Facultat de Matemàtiques
Universitat de Barcelona

ENGINYERIA TÈCNICA EN
INFORMÀTICA DE SISTEMES

Projecte final carrera

Detecció i extracció d'avions a seqüències de vídeo

Marc Garcia i Ramis

Directors: Sergio Escalera Guerrero
Carlos Gallardo García
Alberto Escudero Pardo

Realitzat a: Departament de Matemàtica
Aplicada y Anàlisi. UB

Agraïments

En primer lloc, vull agrair tot l'esforç que ha realitzat al meu tutor de projecte Sergio Escalera, també director i creador del projecte.

Com no mencionar a Alberto Escudero un programador amb molts coneixements i millor persona, que m'ha donat tant i tant suport. També felicitar al meu company Francesc per tota la excel·lent feina paral·lela que ha realitzat.

Sense vosaltres hagués estat impossible, gràcies de tot cor. Ha estat un plaer treballar amb vostès com a final d'aquesta etapa de la meva vida.

Resum

A l'actualitat, cada cop és més usual trobar aplicacions relacionades amb el tractament d'imatges per vídeo, com ara: càmeres de seguretat o sistemes de reconeixement facial amb webcams. Seguint aquesta línea, hem desenvolupat una eina per facilitar el treball de pista als operaris que guien als avions per aparcar. Aquest projecte constaria d'una càmera que amb el codi, aquí presentat, podria detectar un avió en moviment. A més, com a projecte paral·lel el Francesc Xavier Muñoz Romero ha desenvolupat el codi per identificar el model d'un l'aeroplà i una interfície gràfica per una millor configuració a pista.

Resumen

En la actualidad, cada vez es más frecuente encontrar aplicaciones relacionadas con el tratamiento de imágenes por video, como: cámaras de seguridad o sistemas de reconocimiento facial con webcams. Por tanto, hemos desarrollado una herramienta para facilitar el trabajo de los operarios que guían a los aviones al aparcar. Este proyecto contaría de una cámara que con el código, aquí presentado, podría detectar un avión en movimiento. Además, como proyecto paralelo, el Francesc Xavier Muñoz Romero ha desarrollado el código para identificar el modelo de un aeroplano y una interficie gráfica para mejor configuración en la pista.

Abstract

Currently, it is common to find applications related to video image processing, such as: security cameras and facial recognition systems with webcams. We have therefore developed a tool to facilitate the work of the operators who guide the planes to park. This project would have a camera with the code presented here being able of detecting a moving airplane. Francesc Xavier Muñoz Romero has developed in parallel the code to identify the model of an airplane and a graphic interface for a better configuration.

Index

1 – Introducció pàg. 5 - 7

2 – Anàlisi
- 2.1 – Planificació i costs pàg. 8 - 9
- 2.2 – Anàlisi hardware pàg. 9 - 10
- 2.3 – Metodologia i algorítmica pàg. 10 - 19
- 2.4 – Morfologia i blobs pàg. 19 - 22
- 2.5 – Requeriments funcionals pàg. 22 - 23

3 - Disseny
- 3.1 - Diagrama classes pàg. 24 - 26
- 3.2 - Tipus arxius
  - 3.2.1 – hpp pàg. 26 - 28
  - 3.2.2 – cpp pàg. 28 - 33
- 3.3 – Test avions pàg. 34

4 – Resultats
- 4.1 – Resultats per vídeos diürns pàg. 35 – 41
- 4.2 – Resultats per vídeos nocturns pàg. 42 – 47
- 4.3 – Resultats per vídeos del vespre pàg. 48 – 51
- 4.4 – Comparació de resultats pàg. 51 – 56

5 - Conclusions
- 5.1 – Avaluació dels mètodes pàg. 57 - 58
- 5.2 – Propostes de millora i solucions pàg. 58 – 59

6 – Referències pàg. 60

7 – Apèndix pàg. 61

1 - Introducció

Des de ja fa uns anys, les aplicacions informàtiques relacionades amb el tractament d'imatge estan assumint major presència a les noves generacions d'aparells electrònics.

