Universitat de Barcelona

Tecnologies Multimèdia

4t d'Enginyeria Informàtica

Entrega – FX codec

Albert Folch i Xavi Moreno

9/1/2015

Índex

Introducció……3
Objectiu……..3
Descripció detallada de la implementació……3
    Algorisme……3
    Què és un fitxer FX i com ho guardem?.....5
    Color de les tessel·les escollit……..5
Codi font i execució……..5
Resultats de funcionament i discussió……..6
    Anàlisi del funcionament……...6
    Anàlisi del rendiment……7
Conclusió……...8
Referències……...8

Introducció

Aquest informe neix de la necessitat de mostrar el treball pràctic realitzat durant tot el semestre. En el món de les dades és obligatori parlar de com comprimir-les així doncs, aquí veureu el resultat de la implementació del nostre codec juntament amb les comparatives obtingudes en diferents situacions.

Objectiu

L'objectiu d'aquest compressor-descompressor és comprimir a partir d'una tècnica intra-quadre. El mètode el podem trobar explicat en la següent secció. La compressió utilitzada és una compressió amb pèrdua, per tant el que volem es comprimir, però intentar no perdre molta informació, i que al descomprimir tinguem un resultat similar al original.

Descripció detallada de la implementació

Algorisme

Per comprimir, fem servir un algorisme de cerca basada en moviment d'una imatge respecte la seva imatge de referència tal com s'ha explicat a classe. Aquest algorisme és un algorisme de cerca 'circular' de tessel·les i funciona de la següent manera:

- Amb el número de píxels de cerca (a partir d'ara pc) que s'escolleixi farem que el radi de píxels de l'àrea de cerca sigui més petita o més gran.
- Si el número de pc és 4 (exemplificat amb la *k* al dibuix), anirem buscant des de la posició de la tessel·la de dalt a la esquerra fins a 3 (pc - 1) píxels més amunt i més a l'esquerra de la tessel·la. La cerca s'executarà des de la posició de la tessel·la de la imatge de referència i anirà cap a fora ja que és més provable que el moviment hagi sigut petit i estigui a prop la tessel·la més semblant o igual.
- Per cada posició de la tessel·la a comprovar fem una *normalized cross correlation* amb la tessel·la de la imatge de referència. Si són més semblants que el factor de qualitat introduït per l'usuari retornarem la posició. Sinó, continuarem buscant fins que trobi alguna o retornem *null* per indicar que no n'hem trobat cap.

