

Índex

1	Introducció	6
1.1	Orígens i dificultats.....	6
1.2	Objectius	8
1.3	Abast	9
1.4	Aplicacions	9
1.5	Eines de desenvolupament.....	9
1.6	Planificació del projecte.....	10
1.7	Organització del projecte	10
2	Shots	12
2.1	Extracció dels frames.....	12
2.2	Divisió del vídeo en shots	13
2.2.1	L'espai de color RGB.....	13
2.2.2	L'espai de color HSV	15
2.2.3	Histograma de color	17
2.2.4	Algorisme de classificació per shots	17
3	Detecció i descripció de punts d'interès	20
3.1	L'algorisme de detecció de cantonades Moravec:	20
3.2	L'algorisme de detecció de cantonades Harris-Stephens:	23
3.3	L'algorisme MSER	25
3.4	L'operador Harris-Affine.....	26
3.5	SIFT (Scale-Invariant Feature Transform)	28
4	Anàlisi i disseny: Reconeixement de les regions d'interès	32
4.1	Tracking	33

4.2 Clustering.....	36
4.2.1 Clustering jeràrquic	37
4.2.2 Clustering particional	37
4.3 Classificació de regions d'interès noves.....	42
5 Detalls d'implementació	44
5.1 Estructures de dades.....	44
5.1.1 Estructures de dades en Matlab	45
5.1.2 Estructures de dades proposades	48
5.2 Agrupació de punts en objectes	50
5.3 Tècnica per reconèixer grups.....	50
5.4 Detalls sobre les noves regions.....	51
6 Anàlisi de resultats	52
6.1 Proves sintètiques	53
6.1.1 Prova de dos moviments	53
6.1.2 Prova de tres moviments	56
6.1.3 Prova de dos moviments confusos	58
6.2 Proves reals.....	60
6.2.1 Seqüència “Groundhog’s Day” 1	60
6.2.2 Seqüència “Groundhog’s Day” 2	62
6.2.3 Seqüència “Star Wars: Episodi 1” 1	64
6.2.3 Seqüència “Star Wars: Episodi 1” 2.....	66
6.2.4 Seqüència “El tigre y la nieve”.....	68
6.2.5 Seqüència “Me llamo Earl”	70
7 Conclusiones i treballs futurs	72
7.1 Conclusiones.....	72
7.2 Treballs futurs.....	73
8 Bibliografia	74

Índex de figures

Figura 1.1: Exemple de segmentació d'imatges.	7
Figura 1.2: Esquema general del funcionament del PFC.	8
Figura 2.1: Model de color additiu[12].	13
Figura 2.2: Representació de l'espai de color RGB en forma de cub.	14
Figura 2.3: Exemple de la variació d'intensitat d'un color a l'espai RGB[12].	15
Figura 2.4: Representació del model de color HSV[12].	15
Figura 2.5: Diferència entre espais de color.	16
Figura 2.7: Histogrames de les components H,S i V d'una imatge	17
Figura 2.7: Seqüència en la que es veuen dos canvis d'escena.	18
Figura 2.8: Taula per comparar la diferència entre frames.	18
Figura 2.9: Resultat final de separar l'escena de la figura 2.7 en shots.	19
Figura 2.10: Estructura final de les imatges classificades en shots.	19
Figura 3.1: Càlcul de la variació d'intensitat amb una finestra de 3x3 en la direcció de la diagonal superior dreta[14].	21
Figura 3.2: Diferents casos en que es pot trobar l'operador Moravec[14].	22
Figura 3.3: Mapa de grau de cantonada resultant de l'algorisme Moravec sobre una imatge senzilla, on només hi ha intensitat 0 i 255[14].	22
Figura 3.4: Resultats de l'aplicació de l'algorisme MSER sobre dues imatges consecutives d'una escena.	26
Figura 3.5: Tipus de transformació necessària per els canvis de punts de vista.	27

Figure 3.6: Resultats de l'aplicació de l'algorisme Harris-Affine sobre dues imatges consecutives d'una escena.	28
Figura 3.7: Esquema de la creació d'un descriptor per una regió d'interès.	29
Figura 3.8: Esquema SIFT, cada regió detectada de la imatge es representa amb un vector de 128 valors.	31
Figura 4.1: Esquema del funcionament intern del projecte.	33
Figura 4.2: Exemple d'extrapolació de moviment.	34
Figura 4.3: Finestra de tracking per una de les regions de la imatge	35
Figura 4.4: Exemple de clustering jeràrquic[12].	37
Figura 4.5: Exemple de dues iteracions consecutives del mètode 'fuzzy c-means'	39
Figure 4.6: Estat inicial i primera iteració de l'algorisme K-means[18].	40
Figura 4.7: Segona i tercera, i última, iteració de l'algorisme K-means[18].	41
Figura 4.8: Visió global de la classificació de les noves regions.	42
Figura 5.1: Es mostra una idea general de com podria ser una estructuració basada en objectes avaluant les nostres necessitats.	45
Figura 5.2: Exemple de la creació d'una cel·la punt.	46
Figura 5.3: Exemple de creació d'una simple estructura.	46
Figura 5.4: Es mostra com obtenir els atributs d'una estructura	47
Figura 5.5: Exemple de com crear una taula d'estructures i de com obtenir els valors dels objectes en una determinada posició.	47
Figura 5.6: Esquema de l'estructura plantejada a la figura 5.5.	48
Figura 5.7: Esquema de l'estructura de dades global.	49
Figura 5.8: Exemple il·lustrat d'una possible situació de classificació de dos grups.	51
Figura 6.1: Comparació entre els detectors Harris-Affine i MSER a la hora de trobar regions d'interès en dues imatges iguals però desplaçades.	53
Figura 6.2: Resultat obtingut per les seqüència d'imatges de la prova 1 amb el detector Harris-Affine.	54
Figura 6.3: Resultat de l'execució de l'algorisme en la seqüència d'imatges de la prova 1 amb el detector MSER.	55

ÍNDEX DE FIGURES	5
Figura 6.4: Resultat de la segona prova amb el detector Harris-Affine.	56
Figura 6.5: Resultat de la prova número 2 amb el detector MSER	57
Figura 6.6: En aquestes quatre imatges es mostra el punt més crític de la seqüència de la prova 3. Es fa servir el detector Harris-Affine en aquest exemple.	58
Figura 6.7: Punt crític de la seqüència d'imatges de la tercera prova. En aquest cas s'ha utilitzat el detector MSER amb millors resultats.	59
Figura 6.8: Es mostren els frames més significatius resultants de la prova "Groundhog's Day" 1 executada de tres maneres diferents.	61
Figura 6.9: Resultat de l'execució de la prova "Groundhog's Day" 2 amb el detector MSER, amb una mostra de cada 20 frames.	63
Figura 6.10: Frames inicial i final de l'escena de la seqüència "Star Wars: Episoi 1" 1.	64
Figure 6.11: Resultat de l'execució de la seqüència "Star Wars: Episodi 1" 1 amb el detector MSER.	65
Figura 6.12: Frames inicial i final de l'escena de la seqüència "Star Wars: Episodi 1" 2.	66
Figura 6.13: Resultat de l'execució de la prova "Star Wars: Episodi 1" 2 amb el detector MSER.	67
Figura 6.14: Seqüència de la pel·lícula "El tigre y la nieve"	68
Figura 6.15: Resultat de la prova de la seqüència de la pel·lícula "El tigre y la nieve"	69
Figura 6.16: Escena de la seqüència de la prova corresponent a "Me llamo Earl"	70
Figura 6.17: Resultats de la prova sobre la seqüència d'una escena de la sèrie "Me llamo Earl"	71

Capítol 1

Introducció

Aquest projecte es realitza dins el grup de Visió per Computador i Robòtica del departament d'Electrònica Informàtica i Automàtica de la Universitat de Girona, i s'emmarca en la línia de recerca que estudia la segmentació i el reconeixement d'objectes dins d'un vídeo.

1.1 Orígens i dificultats

En la branca de Visió per computador i Robòtica, es treballa amb la segmentació d'objectes en imatges des de fa un temps ja i es van donant passos importants[19]. Segmentar consisteix en separar els diferents objectes d'una imatge. Depenen del tipus de segmentació, obtindrem uns objectes o uns altres. Per exemple, segmentant per contorns o per colors. D'aquesta manera obtindríem o tots els objectes que considerem separats, inclòs algun que sabem que és el mateix però té un contorn molt definit dins seu, o grups d'objectes centrats en el seu color.

Així doncs si disposem d'una seqüència d'imatges o d'un vídeo, podem aprofitar que tenim una dimensió més per segmentar, el moviment. Això no significa que ens haguem de guiar exclusivament per aquest moviment, encara que en aquest projecte si que es faci així. Podem doncs aprofitar aquest moviment, afegir aquesta dimensió de més, i segmentar d'una manera més acurada i robusta.

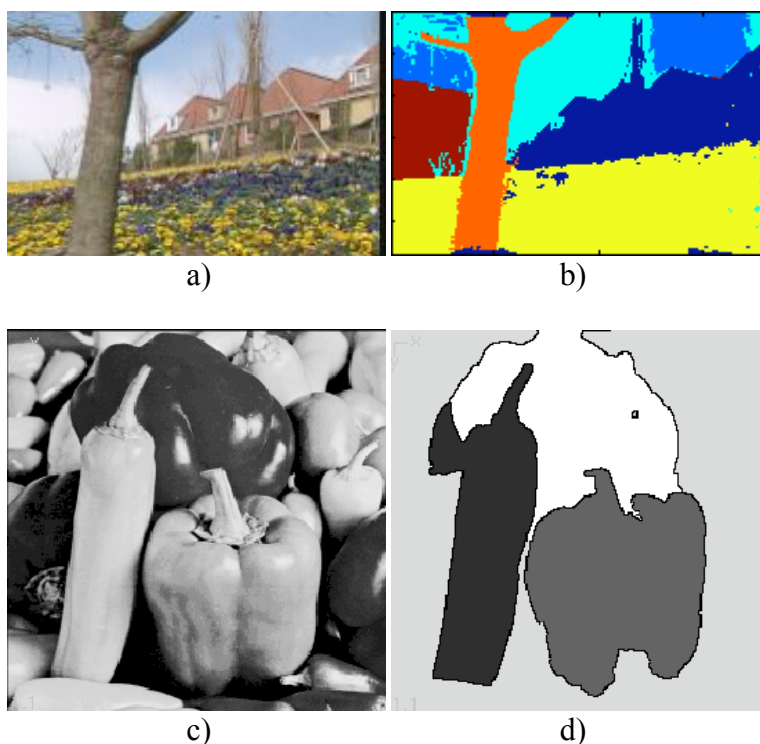


Figura 1.1: Exemple de segmentació d'imatges. a)[2] i c)[3] són les imatges originals, b)[2] és la imatge segmentada a través del color i d)[3] és la imatge segmentada a través del contorn.

Segmentar en un vídeo no és fàcil, ja que hi ha diversos problemes que hem de tenir en compte. Els objectes de la imatge poden variar la seva aparença, degut a la il·luminació i a la oclusió total o parcial d'aquests. La manera de segmentar per moviment es farà a través de regions d'interès. Aquestes seran àrees de la imatge que destacaran certes característiques. Per tant, es veurà afectat d'una manera determinada per els diferents problemes clàssics en la segmentació d'imatges.

- **Oclusió:** Quan un objecte que s'està seguint en una escena s'amaga darrere d'un altre objecte, encara que sigui parcialment, les seves regions d'interès es redueixen o desapareixen. En el cas de que només sigui parcialment, encara podrem seguir identificant l'objecte. Si la oclusió és total, ja no podrem identificar-lo fins que no es torni a mostrar alguna de les seves parts.
- **Variació del punt de vista:** és prudent pensar que si estem segmentant els objectes que es mouen, el punt de vista d'aquests ens pot portar problemes. Si un objecte es gira en l'escena, canviarà la seva forma. Aquesta és una qüestió que ens pot portar problemes només fins a un cert punt. Si un objecte canvia de tamany, ho farà progressivament, permetent la adaptació de les regions detectades a cada frame. Si per exemple hi ha un canvi de punt de vista molt sobtat, es molt possible que no siguem capaços de reconèixer les regions d'interès del frame anterior a l'actual, ja que seran diferents.
- **Il·luminació:** En un vídeo és freqüent que els objectes canviïn d'il·luminació, sobretot en escenes diferents. Això no representa un problema, ja que l'estudi

del moviment es farà per escenes, com s'explica posteriorment. Si hi ha algun canvi d'il·luminació en la mateixa escena, serà progressiu, així que frame a frame les regions d'interès s'adaptaran a aquest.

1.2 Objectius

L'objectiu d'aquest PFC és estudiar la branca de la detecció d'objectes en vídeos segons el seu moviment. Per fer-ho es crearà un algorisme que sigui capaç de tractar un vídeo, separant els objectes de l'escena i obtenint quina és la posició de cada un d'aquests.

Entrada: Seqüència d'entrada



PFC
Segmentació per
moviment



Sortida: Seqüència segmentada

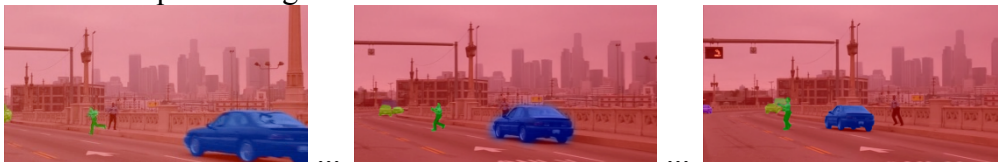


Figura 1.2: Esquema general del funcionament del PFC.

A la figura 1.2 podem veure un esquema de l'objectiu d'aquest projecte. Les imatges de sortida són segmentacions ideals. En elles es destaquen fins a quatre objectes i el fons. Un és el cotxe blau. Aquest es mou en una direcció diagonal esquerra. La

persona pintada de verd s'està movent en direcció esquerra i els cotxes grocs del fons s'estan movent fent una corba a la dreta, seguint la carrera. A més, apareix un últim cotxe pintat de lila. La persona que està pintada de color vermell no s'està movent. Això ens dona un conjunt de cinc objectes que es poden anar analitzant en el transcurs de l'escena.

Tal i com s'ha explicat, això és una representació d'un resultat ideal, el moviment ens pot ajudar a distingir els objectes, de tal manera que si els dos cotxes de l'esquerra pintats de grocs fossin, per exemple, un cotxe estirant un remolc, al estar separats una segmentació normal podria dir que són dos objectes diferents quan en realitat és un de sol. Gràcies a que tenen el mateix moviment en la escena podríem identificar-los com a un sol objecte.

1.3 Abast

La finalitat d'aquest projecte és l'estudi de la detecció d'objectes en vídeo. Intentarem crear un algorisme que ens permeti dur a terme aquest estudi i treure'n conclusions. Pretenem fer un algorisme, o un conjunt d'algorismes, que sigui capaç de donar qualsevol vídeo, pugui retornar un conjunt de imatges, o un vídeo, amb els diferents objectes de l'escena destacats. Es faran proves en diferents situacions, des de objectes sintètics amb un moviment clarament definit, fins a proves en seqüències reals extretes de diferents pel·lícules. Per últim es pretén comprovar l'eficiència d'aquest.

1.4 Aplicacions

Aquest projecte, per si sol, no té moltes aplicacions en la detecció robusta d'objectes. Però és una part molt important a l'hora de reconèixer-los. Cal tenir en compte que estudiar el moviment d'aquests és indispensable per poder evolucionar en l'estudi d'aquesta branca. Així doncs la idea i l'estudi d'aquest projecte pot esdevenir una part molt important a l'hora de seguir estudiant i construir futures implementacions del reconeixement d'objectes en vídeos.

1.5 Eines de desenvolupament

Ja que el projecte s'emmarca en la línia de recerca de robòtica i visió per computador, la tasca principal serà la manipulació d'imatges. Per tant farem servir el *Matlab*, ja que les imatges no són res més que matrius i aquest programa permet el càlcul vectorial i matricial d'una manera senzilla i realment eficient.

Matlab és una de les eines més utilitzades en la investigació i en les universitats, així com també en la UdG. Cal recordar que en aquest projecte no es pretén fer un programa final i tancat que funcioni segons uns requeriments, si no que es pretén investigar. Per tant considerem que el *Matlab* és la eina més adequada per realitzar el projecte de fi de carrera i intentar assolir els nostres objectius.

1.6 Planificació del projecte

La planificació d'aquest projecte s'ha preparat per treballar-hi un any i mig, degut a que estic treballant a mitja jornada. A continuació exposarem les etapes que s'haurien de seguir:

- Primera etapa: Primera presa de contacte amb el projecte.
- Segona etapa: Documentació, cerca de informació de la primera part del projecte. Visita de les pàgines web proporcionades.
- Tercera etapa: Anàlisi i disseny de l'algorisme de divisió de vídeos.
- Quarta etapa: implementació de l'algorisme de divisió de vídeos.
- Cinquena etapa: Documentació, lectura i cerca d'informació sobre detectors i descriptors de regions d'interès. *Harris-Affine*, *MSER* i *SIFT*.
- Sisena etapa: aplicació i experimentació dels detectors i descriptors de regions.
- Setena etapa: anàlisi i disseny de l'algorisme de *tracking*.
- Vuitena etapa: implementació de l'algorisme de *tracking*.
- Novena etapa: proves diverses amb tot el que s'ha fet fins al punt.
- Desena etapa: anàlisi de resultats

Durant el transcurs de totes aquestes etapes s'anirà documentant la memòria.

1.7 Organització del projecte

Capítol 1: Introducció

En aquest capítol hem introduït el projecte, així com esta estructurat, quins són els objectius que ens hem marcat i quin és el seu abast. També es fa una petita explicació de quines parts té la documentació del projecte i què es fa en cada una d'elles, per tenir una visió global de l'estructura d'aquesta.

Capítol 2: Shots

En aquesta part expliquem com separar un vídeo en un conjunt d'escenes que per si soles aportin suficient informació, d'aquestes escenes en direm *shots*. Exposarem també de quina manera podem obtenir-los i com els classificarem. Així com, també expliquem tota la informació necessària per entendre els conceptes i les bases del seu funcionament.

Capítol 3: Detecció i descripció de regions d'interès

Aquí explicarem l'algorisme que hem utilitzat per trobar les regions d'interès de les imatges. També explicarem l'algorisme per descriure aquestes regions i una introducció a aquests algorismes, explicant una mica la seva evolució i per que hem

triat els que utilitzem.

Capítol 4: Anàlisi i disseny: Reconeixement de les regions d'interès

En aquest capítol explicarem el disseny global de l'algorisme. Un cop hem detectat les regions d'interès d'una imatge, explicarem com ho fem per reconèixer-les com a objectes, com les agruparem i com sabrem quines regions ens aporten més o menys informació. També explicarem doncs en detall els fonaments dels mètodes que utilitzem per obtenir informació de les regions. També explicarem com agrupar-les, i proposarem un mètode propi.

Capítol 5: Implementació

En aquest tema explicarem quin entorn de programació hem utilitzat per implementar el projecte, de quines eines disposa i com s'ha dut a terme cada una de les parts de la implementació. També s'explica breument quins problemes han anat sorgint i què s'ha fet per solucionar-los.

Capítol 6: Anàlisi de resultats

En aquest capítol exposarem els resultats de les diferents proves que hem fet amb l'algorisme. També s'explicarà quins problemes hi ha agut en cada cas i com es pot fer per solucionar-los o reduir-ne l'efecte.

Capítol 7: Conclusions i treballs futurs

Finalment exposarem quina opinió ens ha deixat el projecte i quines conclusions en traiem. A més també parlarem de les futures millores i modificacions que es poden implementar en el nostre projecte.

Capítol 2

Shots

2.1 Extracció dels frames

El primer pas, consisteix en separar el vídeo en imatges. D'aquest procés se'n diu extracció de frames. Cal separar el vídeo en imatges en algun format comprimit que no perdi detall.

Un exemple d'imatge comprimida és el JPG (Joint Photographic Experts Group). Les imatges en aquest format es creen amb un algorisme que les comprimeix amb 24 bits de profunditat o en escala de grisos. Aquest format és un dels més utilitzats, ja que és un dels que comprimeix millor les imatges i per tant ocupen menys. L'inconvenient és que la compressió és amb pèrdua. Això significa que si es descomprimeix la imatge no queda tal i com estava abans de comprimir-la.

El format d'imatge BMP (Bitmap) tampoc ens servirà en aquest cas ja que és un format sense compressió i per tant seria més costós per treballar-hi.

El format que hem escollit per separar el vídeo en imatges és el PNG (Portable Network Graphics). Aquest format és el que millor comprimeix la imatge sense pèrdua. Serà molt important per els processos de separació de shots (histograma de color) i extracció de regions d'interès que la imatge tingui molta qualitat.

Les últimes versions del *VirtualDub*[13] ja permeten separar un vídeo en imatges amb format PNG així que aquest és el programa que hem escollit per fer el procés de extracció de frames.

Un punt important a tenir en compte és que no cal extreure TOTS els frames d'un vídeo. Podem extreure una imatge cada 'x' frames. Al ser un algorisme que funciona a través del moviment dels objectes si la 'x' és massa gran pot ser que no funcioni correctament.

2.2 Divisió del vídeo en shots

Un shot és un conjunt d'imatges dins d'un vídeo que formen una seqüència en un cert període de temps. Normalment un shot està filmat amb una sola càmera, ja que en un canvi de càmera en una pel·lícula es considera un altre shot. Encara que hi ha excepcions. Per exemple la pel·lícula *Màtrix*, que té algunes escenes filmades amb dues càmeres de vídeo i moltes càmeres fotogràfiques.

En aquest projecte és molt important aquest procés per el bon funcionament, si es calcula el moviment d'un objecte quan hi ha un canvi de shot, el més probable és que l'objecte en el següent shot ja no estigui al mateix lloc. En aquest cas l'algorisme perdria l'objecte.

Tal i com s'explica a l'apartat 2.2.4, per separar un vídeo en escenes de manera automàtica crearem per cada imatge el seu histograma de color i el compararem amb l'histograma de la imatge següent. Dues imatges consecutives de la mateixa escena tindran histogrames molt semblants, mentre que dues imatges consecutives que pertanyin a dos shots diferents, els seus histogrames seran diferents. Quan això passi es detecta un canvi d'escena.

L'espai de color que s'ha escollit per fer els histogrames és HSV. A continuació s'explica com són els espais de color HSV i RGB i perquè s'ha escollit el primer quan el més comú és el segon.

2.2.1 L'espai de color RGB

La descripció **RGB** (de l'anglès *Red*, *Green*, *Blue*, vermell, verd i blau) d'un color fa referència a la composició del color en termes de les intensitats dels colors primaris amb què es forma: del vermell (*red*), del verd (*green*) i del blau (*blue*). És un model de color basat en la síntesi additiva, amb el que és possible representar un color mitjançant la barreja per addició dels tres colors llum primaris.