Com ara la integració del software a les càmeres fotogràfiques compactes per disparar si detecten una determinada situació. *Smile shutter* és una tecnologia amb la que Sony va innovar al 2007 amb els seus models Sony Cyber.Shot DSC-T70 i DSC-T200 amb les quals es pot programar que dispari automàticament en cas de detectar un somriure. A la imatge Fig.1.1 podem veure un exemple, fins i tot mesurant la intensitat del riure.

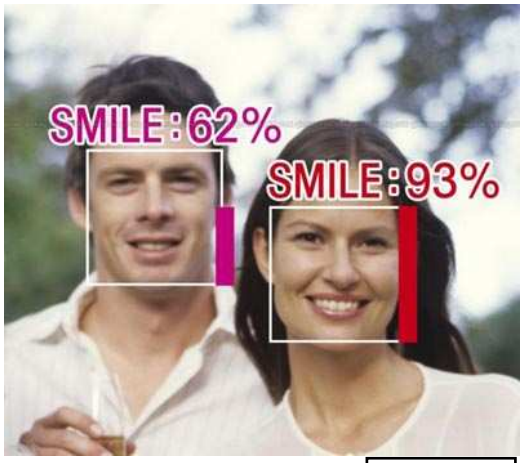


Fig.1.1

També portals web amb tant renom com Youtube o Google han inclòs eines vinculades amb el reconeixement de imatges. D'una banda, el major servidor de vídeos de la xarxa, Youtube, ja fa anys que té diferents programes de reconeixement d'imatges per eliminar vídeos amb continguts pornogràfics o que no respectin drets d'autor (a partir d'una base de dades proporcionada pels propis interessats). D'altra banda, Google al 2006 va adquirir una empresa californiana anomenada Never Vision, que va passar a encarregar-se d'afegir eines de cerca, de seguretat i verificació geomètrica al gestor de fotografies Picassa. A més de les ja anomenades, podem trobar moltes altres utilitats com ara el reconeixement facial en temes de seguretat, cada cop més emprat com substitut de la contrasenya escrita. O les ja famoses càmeres de la policia per detecció de vehicles a grans velocitats, capaces de llegir una matrícula amb detall suficient com per identificar al infractor.

Totes aquestes aplicacions estan clarament relacionades tot i presentar nombroses diferències a nivell algorítmic, però en essència podríem distingir unes passes bàsiques a seguir. En primer lloc, seleccionarem el fons, *background*, com allò que no presenta moviment. Segon, tot el que presenti canvis de color sobre la nostra imatge de *background* representarà un moviment. Aleshores, ens queda seleccionar la nostra imatge en moviment per mida, regió, patrons preestablerts, entre d'altres. A la imatge Fig.1.2 podem veure un clar exemple. Apareix en negre tot el que és el fons, donat que no presenta moviment i en blanc el cotxe.

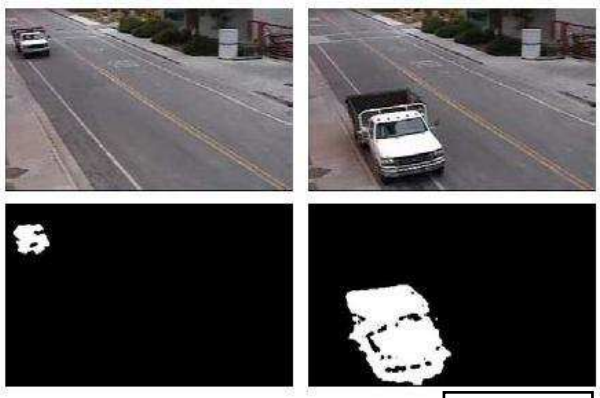


Fig.1.2

Un cop arribat a aquest punt és inevitable imaginar molts altres treballs que amb una solució basada en aquesta tecnologia facilitarien la feina i millorarien els resultats. Així doncs, aquí presentem una proposta innovadora que es podria implementar al món dels aeroports.

Fins al moment, per aparcar un avió es fa mitjançant un operari que guia els pilots mentre aquests mantenen la màquina seguint una línia groga. En aquest projecte, hem implementat el software necessari per identificar un avió en moviment que està aparcant. A més a més, també és capaç de rebutjar qualsevol altre moviment que no ens interessi: persones caminant, altres vehicles, ocells... De manera, que amb molt poc material els avions podrien aparcar de forma autònoma i sense necessitat que l'operari estigui de forma presencial. Donat que la quantitat de personal sol ser ajustada i, en temporada alta, amb una gran quantitat d'arribades alhora.

