Arnau Vancells i Blai Ras

Tecnologies Multimèdia
Projecte Pràctiques – Avconv

Exercici 1. Expliqueu el significat de la informació proporcionada per la notació DEVILS.

```
Codecs:
 D..... = Decoding supported
 .E.... = Encoding supported
 ..V... = Video codec
 ..A... = Audio codec
 ..S... = Subtitle codec
 ...I.. = Intra frame-only codec
 ....L. = Lossy compression
 .....S = Lossless compression
 -------
 D.VI.. 012v                 Uncompressed 4:2:2 10-bit
 D.V.L. 4xm                  4X Movie
 D.VI.S 8bps                 QuickTime 8BPS video
 .EVIL. a64_multi            Multicolor charset for Commodore 64 (encoders: a64multi )
 .EVIL. a64_multi5           Multicolor charset for Commodore 64, extended with 5th color (colram) (e
ncoders: a64multi5 )
 D.V..S aasc                 Autodesk RLE
 D.VIL. aic                  Apple Intermediate Codec
 DEVI.S alias_pix            Alias/Wavefront PIX image
 DEVIL. amv                  AMV Video
 D.V.L. anm                  Deluxe Paint Animation
 D.V.L. ansi                 ASCII/ANSI art
 DEV..S apng                 APNG (Animated Portable Network Graphics) image
 DEVIL. asv1                 ASUS V1
 DEVIL. asv2                 ASUS V2
 D.VIL. aura                 Auravision AURA
 D.VIL. aura2                Auravision Aura 2
 D.V... avrn                 Avid AVI Codec
 DEVI.. avrp                 Avid 1:1 10-bit RGB Packer
 D.V.L. avs                  AVS (Audio Video Standard) video
 DEVI.. avui                 Avid Meridien Uncompressed
 DEVI.. ayuv                 Uncompressed packed MS 4:4:4:4
 D.V.L. bethsoftvid          Bethesda VID video
 D.V.L. bfi                  Brute Force & Ignorance
 D.V.L. binkvideo            Bink video
 D.VI.. bintext              Binary text
 DEVI.S bmp                  BMP (Windows and OS/2 bitmap)
 D.V..S bmv_video            Discworld II BMV video
 D.VI.S brender_pix          BRender PIX image
 D.V.L. c93                  Interplay C93
 D.V.L. cavs                 Chinese AVS (Audio Video Standard) (AVS1-P2, JiZhun profile)
 D.V.L. cdgraphics           CD Graphics video
 D.VIL. cdxl                 Commodore CDXL video
 DEV.L. cinepak              Cinepak
:
```

Cada lletra de la notació DEVILS significa una cosa:

- **D:** significa que suporta la descodificació.
- **E:** significa que suporta la codificació.
- **V** (també pot ser **A** o **S**)**:** Significa còdec de vídeo (Àudio o Subtítols corresponent a la lletra).
- **I:** significa que el còdec és només intra-quadre, o sigui, que es fa servir anàlisis de freqüències de les dades.
- **L:** la compressió té pèrdua.
- **S:** la compressió no té pèrdua.

Arnau Vancells i Blai Ras

Exercici 2. Comproveu quin és el suport que teniu, en el vostre sistema, pels còdecs mjpeg, mpeg1video, mpeg4 i h264. Quins utilitzen compressió intra-quadre, inter-quadre o ambdós? Quin/s és/són més similar/s al còdec que esteu desenvolupant a pràctiques?

- ***mjpeg* (DEVIL.):** còdec de vídeo que suporta la descodificació, la codificació, és únicament intra-quadre i la compressió té pèrdua.
- ***mpeg1video* (DEV.L.):** còdec de vídeo que suporta la descodificació, la codificació, utilitza tant intra com *inter-frame* i la compressió es fa amb pèrdua.
- ***mpeg4* (DEV.L.):** còdec de vídeo que suporta la descodificació, la codificació, utilitza tant intra com *inter frame* i la compressió es fa amb pèrdua.
- ***h264* (DEV.LS):** còdec de vídeo que suporta la descodificació, la codificació, utilitza tant intra com *inter-frame* i la compressió es pot fer amb o sense pèrdua.

