37% |

**ΤΕΛΙΚΑ ΑΠΟΤΕΛΕΣΜΑΤΑ**

Ο τελικός κώδικας είναι ο εξής:

#include <stdio.h>

#include <math.h>

#include <string.h>

#include <stdlib.h>

#define IW 352

#define IH 288

#define EIW 356

#define EIH 292

#define N 5

#define S 0.88

#define K 3

#define filename "ak.yuv"

#define finalfilename "akout.yuv"

int i,j;

int image[IH][IW];

double is[IH][IW];

double g[N][N];

double iun[IH][IW];

int fim[IH][IW];

int Rep[EIH][EIW];

void read()

{

FILE \*frame\_c;

if((frame\_c=fopen(filename,"rb"))==NULL)

{

printf("current frame doesn't exist\n");

exit(-1);

}

for(i=0;i<IH;i++)

{

for(j=0;j<IW;j++)

{

image[i][j]=fgetc(frame\_c);

}

}

fclose(frame\_c);

}

void write()

{

FILE \*frame\_y;

frame\_y=fopen(finalfilename,"wb");

for(i=0;i<IH;i++)

{

for(j=0;j<IW;j++)

{

fputc(fim[i][j],frame\_y);

}

}

fclose(frame\_y);

}

void calfin(){

for(i=0;i<IH;i++){

for(j=0;j<IW;j++){

iun[i][j] = image[i][j] + K\*( image[i][j] - is[i][j] );

if(iun[i][j]>255) fim[i][j] = 255;

else if(iun[i][j]<0) fim[i][j] = 0;

else fim[i][j] = iun[i][j];

}

}

}

void gauss(){

double c = exp(N\*N/(S\*S));

int a = -N/2;

int b;

for(i=0;i<N;i++){

b = -N/2;

for(j=0;j<N;j++){

g[i][j] = c \* exp( -(a\*a+b\*b) / (2\*S\*S) );

g[i][j] = g[i][j]/501187233627271,5;

b++;

}

a++;

}

}

void rot90(){

double tmp;

for(i=0; i<N/2; i++){

for(j=i; j<N-i-1; j++){

tmp=g[i][j];

g[i][j]=g[j][N-i-1];

g[j][N-i-1]=g[N-i-1][N-j-1];

g[N-i-1][N-j-1]=g[N-j-1][i];

g[N-j-1][i]=tmp;

}

}

for(i=0; i<N/2; i++){

for(j=i; j<N-i-1; j++){

tmp=g[i][j];

g[i][j]=g[j][N-i-1];

g[j][N-i-1]=g[N-i-1][N-j-1];

g[N-i-1][N-j-1]=g[N-j-1][i];

g[N-j-1][i]=tmp;

}

}

}

void conv2(){

int center = N/2;

int left = center-1;

int top = center - 1;

int x,y;

rot90();

rot90();

for(i=0;i<IH;i+=8){

for(j=0;j<IW;j+=8){

is[i][j]=0.0;

is[i+1][j+1]=0.0;

is[i+2][j+2]=0.0;

is[i+3][j+3]=0.0;

is[i+4][j+4]=0.0;

is[i+5][j+5]=0.0;

is[i+6][j+6]=0.0;

is[i+7][j+7]=0.0;

Rep[i][j]=0;

Rep[i+1][j+1]=0;

Rep[i+2][j+2]=0;

Rep[i+3][j+3]=0;

Rep[i+4][j+4]=0;

Rep[i+5][j+5]=0;

Rep[i+6][j+6]=0;

Rep[i+7][j+7]=0;

}

}

for(i=IH;i<EIH;i++)

for(j=IW;j<EIW;j++)

Rep[i][j]=0;

for(i=top+1;i<=IH+top;i++){

for(j=1+left;j<=IW+left;j++){

Rep[i][j]= image[i-top-1][j-left-1];

}

}

for(x=0;x<IH;x++){

for(y=0;y<IW;y++){

for(i=0;i<N;i++){

for(j=0;j<N;j++){

is[x][y]=is[x][y]+( Rep[i+x][j+y] \* g[i][j] );

}

}

}

}

}

int main(){

read();

gauss();

conv2();

calfin();

write();

return(0);