Fig.1.3

A la següent imatge, Fig.1.3, podem observar com un operari està ajudant a aparcar l'avió. El punt indicat amb el cercle seria el lloc on s'hauria d'instal·lar el dispositiu, de manera que tindria una captura perfecta de l'arribada de l'avió sobre la línia groga i, d'aquesta forma, mecanitzar l'estacionament.

Com ja hem assenyalat abans, gràcies al projecte complementari del Sr. Francesc Muñoz Romero es podria determinar el model d'avió i així optimitzar l'espai i les característiques necessàries per aparcar un model en concret.

En els pròxims apartats explicarem amb tota claredat quina és la nostra proposta amb els detalls que això suposa.

Al capítol de Anàlisi, mostrarem la planificació que s'ha fet per avaluar i resoldre l'objectiu. I el pressupost calculat en els materials i tecnologia que caldrien per dur-ho tot a terme. A més d'una explicació detallada dels algorismes emprats.

A la secció Disseny, mostrarem la implementació del software i l'explicarem detalladament amb ajuda de taules i diagrames. Per tal de mostrar el funcionament del codi implementat.

També presentarem els corol·laris obtinguts a totes les proves, al capítol Resultats. Com és obvi, aquestes proves seran realitzades amb gravacions de característiques diferents, com ara: de nit, amb excés de llum, amb diferents models d'avió, interferències de gent o cotxes passant

per davant la càmera, entre altres; per assegurar la màxima eficiència en totes les situacions possibles.

A continuació, a l'apartat Conclusions, detallarem quines són les nostres conclusions segons les propostes i resultats obtinguts. Determinant així quina seria la millor opció segons el cas.

Per acabar, anomenarem quines han estat les referències consultades per obtenir la informació idònia i realitzar tot aquest estudi. I al capítol de clausura, Apèndix, exposarem el contingut del CD adjuntat on compactarem totes les dades aquí presentades: vídeos, imatges, codi, memòria, etc...

2 - Anàlisi

Aquesta aplicació ha estat programada en el llenguatge C++ i per facilitar l'ús a l'hora de incloure llibreries i treballar sobre el codi ho hem plantejat des de la interfície Microsoft Visual Studio ©.

Com ja hem explicat breument a la introducció la tècnica de Background segmentation es basa en aconseguir separar el primer pla (l'objecte en qüestió) del segon (fons). Fonamentalment la tasca haurà de seguir i reconèixer l'objecte. Cal assenyalar que al nostre cas la escena és estàtica i *coneguda*.

2.1 – Planificació i costs

En primer lloc, a la següent taula (Fig. 2.1) mostrarem quin temps dedicaríem a cada fase del projecte.

Tasca	Data inici	Duració (dies)	Data final
Anàlisi del problema	06/09/2010	3	09/09/2010
Solucions al problema	10/09/2010	4	15/09/2010
Disseny	16/09/2010	15	06/10/2010
Implementació	06/10/2010	30	19/11/2010
Període de test	22/11/2010	10	03/11/2010
Documentació	06/12/2010	13	22/12/2010

Fig.2.1

Com podem observar, aquest projecte tindria un temps aproximat de 4 mesos, més concretament un total de 75 dies laborables. Tot i que sempre s'han de comptar uns dies per si sorgeixen imprevistos, aquest temps pareix més que suficient.

A la primera fase, dedicaríem un total de 3 dies per citar-nos amb el client i poder conèixer amb més detalls quins són els conflictes i poder focalitzar la nostra aplicació per cobrir aquella tasca d'una forma eficient.

A continuació, passaríem 4 dies per decidir les possibles solucions. A més hauríem de prendre les decisions inicials, com ara: entorn de treball, materials necessaris (vídeos d'avions), etc... També seria la fase on el client coneixeria aquests costos i la planificació del temps.

Amb el següent pas definiríem amb completa profunditat com seria la nostra aplicació: diagrames de classes, diagrames d'ús, algorítmica, entre altres. Cal assenyalar d'una forma especial aquesta etapa del projecte donat que serà la que marcarà el ritme i la consistència del treball, i que si es fa de manera correcte facilitarà tota la feina posterior.