El més similar al còdec de pràctiques és el mpeg1video, on tant es comprimeixen les imatges fent servir l'espai de freqüències, com es fa una predicció del moviment de les imatges entre si per tal de poder codificar informació fent servir comparacions entre *frames*.
El MPEG4 és una versió millorada de MPEG1 i per tant és més complexa, igual que el h264 que també es coneix com a MPEG4 part 10, és semblant però està molt més optimitzat per a tenir més compressió i més qualitat, per tant també és més complexa.

Exercici 3. Compareu els diferents còdecs anteriors de forma quantitativa en quant a temps de CPU, memòria utilitzada en el processat i mida del fitxer resultant (e.g. du –sh o bé ls –la). Observeu el resultat de la compressió en cada cas i ordeneu, de forma qualitativa, la qualitat obtinguda amb cada un dels còdecs.

Còdec	Mida fitxer (kB)	Temps de CPU (s)	Memòria utilitzada (kB)
h264	408	2.471	95440
mpeg4	196	0.343	53064
mjpeg	672	0.353	50832
mpeg1video	232	0.342	5233366

El còdec h264, tot i ser el més lent d'entre els quatre, és el que aconsegueix millor qualitat d'imatge. La compressió de les dades, tot i no ser la més petita de les 4, està en un terme mig que, tenint en compte la qualitat de les imatges, és un resultat molt bo.
Els tres últims còdecs són difícils de diferenciar en termes de qualitat d'imatge, però podem veure que tant mpeg4 com mpeg1video comprimeixen les dades millor que *mjpeg*.
En quant a memòria utilitzada en el processat, podem dir que h264 un altre cop es queda en un punt mig, mpeg4 i *mjpeg* consumeixen poca memòria, i mpeg1video consumeix molta més memòria que la resta.

Arnau Vancells i Blai Ras

Exercici 4. Realitzeu una taula amb els ràtios de compressió que obtenim amb els diferents còdecs respecte a la mida total del fitxer comprimit amb *mjpeg*. Per què prenem aquest fitxer com a referència?

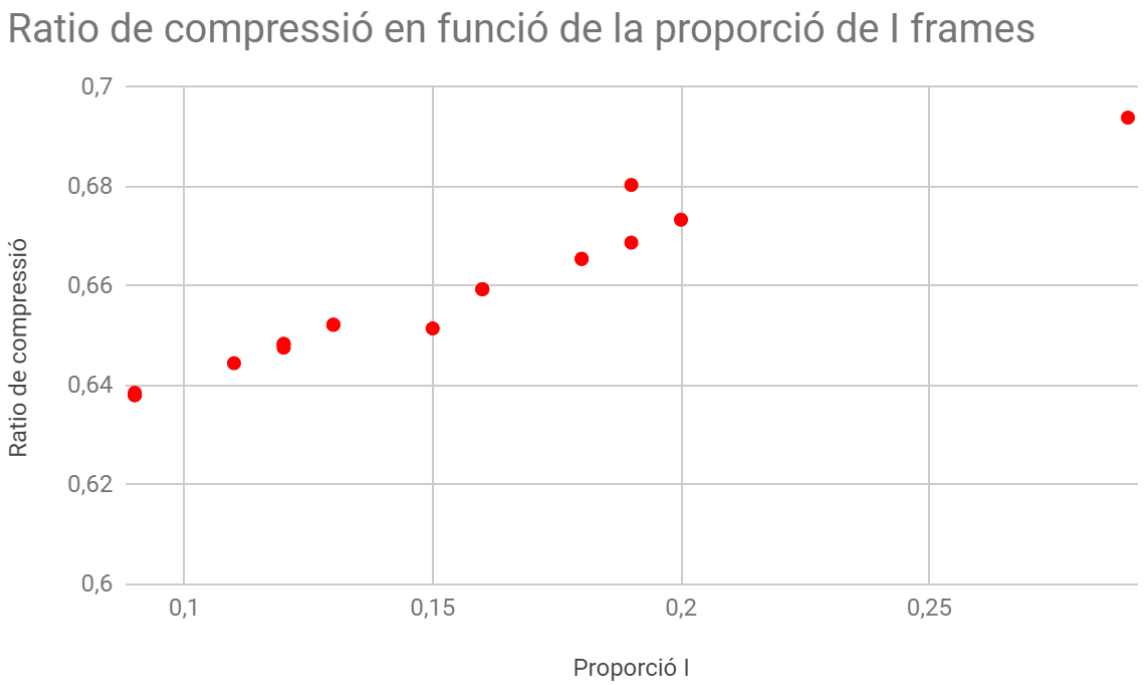
Còdec	**Factor de compressió**
h264	0.607
mpeg4	0.292
mpeg1video	0.345
mjpeg	1