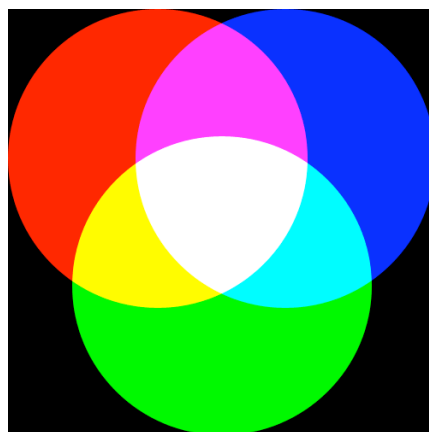


Figura2.1: Model de color additiu[12].

Cal indicar que el model de color RGB, no defineix per si mateix que significa exactament "vermell", "verd" o "blau", pel que els mateixos valors RGB poden mostrar colors notablement diferents en diversos dispositius que emprin aquest model de color. Encara que utilitzin un mateix model de color, els seus espais de color poden variar considerablement.

Per a indicar amb quina proporció barregem cada color, s'assigna un valor a cadascun dels colors primaris de manera que el valor 0 significa que no intervé en la barreja. Com més gran sigui aquest valor s'entén que aporta més intensitat a la barreja. Encara que l'interval de valors podria ser qualsevol (valors reals entre 0 i 1, valors sencers entre 0 i 37, etc), és bastant freqüent que cada color primari es codifiqui amb un byte (8 bits). Així, d'una manera estàndard, la intensitat amb que lluu cada component es mesura segons una escala que va del 0 al 255.

Per tant el vermell és (255, 0, 0), el verd és (0, 255, 0) i el blau (0, 0, 255). L'absència de color - el que impròpiament anomenem color negre - és (0, 0, 0). I el blanc, format pels tres colors primaris a nivell màxim, (255, 255, 255).

El conjunt de tots els colors es pot representar en forma de cub; cada color és un punt de la superfície o de l'interior d'aquest. L'escala de grisos és la diagonal que uneix el blanc i el negre.

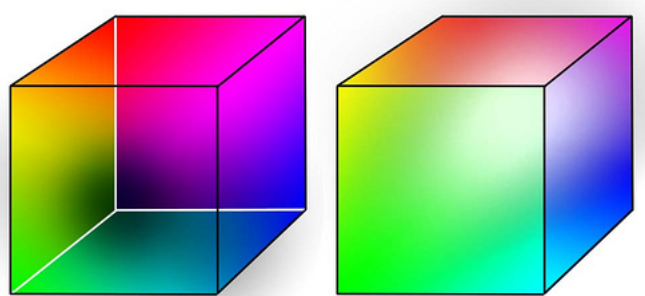


Figura2.2: Representació de l'espai de color RGB en forma de cub. Cada punt a la superfície i l'interior és un possible color. L'escala de grisos està representada a la diagonal.

Un dels problemes d'aquest model és que no permet representar tots els colors que els humans podem percebre, però el problema principal és que la distància perceptual i la distància lineal no son equivalents. Colors que varien lleugerament la intensitat entre ells poden estar a una distància llarga. Per això s'ha decidit no utilitzar aquest model de colors i fer servir el HSV.

Variació de la intensitat

100, 50, 0	#643200	Marró fosc
200, 100, 0	#c86400	Marró
150, 75, 0	#964b00	Marró clar

Figura 2.3: Exemple de la variació d'intensitat d'un color a l'espai RGB[12].
 Veiem que poca variació en la percepció s'expressa de diferents maneres en el vector RGB.

2.2.2 L'espai de color HSV

El model **HSV** (de l'anglès *Hue, Saturation, Value* – Tonalitat, Saturació, Valor), també anomenat **HSB** (*Hue, Saturation, Brightness* – Tonalitat, Saturació, Lluminositat), defineix un model de color en termes dels seus components constituents en coordenades cilíndriques:

- Tonalitat: és el tipus de color (vermell, blau o groc). Es representa com angle els valors del qual poden anar de 0 a 360°. Cada valor correspon a un color. Per exemple: 0 és vermell, 60 és groc i 120 és verd.
- Saturació: es representa com la distància a l'eix central, blanc-negre. Els valors possibles van del 0 al 100%. A aquest paràmetre també se l'anomena "puresa". Quan menor sigui la saturació d'un color, major serà la tonalitat grisenc i estarà més descolorit.
- Valor del color: és els lluminositat del color. Representa la alçada en l'eix blanc-negre. Els valors possibles van del 0 al 100%. 0 sempre és negre. Depenent de la saturació, 100 podria ser blanc o un color més o menys saturat.

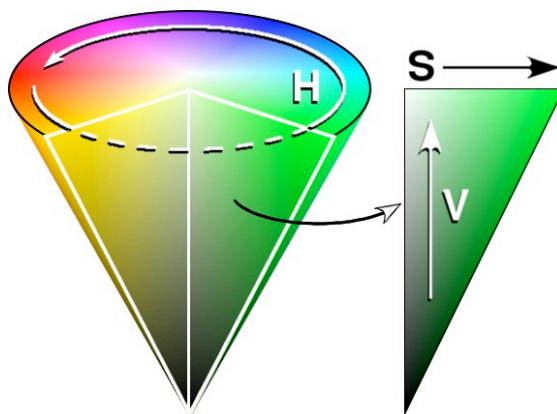
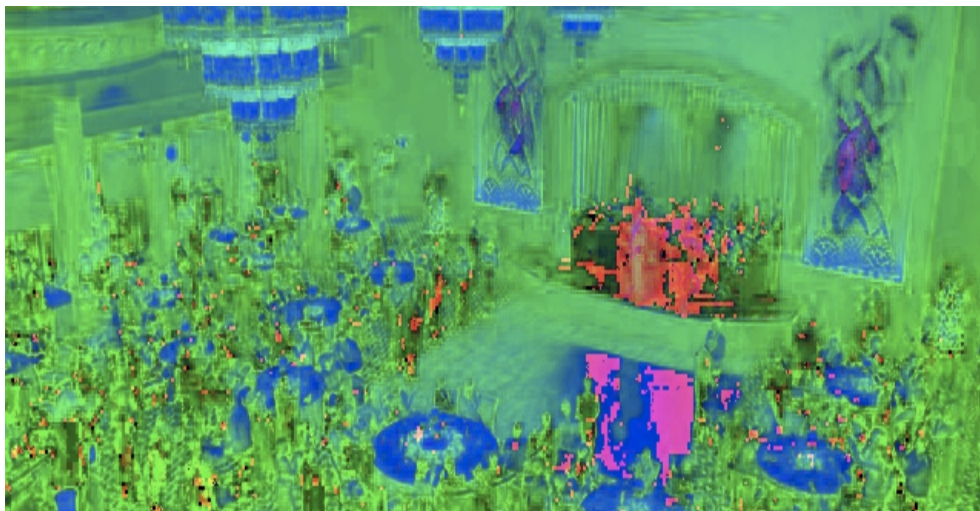


Figura 2.4: Representació del model de color HSV[12].

Així doncs el model de color HSV ens servirà millor que el RGB ja que es una representació més acurada de la percepció humana del color i així si dos colors son pròxims segons la vista, també ho seran en distància segons el gràfic de l'espai de color.



(a)



(b)

Figura 2.5: Diferencia entre espais de color. En l'apartat (a) podem veure una imatge d'una pel·lícula amb l'espai de color RGB, mentre que en la figura (b) es mostra la mateixa imatge transformada a l'espai HSV.

2.2.3 Histograma de color

En estadística, un histograma és una representació gràfica d'una variable en forma de barres, on la superfície de cada barra és proporcional a la freqüència dels valors representats. A l'eix vertical es representen les freqüències, i a l'eix horitzontal els valors de les variables.

Això significa que per cada variable, representada en l'eix horitzontal, en l'eix vertical se'n mostra el nombre de vegades que apareix. En el nostre cas, no serà més que un vector amb la freqüència que apareixen els colors a la imatge.

En el cas d'un histograma d'una imatge, tindrem tants histogrames com dimensions tingui el model de color. Per tant en el nostre cas tindrem tres histogrames per cada una de les imatges, un per el component H, un altre per l' S i un altre per el V. Per tal de poder-hi treballar còmodament els haurem de desratitzar, agrupant els valors en grups. Aquestes agrupacions no són res més que la descretització en *bits*. Així que tindrem 64 bits per l' histograma H i l'S i 32 per l' V.

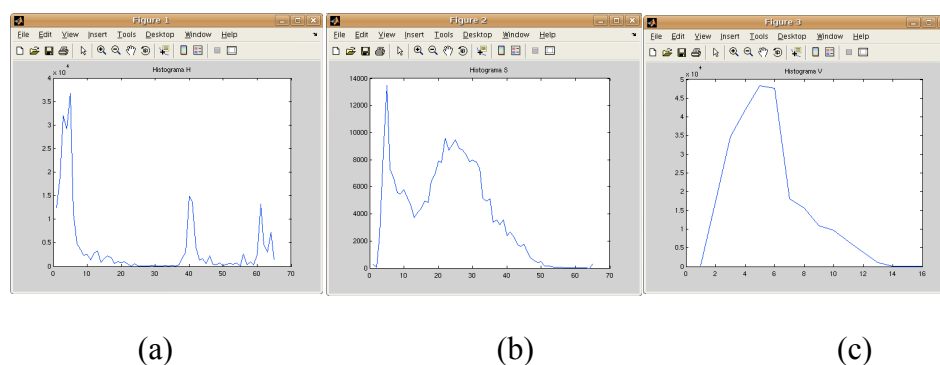


Figura 2.6: Histogrames de les components H,S i V d'una imatge, representades per (a),(b) i (c) respectivament

2.2.4 Algorisme de classificació per shots

Com hem explicat anteriorment, l'algorisme de classificació per shots separarà el vídeo en escenes. Això es farà de la següent manera.

Primer de tot passarem la imatge que volem tractar del model de color RGB al HSV per les raons ja explicades. A continuació crearem un histograma per cada component del model de color, o sigui un per la component H, un altre per la component S, i un últim per la component V. Llavors farem exactament el mateix per la imatge següent. Ara ja les tenim llestes per poder-les comparar. Això ho farem amb la distància

euclidiana entre cada una de les components de les dues imatges. Ens interessa tenir un únic valor representatiu així que sumarem els valors resultants de les distàncies obtingudes.

La formula de la distància euclidiana és la següent:

$$d(p, q) = \sqrt{\sum_{i=0}^n (p_i - q_i)^2} \quad (2.1)$$

A continuació, empíricament buscarem en el vídeo a on hi ha els canvis d'escena i així podrem trobar un llindar òptim, o sigui, quina és la distància que indica que comença un nou shot.



Figura 2.7: Seqüència en la que es veuen dos canvis d'escena.

En l'exemple que és mostra a la figura 2.7 podem veure que hi ha dos canvis d'escena. Per fer l'anàlisi empíric podem observar els valors de cada frame i comparar-los.

Frames	1	2	...	11	12	13	14	...	31	32	33
% Distància	11.5820			1.7380	43.6284	3.0708			38.4549	4.9052	

Figura 2.8: Taula per comparar la diferència entre frames.

Com es pot veure a la figura 2.9, en el vídeo hi ha dos canvis de shots. Un està situat entre els frames 12 i 13 i l'altre entre els frames 31 i 32. Veiem doncs que entre els frames 12 i 13 hi ha una diferència de un 43,6% i entre els frames 31 i 32 hi ha un 38,4%. El valor major entre dues imatges que no són canvi de shot és de 11,58%, per tant veiem que hi ha una diferència notable quan hi ha un canvi.

Cal destacar que aquestes proves s'han fet agafant quatre frames per segon del vídeo, si es fes a més frames per segon veuríem que els valors entre frames que no són canvi de shot, són més baixos.

En aquest cas un llindar vàlid és un 15%-20%. S'ha d'anar en compte a l'hora d'escollir un llindar, ja que si s'escull massa alt l'algorisme no detectarà tots els canvis d'escenes. Si per el contrari s'escull un llindar massa baix, detectarà canvis d'escena on no n'hi ha.

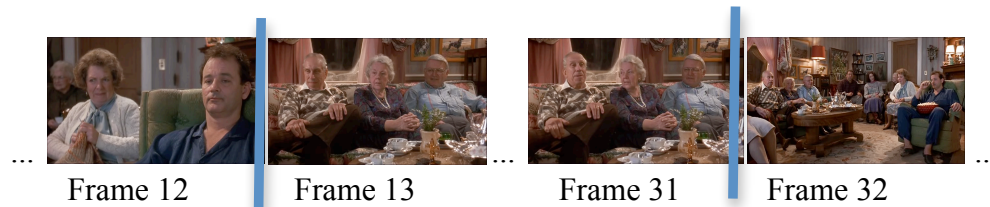


Figura 2.9: Resultat final de separar l'escena de la figura 2.7 en shots.

El resultat final de l'execució és una copia de cada imatge guardada en una estructura de carpetes. Es crea una carpeta per cada shot i es guarda dins una copia de les imatges que hi pertanyen. L'estructura final seria com la de la figura 2.10.

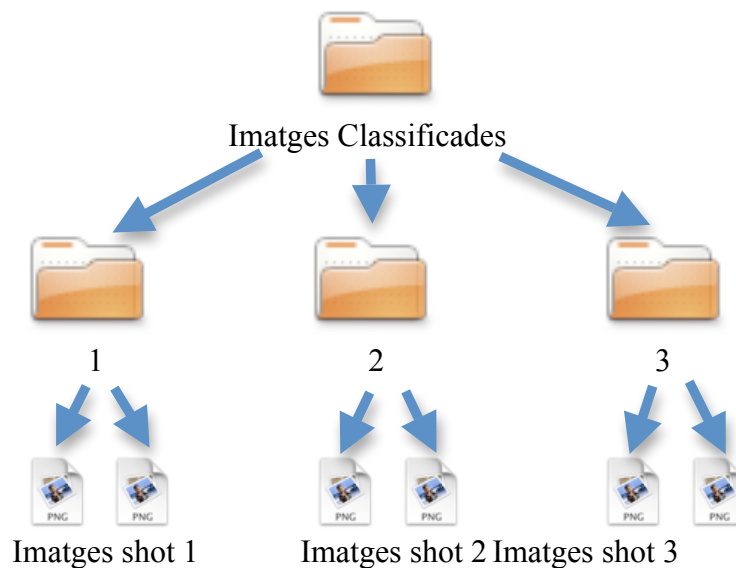


Figura 2.10: Estructura final de les imatges classificades en shots.

Capítol 3

Detecció i descripció de punts d'interès

Un punt d'interès és un punt en un imatge que té una posició ben definida i que pot ser detectat robustament. Així que un punt d'interès pot ser una cantonada, però també pot ser, per exemple, un punt aïllat d'intensitat màxima o mínima, el final d'una línia, o un punt de màxima inflexió d'una corba. A la pràctica, molts dels mètodes que s'anomenen detectors de cantonades, detecten punts d'interès generals i no només cantonades. Els detectors de cantonades normalment no són gaire robustos i sovint requereixen de supervisió o un gran nombre de redundàncies introduïdes per evitar que un error afecti a la detecció d'aquests punts.

La qualitat d'un detector de regions està basada en la capacitat de detectar el mateix punt en múltiples imatges similars però no idèntiques, com per exemple amb diferent il·luminació, translació, rotació o altres transformades. Una aproximació a la detecció de cantonades en una imatge és amb l'ús de la correlació, per el seu cost computacional és elevat.

Com es comenta a l'apartat de anàlisi de resultats, en aquest projecte s'ha començat utilitzant el mètode *Harris-Affine*[5] i finalment s'ha optat per el mètode *MSER*[5]. Per entendre com funcionen primer de tot s'explicarà el mètode de *Moravec*[1] que és el pas previ en la evolució d'aquests.

3.1 L'algorisme de detecció de cantonades

Moravec:

Aquest algorisme va ser desenvolupat per Hans P. Moravec al 1977 i es descriu a Moravec[1]. Aquest és un dels primers algorismes d'aquest tipus. El mètode es basa en trobar un grup de punts que tenen una gran variació de intensitat en diferents direccions.

Moravec va proposar mesurar la variació d'intensitat utilitzant una finestra (usualment de 3×3 , 5×5 , o 7×7 punts) centrada en el punt P i després desplaçant aquesta finestra un punt en les vuit direccions (horitzontalment, verticalment, i en les quatre diagonals). La variació d'intensitat donada per cada desplaçament es calcula amb la suma de quadrats de la diferència d'intensitats dels punts de les dues finestres. La figura 3.1 mostra aquest càlcul per un desplaçament diagonal per un punt negre aïllat (intensitat igual a 0) en un fons blanc (intensitat igual a 255) en una cantonada ideal. El requadre vermell indica la finestra original i el blau indica la desplaçada.

La variació d'intensitat en el punt P és la variació d'intensitat mínima calculada sobre les vuit direccions principals.

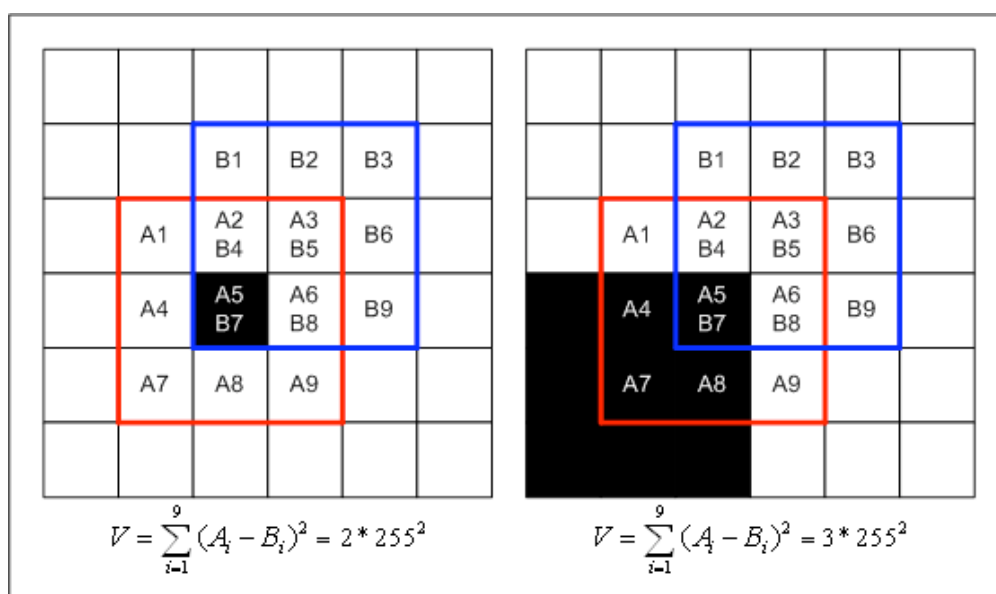


Figura 3.1: Càlcul de la variació d'intensitat amb una finestra de 3×3 en la direcció de la diagonal superior dreta[14].

L'exemple de perquè l'operador *Moravec* és un detector de cantonades el podem veure a la figura 3.2. Aquesta figura mostra finestres a els quatre tipus de posicions diferents. La posició A és l'interior d'un objecte (o del fons) on assumim que la intensitat de la imatge serà relativament constant dins la finestra, així que desplaçant-la, la variació d'intensitat serà mínima. Per una finestra situada en un contorn, com a la posició B, movent la finestra perpendicularment al contorn resultarà una gran variació d'intensitat. Però desplaçant-la al llarg d'aquesta només es detectarà una petita variació. Les dues posicions C i D, corresponent respectivament a una cantonada i a un punt aïllat, al desplaçar la finestra obtindrem una gran variació d'intensitat en qualsevol direcció. Això demostra que l'operador *Moravec* és un detector de cantonades, però que també és sensible als punts aïllats.

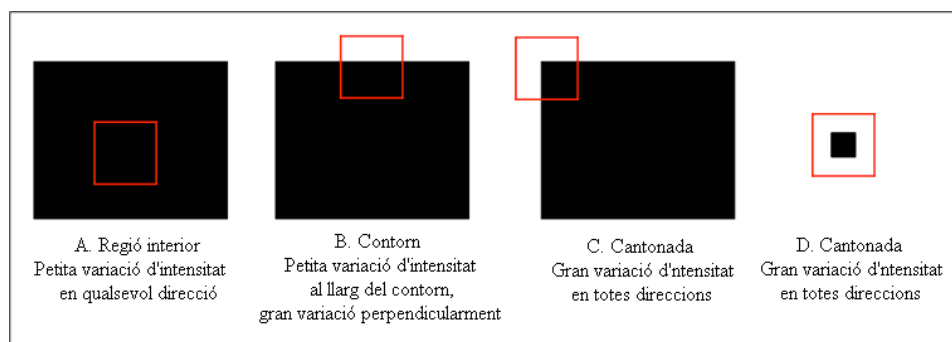


Figura 3.2: Diferents casos en que es pot trobar l'operador Moravec[14].

L'operador *Moravec* es pot utilitzar per donar a cada punt de la imatge un grau de cantonada. Aquesta mesura és el valor mínima de la intensitat del desplaçament en les vuit direccions. Aplicant *Moravec* a cada punt de la imatge crea un mapa de grau de cantonada (*cornerness map*). La figura 3.3 mostra un mapa de grau de cantonada per una imatge simple amb una finestra de 3x3. Aquest mapa destaca un quants punts importants de l'operador Moravec (el grau de cantonada de la figura 3.3 s'ha dividit per 255 al quadrat, així que el valor 2 representa en realitat $2 \cdot (255^2)$):

- 1 – Una cantonada és una variació local màxima.
- 2 – El grau de cantonada d'un punt aïllat i una cantonada són iguals
- 3 – A la regió exterior de la imatge no es pot aplicar l'operador, es representa amb una X.

X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
X	X	0	0	0	0	0	0	0	0	0	1	1	1	X	X
X	X	0	0	0	0	0	1	1	0	0	1	2	1	X	X
X	X	0	0	0	0	0	2	1	0	0	1	1	1	X	X
X	X	0	0	0	0	0	0	0	0	0	0	0	0	X	X
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Figura 3.3: Mapa de grau de cantonada resultant de l'algorisme Moravec sobre una imatge senzilla, on només hi ha intensitat 0 i 255[14].

El primer punt indica que les cantonades són màxims locals en el mapa de grau de cantonades. La màxima local es pot identificar utilitzant una supressió no-màxima. Encara que, això resultarà que trobarà cantonades com a punts aïllats. Per aquesta raó, l'operador *Moravec* és sensible al soroll. Utilitzant finestres majors fa l'algorisme

més robust al soroll, ja que la cantonada real tindrà una variació d'intensitat major que un punt aïllat. Però un punt aïllat encara serà un màxim local. Ja que cada punt se l'hi assigna un valor, és fàcil imaginar que en una imatge en escala de grisos hi haurà algunes màximes locals que no corresponen a cantonades. Això es pot solucionar aplicant un llindar. Seleccionar aquest llindar és una tasca complicada, ja que ha de ser prou alt com per evitar les falses cantonades, però prou baix com per detectar-ne el màxim possible de reals. Finalment els punts del voltant de la imatge, on l'operador *Moravec* no hi pot arribar, es poden ignorar posant el seu valor a 0.