Per continuar, implementaríem el codi. Gràcies a les fases anteriors aquesta etapa hauria de ser bastant concreta i relativament ràpida. Però també és la fase en la que es troben contratemps inesperats, això fa que ho haguem determinat en 30 dies donat que es podria allargar.

En darrer lloc, només quedaria fer les proves necessàries a nivell de pista, comprovant que l'aplicació resol exactament la tasca requerida i contrastar amb l'usuari la usabilitat. Seria el moment per fer els darrers

retocs, com ara de la interfície gràfica per fer-la més còmode/intuïtiva per l'usuari.

Per acabar, finalitzaríem la documentació i la entregaríem.

A continuació, mostrem d'una forma gràfica, diagrama de Gannt (Fig. 2.2), la distribució del temps:

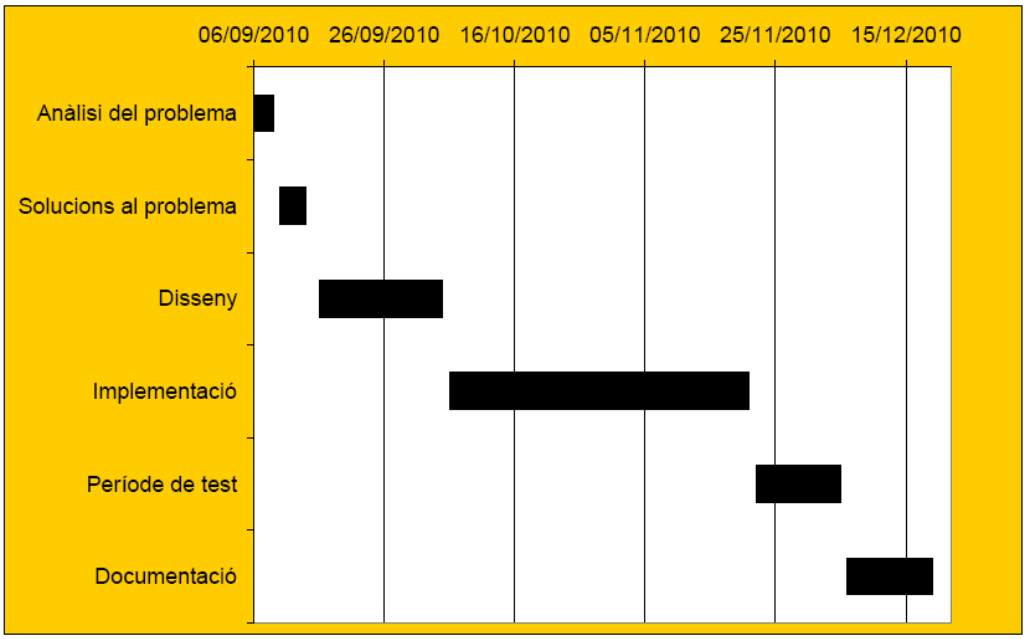


Fig.2.2

Finalment, ens quedaria calcular les despeses econòmiques (Fig. 2.3) tenint en compte les hores de treball dels programadors, llicències de la interfície i material informàtic:

Concepte	Quantitat	Preu	IVA	Preu final
Llicència Visual Studio	1	713,15 €	114,10 €	827,25 €
Material informàtic	1	1.000 €	160 €	1.160,00 €
Hores de treball	600	17€/h	-	10.200,00 €
			Total	12.187,25 €

Fig.2.3

2.2 – Anàlisi hardware

Tota la informació dels aeroports es troba a una base de dades coneguda com AODB Airport Operational Data Base que junt amb un canal de comunicacions d'AENA, llenguatge AOML i basat en esquemes XML, que amb un bus de comunicació TIBCO assegura el transport de la informació. De manera que aquest és l'esquelet de la comunicació aeroportuaria.

Així doncs obtenint la informació de tots els punts, el nostre agent podria conèixer la programació de vols i models que se dirigeixen a la plaça d'estacionament en concret.

Finalment, la sortida del sistema es connecta a la guia d'estacionament i permet la introducció del model d'aeroplà reconegut automàticament per

poder accionar la guia. Tot això es veu representat a la següent imatge (Fig.2.4).