Agafem aquest fitxer com a referència perquè és l'únic que no realitza compressió *inter-quadre*, de forma que per a comparar l'ús d'aquesta tècnica de compressió és millor comparar-ho amb un fitxer on hi ha compressió només intra-quadre.
Cal tenir en compte que si la compressió és massa baixa, estarem perdent molta informació de les dades, i per tant qualitat d'imatge. L'exemple clar és h264, que amb un *ratio* de compressió pitjor a nivell de memòria que els de mpeg4 i mpeg1, aconsegueix una qualitat d'imatge molt millor.

Exercici 5. Aneu modificant el GOP mitjançant el paràmetre corresponent (e.g. entre 1 i 100), mantenint el nombre d'imatges de referència a 0 i el *frame-rate* a 25. Fixeu-vos en la sortida per tal de visualitzar la proporció de *frames* I i P. Feu una gràfica de com és modifica el ràtio de compressió en funció d'aquesta proporció. Expliqueu el resultat.

GOP	**Mida fitxer (kB)**	**I-Frames**	**P-frames**
0	460,2	100	0
5	466,3	29	71
10	437,8	15	85
15	433,1	11	89
20	435,2	12	88
25	452,5	20	80
30	449,4	19	81
35	457,2	19	81
40	447,2	18	82
45	447,2	18	82

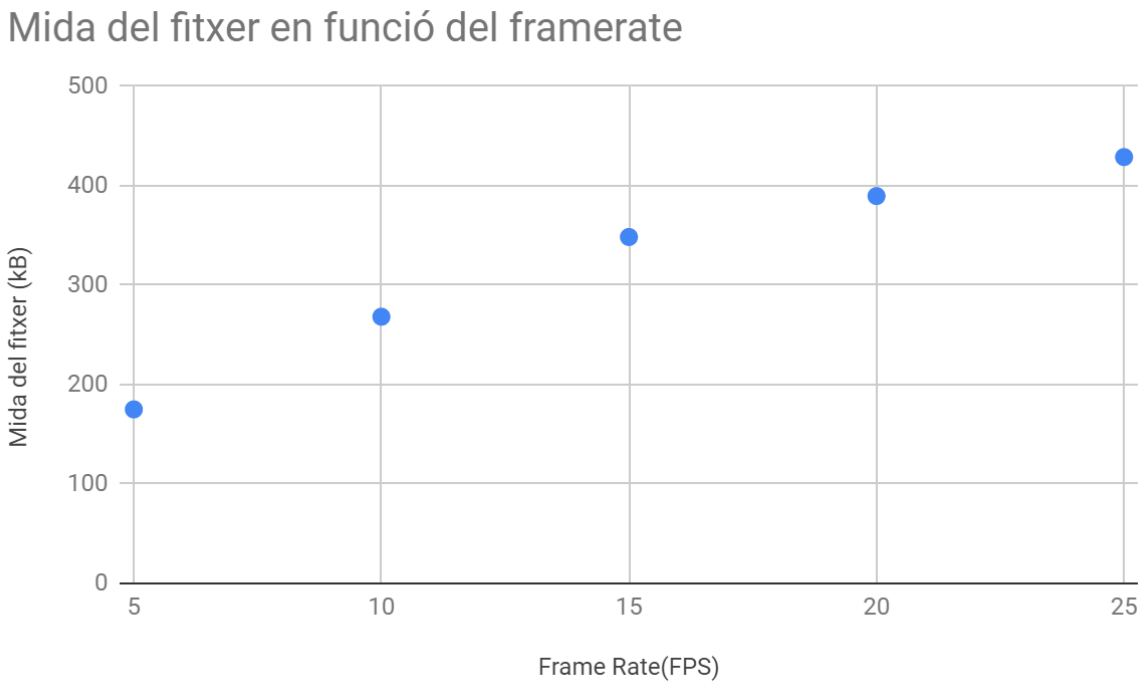
GOP	**Mida fitxer (kB)**	**I-Frames**	**P-frames**
50	443,1	16	84
55	443,1	16	84
60	438,3	13	87
65	438,3	13	87
70	435,7	12	88
75	435,7	12	88
80	428,8	9	91
85	428,8	9	91
90	428,8	9	91
95	428,8	9	91
100	429,1	9	91