Així doncs podríem descriure l'algorisme[14]*Moravec* de la següent manera:

Definim la intensitat d'un punt de la imatge a (x,y) amb $I(x,y)$.

Entrada: una imatge en escala de grisos, tamany de la finestra, llindar.

Sortida: Mapa amb la posició de les cantonades detectades.

- 1 – Per cada punt (x,y) de la imatge es calcula la variació d'intensitat per un desplaçament (u,v) com:

$$V_{u,v}(x,y) = \sum_{\forall a,b \text{ de la finestra}} (I(x+u+a, y+v+b) - I(x+a, y+b))^2 \quad (3.1)$$

Considerant els desplaçaments (u,v) com:

$$(1,0), (1,1), (0,1), (-1,1), (-1,0), (-1,-1), (0,-1), (1,-1)$$

- 2 – Construir el mapa de grau de cantonada calculant aquesta mesura $C(x,y)$ per cada punt (x,y):

$$C(x,y) = \min (V_{u,v}(x,y)) \quad (3.2)$$

- 3 – Aplicar el llindar al mapa posant tots els valors $C(x,y)$ per sota d'aquest a zero.

- 4 – Aplicar una supressió no-màxima per trobar les màximes locals.

Tots els punts resultants que no siguin zero, són cantonades.

3.2 L'algorisme de detecció de cantonades Harris-Stephens:

Harris i Stephens [4] van millorar l'algorisme de Moravec considerant directament la diferència de puntuació de la cantonada respecte a la direcció d'aquesta, enlloc d'utilitzar una finestra d'avaluació. Aquesta puntuació s'anomena, a vegades, autocorrelació, ja que aquest terme s'utilitza en el document on es descriu aquest detector. Encara que la manera en que s'explica a el document matemàticament és amb SSD, *Sum of Squared Difference*.

Per fer més amena l'explicació suposarem un imatge de dues dimensions i a escala de grisos. La imatge l'anomenarem I . Considerem una regió d'aquesta (u,v) i la desplacem cap a (x,y) . La suma de la diferència al quadrat entre les dues regions, l'anomenarem S , serà:

$$S(x,y) = \sum_u \sum_v w(u,v) (I(u,v) - I(u-x,v-y))^2 \quad (3.3)$$

La matriu de Harris, denominada A , es pot trobar a través de derivades de I amb $(x,y)=(0,0)$.

$$A = \sum_u \sum_v w(u,v) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = \begin{bmatrix} \langle I_x^2 \rangle & \langle I_x I_y \rangle \\ \langle I_x I_y \rangle & \langle I_y^2 \rangle \end{bmatrix}, \quad (3.4)$$

on els parèntesis angulars indiquen que s'ha de realitzar la mitjana (sumant sobre (u,v)), i s'utilitza la notació típica per derivades parcials.

La mesura d'una cantonada es fa amb l'estudi dels valors propis de A (λ_1, λ_2). Basant-nos en la magnitud d'aquests podem dir:

- 1 – Si $\lambda_1 \approx 0$ i $\lambda_2 \approx 0$ llavors no hi ha cap característica d'interès en el punt.
- 2 – Si $\lambda_1 \approx 0$ i λ_2 és un valor gran i positiu, llavors s'ha trobat un contorn.
- 3 – Si λ_1 i λ_2 són valors diferents, grans i positius, llavors s'ha trobat una cantonada.

Harris i Stephen es van adonar que el càlcul dels valors propis és costós, ja que requereix fer servir l'arrel quadrada. Llavors van suggerir la següent funció M_c , on κ és un paràmetre variable.

$$M_c = \lambda_1 \lambda_2 - \kappa (\lambda_1 + \lambda_2)^2 = \det(A) - \kappa \text{trace}^2(A) \quad (3.5)$$

El valor de κ s'ha de determinar empíricament, però el valor proposat per Harris està entre 0.04 i 0.15, on 0.04 és el valor agafat per defecte.

Per veure-ho d'una manera més senzilla, treballarem amb una imatge en escala de grisos. L'algorisme seria el següent[14]:

Entrada: Imatge en escala de grisos, variància gaussiana, valor de k, llindar

Sortida: Mapa amb la posició de cada punt detectat.

1 –Per cada punt (x,y) de la imatge, calcular la matriu d'autocorrelació M:

$$M = \begin{bmatrix} A & C \\ C & B \end{bmatrix}$$

$$\text{on: } A = \left(\frac{\partial I}{\partial x} \right)^2 \otimes w, B = \left(\frac{\partial I}{\partial y} \right)^2 \otimes w, C = \left(\frac{\partial I}{\partial x} \frac{\partial I}{\partial y} \right) \otimes w$$

(3.6)

$\otimes$ és l'operador de la convolució
 w és la finestra gaussiana

2 –Construir el mapa resultant final calculat el grau de cantonada $C(x,y)$ per cada punt (x,y)

$$C(x,y) = \det(M) - k(\text{trace}(M))^2 \quad (3.7)$$

$$\det(M) = \lambda_1 \lambda_2 = AB - C^2 \quad (3.8)$$

$$\text{trace}(M) = \lambda_1 + \lambda_2 = A + B \quad (3.9)$$

on k = constant

3 – Aplicar el llindar al mapa posant tots els valors $C(x,y)$ per sota d'aquest a zero.

4 – Aplicar una supressió no-màxima per trobar les màximes locals.

Tots els punts resultants que no siguin zero, són cantonades.

3.3 L'algorisme MSER

Una *Maximally Stable Extremal Region*[5] és un component connectat d'una imatge que se l'hi ha aplicat un llindar apropiat. La paraula “*Extremal*” es refereix a la propietat de que tots els píxels que estan dins de la regió tenen la més alta o la més baixa intensitat que tots els píxels en el seu voltant. “*Maximally stable*” descriu la propietat de la optimització en el procés de selecció del llindar.

El conjunt de regions extremes obtingudes en una imatge tenen un conjunt de propietats. Un canvi monòton de la intensitat de la imatge fa que el conjunt no variï, ja que depèn només de l'ordre de la intensitat dels punts que es manté en transformacions monòtones. Això assegura que canvis d'intensitat comuns modelats com a linears o afins, no afectin al conjunt de regions, encara que la càmera sigui no lineal (amb correcció de gamma). Finalment no hi ha més regions que píxels a la imatge. Així que un subconjunt definit de regions es manté sota una ampla varietat de canvis d'intensitat, de posició, orientació i escala, i a més, manté la seva cardinalitat.

El conjunt de regions extremes és molt eficient, quasi lineal al nombre de punts de la

imatge. La enumeració procedeix de la següent forma. Primer de tot, els punts s'ordenen per intensitat. Tot seguit, els punts es marquen a la imatge (en ordre creixent o decreixent) i la llista dels components connectats que creixen o s'ajunten i les seves àrees es mantenen utilitzant l'algorisme de unió-cerca [15]. Durant el procés d'enumeració, l'àrea de cada component connectat es guarda com una funció d'intensitat. Els components connectats que són extrems es consideren regions. Entre les regions extremes, les “*maximally stable*” són les corresponents als lindars que el canvi de l'àrea relativa en funció del llindar és mínim. En altres paraules, les *MSER* són parts de la imatge on la binarització local és estable sobre un gran rang de llindars. La definició de *MSER* basada en el canvi d'àrea relativa és invariant afi (tant per canvis d'intensitat o de geomètrics, de posició, orientació, o escala). Conseqüentment, el procés de detecció *MSER* és covariant afi.

La detecció de regions *MSER* està molt relacionada al llindar, ja que cada regió extrema és un component connectat d'una imatge amb llindar. Encara que es busca un llindar òptim o no global, es proven tots els llindars i s'avalua l'estabilitat de cada component connectat. Per algunes parts de la imatge hi poden haver múltiples llindars estables, i en aquest cas, es retorna un sistema de subgrups aniuats.

Finalment cal afegir que si es canvia la funció d'ordenació de punts, canviarien completament les regions extremes trobades.

El detector *MSER*[5] es farà servir en els experiments d'aquest projecte com s'explica a la part d'implementació.



Figura 3.4: Resultats de l'aplicació de l'algorisme *MSER* sobre dues imatges consecutives d'una escena.

3.4 L'operador Harris-Affine

Quan una imatge es veu des de varis punts de vista, no podem traçar un cercle al seu voltant de les mateixes característiques ja que no seria igual, no s'emmarcaria la mateixa porció de fons en cada punt de vista. Quan en lloc de traçar un cercle fem una

el·lipse, en direm afinitat. Com es pot veure a la figura 3.6.a i 3.6.b el fons que s'emmarca dins el cercle no és el mateix. Però si utilitzem una el·lipse, podem emmarcar el mateix tall de fons, com es veu a la figura 3.6.c. És per això que disposem del detectors afins.

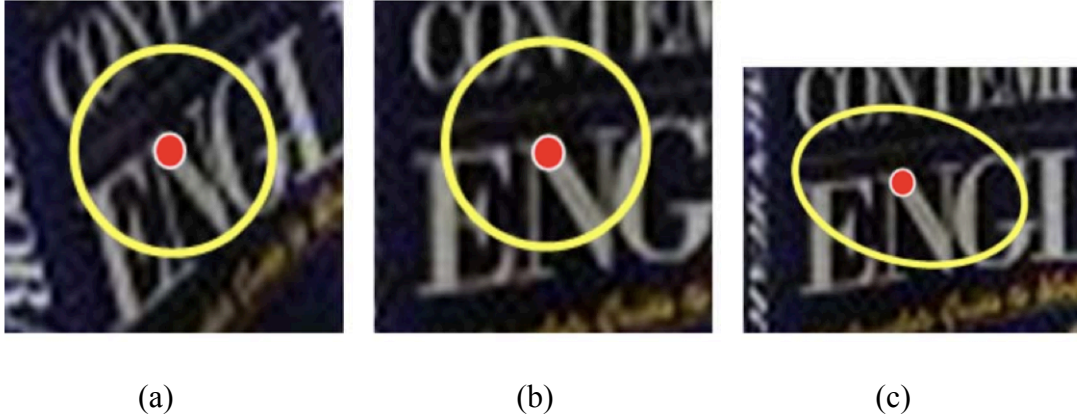


Figura 3.5: Tipus de transformació necessària per els canvis de punts de vista. En (a) es veu una regió amb un punt de vista. En (b) es veu la mateixa regió en un altre punt de vista, podem veure que el cercle no engloba el mateix tall de regió que en (a). En (c) podem veure com el descriptor de la regió afí en forma el·líptica compensa el canvi de punt de vista[5].

El detector afí *Harris* detecta regions d'interès en un espai escalar, basada en *Laplace*. Aquestes regions es representen amb una el·lipse que es calcula amb una matriu de segon moment de la intensitat gradient (Baumberg, 2000[6]; Lindeberg and Gårding, 1997[7]).

La matriu de segon moment també s'anomena matriu d'autocorrelació i s'utilitza freqüentment per a la detecció de característiques en les imatges. La següent matriu descriu la distribució del gradient en el veïnatge d'un punt:

$$\begin{aligned}
 M = \mu(\mathbf{x}, \sigma_I, \sigma_D) &= \begin{bmatrix} \mu_{11} & \mu_{12} \\ \mu_{21} & \mu_{22} \end{bmatrix} \\
 &= \sigma_D^2 g(\sigma_I) * \begin{bmatrix} I_x^2(\mathbf{x}, \sigma_D) & I_x I_y(\mathbf{x}, \sigma_D) \\ I_x I_y(\mathbf{x}, \sigma_D) & I_y^2(\mathbf{x}, \sigma_D) \end{bmatrix}
 \end{aligned} \tag{3.10}$$

Les derivades locals de la imatge es componen de nuclis gaussians de escala σ_D (escala de la diferència). Llavors es fan les mitjanes de les derivades dels punts veïns i es suavitzen amb una finestra gaussiana de escala σ_I (escala d'integració). Els valors propis d'aquesta matriu representen dos cavis principals de senyal en el veïnatge del punt. Aquesta propietat permet l'extracció de punts amb les dues curvatures significants, ja que el canvi de senyal és significant en les direccions ortogonals. Aquests punts són estables en les variacions d'il·luminació i per tant són representatius de la imatge.

El mètode troba la matriu que normalitza la regió trobada. Fa servir la matriu adaptativa U per transformar la imatge en un espai normalitzat. Els paràmetres del punt d'interès (posició i escales d'integració i diferencial) es refinen amb mètodes similars al detector *Harris-Laplace*. Aquest algorisme funciona de la següent manera:

- 1 – Es seleccionen els punts amb el detector *Harris-Stephens*.
- 2 – Es normalitza la regió, la primera vegada amb $U^{(0)}$.
- 3 – S'escull l'escala de integració que maximitza el laplacà del gaussià i, l'escala diferencial.
- 4 – Es calcula la forma de la regió amb la matriu de segon moment.
- 5 – Es normalitza la regió en forma el·líptica i s'actualitza la matriu U .
- 6 – Si els valors propis no són iguals es torna al punt 2.

Per veure una mica més clar el funcionament del mètode *Harris-Affine* a la figura 3.8 tenim un exemple.



Figura 3.6: Resultats de l'aplicació de l'algorisme Harris-Affine sobre dues imatges consecutives d'una escena.

3.5 SIFT (Scale-Invariant Feature Transform)

Les operacions anteriors han assignat una posició, una escala i una orientació per cada regió d'interès de la imatge. Aquests paràmetres imposen un sistema coordinat i repetible en dues dimensions el qual descriu les regions de la imatge i els proporciona invariància. El següent pas és calcular un descriptor per a cada regió de la imatge que sigui molt descriptiu, i tant invariable com sigui possible a les variacions d'il·luminació i canvi de perspectiva.

Una primera aproximació seria agafar les intensitats dels punts al voltant de la regió d'interès amb l'escala apropiada i relacionar-los amb una mesura de correlació

normalitzada. Encara que, la correlació simple de les regions d'interès de la imatge són molt sensibles al canvis de punt de vista o deformacions. Una millor implementació és la de *Edelman, Intrator, i Poggio*[8]. La seva proposta de representació està basada en un model de visió biològica, en particular, de neurones complexes en el còrtex visual primari. Aquestes neurones responen al gradient d'una certa orientació i espai freqüencial. Edelman *et al* van fer la hipòtesis que la funció d'aquestes neurones permetia relacionar i reconèixer objectes en 3D des de diferents punts de vista. Van fer proves detallades utilitzant models d'objectes i animals en 3D que mostren que relacionar amb gradients té millors resultats si hi ha canvis de posició que no si hi ha canvis en la rotació. Per exemple, la precisió de reconeixement d'un objecte en 3D rotat en profunditat 20 graus incrementa d'un 35% en correlació de gradients a un 90% amb el sistema de neurones. Lowe [9], es va inspirar en aquesta idea per crear el descriptor *SIFT*, però permet canvis de posició utilitzant un mecanisme de càlcul diferent.

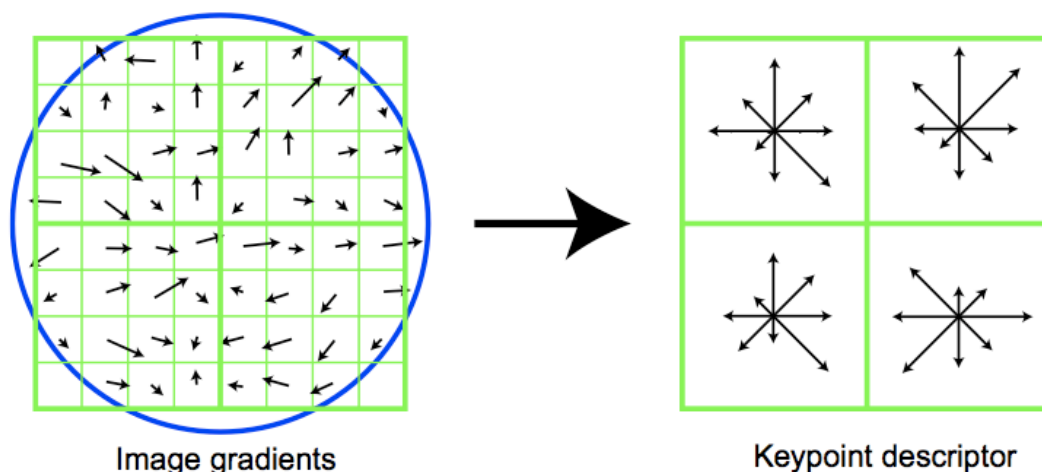


Figura 3.7: Esquema de la creació d'un descriptor per una regió d'interès.

En la figura 3.7 es crea un descriptor calculant la magnitud i la orientació del gradient per cada punt en una regió al voltant del punt d'interès, com es mostra a la part esquerra. A aquests se'ls assigna un pes amb una finestra gaussiana indicada amb el cercle blau de l'esquerra. Aquestes mostres s'acumulen en histogrames d'orientació resumint els continguts en 4x4 subregions com es mostra a la part dreta, amb l'allargada de cada fletxa corresponent a la suma dels gradients pròxims en aquella direcció dins la regió[9].

La figura 3.9 mostra el càlcul d'un descriptor. Primer de tot, es prenen mostres de la magnitud i orientació del gradient dels punts al voltant del punt d'interès, utilitzant l'escala d'aquest per seleccionar el nivell de difusió gaussiana, o "*gaussian blur*", de la imatge. Per aconseguir una orientació invariant, a les coordenades del descriptor i la orientació del gradient se'ls fa una rotació relativament amb la orientació del punt d'interès.

Per assignar un pes a la magnitud de cada punt de la mostra, es fa servir una funció gaussiana amb σ igual a la meitat de l'amplada de la finestra del descriptor. Aquesta

finestra està il·lustrada a la part esquerra de la figura 3.9. La funció d'aquesta finestra gaussiana és evitar canvis sobtats en el descriptor per petits canvis en la posició de la finestra, donant menys èmfasi al gradient que està més allunyat del centre d'aquesta.

El descriptor del punt d'interès es mostra a la part dreta del punt. La figura mostra vuit direccions per cada histograma d'orientació, amb la longitud de la fletxa corresponent a la magnitud d'aquest. Un gradient de la part esquerra pot canviar fins a quatre vegades de posició sense que variï l' de la part dreta, permetent així reconèixer punts d'interès amb cert canvi de posició.

El descriptor es forma d'un vector que conté totes les orientacions de les entrades dels histogrames, o sigui, la longitud de les fletxes com es mostra a la figura 3.9.

Tot seguit, el vector de característiques es modifica per reduir els efectes del canvi d'il·luminació. Primer de tot, es normalitza el vector. Un canvi en el contrast de la imatge, en el que cada píxel es multiplica per un valor constant, també multiplicarà el gradient per un valor constant, així el canvi de contrast es cancel·larà per la normalització del vector. Un canvi de brillantor, que suma una constant a cada píxel, no afectarà als valors dels gradients, ja que es calculen a partir de la diferència de píxels. Així que el descriptor és invariant a canvis afins en la il·luminació.

Per acabar, un cop s'aplica l'algorisme *SIFT* en un imatge, s'obté un vector de 128 posicions per cada una de les regions d'interès. Els resultats són tal i com es mostren a la figura 3.10.

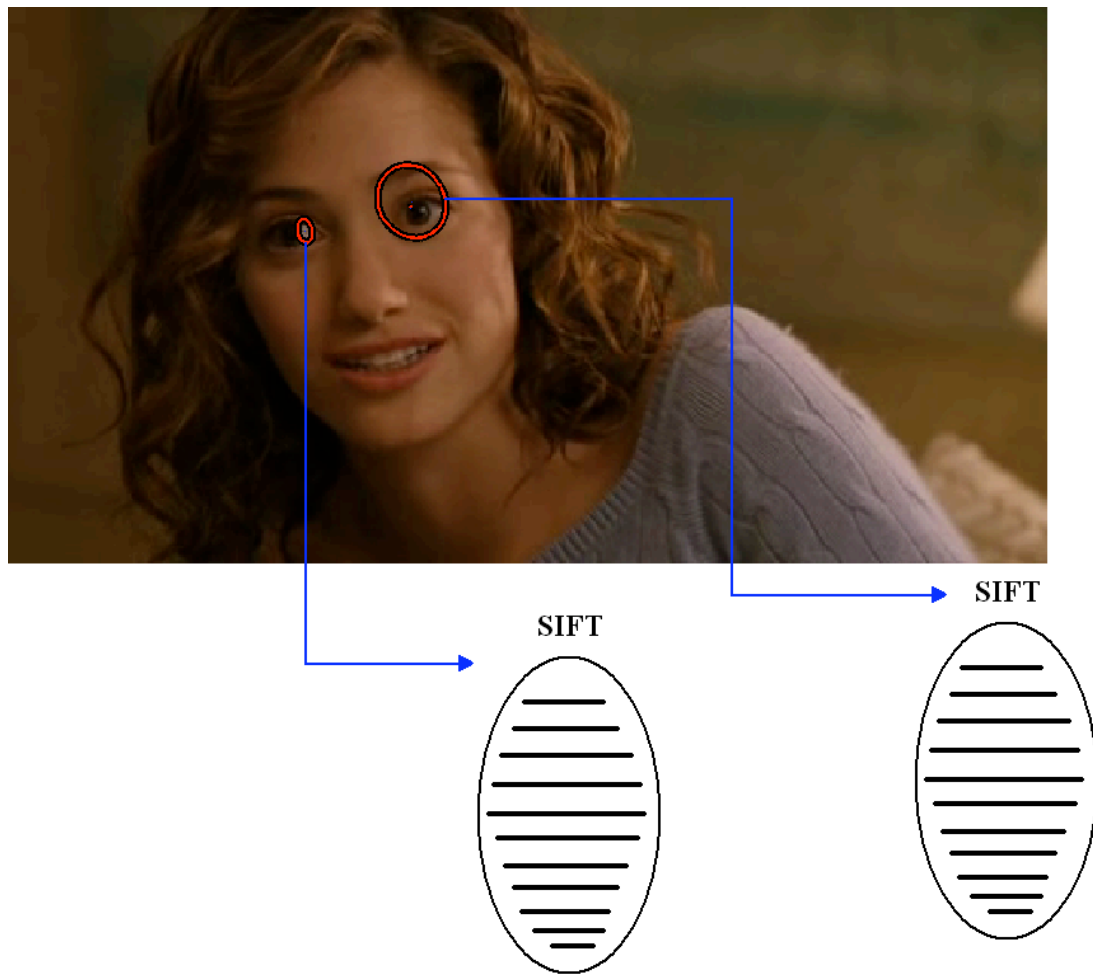


Figura 3.8: Esquema SIFT, cada regió detectada de la imatge es representa amb un vector de 128 valors.