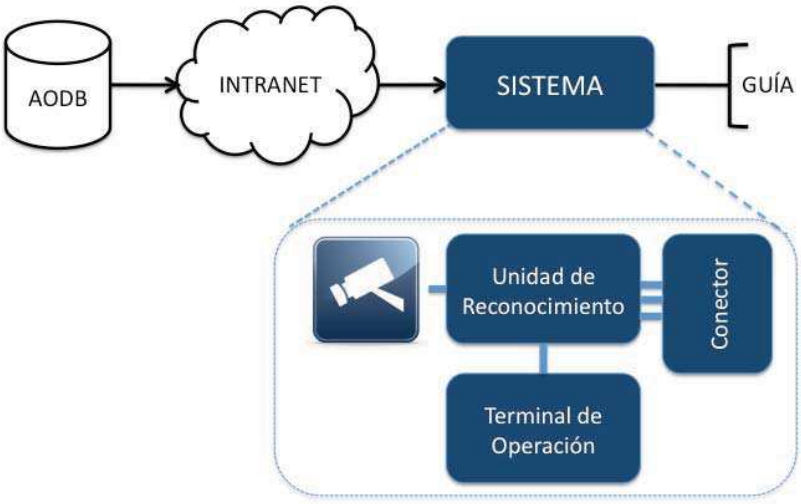


Fig.2.4

Amb més profunditat, a nivell de hardware podem dividir el nostre projecte en quatre parts fonamentals:

- **Sistema de detecció d'apropament:** amb una càmera digital de poca resolució bastaria per detectar l'arribada d'un avió a la nostra zona i així prevenir o activar la resta del sistema.
- **Unitat de reconeixement:** un PC implementat i ajustat amb la tasca única i exclusiva d'enregistrar i processar a temps real el vídeo per poder identificar el model d'aeroplà.
- **Terminal d'operació:** sistema amb una interfície per permetre a l'operari certificar la correcta detecció de la nau.
- **Connexió amb la guia:** Interfície per permetre retornar el model d'avió determinat i activar la senyalització pertinent.

2.3 – Metodologia i algorítmica

A continuació explicarem els diferents mètodes que hem desenvolupat i emprat des de la idea més bàsica fins al seu algoritme matemàtic pur, i aclarirem com cada un d'ells resol la nostra problemàtica. Molts d'aquests mètodes es basen en característiques estàtiques, com ara: intensitat o color, per modelar cada píxel de forma independent. El clar avantatge que això ens proporciona és la detallada silueta obtinguda. Per contra, aquests mètodes són sensibles a canvis d'iluminació, *renou* i moviments del fons.

Cal assenyalar que el nom d'algun dels següents algoritmes és el nom del seu creador. En la seva majoria investigadors.

Al següent capítol, Disseny, podrem veure tota l'estructuració del codi, com funcionen i la implementació d'aquests mètodes.

Mètode de Eigen Background-substraction [1]

Eigen que ve de l'alemany, significa propi. Donat que un frame conté la seva informació d'una manera matricial, en aquest mètode ens servim dels eigenvectors i els seus eigenvalues que anirem obtenint de cada pas. Un eigenvector (o vector propi) és aquell que no es veu afectat per les transformacions lineals i que en qualsevol cas no varia la seva direcció tot i ser multiplicat per un escalar. Alhora, un eigenvalue (o valor propi) serà aquell escalar pel qual es multiplica el eigenvector associat. Així doncs, aquest entorn és conegut com un eigenspace.

Com ja sabem, una imatge digital és una matriu amb una informació determinada. Aleshores anirem obtenint frame a frame aquesta matriu on es delimita un eigenspace. Cal assenyalar que la major part del pes d'una imatge està continguda en pocs eigenvalues, de manera que els de menys pes seran rebutjables.

Tot i que al principi d'aquest apartat avançàvem que molts mètodes es basen en dades estàtiques del píxel, Eigen en canvi es basa en dades estadístiques de l'escala de grisos sobre el temps.

A nivell matemàtic:

- Primer prenem *N* imatges d'exemple:

$$[I_t]_t = 1,2,\dots N$$

- D'aquestes obtenim la "mitjana" μ_b de l'imatge-fons, així com:

$[X_i]_i = 1,2,\dots N$ tal que $X_t = I_t - \mu_b$

- Aleshores calculem la matriu diagonal, L_b, amb els eigen values a partir de:

$$L_b = \Phi_b C_b \Phi_b^T$$

on C_b és la matriu de covariància i Φ_b l'eigenvector calculat amb la descomposició de valors singulars (SVD).