En el gràfic es pot observar que com més petita és la proporció de I *frames*, millor és el factor de compressió. Aquests valors variaran en funció del vídeo amb el que estem treballant, en l'exemple de la seqüència de "cubo.zip" dels 100 fotogrames que té obtenim la millor compressió fent servir 9 *frames* I i la resta com a P.
Això és degut a la redundància que hi ha entre les imatges del vídeo. Al ser bastant repetitiu, utilitzar gran quantitat de *frames* I no és eficient, ja que estarem guardant informació repetida per cada frame I, que podria codificar-se perfectament amb un GOP més gran.

Exercici 6. Mantenint el GOP i el nombre d'imatges de referència constants, i sense modificar el *frame-rate* a l'entrada, modifiqueu el de sortida en valors entre 5 i 25. Genereu una gràfica amb la mida dels fitxers resultants. És el que esperàveu? Quin creieu que és el motiu? Observeu alguna relació quantitativa entre el *frame-rate* i la mida del fitxer?
Hem pres un GOP de 85, ja que és un dels valors que hem utilitzat que obtenia millors resultats (també podríem haver agafat 80, 90 o 95).

GOP	Frame-rate Sortida	Mida fitxer (kB)
85	25	428,8
85	20	389,6
85	15	348,4
85	10	268,2
85	5	175,0



El resultat és el que esperàvem i el que té tot el sentit del món: reduir la quantitat d'imatges per segon que conté el vídeo implica una reducció del fitxer de sortida. Hi haurà menys imatges a reproduir.
Així que, evidentment, un major *frame-rate* comporta una major mida del fitxer. Per tant, sí que observem una relació quantitativa entre aquestes dues variables.

Exercici 7. Compareu el temps de processat respecte a la mida del fitxer obtingut amb les funcions de comparació *sum* of *absolute differences* (*sad*), *sum* of *squared* errors (*sse*), *sum* of *absolute* Hadamard *transformed differences* (*satd*) i *chroma* (igual que *sad* però utilitzant la informació del color, en comptes de només la lluminositat). Segons aquests resultats, quina és la més convenient a utilitzar? Expliqueu breument també el funcionament de les 4 opcions.

GOP	***Frame-rate* de Sortida**	**Funció de Comparació**	**Temps de processat (s)**	**Mida fitxer (kB)**
85	25	sad	3,249	429,7
85	25	sse	3,204	429,7
85	25	satd	3,207	429,7
85	25	chroma	3,841	428,8

La funció de comparació més convenient a utilitzar és la *chroma*, ja que, tot i que té un temps de processat major, la mida del fitxer és menor. A més, en comparació amb els altres mètodes no hi ha un augment excessiu de temps de processat.

- **SAD:** és una mètrica calculada fent la suma del valor absolut de les diferències entre píxels, conegut també com a distància de Manhattan.
- **SSE:** és una mètrica molt usada en estadística. Es calcula fent la suma de diferències al quadrat entre els píxels dels blocs a comparar.
- **SATD:** és una mètrica molt usada en estimació de moviment. Es calcula aplicant una transformació Hadamard a la diferència entre píxels dels diferents blocs.
- **Chroma:** funciona de la mateixa forma que SATD però fent servir els valors de cada canal de color, en lloc de tant sols el valor de intensitat de les imatges.

Exercici 8. Compareu el temps de processat amb la mida del fitxer resultant dels algoritmes de cerca de desplaçament *diamond* (dia), *hexagon* (hex), *uneven multi-hexagon* (umh) i complert (*full*). Quin considereu més òptim en el nostre cas? Expliqueu breument també el funcionament de les 4 opcions.