Capítol 4

Anàlisi i disseny:

Reconeixement de les regions d'interès

Un cop extretes les regions d'interès de les imatges del vídeo i definides amb descriptors *SIFT*, cal fer un seguiment d'aquestes a través de les imatges consecutives del vídeo. Una regió que es mantingui en un seguit de fotogrames serà una part d'un objecte.

El reconeixement d'objectes es farà a través d'algorismes de “*clustering*”. Aquests algorismes, el que fan, és agrupar uns valors en un nombre determinat de grups. Aquests valors poden ser coordenades d'objectes, poden ser nivells d'intensitat de gris o colors(en un espai que permeti fer un bon “*clustering*”), o en el nostre cas, moviments.

Així doncs combinant aquestes dues parts podem saber com es mouen aquests punts en el vídeo i quins objectes componen. En aquest apartat parlarem de les diferents maneres de seguir aquests punts d'un fotograma al següent, i de com es poden agrupar per formar objectes.

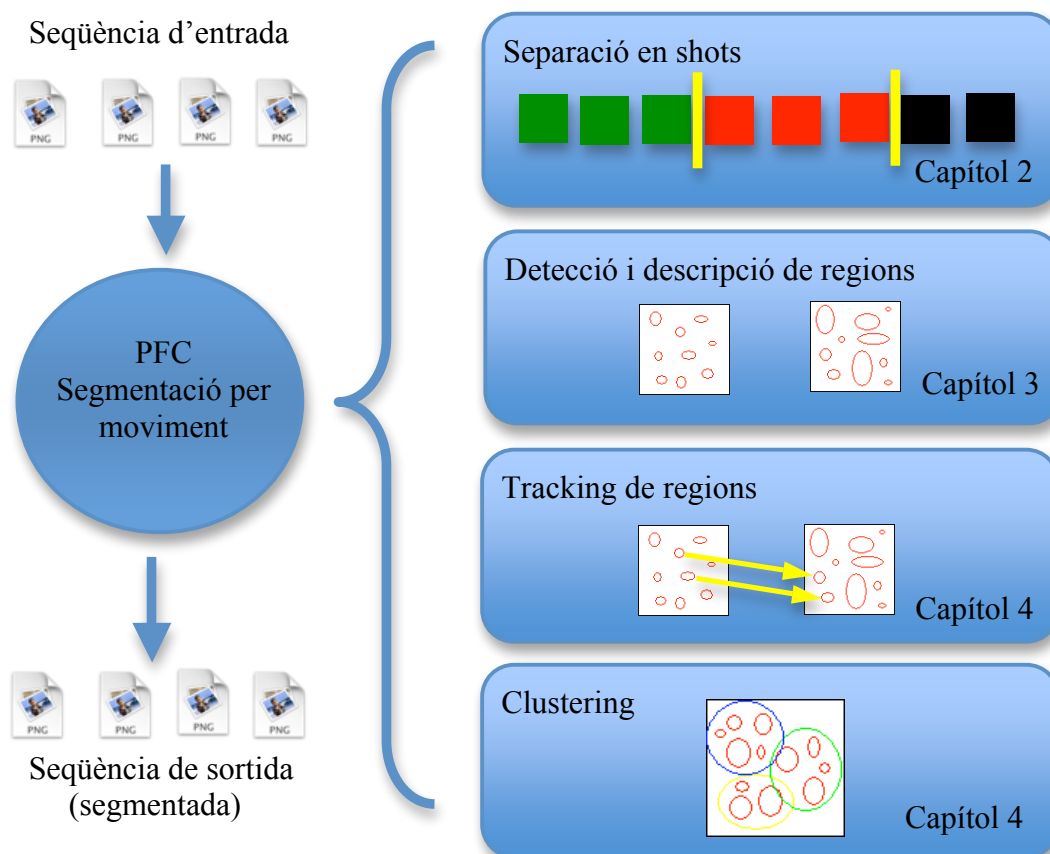


Figura 4.1: Esquema del funcionament intern del projecte.

A la figura 4.1 podem veure un esquema de com s'estructura el funcionament del projecte. Les quatre etapes més importants són les que es mostren a la part dreta de la figura. La primera és la separació del vídeo en shots, que hem explicat al tema 2, la segona és la detecció i la descripció de les regions d'interès amb els algorismes de *Harris-Affine*, *MSER* i *SIFT*, que hem explicat al tema 3. La tercera part és el *tracking* de les regions que hem obtingut i la quarta etapa és el *clustering*, l'agrupament d'aquestes regions. Aquestes dues últimes parts les explicarem a continuació.

4.1 Tracking

El *tracking* consisteix en seguir un objecte a través dels frames d'una escena. En aquest projecte el *tracking* és una de les parts més importants, un bon *tracking* fa molt més fiable la informació de com es mouen els objectes. D'aquests mètodes n'hi ha de diferents tipus, i la seva implementació és diferent en cada cas. En el nostre, hem optat per crear un mètode propi.

La idea general és analitzar els frames un per un i descriure com s'estan movent els objectes, ja que la visió general del projecte estaria enfocada a reconèixer els objectes en temps real. Una possible manera d'enfocar un algorisme de tracking és estudiar com ho fem les persones. Hi ha certs punts a tenir en compte que ens poden ajudar a implementar l'algorisme. Primer de tot quan nosaltres estem mirant un objecte que està parat i aquest es comença a moure, hi ha un petit temps de reacció en que

estudiem primer de tot, que està passant. Un cop el cervell ha processat que l'objecte que estàvem mirant s'està movent, comencem a seguir-lo amb la mirada. Un altre punt a tenir en compte és que nosaltres som capaços d'intuir on serà l'objecte que s'està movent d'aquí a un segon, ja que tenim en compte quin és el seu moviment actual, perquè sabem quina velocitat, acceleració, etc. té.



Figura 4.2: Exemple d'extrapolació de moviment.

En la figura 4.2 es pot veure el moviment que han fet els objectes amb una tonalitat més transparent, i de color groc quina serà la posició d'aquests en el següent frame.

Tenint en compte aquests factors, l'algorisme disposarà d'una petita memòria de com es mou cada punt, tant per millorar l'eficiència com per utilitzar aquesta informació en la següent fase. A més, l'algorisme funcionarà des del primer frame, però no donarem una resposta segura fins que no passi aquest temps de reacció del que hem parlat, hem optat per cinc frames.

L'algorisme compararà cada punt del frame anterior amb el frame actual per saber on és la nova posició d'aquest. A l'hora de comparar les seves característiques només es tindran en compte els punts del frame actual que estiguin a prop d'aquest. Això ho farem obrint una finestra de $p \times p$ al voltant de la posició on era el punt anterior en el frame actual. Ja que disposem d'una informació de com s'està movent el punt que estem tractant, aquesta finestra es pot obrir centrada en el punt, o bé desplaçada extrapolant la posició d'on serà aquest punt. Així és poden obrir finestres més petites i agilitzar l'algorisme de tracking.

Un cop hem decidit quins punts del frame actual poden ser el punt que estem tractant del frame anterior, cal comparar-los per saber quin d'ells es més semblant. Aquesta operació serà una distància euclidiana amb els 128 valors del descriptor *SIFT* de cada punt, així el que estigui més a prop serà el nou punt. Gràcies al càlcul vectorial del *Matlab* aquesta operació és viable, ja que sense ell, seria massa costós.

Ara ja disposem d'una estructura amb tots els punts del frame anterior situats en el frame actual, així que ja es pot tractar aquesta informació com es desitgi. Cal tenir en compte també que pot ser que per un punt del frame anterior no se'n trobi cap en l'actual, llavors direm que el punt s'ha perdut. Per els punts perduts també es perd la seva informació, o sigui el seu moviment. Però gràcies a que guardem la informació del moviment també podem afegir punts nous a la nostra estructura, però no tenir-los en compte per el clustering, per exemple, si no tenen un moviment suficientment acurat, per exemple un moviment emmagatzemat de tres frames enrere.

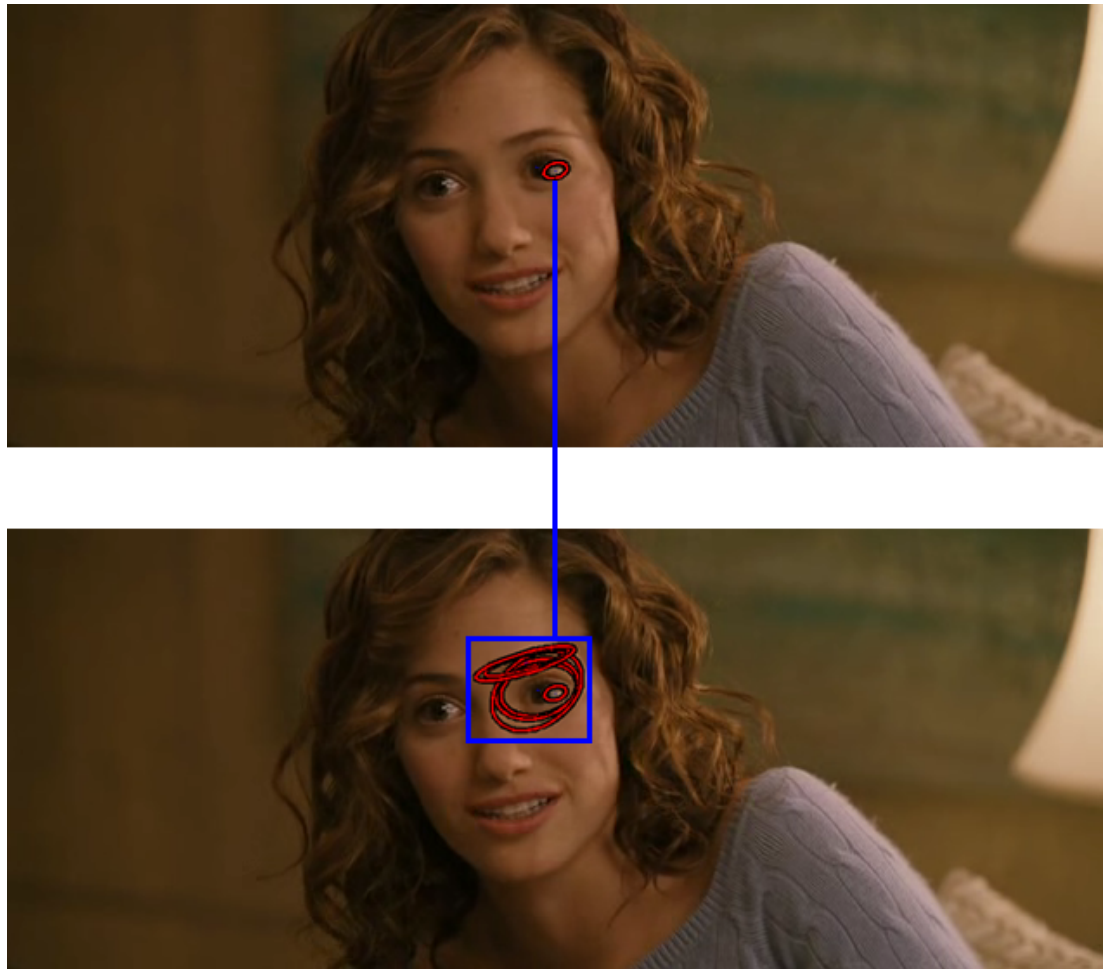


Figura 4.3: Finestra de tracking per una de les regions de la imatge

Els únics punts que es mostren a la figura 4.3 són els que tenen significança per l'exemple. Per el punt de la imatge superior s'obre una finestra de $p \times p$ com es veu a la imatge inferior. Es tractaran tots els punts que el seu centre caigui dins la finestra. En aquest cas els punts que es compararan són tots els que es mostren a la imatge inferior.

4.2 Clustering

El “*clustering*”[12] és la classificació d’objectes en diferents grups, o més precís, el particionament de grups de dades en subgrups (clúster), en que la informació de cada element del subgrup sigui comuna. El “*clustering*” de la informació és una tècnica per l’anàlisi estadístic utilitzat en molts camps, com l’aprenentatge de màquines, la mineria de dades, el reconeixement de patrons, l’anàlisi d’imatges o la bioinformàtica. La tasca d’agrupar objectes en k clústers, o grups, se’n diu *k-clustering*.

Els algorismes de “*clustering*” poden ser jeràrquics o particionals. Els algorismes jeràrquics funcionen troben clústers successivament utilitzant clústers prèviament establerts. Poden ser aglomeratius (“*bottom-up*”[22]) o divisius (“*top-down*”[21]). Els aglomeratius comencen amb cada element com un clúster separat i es van unint successivament en clústers majors. Els divisius funcionen a la inversa, comencen amb un sol clúster i es va separant en clústers més petits.

Els algorismes particionals normalment determinen els clústers tots de cop, a més, poden usar-se com a algorismes divisors en el clúster jeràrquic.

El clustering a dues bandes (*two-way clustering*), *co-clustering*, o *biclustering*, són mètodes on no només s’agrupen els objectes, si no també les característiques d’aquests. Per exemple, si les dades es representen en una matriu, s’agrupen simultàniament les files i les columnes.

Una distinció més entre els mètodes de clustering és si aquests utilitzen distàncies simètriques o asimètriques. Una propietat de l’espai euclidià és que les distàncies són simètriques, mentre que en altres aplicacions, això no és així. Per exemple en mètodes “*sequence-alignment*” (veure Prinzie & Van den Poel[16]).

Un pas important en el procés és la elecció d’un mesura de distància, la qual determina quan similar són dos elements. Això variarà la forma dels clústers, ja que uns elements poden estar a prop d’uns altres amb un tipus de distància, però lluny a través d’un altre tipus. Els diferents camps d’investigació fan servir varis tipus de mesures. D’aquestes, n’hi ha moltes, i les més comuns són:

- Distància euclidiana: també anomenada norma 2, és el mètode més comú en el clustering. La seva formula és $\sqrt{x^2+y^2}$
- Distància manhattan: també anomenada la norma de taxi o norma 1.
- Norma màxima: també anomenada norma uniforme
- Distància Mahalanobis: corregeix dades de diferents escales i correlacions de les variables.
- L’angle entre dos vectors: es pot fer servir de mesura en un clustering quan les dades tenen moltes dimensions[12].
- Distància Hamming: mesura el nombre mínim de substitucions requerides per canviar d’un membre a un altre.

En el nostre cas, el que s'ha de mesurar és el moviment de cada punt en el transcurs dels frames. L'espai és en dues dimensions i les dades seran vectors de dues dimensions amb la mitjana de velocitat d'una finestra de cinc frames. Així que la distància que hem escollit és la Distància Euclidiana.

4.2.1 Clustering jeràrquic

Els clústers jeràrquics construeixen (els aglomeratius) o trenquen (els divisius) una jerarquia de clústers. La representació tradicional d'aquesta jerarquia és un arbre, amb elements individuals a un extrem i un sol clúster contenint tots els elements a l'altre. Els algorismes aglomeratius comencen a les fulles de l'arbre, mentre que els divisius ho fan a l'arrel. Tallant l'arbre a una certa altura és el que ens donarà els clústers, depèn del nombre que en volem, es farà a una altura determinada.

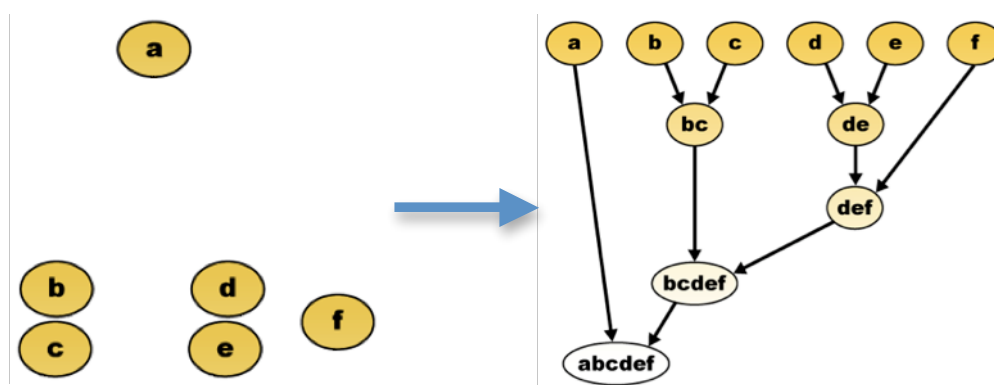


Figura 4.4: Exemple de clustering jeràrquic[12].

A la part esquerra de la figura 4.4 es mostren els elements a classificar. Els elements es classificaran segons la proximitat entre ells. A la part dreta es mostra el procés de classificació, l'arbre resultant. En el cas d'un algorisme aglomeratiu el procés es tal com es mostra a la part dreta, si es tractés d'un divisiu, el procés seria a l'inversa, es començaria per l'arrel (abcdef) i s'acabaria a les fulles (a,b,c,d,e,f).

4.2.2 Clustering particional

D'algorismes de clustering particional n'hi ha uns quans, però només entrarem en detall en dos. El clustering fuzzy c-means i el clustering k-means. Tant un com l'altre estan implementats en *Matlab* per defecte així que qualsevol de les dues opcions seria vàlida. En la fase de implementació s'explica perquè s'ha escollit el *K-means* per el projecte.

4.2.2.1 Fuzzy c-means

En el clustering difús[12], cada punt té un nivell de pertinència a un clúster, com a la lògica difusa, enlloc de pertànyer completament a un clúster. Els punts més allunyats del centre d'un clúster formaran part d'aquest en un menor grau que els punts que estan més a prop. Per cada punt x tenim un coeficient de pertinença a el ' k ' clúster $u_k(x)$. Normalment la suma d'aquests coeficients és 1.

$$\forall x \quad \sum_{k=1}^{\text{num. clusters}} u_k(x) = 1. \quad (4.1)$$

En el clustering difús, el centroide del clúster és a la posició de la mitjana de tots els punts, tenint en compte el seu grau de pertinença a aquest.

$$\text{center}_k = \frac{\sum_x u_k(x)^m x}{\sum_x u_k(x)^m}. \quad (4.2)$$

El grau de pertinença esta relacionat inversament amb la distància al clúster

$$u_k(x) = \frac{1}{d(\text{center}_k, x)}, \quad (4.3)$$

llavors a els coeficients se'ls aplica una normalització i una difusió amb el paràmetre real $m > 1$ perquè la seva suma sigui 1.

$$u_k(x) = \frac{1}{\sum_j \left(\frac{d(\text{center}_k, x)}{d(\text{center}_j, x)} \right)^{2/(m-1)}}. \quad (4.4)$$

Per $m = 2$, és equivalent a normalitzar el coeficient linearmet per fer la seva suma 1. Quan m s'acosta a 1, al centre del clúster més proper se li dona molt més pes i l'algorisme és similar al *K-means*.

L'algorisme de clustering difús és molt semblant al *K-means* que s'explica a continuació però canviant la manera d'avaluar els punts utilitzant lògica difusa.

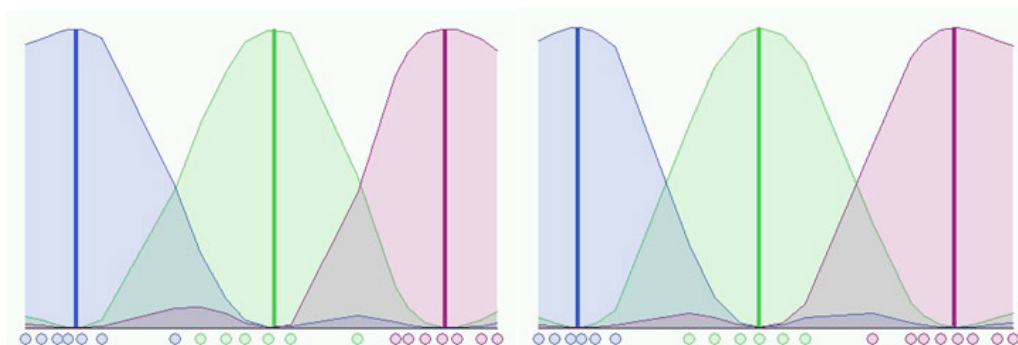


Figura 4.5: Exemple de dues iteracions consecutives del mètode 'fuzzy c-means'

En l'exemple de la figura 4.5 es pot veure un pas de l'algorisme de clustering difús. Primer de tot, els punts a classificar són les rodones de la part inferior de les imatges, punts en una sola dimensió. A la part esquerra podem veure la classificació dels punts al clúster corresponen del pas anterior. A la part dreta podem veure el reposicionament dels centroides i la nova classificació dels punts.

4.2.2.2 K-means

L'algorisme *K-means*[12] assigna a cada punt el clúster amb el centre (anomenat centroide) més proper. El centroide és la mitjana de tots els punts del clúster, que és la mitjana aritmètica de totes les dimensions dels punts. Per exemple: Les dades tenen dues dimensions i el clúster té dos punts $X = (x_1, x_2, x_3)$ i $Y = (y_1, y_2, y_3)$. Llavors el centroide Z serà $Z = (z_1, z_2, z_3)$, on $z_1 = (x_1 + y_1)/2$, $z_2 = (x_2 + y_2)/2$ i $z_3 = (x_3 + y_3)/2$.

L'algorisme funcionarà de la següent manera[17]:

Primer de tot s'escull un nombre de clústers k . Tot seguit es determinen un conjunt de k punts que seran els centroides. La localització d'aquests pot ser tant aleatòria com heurística. Els resultats més idonis s'obtenen quan aquests estan suficientment separats. També cal comentar que els resultats varien segons la posició inicial d'aquests centroides. A continuació s'assigna cada element al centroide més proper. Un cop assignats tots els elements la posició dels centroides es recalcula tenint en compte tots els punts que formen el clúster per aconseguir una distància mínima. Cal repetir els dos últims passos, reassignar elements i recalcular centroides, fins que el moviment de tots els centroides sigui pràcticament nul. La funció d'aquest algorisme és minimitzar la funció d'error:

$$V = \sum_{i=1}^k \sum_{x_j \in S_i} (x_j - \mu_i)^2 \quad (4.5)$$

on hi ha k clústers S_i , $i = 1, 2, \dots, k$ i μ_i és el centroide de tots els punts $x_j \in S_i$.