A partir d'aquest punt només treballarem amb els *M* eigenvalues de major pes, ΦM_b.

- Per cada imatge nova, I_t, la projectarem sobre l'espai dimensional més baix d'eigenbackgrounds, de manera que només es remarcaran els punts estàtics de la imatge.

$$I_t^B = \Phi_{Mb}(I_t - \mu_b)$$

- Finalment, podrem determinar l'objecte comparant diferències entre la imatge nova i el background previ sobre un threshold θ:

$$D_t = \left| I_t - I_t^B \right| > \theta$$

Pros: en comparació amb els altres mètodes aquí exposats, els fonamentats en càlculs amb la densitat o color del píxel, Eigen és pot considerar més senzill i ràpid.

Cons: Perdria precisió i seria més dificultós treballar amb dades a temps real donat a l'alt cost computacional de l'algoritme.

Mètode de Mean [1]

Aquesta funcionalitat aprofitarà, com ja hem parlat de forma general, les característiques estàtiques per modelar cada píxel a nivell individual. A cada iteració els valors Gaussians adaptats seran avaluats de forma heurística i els píxel que no coincideixin amb els del fons "Gaussià" seran classificat com a primer pla.

Així doncs, aquest mètode ,i alguns dels següents, es basarà en el pes estimat de la densitat, és a dir, en la probabilitat d'una intensitat o color de cada píxel.

A nivell matemàtic:

- Captem *N* frames sense l'objecte a detectar:

$$[I_t]_t = 1,2,\dots N$$

- Amb aquestes construirem el model de probabilitat de densitat (*density probability model*) . Crearem aquest espai amb una normalització espacial (*r*,*g*). On:

$$r = R/(R+G+B) \qquad g = G/(R+G+B)$$

- Dels frames d'aprenentatge, obtindrem el paràmetre del model composat per un píxel i de l'espai (*r*,*g*), a partir de la mitjana μ_i i la covariància σ_i.

Quan apareix un nou píxel I_t a la nostra distribució gaussiana, actualitzem els paràmetres en aquell punt, tal que:

$$\mu_{i,t} = (1-\alpha)\mu_{i,t-1} + \alpha(I_{i,t})$$
$$\sigma^2_{i,t} = (1-\alpha)\sigma^2_{i,t-1} + \alpha(I_{i,t}-\mu_{i,t})^T(I_{i,t}-\mu_{i,t})$$

On α serà la taxa d'aprenentatge, que determina la velocitat dels paràmetres distribuïts respecte al canvi.

Pros: Cal anotar que al fer-ho pels diferents canals redueix l'esforç de còmput. Detectarà de forma eficient píxels-ombra (*shadow-pixel*).

Cons: Alta sensibilitat a les variacions de llum i brill, apareixeran gran quantitat de píxels morts, soroll.

Mètode Adaptive [1]

Aquest mètode és el resultat de la combinació dels dos mètodes anteriors. Treballa de manera que si detecta variacions considerables d'intensitat ens centrarem més al model Eigen, per incrementar l'estabilitat enfront als canvis de lluminositat; mentre que ens centrarem en el model mean per detectar els canvis als canals de color R,G,B. Aquesta compensació entre els 2 mètodes es realitza mitjançant la següent algorítmica:

- D_i *(E)* és el cost de fons calculat pel mètode Eigen per cada píxel, tal que:

$$D_t = \left| I_t - I_t^B \right| > \theta$$

- Aleshores podrem calcular $D_i(B)$ que correspon a la combinació de costs de cada píxel que pertanyi al fons, mitjançant:

$$D_i(B) = \lambda D_i(E) + (1 - \lambda)$$

- Així doncs, calcularem $D_i(G)$, que és el cost de fons mitjançant l'ús del mètode adaptatiu, segons:

$$D_i(E)\, D_i(G) = \left| I_i - B_i \right|$$

$$D_i(G) = \frac{(I_{r,i} - \mu_r)^2}{\sigma_r^{\,2}} + \frac{(I_{g,i} - \mu_g)^2}{\sigma_g^{\,2}}$$

On I_i, I_r,i y I_g,i són la il·luminació i els valors r,g normalitzats, respectivament.