GOP	***Frame-rate* de Sortida**	**Funció de comparació**	**Algoritme desplaçament**	**Temps de Processat (s)**	**Mida Fitxer (kB)**
85	25	chroma	dia	3.684	433,3
85	25	chroma	hex	3,833	428,8
85	25	chroma	umh	4.442	425,9
85	25	chroma	full	5.244	426,0

En el nostre cas, el algoritme de desplaçament més òptim és el *umh*, ja que té un temps de processat que no és ni molt gran ni molt petit(en comparació amb els demés es troba en un punt mig) i a més és el que aconsegueix un fitxer amb menor mida.

- **DIA:** es realitza una cerca seguint un patró en forma de diamant, o sigui, es busca en els 4 veïns més propers(situats horitzontal i verticalment). Utilitzat per a vídeos amb pocs canvis en el moviment.
- **HEX:** cerca en dos passos, primer es busca en píxels a una determinada distància (seguint un patró hexagonal), i es busca en els veïns d'aquests píxels, per tal de veure si hi ha moviment. Permet trobar moviment en més direccions que DIA, però tot i així és limitat.
- **UMH:** seria com una espècie de cerca en hexàgon però aquest té diferents mides, és a dir, per cada '*good hit*' reduïm l'espai de cerca (reduïm la mida del hexàgon) i un cop hem arribat al mínim, busquem de manera recursiva el següent hexàgon (distància ortogonal) a iterar. Permet trobar moviments amplis i aleatoris de forma eficient.
- **FULL:** es busca de forma exhaustiva en els veïns del píxel. Molt costós a nivell computacional. Útil per a comparar l'eficiència d'altres tècniques de cerca.

Arnau Vancells i Blai Ras

Exercici 9. Fixant el mètode d'estimació de desplaçament a *umh*, proveu diferents valors del rang de cerca de desplaçament. Quin valor resulta més òptim?

Desplaçament	**Temps (s)**	**Mida (kB)**
0	4,180	464,8
15	4.353	426,2
25	4,462	423,7
50	5,1	424,4
75	5,520	424,5
100	5,961	425,4
200	7,143	425,5
500	8.170	425,5
1000	10,836	425,5
3000	11,352	425,5

Tal i com veiem en la taula superior, la mida de desplaçament 'perd sentit' a mesura que l'augmentem a partir d'un cert valor, ja que augmentar el desplaçament no fa que es redueixi la mida del fitxer mentre que el temps augmenta quantitativament (estarem fent una cerca a una distància molt gran, quan la imatge no té les mides suficients per a realitzar-la).
El valor més òptim és el desplaçament de 25, ja que obtenim la mida del fitxer més petita. Tot i així, a l'exercici 10 hem obtingut més compressió amb mida 50, per tant, el valor òptim oscil·larà entre aquests valors.

Arnau Vancells i Blai Ras

Exercici 10. Tenint en conte els resultats de tots els exercicis anterior, proposeu quin creieu que és i per què el millor set de paràmetres a utilitzar amb el còdec h264. Utilitzeu el *avconv* per generar aquest vídeo i comproveu visualment que la qualitat de la imatge és l'adequada.
Amb el còdec h264, mencionat ja anteriorment com el còdec que millor qualitat ens generava, hem arribat a la conclusió de que els següents paràmetres són els més òptims i idonis:

1. Group Of Pictures: 85
2. Frame-rate de sortida: 25
3. Funció de comparació: chroma
4. Algoritme de desplaçament: umh
5. Desplaçament: 25

El fitxer generat triga 4,311 segons a generar-se i pesa 426,8 kB. En la pregunta 6 hem mencionat que ens era indiferent seleccionar un GOP de 80, 85, 90 o 95, ja que tots trigaven aproximadament el mateix i pesaven igual. Així doncs, hem canviat només el paràmetre GOP amb aquests valors per veure si vèiem una diferència substancial de temps o pes, però no ha sigut així. En conseqüència, hem triat el 85 mateix.
Hem generat el vídeo i la qualitat de la imatge considerem que és la màxima possible tenint en compte també el temps d'execució i mida del fitxer resultant.
Al principi hem optat per agafar un desplaçament de 50, el qual triga una mica més (5,1 segons) però ocupava una mica menys (424,4 kB). No hi ha una gran diferència, així que hem optat per la opció amb menys temps d'execució.

**Tots els exercicis anteriors s'han realitzat en Linux Ubuntu 16.04*