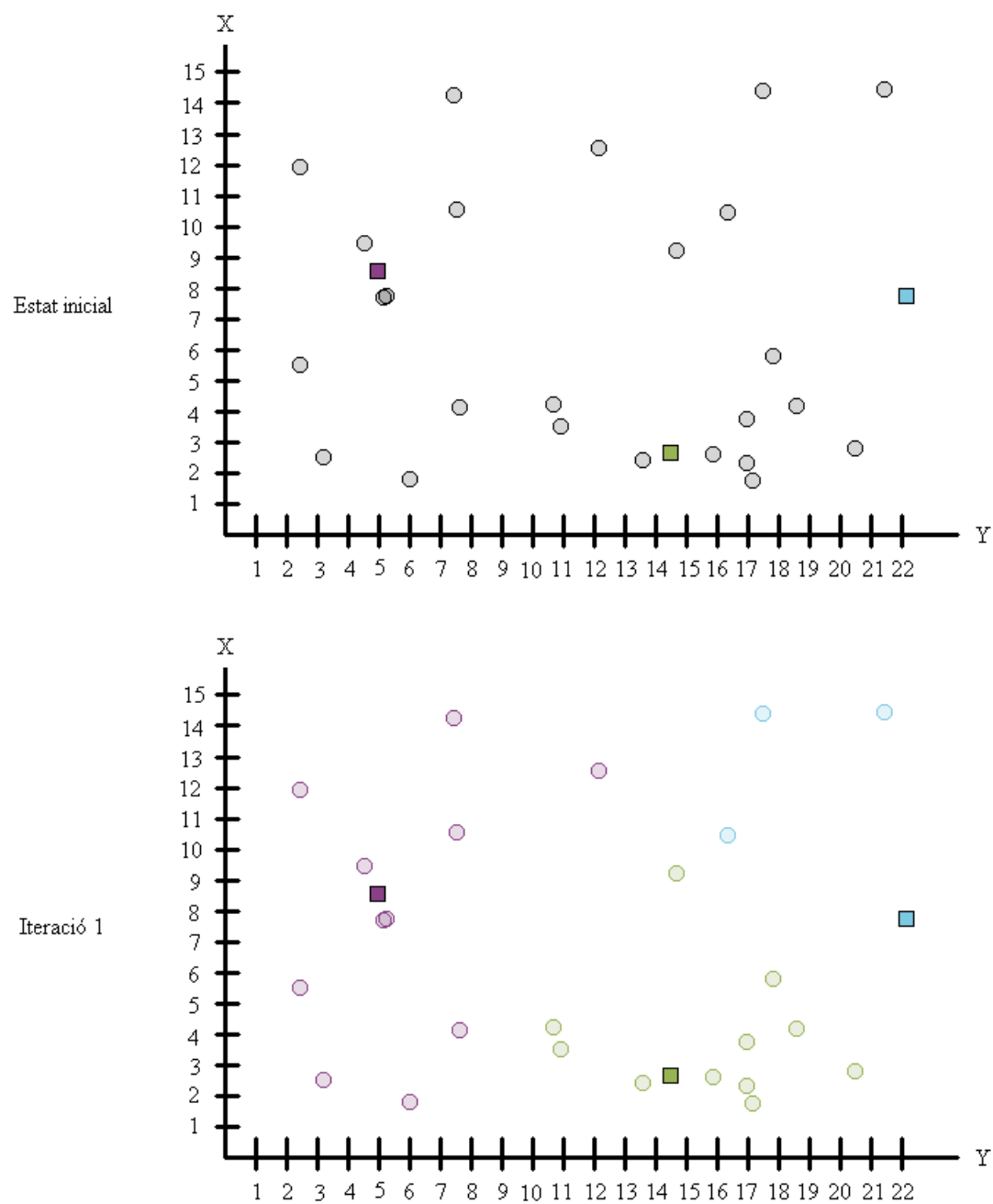


Figura 4.6: Estad inicial i primera iteració de l'algorisme K-means[18].

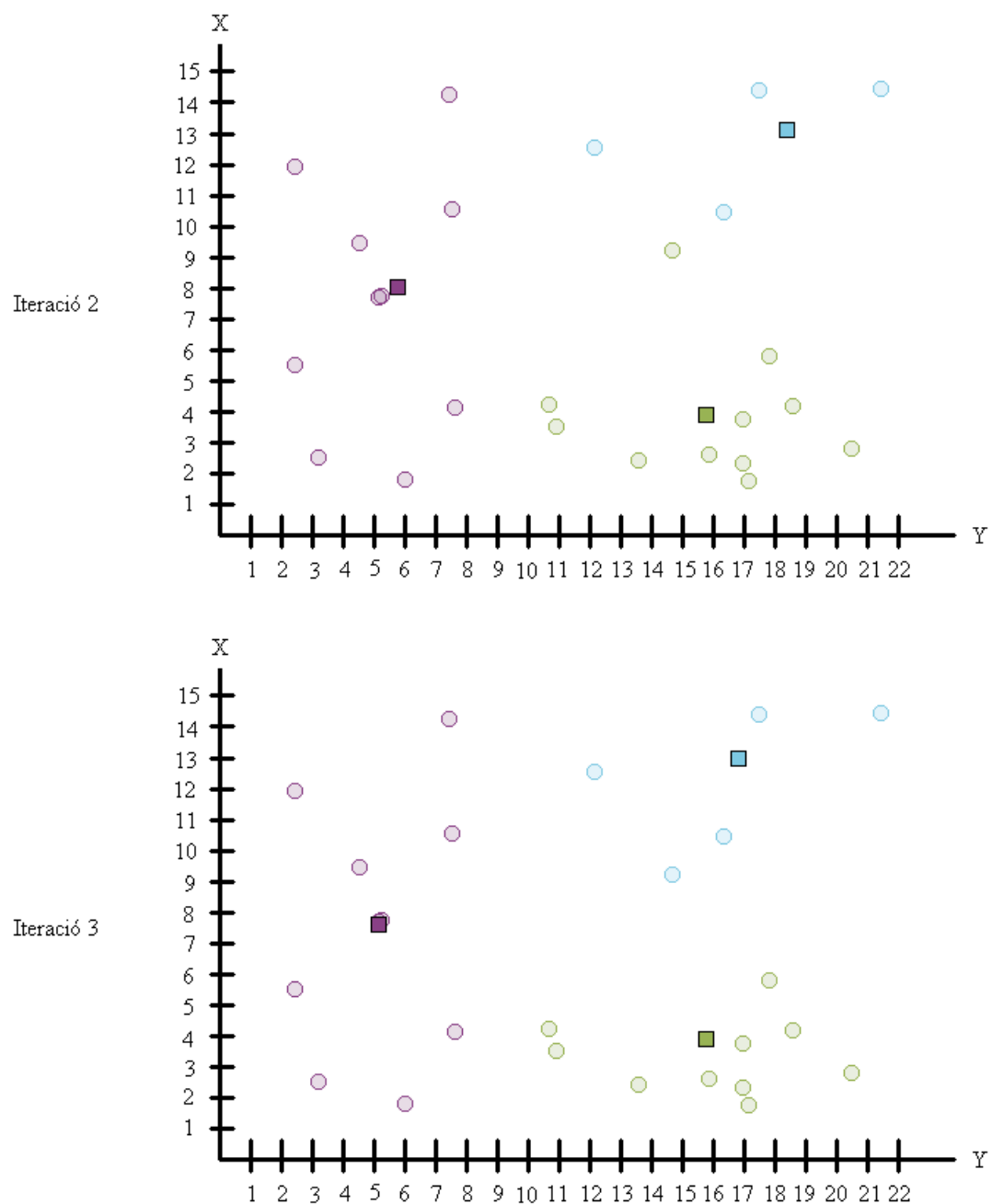


Figura 4.7: Segona i tercera, i última, iteració de l'algorisme *K-means*[18].

A la figura 4.6 i 4.7 podem veure un exemple de l'algorisme *K-means*. A l'inici, els centroides es distribueixen aleatòriament per l'espai. En el primer pas s'assigna cada punt al centroeide més proper utilitzant la distància euclidiana. En les iteracions següents es recalcula la posició de cada centroeide per situar-lo al centre del clúster i es tornen a assignar els punts als centroides. Quan no es produeix cap desplaçament de cap centroeide es dona l'algorisme per finalitzat.

4.3 Classificació de regions d'interès noves

Amb el clustering només agruparem les regions que han tingut un moviment a través dels frames. Si hi ha alguna regió que s'ha perdut en el tracking, perquè en algun frame no s'ha detectat, s'afegirà després. Hi ha diferents tipus de classificacions per afegir aquestes regions a els grups que ja tenim. La que s'ha escollit en aquest cas és classificar per proximitat.

Una regió pertanyerà al grup que estigui més a prop seu. Per tant s'avaluaran els tres veïns més pròxims de cada grup a la regió a classificar. A partir d'aquesta idea, es pot fer de dues maneres. La primera és només tenint en compte els conjunts inicials per totes les classificacions. I la segona és tenir en compte el nou conjunt, amb la regió afegida a la última iteració per comparar la pròxima regió. Així seria la idea de regions creixents.

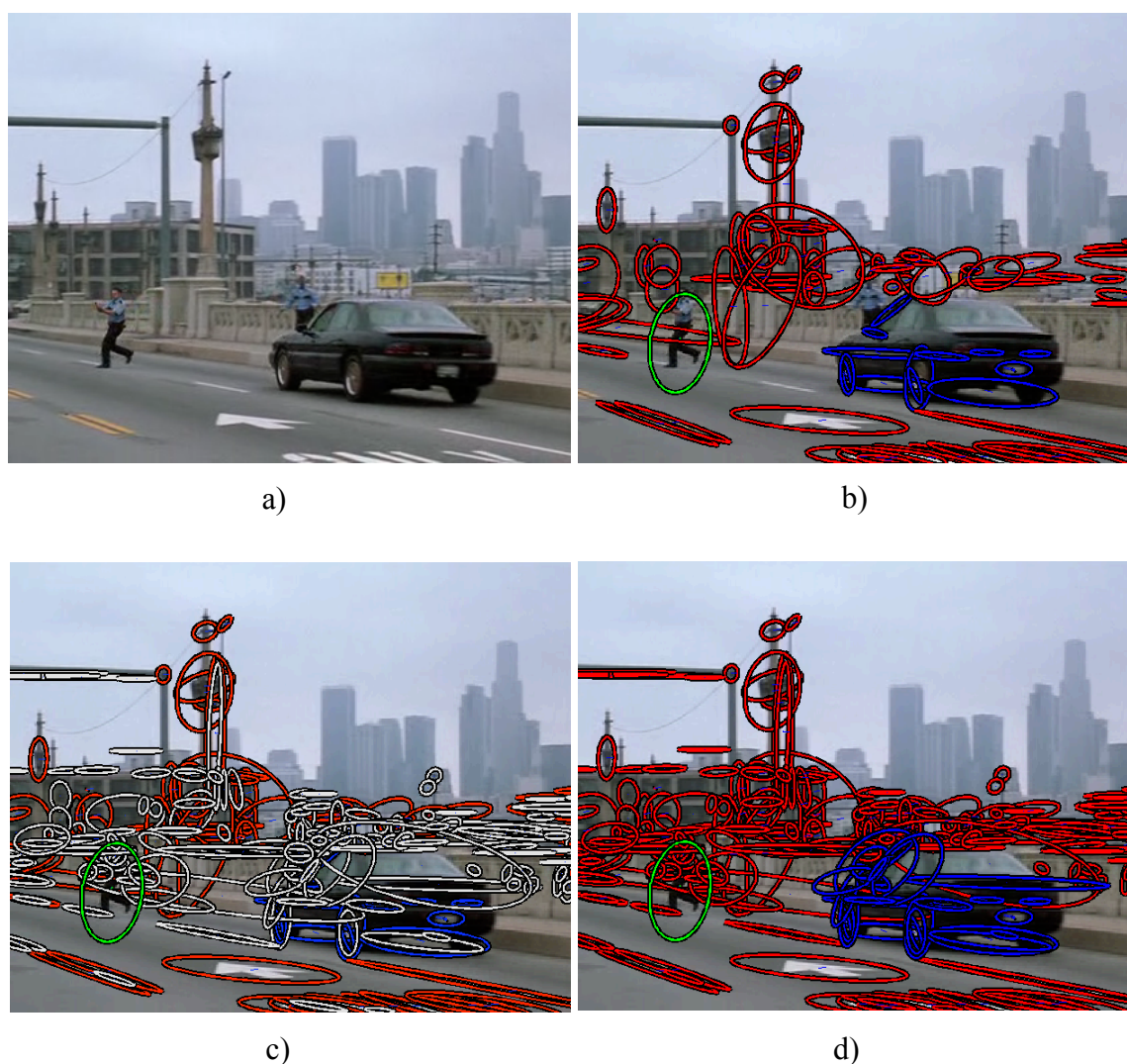


Figura 4.8: Visió global de la classificació de les noves regions. a) imatge original. b) imatge amb regions detectades. c) regions no classificades pintades en blanc. d) resultat final

En la figura 4.8 podem veure el procés que passen les regions que no tenen un moviment suficientment definit, per exemple de 3 frames enrere o menys. A l'apartat a) es pot veure la imatge sense tractar. En el b) veiem les regions d'interès que tenen un vector de moviment suficientment gran. En l'apartat c) es poden veure totes aquelles regions que no s'han utilitzat per el clustering pintades de blanc. I en l'apartat d) es pot veure el resultat final.

Capítol 5

Detalls d'implementació

En aquest capítol explicarem com hem implementat a la pràctica el que hem analitzat i dissenyat, també explicarem les tècniques que hem utilitzat per gestionar les dades, per reconèixer el moviment dels punts, i per agrupar-los en objectes. Així doncs s'explica en detall el funcionament general de l'algorisme i també els problemes i solucions trobades per cada una de les parts.

5.1 Estructures de dades

Per tal de tenir les dades ben estructurades i fàcils d'accedir entre els diferents mètodes de l'aplicació cal tenir-les estructurades d'una forma correcta. Una manera ordenada visualment de treballar amb les dades és a través d'objectes, com en la programació orientada a objectes, els quals tenen diferents atributs. En el nostre cas tenim dues classes d'objectes, els punts i els objectes(grups de punts).

Els objectes punts tenen els atributs recollits en el descriptor SIFT de cada regió d'interès de la imatge, X,Y,R1,R2,R3, Vector de moviment i Vector de 128 característiques, mentre que els grups tenen un conjunt de punts i un color per pintar-los a la imatge.

Per poder aplicar aquesta representació dins al *Matlab* disposem de varies eines, les estructures i les cel·les.

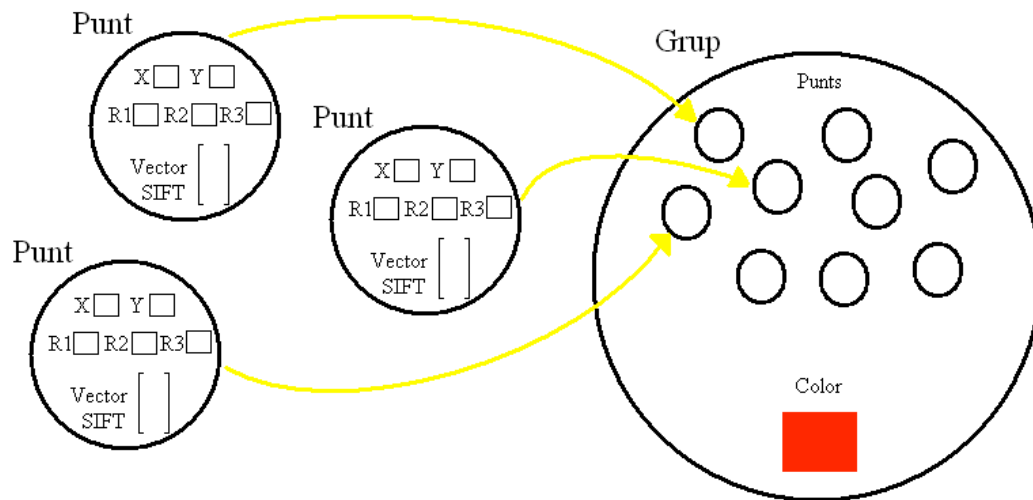


Figura 5.1: Es mostra una idea general de com podria ser una estructuració basada en objectes avaluant les nostres necessitats.

5.1.1 Estructures de dades en Matlab

5.1.1.1 Cel·les

Les cel·les són una de les múltiples formes de guardar la informació de les que disposa el *Matlab*. La idea principal és que són com un vector (o matriu si hi ha més d'una dimensió) on guardar la informació, però a diferència dels vectors, a cada posició de la cel·la hi pot haver un tipus diferent d'informació, des de nombres, caràcters, inclòs es poden guardar cel·les dins de caselles d'altres cel·les. Per tant es poden guardar els atributs d'un objecte de forma ordenada i consecutiva.

```
>> punt{1}=10
```

```
punt =
```

```
[10]
```

```
>> punt{2}=14
```

```
punt =
```

```
[10] [14]
```

```
>> punt{3}=[1 1; 1 2; 2 1; 2 2]
```

```
punt =
```

```
[10] [14] [4x2 double]
```

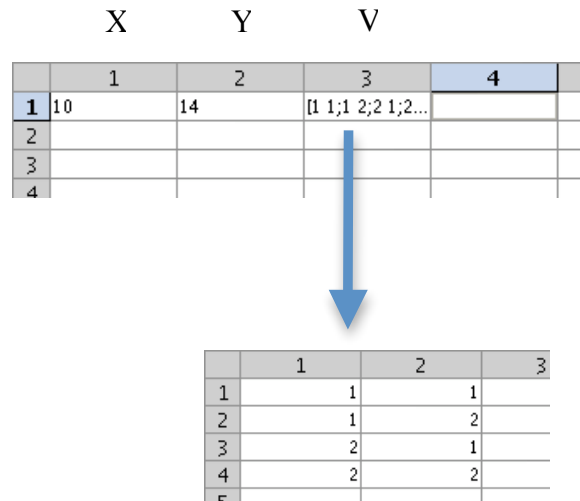


Figura 5.2: Exemple de la creació d'una cel·la punt.

En l'exemple de la figura 5.2 es construeix una cel·la anomenada punt. Ja que els atributs no es poden nombrar en la estructura, els hem de tenir presents nosaltres. El primer atribut serà la X, el segon serà la Y i el tercer serà un vector amb el moviment d'aquest punt.

5.1.1.2 Estructures

Les estructures del *Matlab* estan orientades al principi d'objecte, com en la programació orientada a objectes, però en realitat, són com les estructures de C/C++, structs. Aquestes estructures s'han de declarar per poder-les utilitzar. Això es fa amb la comanda 'struct', i se li passen els paràmetres que ha de tenir l'struct, en parelles de variable – valor. Per que quedi més clar es pot il·lustrar amb un exemple, com el de la figura 5.3.

```
>> punt = struct ('X', 10, 'Y', 14);
>> punt
```

```
punt =
```

```
X: 10
Y: 14
```

Figura 5.3: Exemple de creació d'una simple estructura.

Aquest és un exemple d'una estructura senzilla, un punt en un espai en dues dimensions. Per accedir als seus atributs, es pot fer igual que en C/C++, i la crida seria com es mostra a la figura 5.4.

```
>> punt.X
ans =
    10
>> punt.Y
ans =
    14
```

Figura 5.4: Es mostra com obtenir els atributs d'una estructura

Aquesta estructura només conté un sol punt, però per poder guardar tots els de la imatge hem de fer un pas més, transformar-la en un vector de estructures (o de punts). Es pot inicialitzar l'estructura en la mateixa declaració, com s'ha vist en l'exemple anterior, o es pot declarar una estructura en blanc. En l'exemple següent podem veure els dos casos:

```
>> punts = struct ( 'X', {}, 'Y', {} )
punts =
0x0 struct array with fields:
    X
    Y
>> punts = struct ( 'X', {10,11}, 'Y', {14,15})
punts =
1x2 struct array with fields:
    X
    Y
>> punts(1)
ans =
    10
```

Figura 5.5: Exemple de com crear una taula d'estructures i de com obtenir els valors dels objectes en una determinada posició.

El *Matlab* té varies funcions per treballar amb estructures, però les més útils en el nostre cas seran: les d'accedir a la informació, `vector_estructura(posició)` i `estructura.atribut`, i les que proporcionen informació, com la funció `isstruct`, que

retorna cert o fals (1 o 0).

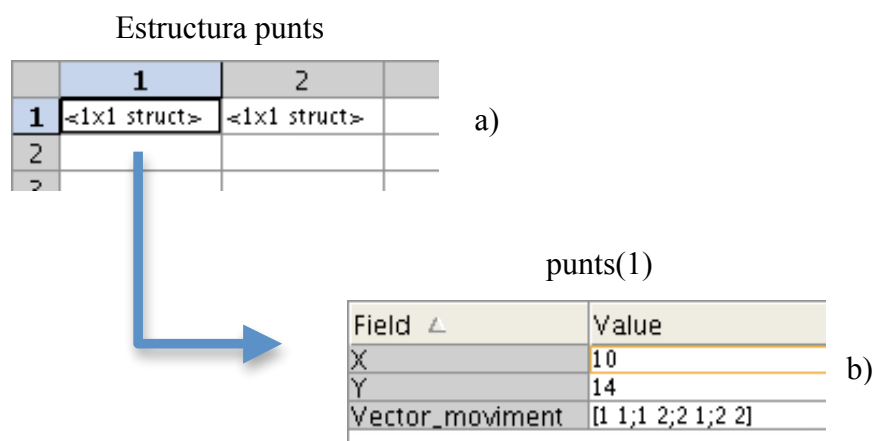


Figura 5.6: Esquema de l'estructura plantejada a la figura 5.5. a) és el la taula d'estructures punts i b) és el contingut d'un element.

En l'exemple de la figura 5.6 es crea l'estructura de l'exemple anterior, figura 5.5, però a més si afegeix un vector de moviment del punt. Podem veure a la informació de l'estructura (apartat a) que en la taula d'estructures només es veu quantes n'hi ha, però no que són. Per saber-ho hem de visualitzar-les per separat, com es veu a l'apartat b, on si que podem veure els atributs de la estructura, amb el seu nom.

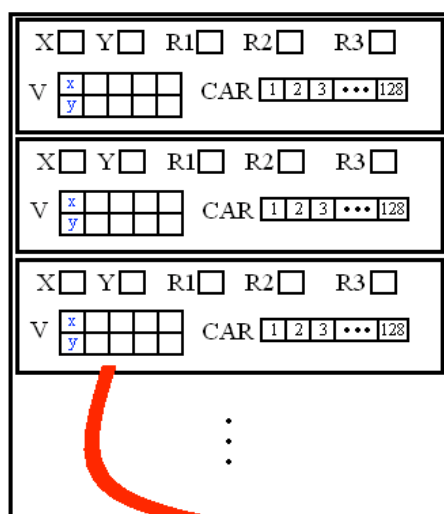
5.1.2 Estructures de dades proposades

La estructura de dades proposada compta de dues estructures, la primera per guardar les regions d'interès de la imatge, i la segona per guardar els grups de regions, o sigui els diferents objectes que es detecten a l'escena.

De la primera estructura, la de les regions d'interès, n'hi ha dues, una per la imatge anterior i una altra per la imatge actual. Els seus atributs són: X, Y, que és la posició central de la regió d'interès. R1, R2, R3, que és la forma de la el·lipse de la regió afí. CAR, és el vector de 128 característiques que defineixen el descriptor SIFT de la regió. I V, que és una matriu que conté una finestra de cinc posicions del moviment de la regió en la escena, 5 posicions x 2 valors (X,Y). Si una regió s'ha detectat com a nova, pot ser que no s'emplenin les 5 posicions, llavors la matriu podria ser de 1x2.