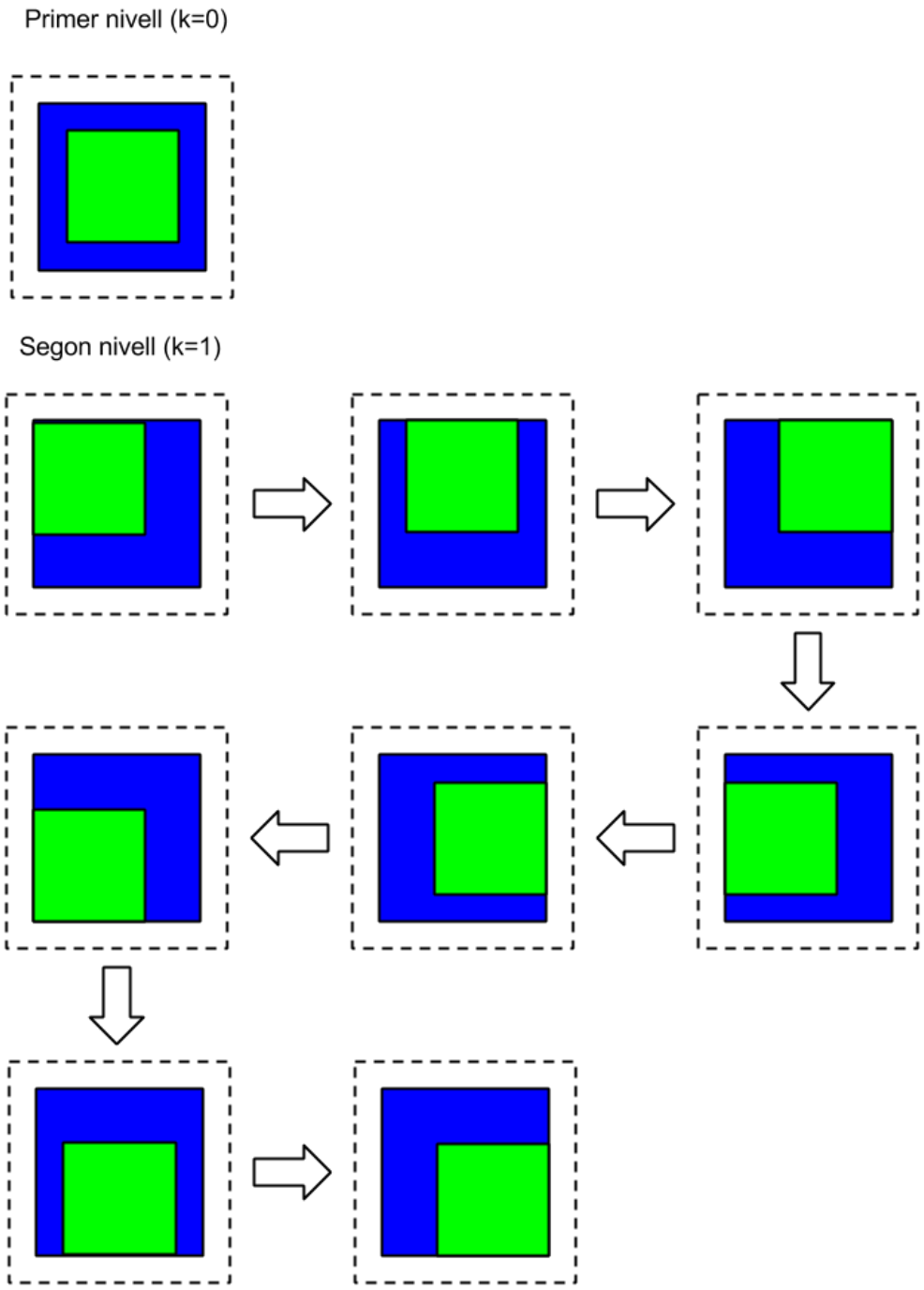

Figura 1 - Dibuix de l'algorisme. Font: pròpia.

Què és un fitxer FX i com ho guardem?

Un fitxer FX és un fitxer on guardem dades sobre la compressió. La informació que guardem:

- Mida de la tessel·la (sempre es quadrada)
- Group of Pictures (GoP): són les imatges de referències que hi han (la primera es una imatge de referència, a partir de llavors es reparteixen segons el GoP escollit)
- Conjunt de coincidències de tessel·les per frame. És a dir, guardem una llista de frames on cada frame té una hash de tessel·les amb la posició destí corresponent. Ex.: 4,(56,78), la tessel·la 4 de la imatge origen es troba a la posició (56,78) de la imatge destí.

El format escollit per guardar tota aquesta informació ha sigut un fitxer JSON. Ens tirava enrere perquè es repetien masses símbols tals com: '{', '}', ':', … Però alhora ens donava la comoditat de guardar valors i estructures de dades més complexes.
Per resoldre el problema de la repetició de símbols, el que hem fer és comprimir el JSON en gzip. D'aquesta manera obtenim bons resultats (els resultats es poden veure més abaix)

Color de les tessel·les escollit

Quan esborrem una tessel·la de la imatge de destí, l'omplim del color mig de tota la imatge. Aquest color mig és un Color que tindrà per cada canal la mitja de cada canal de la imatge. Fent això aconseguim que al fer la compressió amb JPG sigui més alta ja que la transició és menys brusca.

Codi font i execució

Per tal d'executar el programa i poder obrir o guardar un fitxer FX, cal llegir abans aquesta part.