}

Η διαφορά με την μη βελτιστοποιημένη έκδοση είναι:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Προηγούμενο | Βελτιστοποιημένο | Απόδοση | Διαφορά |
| Instructions | 391152159 | 374045573 | 4.37 % | 17106586 |
| Core\_Cycles | 567042251 | 540952035 | 4.6% | 26090216 |
| S\_Cycles | 434982760 | 415335665 | 4.5% | 19647095 |
| N\_Cycles | 99831390 | 94593002 | 5.25% | 5238388 |
| I\_Cycles | 103426633 | 101990451 | 1.39% | 1436182 |
| C\_Cycles | 0 | 0 | 0% | 0 |
| Total | 638240783 | 611919118 | 4.12% | 26321665 |
| Grand Totals | | | |  |
| Code | 24200 | 18968 | 21.62% | 5232 |
| RO Data | 606 | 470 | 22.44% | 136 |
| RW Data | 0 | 0 | 0% | 0 |
| ZI Data | 3660348 | 2849340 | 22.16% | 811008 |
| Debug | 15128 | 13340 | 11.82% | 1788 |

Για την επιλογή οξύτητας αλλάζουμε το Κ ως εξής:

Κ=0 : Καμία Αλλαγή

Κ=1 : 1ος βαθμός οξύτητας

Κ=2 : 2ος βαθμός οξύτητας

Κ=3 : 3ος βαθμός οξύτητας

**ΠΙΝΑΚΕΣ ΚΑΙ ΠΡΟΣΠΕΛΑΣΕΙΣ**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Bit | Διαστάσεις | Μέγεθος | Προσπελάσεις |
| image | 32 | 288x352 | 396ΚΒ | 405504 |
| is | 64 | 288x352 | 792ΚΒ | 5182848 |
| g | 64 | 5x5 | 200Β | 2534667 |
| iun | 64 | 288x352 | 792ΚΒ | 403487 |
| fim | 32 | 288x352 | 396ΚΒ | 202752 |
| Rep | 32 | 292x356 | 406ΚΒ | 2648464 |
| Σύνολο |  |  | 2.72ΜΒ | 11377722 |

**SCATTER - HEAP - MAP**

Φτιάχνουμε τα εξής αρχεία scatter, heap, map:

Αρχικά διαλέξαμε τον τύπο 2, «μια περιοχή στην προβολή φόρτωσης και τρεις μη συνεχόμενες περιοχές στην προβολή εκτέλεσης». Αυτή η προσέγγιση χρησιμοποιείτε, για παράδειγμα, σε ROM-based ενσωματωμένα συστήματα, που τα RW δεδομένα αντιγράφονται από τη ROM στη RAM κατά την εκκίνηση.

Η μνήμη ROM μας είναι **19438**Bytes, οπότε σε δεκαεξαδικό είναι **4BEE** και τα ΖΙ και RW δεδομένα είναι **2849340**Bytes, οπότε σε δεκαεξαδικό είναι **2B7A3C**.

Για τα δεδομένα heap και stack επιλέξαμε **6.98**ΜΒ, δηλαδή **700000-4BEE=6FB412**.

Τέλος επιλέξαμε για συχνότητα ρολογιού 50ΜΗz.

scatter.txt

ROM 0x00000000 0x00004BEE

{

ROM 0x00000000 0x00004BEE

{

\*.o ( +RO )

}

SRAM 0x00700000 0x002B7A3C

{

\* ( +RW )

\* ( +ZI )

}

}

heap.c

#include <rt\_misc.h>

\_\_value\_in\_regs struct \_\_initial\_stackheap \_\_user\_initial\_stackheap(

unsigned R0, unsigned SP, unsigned R2, unsigned SL){

struct \_\_initial\_stackheap config;

config.heap\_base = 0x00004BEE;

config.stack\_base = 0x00700000;

return config;}

memory.map

00000000 00004BEE ROM 4 R 1/1 1/1

00700000 002B7A3C SRAM 4 RW 250/50 250/50

|  |  |  |  |
| --- | --- | --- | --- |
|  | Προηγούμενο | Βελτιστοποιημένο | Απόδοση |
| Instructions | 374045573 | 374045542 | 0% |
| Core\_Cycles | 540952035 | 540951987 | 0% |
| S\_Cycles | 415335665 | 415335647 | 0% |
| N\_Cycles | 94593002 | 94592981 | 0% |
| I\_Cycles | 101990451 | 101990442 | 0% |
| C\_Cycles | 0 | 0 | 0% |
| Wait\_States | - | 259734510 | - |
| Total | 611919118 | 871653580 | -42.46% |
| True\_Idle\_Cycles | - | 38769026 | - |
| Grand Totals | | | |
| Code | 18968 | 18936 | 0.17% |
| RO Data | 470 | 450 | 4.26% |
| RW Data | 0 | 0 | 0% |
| ZI Data | 2849340 | 2849340 | 0% |
| Debug | 13340 | 14772 | -10.73% |