La segona estructura és la que conté els grups de regions o objectes, aquests grups tenen com a atributs, les dades de tots els punts del grup, i a més el color en que es pintarà el grup sobre la imatge. Per poder aprofitar el càlcul vectorial del *Matlab*, els punts de cada grup no es guardaran com estructures dins d'estructures, si no que es transformaran a vectors i es guardaran com una sola matriu, una fila per regió d'interès i una matriu per cada grup.

Taula d'estructures de regions



Mitjana del vector

Matriu de dades
de les regions

	Regio 1	Regio 2	...	Regio n
X				
Y				
R1				
R2				
R3				
V _x				
V _y				
1				
2				
3				
...				
CAR				
128				

Taula d'estructures de grups

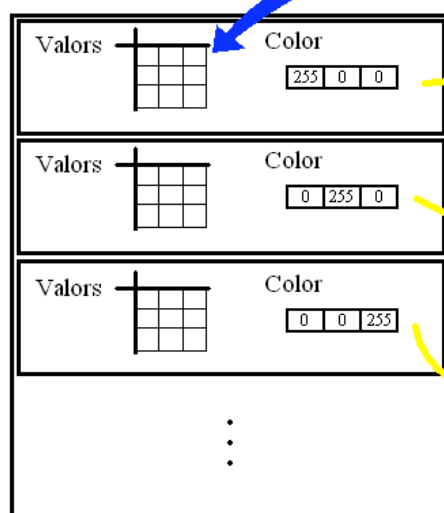


Figura 5.7: Esquema de l'estructura de dades global.

La primera part de la figura 5.7 és una taula on es guarden totes les estructures corresponents a cada regió d'interès trobada per l'algorisme MSER i transformada a SIFT. Guardar les dades d'aquesta manera fa l'algorisme més amè per la seva comprensió i facilita el treball a l'hora de comparar les regions del frame anterior i actual. La segona part (la taula situada al mig, a la part dreta) consisteix en una taula per emmagatzemar totes les regions de l'escena actual en forma de matriu per agilitzar tots els càlculs vectorials a l'hora de processar dades amb el *Matlab*. Un cop s'han calculat els grups es retorna una taula d'estructures grup. Cada estructura grup conté en forma de taula igual que la de la segona part (també per agilitzar càlculs posteriors) i un color assignat, en forma de vector RGB. A partir els punts que s'han trobat nous s'afegeixen als grups amb la tècnica de veïnatge, i ja es pot dibuixar el resultat.

5.2 Agrupació de punts en objectes

Un cop hem localitzat les regions d'interès a la imatge, i ja sabem com guardar les dades de forma que siguin accessibles, hem de transformar aquestes regions en objectes. De les regions que es van detectant a través de la seqüència amb la tècnica de tracking explicada anteriorment, es guarden les seves dades i a més es va emplenant un vector de moviment, que és una matriu on es guarda el nombre de punts que s'ha mogut la regió en X i Y en el transcurs del frame anterior a l'actual. Aquest vector és una finestra d'amplada cinc, així que quan han transcorregut cinc frames, al següent es perdran les dades de la primera posició.

Un cop tenim les regions guardades amb el moviment que han anat fent en un seguit de cinc frames, és hora d'agrupar les que es mouen en la mateixa direcció. Aquí és on entra la tècnica de clustering *K-means* que s'ha explicat anteriorment. Per fer aquest pas, degut a que en el transcurs dels frames es perden i s'afegeixen regions, només es tenen en compte aquelles que es mantenen en quatre frames com a mínim, o sigui, que tenen un vector de moviment de quatre o cinc posicions. Com ja s'ha comentat, el *Matlab* té implementada una funció *K-means*, que té diverses opcions. Una d'elles és poder escollir la ubicació inicial dels centroides, però això és una tasca difícil quan el moviment de les regions pot anar en qualsevol direcció. Per tant no la hem utilitzat, això però pot portar problemes, per que una mala inicialització dels centroides pot provocar que el *K-means* doni un resultat incorrecte. Per aquest cas l'algorisme del *Matlab* té una altra opció que repeteix l'algorisme X vegades i retorna el millor resultat. La opció més important però del *K-means* és la del nombre de clústers que volem. Existeixen diverses tècniques per estimar el nombre de clústers òptims, però en aquest projecte simplement s'ha fet sabent-ho amb anterioritat.

Ara ja tenim les regions organitzades en objectes de l'escena. Les regions d'aquests objectes es guarden en grups, com s'ha vist en l'apartat anterior. El següent pas doncs és la gestió d'aquests grups.

5.3 Tècnica per reconèixer grups

Durant el transcurs de l'algorisme, entre frames, els grups retornats per el *K-means* no tenen per que ser els mateixos que ja tenim del frame anterior. Depenen del primer punt de la llista. Per això s'ha de buscar alguna tècnica que ens relacioni els grups que teníem abans amb els grups retornats per el *K-means*.

La idea consisteix en avaluar els grups un a un entre ells, comparant els que ja tenim amb els nous. És possible que els grups hagin canviat considerablement, degut a pèrdues de regions en el tracking o incorporació de noves.

L'algorisme de reconeixement de grups anirà avaluant els grups un a un segons dues regles. La primera regla és segons el tamany del grup. Se suposa que un grup pot canviar de tamany de un frame a un altre, com ja s'ha comentat, però aquest canvi no serà molt gran, un grup amb pocs punts, o regions, seguirà tenint més o menys el mateix nombre. Per tant es donarà un tant per cent de punts als grups que tinguin més

o menys el mateix nombre de regions, relacionat amb el nombre total de regions.

Per exemple, com es pot veure a la figura 5.8, si hi ha dos grups, el fons amb 100 regions i un objecte amb 15 regions, el total és de 115. En el següent frame, el fons té 98 punt i l'objecte 16. En relació al total, si comparem el fons anterior amb l'objecte actual, es compararà el tant per cent de punts del fons anterior amb el total anterior, i el tant per cent de punts de l'objecte respecte al total actual. Llavors es compararan els dos tants per cent i s'obté el resultat de la prova.

La segona regla és comparar els punts de cada grup. Es compara la posició X de cada punt d'un grup amb l'altre. Els valors mínims es guarden, i al final es fa una suma de quants punts tenen en comú.

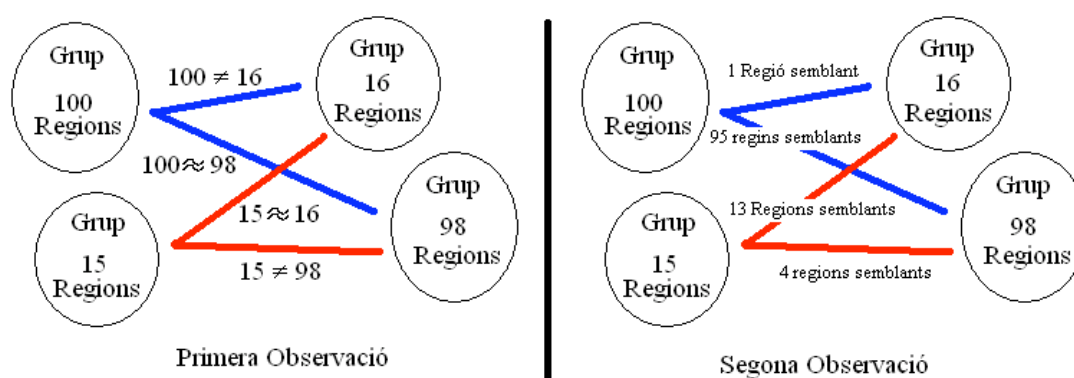


Figura 5.8: Exemple il·lustrat d'una possible situació de classificació de dos grups.

A la figura 5.8, veiem dos detalls a l'hora de reconèixer grups. Per exemple es podria donar la situació de que un grup perdés molts punts i un altre en guanyés forces i donar una situació confusa. Aplicant les dues tècniques els resultants són més bons. Finalment s'agafen els dos resultats i es sumen, cada un amb un pes del 50%.

5.4 Detalls sobre les noves regions

Aquesta és la part final i com s'ha explicat en el tema anterior consisteix en afegir les regions que no tenen un moviment suficientment definit. A la hora d'implementar aquesta part, s'ha fet servir una tècnica de agrupament per veïnatge que compara cada regió nova amb tots els grups. De cada grup, mira quines tres regions hi ha més a prop seu, i el grup que tingui més a prop, se l'hi assigna. Com hem vist a la figura 4.8 el resultat és molt bo.

Capítol 6

Anàlisi de resultats

En aquesta secció executarem l'algorisme amb diferents proves sintètiques i reals sobre diferents vídeos i pel·lícules, per comprovar com són els resultats i com es comporta l'algorisme. Cal destacar que aquestes proves estan fetes sobre escenes seleccionades de diferents pel·lícules que no tenen cap ordre en particular. A més, en aquesta memòria no es mostren totes les imatges obtingudes de l'execució de l'algorisme, si no que només conté les més significatives per la comprensió d'aquest.

Un dels resultats més significatius que hem obtingut en les proves de l'algorisme és la diferència entre el detector *Harris-Affine* i el detector *MSER*. En un principi es va començar utilitzant el detector *Harris-Affine*, i els resultats que s'obtenien, tot i que eren suficientment bons, no acabaven de ser acurats del tot. La qüestió està en que el detector *Harris-Affine* no detecta les mateixes regions d'interès en un objecte quan aquest es desplaça. Sí que detecta la majoria d'elles, però canvien sobtadament d'un frame al següent. En canvi, amb el detector *MSER* això no passa, si un objecte és igual a tots els frames, encara que variï la seva posició les regions obtingudes seran totes les mateixes en cada frame. Podem veure, doncs, la comparació en la figura 6.1. Per tant, en els primers resultats de cada tipus, sintètics i reals, mostrarem la diferència entre els dos detectors, per poder veure quin resultat dona cada un.



Figura 6.1: Comparació entre els detectors Harris-Affine i MSER a la hora de trobar regions d'interès en dues imatges iguals però desplaçades.

A la figura 6.1 podem veure dues imatges idèntiques, desplaçades per un espai buit, que no representarà cap regió d'interès. Les dues primeres és el resultat del detector *Harris-Affine* i les inferiors de l'*MSER*. Podem veure doncs que els resultats del detector *Harris-Affine* varien depenent de la posició de l'objecte, mentre que amb el detector *MSER* això no passa. Per l'ús que volem donar a les regions detectades això és el pitjor que ens pot passar, ja que necessitem que si un objecte s'ha desplaçat molts pocs punts d'un frame al següent, el detector ens retorni el major nombre de regions iguals possibles.

6.1 Proves sintètiques

Les proves sintètiques són les primeres que s'han fet amb l'algorisme. Ens han servit tant per fer-lo com per ajustar-ne els paràmetres. A continuació podrem veure algunes d'aquestes proves explicades en detall.

6.1.1 Prova de dos moviments

En aquesta prova s'ha utilitzat una seqüència d'imatges amb tres objectes iguals, que es mouen de dues maneres. Els superior i inferior, es mouen en la mateixa direcció, mentre que el del mig es mou en l'oposada. L'objectiu d'aquesta prova és que

l'algorisme detecti els objectes dels extrems com un de sol i el del mig com un altre objecte diferent.

Primer de tot, s'ha realitzat la prova amb el detector *Harris-Affine* i els resultats obtinguts són els desitjats. Els objectes queden correctament separats en els dos grups esperats. Però seguint la seqüència detalladament es pot veure com es van trobant molts regions diferents a cada frame. El mateix conjunt d'imatges s'ha provat, després, amb el detector *MSER*. El resultat final de la prova és el mateix, els objectes es separen de la mateixa manera que amb el detector *Harris-Affine*, però en aquest cas podem veure que els objectes menten tots els seus punts en cada un dels frames. Podem veure la comparació en la figura 6.2 i 6.3.

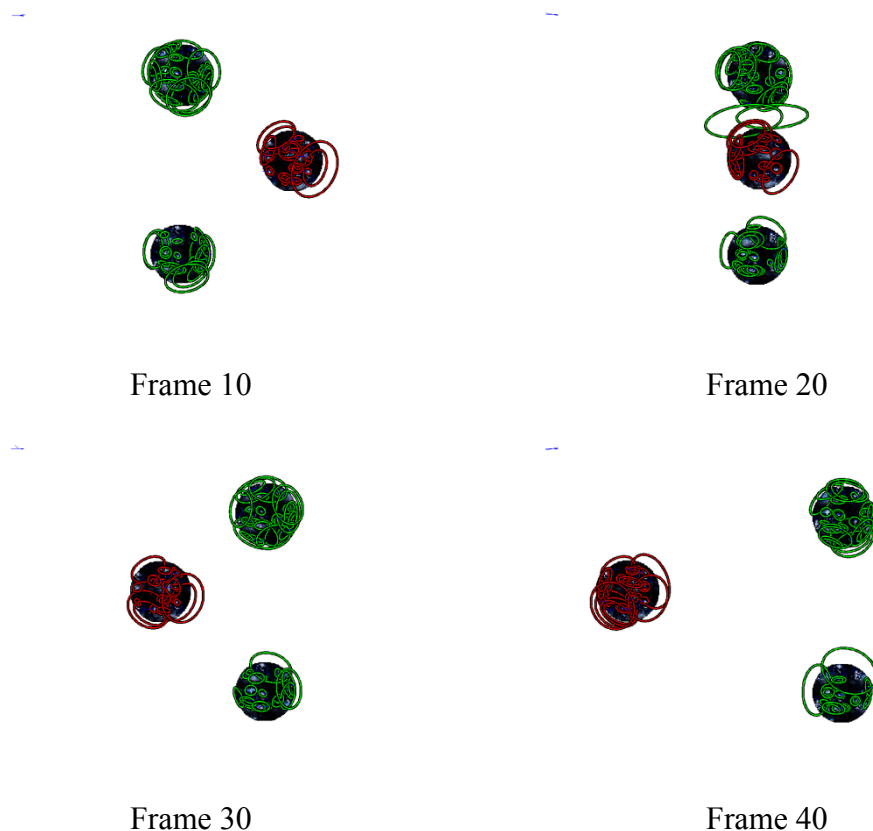


Figura 6.2: Resultat obtingut per les seqüència d'imatges de la prova 1 amb el detector Harris-Affine. Els objectes queden clarament separats segons el seu moviment. Però podem veure que es detecten diferents regions per cada objecte en els diferents frames.

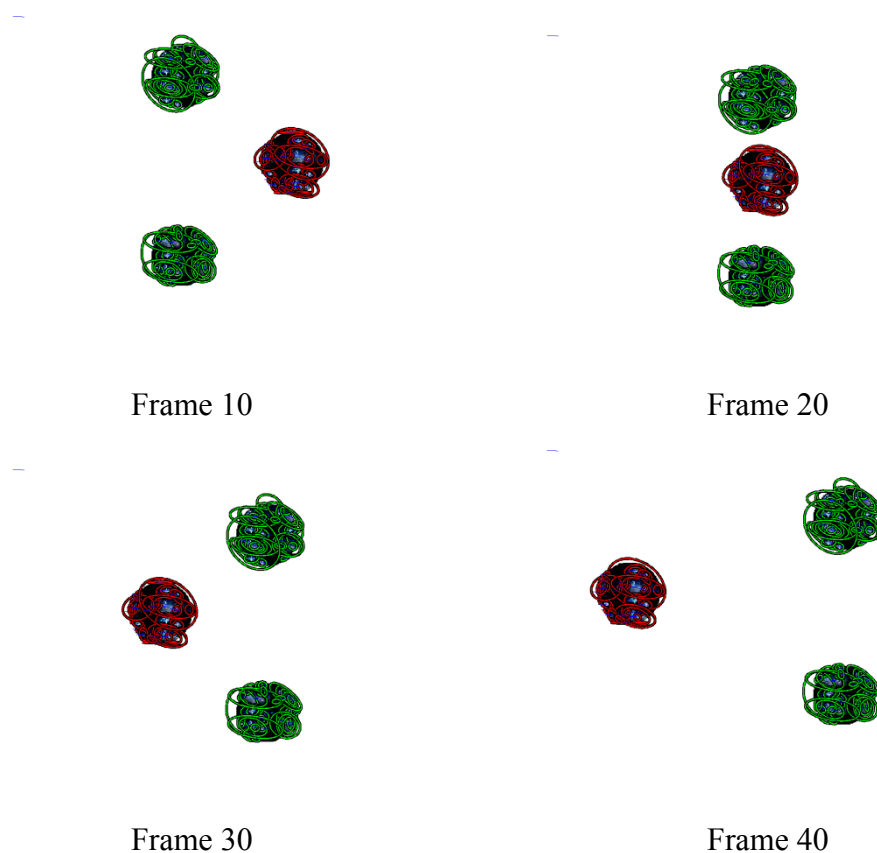


Figura 6.3: Resultat de l'execució de l'algorisme en la seqüència d'imatges de la prova 1 amb el detector MSER. Els objectes estan clarament separats segons el seu moviment.

Podem veure a la figura 6.3 que les regions d'interès detectades a cada un dels objectes en els diferents frames són les mateixes. Així doncs s'afavoreix el tracking i ens permet utilitzar un llindar en la distància euclidiana del descriptor SIFT molt més petit. En l'objecte superior del frame 20 de la figura 6.3, podem veure que regió de la part superior d'aquest, no es detecta en aquest frame. Això és degut al resultat del detector *MSER* que de totes les imatges, només en una i en una sola regió no ha retornat el mateix resultat.

Per tant podem concloure d'aquest experiment que, els objectes es detecten adequadament, els moviments es reconeixen de forma correcta i que utilitzant el mètode *MSER* podem fer un tracking molt més acurat.

6.1.2 Prova de tres moviments

En la seqüència d'imatges d'aquesta prova, tenim tres objectes amb la mateixa forma, que es mouen de tres maneres diferents. Per tant el nostre algorisme hauria de detectar que són tres objectes separats i hauria de mostrar-los de colors diferents. En aquesta prova també s'ha utilitzat els dos detectors, el *Harris-Affine* i l'*MSER*, per tant podem veure les diferències en el resultat que hi ha entre ells.

En la figura 6.4 es mostra el resultat amb el detector *Harris-Affine*, i en la 6.5 l'obtingut amb el detector *MSER*.

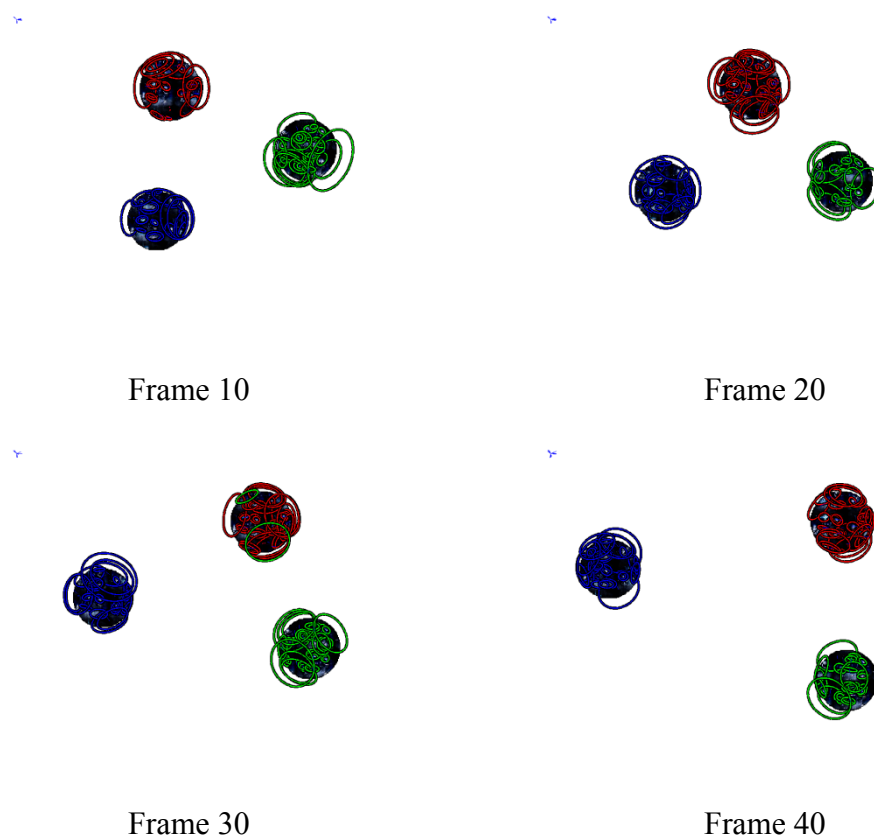


Figura 6.4: Resultat de la segona prova amb el detector *Harris-Affine*.

Podem veure en figura 6.4 que els resultat és correcte en un 95-98% de les regions detectades en cada frame. Podem considerar aquest resultat entre bo i molt bo, ja que la gran part de cops que es classifiquen regions per el seu moviment sempre n'hi poden haver algunes que no queden al grup on se suposa que han d'anar. Aquest cas és diferent però, aquí el que passa és que el detector *Harris-Affine* no ha trobat les mateixes regions en les imatges al voltant del frame 30. Per tant s'han assignat

regions que al no ser detectades iguals, no estan a la mateixa posició, per tant s'altera el vector de moviment i es dona aquesta confusió.

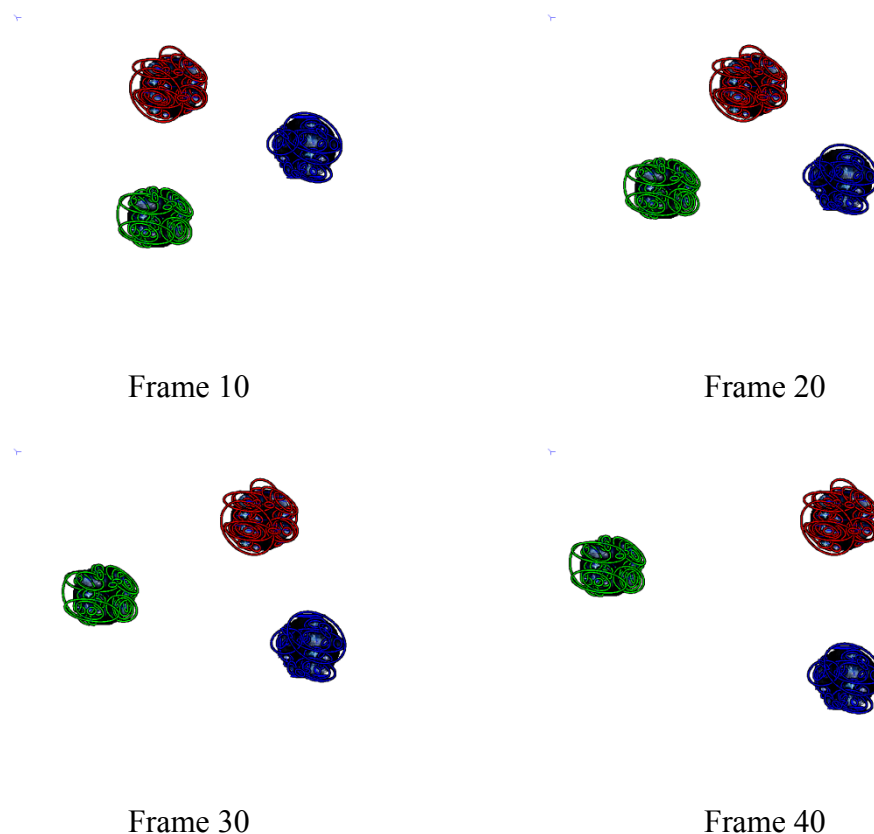


Figura 6.5: Resultat de la prova número 2 amb el detector MSER

En la figura 6.5 veiem que el resultat final amb el detector *MSER* és molt millor que amb el detector *Harris-Affine*. El resultat obtingut en aquesta prova és un resultat ideal que gairebé només passa en les proves sintètiques en un entorn controlat. Veiem doncs que els tres objectes queden separats de forma correcta i destacats cada un amb el color corresponent. A diferència del detector *Harris-Affine*, en la figura 6.4, aquí podem veure que totes les regions detectades de cada objecte es mantenen intactes fins al final. Cosa que fa que no es produeixin confusions de regions en grups com passava a la figura 6.4.