En el menú 'File' trobem tot allò referent a guardar i obrir fitxers. Si el que volem es guardar amb FX, primer de tot hem de tenir un vídeo carregat en el reproductor. Ara si cliquem a: 'Save FX …' ens demanarà que seleccionem una carpeta. En aquesta carpeta es crearà un fitxer 'fxf' que es un gzip, i un fitxer 'images.zip'. Per tant si ara volem obrir un fitxer FX hem de seleccionar una carpeta que contingui un fitxer 'fxf' i un: 'images.zip'.

El codi font del projecte desenvolupat es pot veure a: https://svn.mat.ub.edu/svn/TM/7z/

Resultats de funcionament i discussió

Anàlisi del funcionament

La comparativa de compressió la fem respecte el zip d'imatges del cub en format JPG que ocupen un total de 1.061.650 bytes. En la següent taula veiem per cada prova els paràmetres seleccionats:

- GoP: és el número d'imatges de referència que s'agafaran, *group of pictures.*
- ST: és la mida quadrada d'una tessel·la en píxels, *size of region.*
- RP: és el número de píxels de l'àrea de cerca respecte la posició de la tessel·la de la imatge de referència, *radius of pixels.*
- QF: és el factor de qualitat quant de semblants són una tessel·la amb una altra, com més alta més iguals hauran de ser, *quality factor.*
- Imatges: és la quantitat de bytes que ocupen les imatges en .zip amb els forats de les tessel·les eliminades.
- FX: és la quantitat de bytes que ocupa el fitxer FX.
- Total: és la suma d' Imatges + FX

En la taula hem marcat en vermell la prova que ha comprimit menys i hem marcat amb verd la que ha comprimit més i es veia força bé.

Prova	GoP	ST	RP	QF	Imatges	FX	Total	Comprimit
prova2	30	15	5	0.70	1.054.412	7.990	1.062.402	−0,07%
prova3	15	12	5	0.55	1.060.116	19.780	1.079.896	−1,72%
prova4	10	18	6	0.60	1.032.454	7.346	1.100.991	2,06%
prova5	15	20	20	0.70	1.046.055	6.936	1.052.991	0.82%
prova7	10	40	40	0.75	1.015.909	2.050	1.017.959	4.12%
prova9	10	20	3	0.25	929.438	14.433	943.871	11.04%
prova10	5	20	20	0.20	843.497	38.945	882.442	16.88%
prova11	10	20	3	0.50	1.011.294	4.544	943.871	4.32%

Taula 1 - Taula d'anàlisis de funcionament. Font: pròpia

Anàlisi del rendiment

En el nostre anàlisi de rendiment hem calculat el temps que triga en executar-se en la compressió i em fet el quocient del que s'ha comprimit respecte el temps que ha trigat. En verd marquem el que té un quocient més alt i en vermell el pitjor.

Prova	**Temps (segons)**	**(% Comprimit) / Temps**
prova2	350	-0,0002
prova3	523	-0,003288719
prova4	673	0,003060921
prova5	2645	0,000310019
prova7	5850	0,000704274
prova9	228	0,048640351
prova10	886	0,019051919
prova11	280	0,015428571

Taula 2 - Taula d'anàlisis de rendiment. Font: pròpia

Conclusió

En conclusió, jo Albert Folch veig que el nostre codec basat en pèrdues és prou bo en alguns casos com per exemple els mostrats en verd anteriorment i amb l'exemple del cub. Cal dir que no podem posar una àrea de cerca massa gran ja que triga massa estona, hores. Aquests valors no s'haurien d'aplicar normalment ja que ajustant la resta obtenim una reproducció del cub prou fluïda i amb pocs errors (ja esperats) en cantonades.

En definitiva, aconseguim comprimir fins a poc més del 11% (prova9) de la mida original amb un algorisme de cerca només basat en moviment i amb pèrdua amb un temps acceptable. Com a millora jo canviaria el codec per un algorisme sense pèrdua que fos més eficaç.

*La conclusió de Xavi Moreno Liceras es troba en el seu informe.

Referències

Per aquest projecte només hem utilitzat dues llibreries:

- jlfgr-1_0: Llibreria proporcionada per la Universitat de Barcelona que conté una serie d'icones.
- org.json: Aquesta llibreria la utilitzem a l'hora de guardar el FXFile, ja que com hem explicat guardem la informació en JSON.