Cal exposar també que el temps d'execució de l'algorisme amb el detector *MSER* ha estat lleugerament més ràpid en aquest cas. Amb el detector *Harris-Affine* ha tardat 158 segons en processar la seqüència, mentre que amb el *MSER* 148 segons. És un diferència de 10 segons que, tot i que ha trobat més punts, sigui una mica més àgil aquest detector.

6.1.3 Prova de dos moviments confusos

Quan un objecte es para, la mitjana del seu vector de moviment es va fent cada cop més petita fins que arriba a 0. En aquest moment és quan ens en adonem que l'objecte s'ha parat. En aquest projecte no s'ha implementat aquesta opció, quedaria pendent per futures millores, però s'ha estudiat què passa en aquestes situacions.

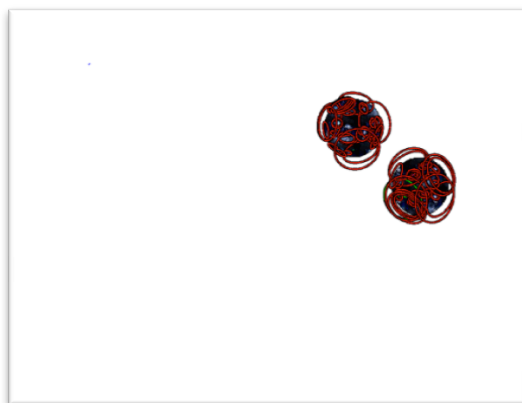
En aquesta prova es presenta un cas molt particular. Dos objectes es mouen en dues direccions diferents, llavors, els dos de cop canvien sobtadament de direcció i es mouen inversament. Això provoca que hi hagi un punt on la mitjana de la velocitat de cada punt és pràcticament nul·la. Així d'aquesta manera es posa a prova l'algorisme *K-means* a l'hora de dividir les regions en els dos grups quan aquesta mitjana de moviment de cada regió és tant petita.



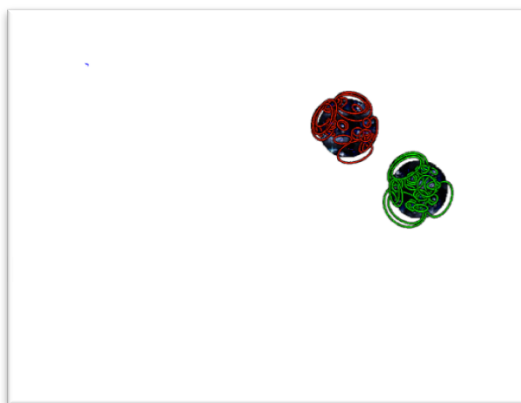
Frame 40



Frame 41



Frame 42



Frame 43

Figura 6.6: En aquestes quatre imatges es mostra el punt més crític de la seqüència de la prova 3. Es fa servir el detector Harris-Affine en aquest exemple.

En la figura 6.6 podem veure el punt més crític de la seqüència que estem provant, és quan la mitjana del moviment de cada regió és pràcticament nul·la. Podem veure doncs en el frame 42 que es confonen quasi tots els punts del segon objecte menys un. Aquí es produeix una situació delicada, ja que quan s'escullen quins grup és cada conjunt de punts en el frame 43 poden haver-hi errors. Si només hi ha un punt en el conjunt 2, aquest punt també serà en el següent frame, per tant és un 100% d'encert en el grup. El mateix passa per l'altre grup, on tots els seus punts seran també a un dels conjunts. Així es pot decidir per el número de punts també, on es farà la distinció dels grups.

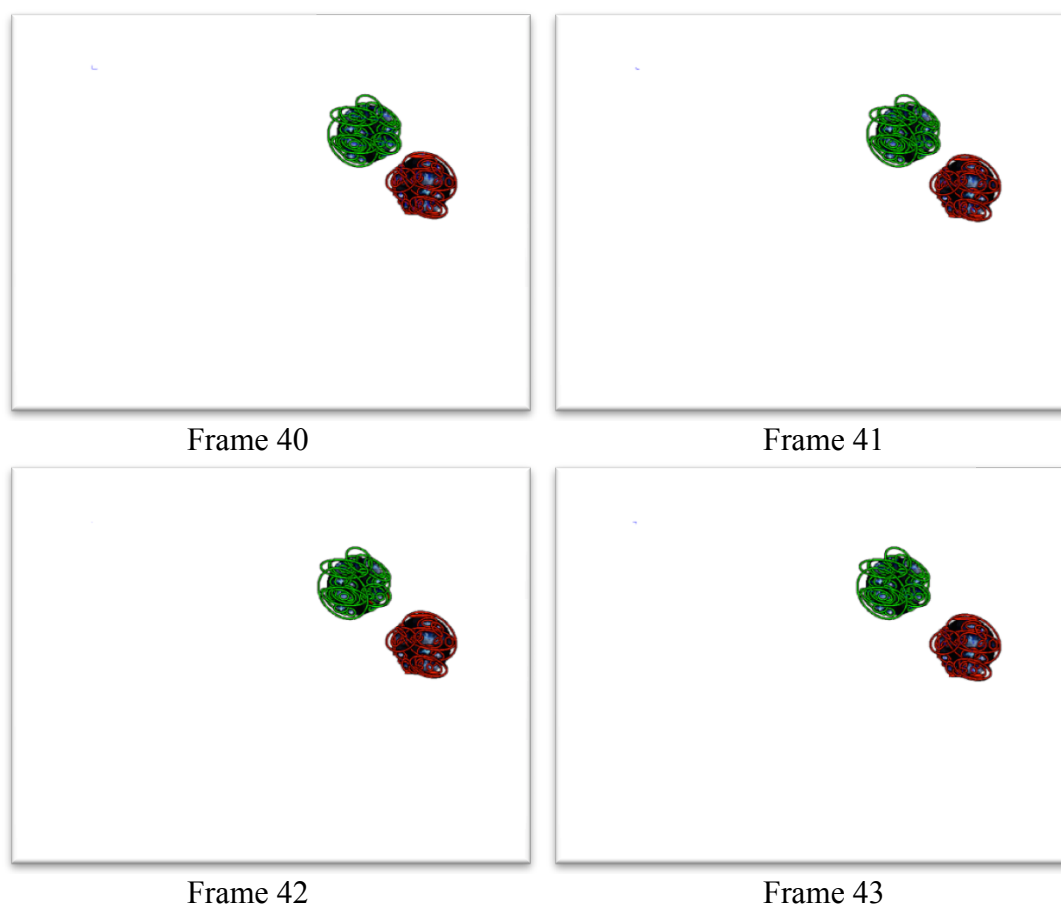


Figura 6.7: Punt crític de la seqüència d'imatges de la tercera prova. En aquest cas s'ha utilitzat el detector MSER amb millors resultats.

Com es pot veure a la figura 6.7, els resultats són molt millors utilitzant el detector *MSER*. En el frame més crític, el 42, veiem si mirem molt bé el primer objecte, de color verd, que una de les seves regions s'ha assignat a l'altre grup. Podem dir doncs que el resultat és molt millor en aquest cas, ja que el moviment de totes les regions s'anul·la per igual. Al no confondre cap de les regions tot s'escala per igual i per això obtenim tant bon resultat.

Situacions d'aquestes en imatges reals, sempre solen donar problemes, ja que sempre hi ha regions de més a més que es detecten amb un moviment quasi nul. Aquestes solen ser regions del fons.

6.2 Proves reals

Un cop hem observat el comportament de l'algorisme amb proves sintètiques i controlades, ja podem passar a experimentar amb vídeos reals. Aquests, seran fragments extrets de diferents pel·lícules. A continuació es mostren algunes de les proves que s'han fet i quins resultats han donat. Cal recalcar que no mostrarem totes les imatges que retorna l'algorisme, si no les més significatives per la seva explicació.

6.2.1 Seqüència “Groundhog’s Day” 1

En aquesta prova separarem el fons, d'un objecte, un cotxe que es mou per una carretera. Aquesta és la primera seqüència que vam escollir per fer el pas de sintètic a real. Per aquest motiu és una de les proves que hem repetit més vegades. El problema principal estava en que el detector *Harris-Affine* retornava regions d'interès lleugerament diferents en frames consecutius.

Compararem tres casos diferents. El primer és utilitzant el detector *Harris-Affine* i tenint en compte per el clúster els punts que tinguin una finestra de moviment de com a mínim de 4 frames. En el segon cas s'utilitza també el detector *Harris-Affine* i una finestra de 5 frames. En l'últim cas utilitzem el detector *MSE* amb una finestra de 4.

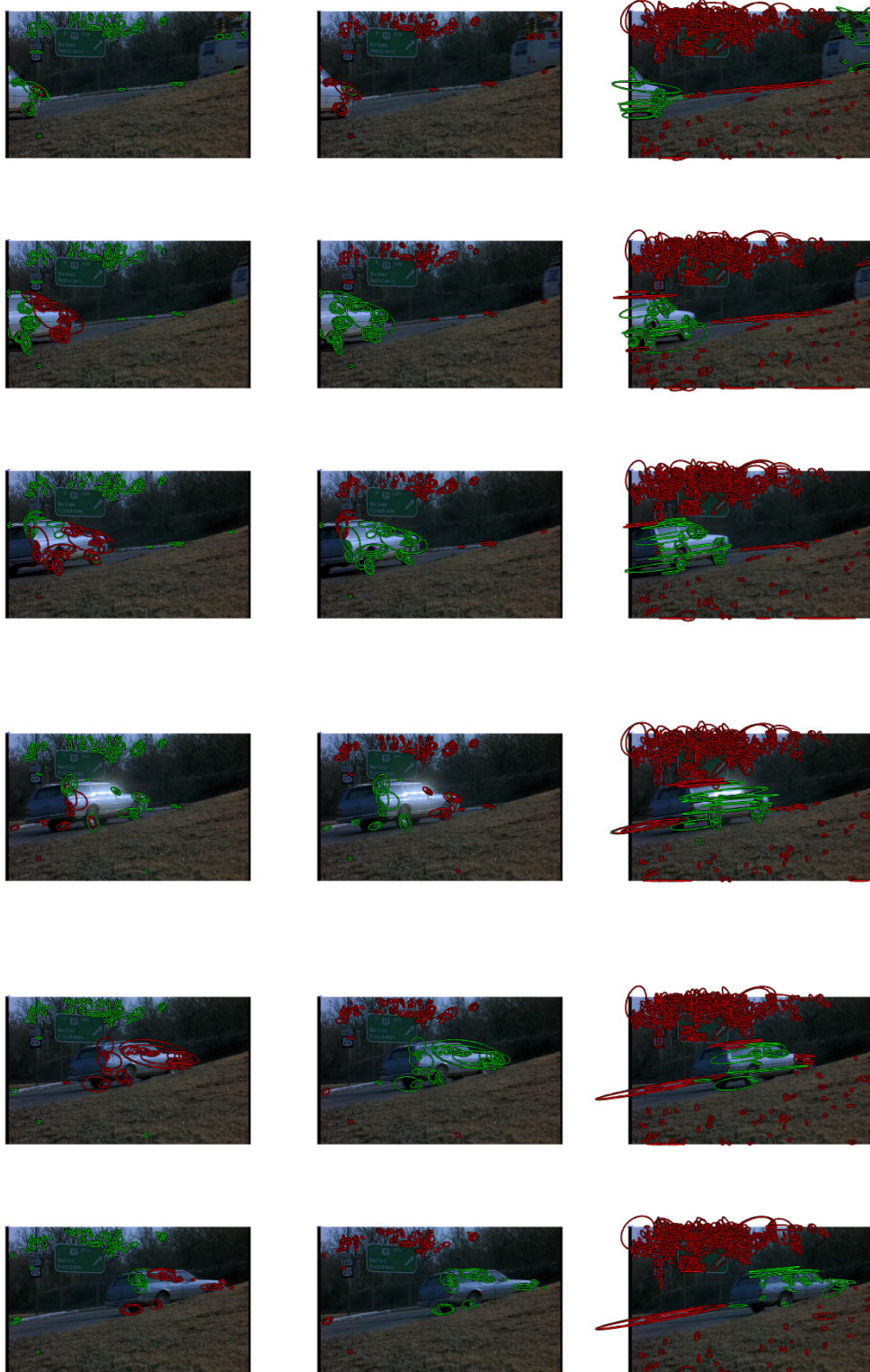


Figura 6.8: Es mostren els frames més significatius resultants de la prova “Groundhog’s Day” i executada de tres maneres diferents.

A la primera columna de la figura 6.8 s'ha utilitzat el detector *Harris-Affine* amb una finestra de moviment de 4 posicions com a mínim. A la columna central s'ha utilitzat el detector *Harris-Affine* també, amb una finestra de moviment de 5 posicions com a mínim. I a la columna de la dreta, s'ha utilitzat el detector *MSER* amb una finestra de 4 posicions com a mínim.

Si comparem a simple vista el detector *Harris-Affine*, figura 6.8, columna 1 i 2, amb finestra de 4 i de 5 entre ells veiem que hi ha més frames encertats en el de 5. Això és degut al veïnatge i a la falta de punts en el fons. Al voltant del cotxe hi ha relativament poques regions d'interès amb el detector *Harris-Affine*. Per tant, en el moment en què es troben tres punts dins el cotxe, la resta de punts s'assignen per veïnatge. En el cas de la finestra de 5 el moviment és molt més robust i només detecta punts que classifica com a cotxe. En el cas de la finestra de 4, les regions que es detecten no tenen un moviment tant robust i per tant algunes s'assignen al cotxe i altres al fons. Això fa que l'assignació de punts restants per veïnatge no sigui tant acurada, ja que hi ha regions dels dos tipus dins el cotxe. Tot i això la execució amb una finestra de cinc ens dona un resultat força acurat, en aquest cas.

Si ara donem un cop d'ull al detector *MSER*, figura 6.8, columna 3, veiem que el resultat és molt més acurat en quasi tots els frames, i molt més robust. Si observem el segon frame veurem que hi passa alguna cosa estranya. Mentre els altres es separa clarament el cotxe del fons, en el segon no és així, si no que a més també es confonen més punts. Bé, això té una explicació senzilla i és que, la part posterior del cotxe entre en escena. Aquesta part del cotxe és molt útil per generar regions d'interès per la forma que té, llavors tots els punts del cotxe es reestructuren de tal manera que tinguin en compta tota aquesta part. Degut a això en canvien molts i llavors és quan la majoria es perden i tarden uns frames a tornar a ser vàlids per el clustering. Degut a que el moviment del punt que queda dins el cotxe no és molt destacat, l'algorisme *K-means* agrupa també algun punt del fons, ja que intenta ser el més eficient possible, grups tant grossos com es pugui.

La diferència entre el temps de la prova amb el detector *Harris-Affine* i *MSER* és bastant notable. El detector *Harris-Affine* ja que ha detectat menys punts només ha tardat 285 segons, mentre que el detector *MSER*, al trobar moltes més regions n'ha tardat 1141. Així doncs veiem que el temps es desmarca no en la fase de trobar les regions d'interès si no en la fase de processar-les.

6.2.2 Seqüència “Groundhog’s Day” 2

En aquesta prova es testja una escena llarga de la pel·lícula “Groundhog’s Day” on la furgoneta dels protagonistes arriba al poble i es para. Es separen dos tipus d'objectes en particular, el fons, dels objectes que es mouen. Es pot veure que a la seqüència hi apareixen els crèdits, que també es detecten com a un objecte que es mou.

En aquesta escena, només s'ha provat amb l'algorisme detector de regions *MSER*. Ja que hem decidit que ens funciona molt millor que el *Harris-Affine*. A continuació, a la figura 6.9 es mostra el resultat d'aquesta prova.

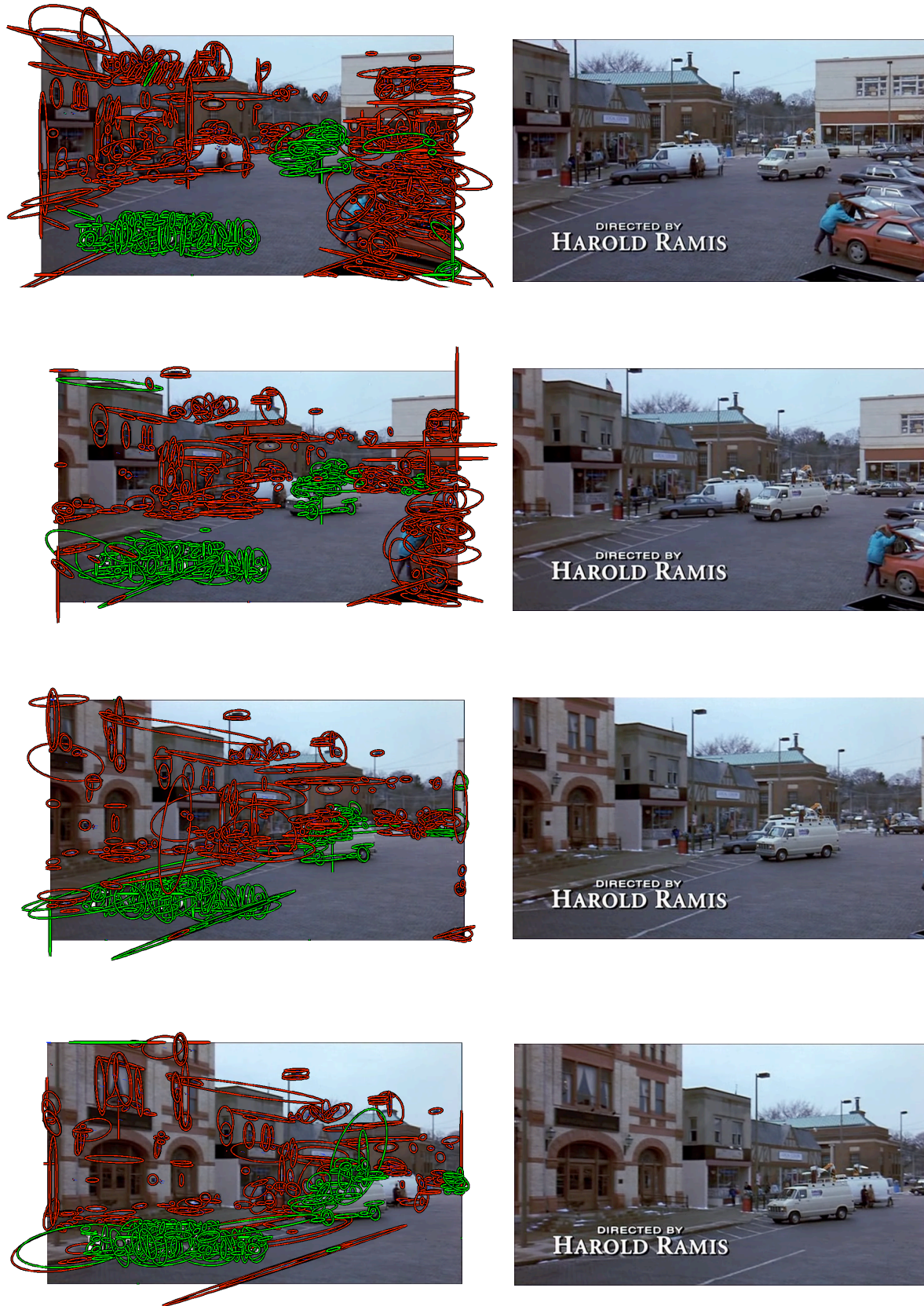


Figura 6.9: Resultat de l'execució de la prova "Groundhog's Day" 2 amb el detector MSER, amb una mostra de cada 20 frames.

Com es pot veure a la part esquerra de la seqüència de la figura 6.9, els objectes queden destacats en verd sobre el fons. Podem veure un objecte principal, que seria la furgoneta. També es veuen els crèdits, que es consideren com a part de l'objecte que es mou. També es pot veure en els quatre primers frames que es detecta un cotxe que es mou al fons de l'escena.

Així doncs podem dir que per una escena on hi ha varis objectes rígids que es mouen, l'algorisme es capaç de separar-los del fons.

6.2.3 Seqüència “Star Wars: Episodi 1” 1

Aquesta prova s'ha fet sobre objectes no rígids. Necessitem saber si l'algorisme es capaç de detectar tot tipus d'objectes. En concret s'ha fet sobre un grup de persones que es mouen en la mateixa direcció. És una escena de la pel·lícula “La guerra de les galàxies: Episodi 1”. El tracking separa el grup de persones del fons, però aquí no es diferencia quantes persones hi ha.



Figura 6.10: Frames inicial i final de l'escena de la seqüència “Star Wars: Episodi 1” 1.

La seqüència de la figura 6.10 és sobre la que es farà la prova. Es mostra només la imatge inicial i la final de l'escena per veure el contingut d'aquesta abans de fer-la. A continuació es mostraran els resultats de l'execució de l'algorisme, mostrejant cada cinc frames.

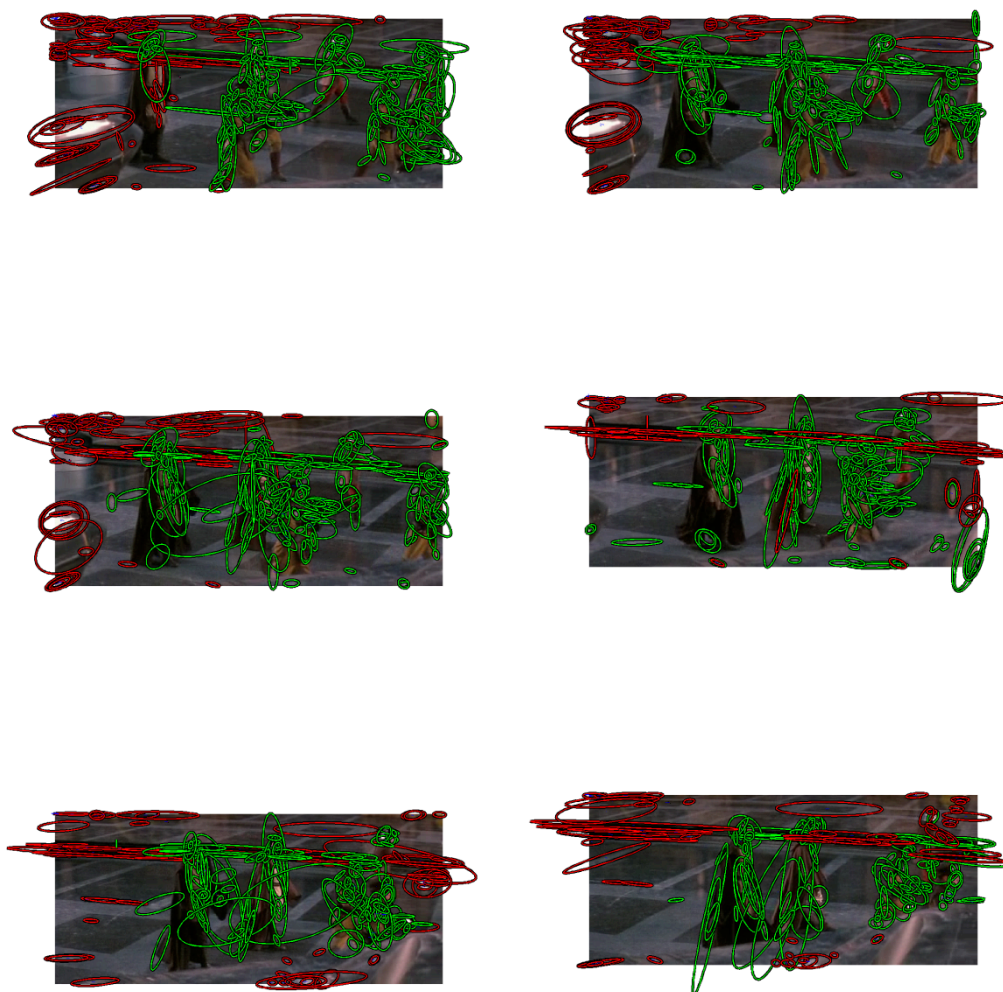


Figura 6.11: Resultat de l'execució de la seqüència "Star Wars: Episodi 1" 1 amb el detector MSER.

En la seqüència d'imatges del resultat de l'escena que podem veure a la figura 6.11, observem que encara que es confonen algun punts, el resultat final és prou bo i acceptable. La raó principal de que es confonguin tants punts és que a la part superior i inferior se la imatge no es detecten prou regions. Això fa que el fons d'aquesta zona no tingui prou veïns com per classificar els punts que s'han perdut correctament. Un altre factor és la línia més fosca que hi ha a l'altura dels caps de les persones. Aquesta línia és una regió de molt d'interès ja que es troben molts descriptors. Això provoca a vegades un efecte negatiu en el veïnatge dels punts de les persones.

6.2.3 Seqüència “Star Wars: Episodi 1” 2

Aquesta prova segueix amb la dinàmica de l'anterior, només per mostrar la escena següent. En aquest cas només surten dues persones i la càmera és mou. A continuació podem veure l'escena sobre la que es treballa, a la figura 6.12.

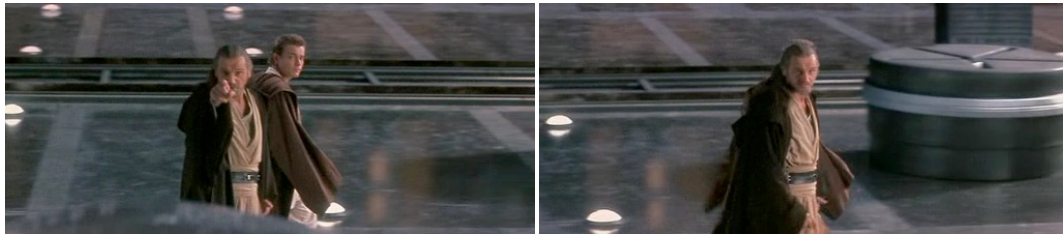


Figura 6.12: Frames inicial i final de l'escena de la seqüència “Star Wars: Episodi 1” 2.

Veiem doncs que l'escenari és el mateix, però l'escena o shot és diferent. L'algorisme és capaç de detectar aquest canvi i així classificar-lo com a un shot diferent. A continuació veurem els resultats mostrejant de tres en tres frames.

En els resultats podem veure que en la figura 6.13 es distingeixen millor les persones del fons, ja que el resultat és una mica més acurat. Encara es confonen alguns punts del fons però cap o molt pocs de les persones. En aquest cas l'escena és més rica en objectes i la detecció de regions d'interès dona millors resultats. Així doncs hem pogut veure que la detecció d'objectes no rígids és possible i dona resultats suficientment bons.



Figura 6.13: Resultat de l'execució de la prova "Star Wars: Episodi 1" 2 amb el detector MSER.

6.2.4 Seqüència “El tigre y la nieve”

En aquesta prova s'intentarà identificar tres objectes diferents. L'escena és de la pel·lícula “El tigre y la nieve”. Com el pot veure a la figura 6.14 en la seqüència surten dues persones creuant un carrer en una direcció i un cotxe circulant en l'altra. L'objectiu és aconseguir separar aquestes dues persones, el cotxe i el fons en tres grups diferents.



Figura 6.14: Seqüència de la pel·lícula “El tigre y la nieve”

Així doncs s’ha fet aquesta prova tant amb el detector *Harris-Affine* com amb el detector *MSE*. Podem observar a la figura 6.15 que els resultats no són en absolut el que esperàvem. Les dues persones que travessen el carrer no s’han pogut detectar.

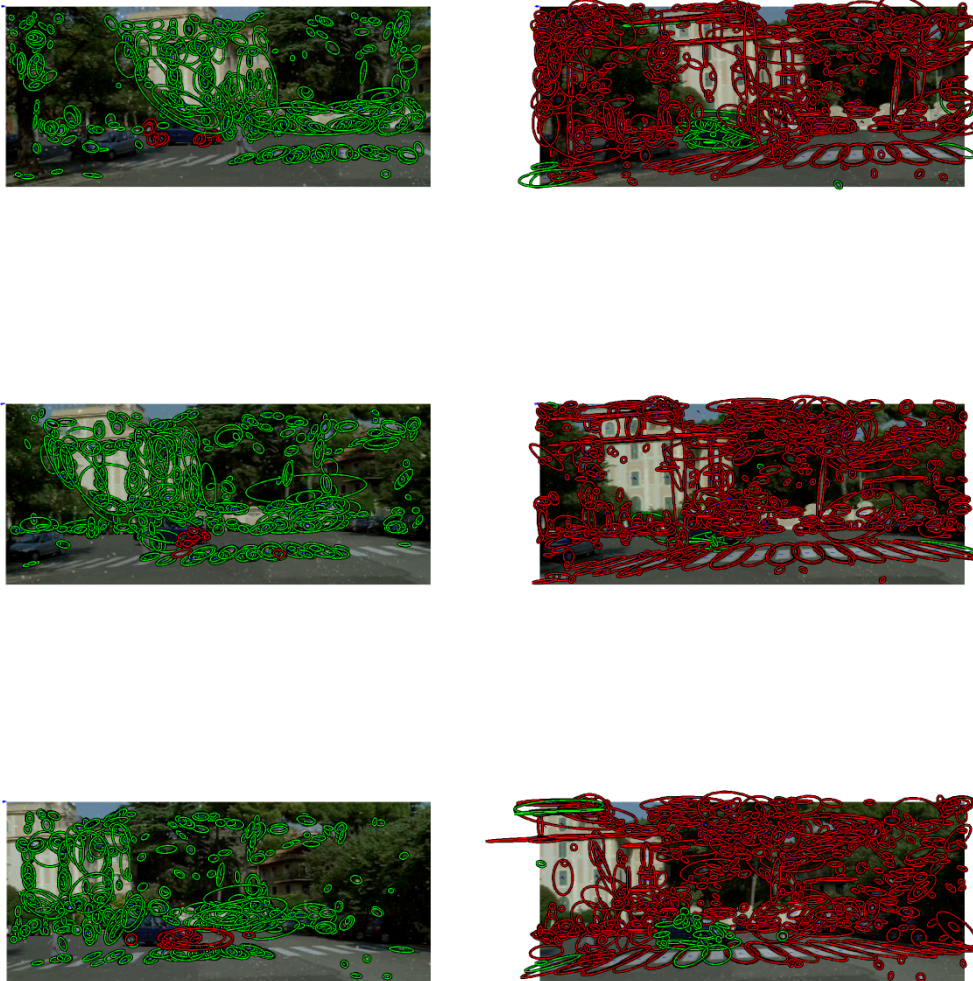


Figura 6.15: Resultat de la prova de la seqüència de la pel·lícula “El tigre y la nieve”

Els motius per els quals no es detecten les dues persones són els següents. Primer de tot, les característiques d'aquestes no són les adequades, o sigui, no estan prou ben definides i els detectors no aconsegueixen trobar cap regió dins seu. Això podria ser degut a que van vestides de gris uniformement. Llavors les regions detectades en aquestes persones són les del voltant seu. Això es nefast per l'algorisme ja que aquestes regions canvien a cada frame, i és pràcticament impossible seguir el seu moviment.

Concloem doncs que és molt difícil detectar el moviment d'objectes que no queden suficientment definits i només obtenim regions en el seu contorn. L'algorisme és incapaç de fer una detecció robusta en aquests casos, i la detecció casual de punts, per afegir-los després a través del veïnatge, no ens serveix sense una base.

6.2.5 Seqüència “Me llamo Earl”

En aquesta prova, al igual que en la prova anterior intentarem separa dos objectes diferents entre ells i del fons. S’ha escollit una escena on els objectes tenen una mica més de detall intern per que els detectors aconseguixin distingir-los correctament. És una escena de la sèrie de televisió “*Me llamo Earl*” on es veu a quatre persones, agrupades de dos en dos, desplaçar-se cap a dues direccions oposades. En la figura 6.16 podem veure les imatges inicials i finals de l’escena.



Figura 6.16: Escena de la seqüència de la prova corresponent a “Me llamo Earl”

El resultat d’aquesta seqüència hauria de ressaltar les dues persones de l’esquerra d’un color, les dues persones de la dreta d’un altre i el fons d’un tercer. Com es pot veure a la figura 6.17 el resultat final és l’esperat en gairebé tots els frames. Només s’ha fet servir el detector *MSER* per aquesta prova.

Veiem doncs en la figura 6.17 que els dos objectes ressalten sobre el fons en colors diferents, per tant podem dir que el resultat és bo. En la primera imatge de la figura 6.17 veiem però que no es detecten correctament els grups que toquen. Això és degut a que és l’inici de la seqüència i encara no s’ha definit del tot el moviment de cada objecte. En algun dels frames de l’escena es perden regions importants per després poder fer l’agrupament per veïnatge, i això fa que algun dels frames no surti bé del tot. Això s’arregla gairebé al següent frame quan ja s’emplena el vector de moviment d’aquests punts.

Per concloure aquests experiments direm doncs que en general hem obtingut un resultat relativament bo. Concloem doncs que l’algorisme té un resultat millor en proves sintètiques i que malgrat que és una part molt important no s’ha acabat d’aconseguir que funcioni per múltiples objectes. Només s’ha arribat a diferenciar-ne dos i el fons i no en totes les escenes. Separar qualsevol objecte del fons però, si que és una tasca que l’algorisme pot aconseguir de manera satisfactòria.

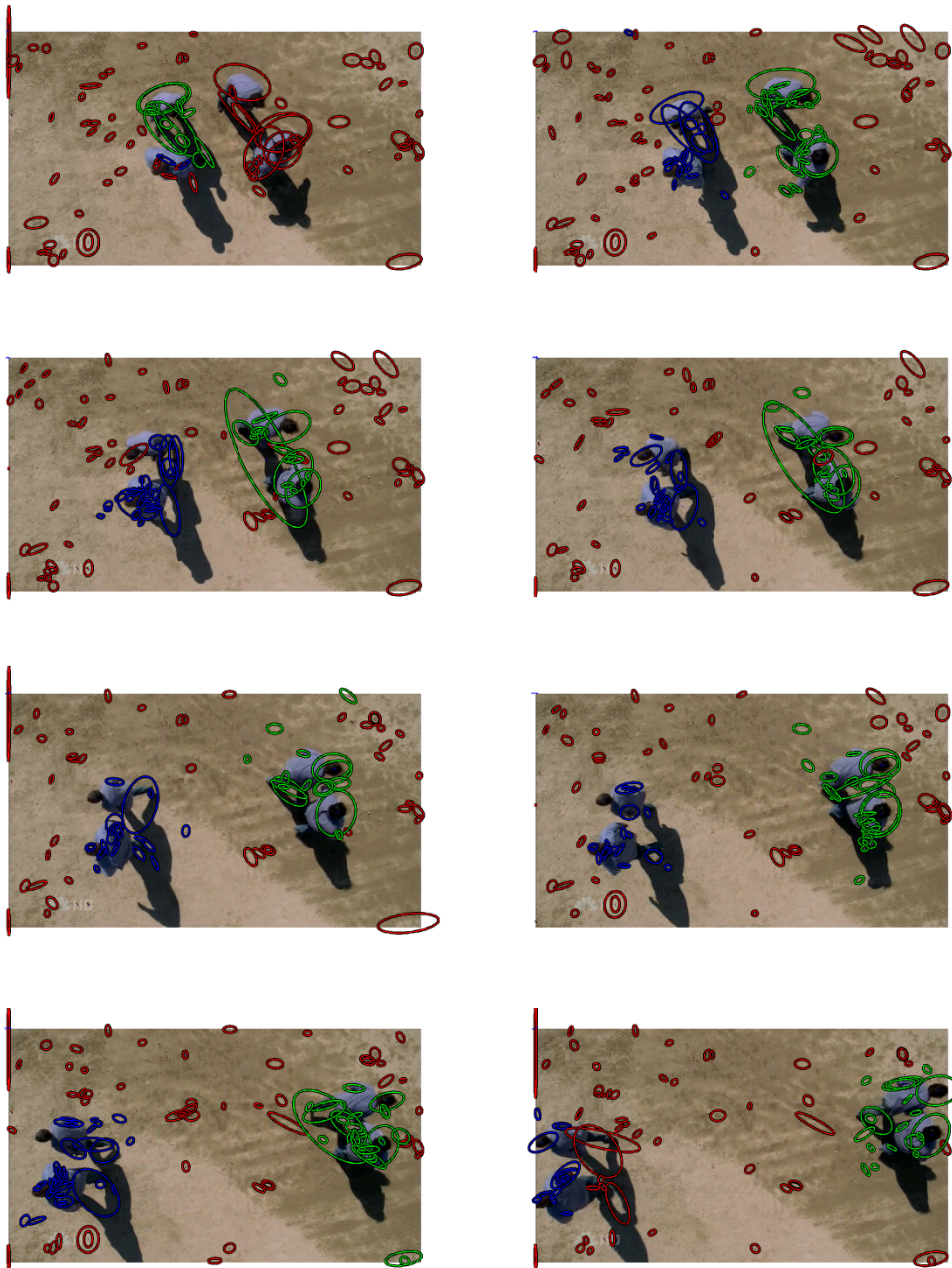


Figura 6.17: Resultats de la prova sobre la seqüència d'una escena de la sèrie "Me llamo Earl"

Capítol 7

Conclusions i treballs futurs

7.1 Conclusions

En aquest apartat exposarem la opinió que ens aporta el projecte final de carrera i les impressions que ens ha deixat. També parlarem de les conclusions dels resultats finals i que es pot fer en el futur.

Podem dir que s'han aconseguit quasi tots els objectius proposats en el projecte, encara que hem agut de deixar-ne algun. Ens hem introduït en el món de la Visió per Computador i hem après les tècniques per segmentar objectes en vídeos. Concretament hem après com s'obtenen característiques de les imatges, com funcionen els algorismes de *tracking* i n'hem implementat un de propi enfocat a la visió global del projecte. També hem vist quins problemes porta tot això i quines solucions es poden implementar per solventar-los i seguir endavant.

El reconeixement d'objectes en vídeos de la branca de Visió per Computador és un tema en que no hi ha tantes investigacions com en el reconeixement d'objectes en imatges. Tot i que el resultat final d'aquest projecte, en sí, no és ni tant eficient ni tant bo que es podia haver esperat, no és més que una fase en el desenvolupament d'aplicacions relacionades amb la detecció d'objectes, i combinant aquesta tècnica exposada amb altres que s'han explorat més, es poden aconseguir uns resultats molt més satisfactoris.

Cal fer un incís en el temps computacional de l'algorisme. Totes les proves s'han executat sobre una màquina virtual, cosa que fa que aquest creixi. En una màquina dedicada al càlcul és possible que s'obtinguessin millors resultats. Tot i així gracies a diverses tècniques utilitzades, s'ha aconseguit reduir el temps de processament significativament.

7.2 Treballs futurs

En aquest apartat explicarem possibles modificacions futures de l'algorisme que no s'han pogut dur a terme en aquest projecte. Durant el seu desenvolupament han anat sorgint diverses idees enfronts als diferents problemes, així com també altres idees per futures aplicacions, ja sigui la expansió d'aquesta o la seva incorporació en altres projectes.

- Primer de tot i quasi la més important és que en el nostre projecte, totes les proves s'han realitzant coneixent i imposant quants objectes hi ha a l'escena. Per tant seria molt millor, utilitzar algun algorisme d'estimació de clústers per determinar-ne el nombre i veure així com varien els resultats.
- Caldria veure també, si es volgués utilitzar l'algorisme en temps real, què faria falta millorar i com es podria fer. El temps de càlcul esta repartit de forma similar entre les fases, però s'eleva amb quantes més regions es detectes.
- La idea general és que el moviment dels objectes ens pot ajudar a reconèixer-los, però al igual que en el cervell humà, fa falta més informació per arribar reconèixer satisfactòriament un objecte. Així que és una possible idea integrar la informació que dona aquest tipus de reconeixement, per moviment, amb tot el que s'ha estudiat fins ara. Creant així algorismes molt més robustos.
- La opció més immediata seria la implementació d'aquest algorisme en llenguatge C. Podríem veure com el rendiment milloraria molt i es podria compara els resultats o temps d'execució entre les dues opcions.
- Quan un objecte deixa de moure's, l'algorisme el perd, ja que no hi ha moviment per classificar, una opció futura seria la de recuperar objectes que s'han deixat de moure un cop ho tornen a fer.

Bibliografia

- [1] H. P. Moravec. Towards automatic visual obstacle avoidance. In Proc. of the International Joint Conference on Artificial Intelligence, 1977.
- [2] University of Central Florida, Computer Vision Lab, Sohaib Khan's Research
<http://www.cs.ucf.edu/~khan/video-segmentation.htm>
- [3] Computer System Support, College of Engineering, The University of Iowa
<http://www.icaen.uiowa.edu/~dip/LECTURE/Segmentation3.html>
- [4] C. Harris and M. Stephens. In Proc. of The Fourth Alvey Vision Conference, Manchester, pp 147-151, 1988.
- [5] K. Mikolajczyk and C. Schmid. Scale and Affine invariant interest point detectors. In IJCV 60(1):63-86, 2004.
- [6] A. Baumberg. Reliable feature matching across widely separated views. In Proc. of CVPR, pp. 774-781, 2000.
- [7] T. Lindeberg and J. Garding. Shape-adapted smoothing in estimation of 3-D depth cues from affine distortions of local 2-D brightness structure. In Image and Vision Computing, vol. 15, pp. 415-434, 1997.
- [8] Edelman, S., N. Intrator, and T. Poggio. Complex cells and object recognition, unpublished manuscript.
- [9] D. Lowe. Distinctive image features from scale invariant keypoints. In IJCV 60(2):91-110, 2004.
- [10] Vision Geometry Group. Affine Covariant Region Detectors and Region Descriptors.
<http://www.robots.ox.ac.uk/~vgg/research/affine/detectors.html#binaries>
<http://www.robots.ox.ac.uk/~vgg/research/affine/descriptors.html#binaries>
Visitat tardor del 2007, primavera del 2008 i estiu del 2008

- [11] The MathWorks - MATLAB and Simulink for Technical Computing.
<http://mathworks.com/> Visitat primavera del 2008 i estiu del 2008.
- [12] Wikipedia - The Free Encyclopedia.
<http://en.wikipedia.org/>
<http://es.wikipedia.org/>
Visitat tardor del 2007, primavera del 2008 i estiu del 2008.
- [13] VirtualDub. <http://www.virtualdub.org/> Visitat tardor del 2007.
- [14] Donovan H. Parks, Centre for Intelligent Machines.
<http://www.cim.mcgill.ca/~dparks/CornerDetector/mainMoravec.htm> Visitat tardor del 2007.
- [15] Sedgewick, R. 1988. *Algorithms*, 2nd edition. Addison-Wesley.
- [16] Prinzie A., D. Van den Poel (2006), Incorporating sequential information into traditional classification models by using an element/position-sensitive SAM. *Decision Support Systems* 42 (2): 508-526.
- [17] MacQueen, J. B. (1967). Some Methods for classification and Analysis of Multivariate Observations, Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, University of California Press, 1:281-297
- [18] Dipartimento Di Elettronica e Informazione, Politecnico Di Milano, K-means - Interactive demo.
http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/AppletKM.html
- [19] Linda G. Shapiro i George C. Stockman (2001): "*Computer Vision*", pp 279-325, New Jersey, Prentice-Hall, ISBN 0-13-030796-3.
- [20] Joan Martí, Jordi Freixenet, Xavier Lladó, Xevi Muñoz, Xevi Cufí. *A new Approach to Natural Object Labelling in Colour Aerial Images*. Proc. of the IX Simposium Nacional en Reconocimiento de Formas y Análisis de Imagen (SNRFAI). Vol. I, pp. 67-72. Benicàssim (Castelló). Maig 2001.
- [21] J. Martí, J. Freixenet, J. Batlle, A. Casals. *A new approach to outdoor scene description based on learning and top-down segmentation*. Image And Vision Computing. Elsevier Science. 19(14): pp1041-1055, Dec. 2001.
- [22] X.Lladó, J.Freixenet, J.Martí, J.Forest, J.Salvi. *Reconeixement D'objectes Naturals En Escenes D'exteriors Basat En Una Estratègia Bottom-Up*. 2on Congrés Català d'Intel·ligència Artificial, Girona, 1999, Pp: 299-307.