- I finalment:

$$\lambda = 1 - \exp\left(\frac{-(I_i - \mu_i)^2}{\eta^2}\right)$$

On λ és el factor de pèrdua que reflexa la confidència a l'Eigenspace. η és una constant predefinida que modela el soroll.

Cal assenyalar que quan el valor d'intensitat I_i és considerablement petit i la imatge d'entrada és obscura, seria més òptim utilitzar només $D_i(E)$, donat que el valor (R+G+B) seria tan petit que es perdria precisió.

Mètode Grimson [2]

En aquest mètode únicament es modelen els valors del píxel per individual com a mescla dels valors Gaussians. Es basa en la persistència i la variància de cada un dels Gaussians de la composició, i d'aquests determinarem quins colors pertanyen al fons, *background*. Cada píxel resultant, d'una superfícies particular, sota una lluminositat concreta basta per modelar el valor Gaussià del píxel.

A nivell pràctic, apareixeran múltiples superfícies en la "vista troncal" (*view frustum*) d'un píxel amb condicions de llum canviant. Aquestes múltiples superfícies són necessàries donat que s'utilitzaran en la barreja Gaussiana a l'aproximació durant el procés.

Així doncs cada vegada que actualitzem els paràmetres Gaussians s'avaluaran, amb una simple heurística, per deduir quins pareixen part del fons. Així doncs tots els píxels que no quadrin amb el fons seran units, i formaran el component *foreground*.

A nivell matemàtic:

- Volem avaluar un històric pels valors de cada píxel. Aquesta informació la podem obtenir en qualsevol moment, per un píxel $\{x_0,y_0\}$ és:

$$\{X_1, ..., X_t\} = \{I(x_0, y_0, i) : 1 \leq i \leq t\}$$

On *t* indica temps i *I* el frame.

- Aquesta història, $\{X_1,...,X_t\}$, la modelarem amb una barreja de *K* distribucions Gaussianes. La probabilitat d'observar el valor del píxel actual és:

$$P(X_t) = \sum_{i=1}^{K} \omega_{i,t} * \eta(X_t, \mu_{i,t}, \Sigma_{i,t})$$

On *K* és el nombre de distribucions, $\omega_{i,t}$, és una aproximació del pes i del i^{th} Gaussià de la barreja al moment *t*. $\mu_{i,t}$ la mitjana del i^{th} Gaussià de la barreja al moment *t*. El sumatori $\Sigma_{i,t}$ refereix a la covariància .

- A continuació amb η, calcularem la probabilitat de densitat

$$\eta(X_t, \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(X_t-\mu_t)^T \Sigma^{-1} (X_t-\mu_t)}$$

Cal assenyalar que *K* vendrà condicionada pel poder computacional i la memòria disponible.

- Per motius computacionals la matriu de covariància por assumir la forma:

$$\Sigma_{k,t} = \sigma_k^2 \mathbf{I}$$

En aquest cas, estarem assumint que tant *R,G* com *B* són independents i tenen la mateixa variància.

- Si cap del les *K* distribucions troba el píxel actual, se reemplaçarà la menys probable amb una amb el nou valor del píxel, una variància alta i pes baix. El pes de les *K* distribucions en un moment *t*, $\omega_{i,t}$, s'ajustarà:

$$\omega_{k,t} = (1-\alpha)\omega_{k,t-1} + \alpha(M_{k,t})$$

On α és el coeficient d'aprenentatge i $M_{k,t}$ serà 1 pels models trobats i 0 pels que estan en reserva.

- Finalment μ i σ per distribucions no trobades seran equivalents. Però per les que sí es troben la nova observació és modularà tal que:

$$\mu_t = (1-\rho)\mu_{t-1} + \rho X_t$$

$$\sigma_t^2 = (1-\rho)\sigma_{t-1}^2 + \rho(X_t-\mu_t)^T(X_t-\mu_t)$$

On:

$$\rho = \alpha\eta(X_t|\mu_k, \sigma_k)$$

Pros: Quan s'ha determinat que algo pot formar part del background, aquest no destruirà el model ja existent, si no que s'integrarà. (Com anotació puntual vull aclarir que es podria fer aquesta integració amb un nou color i així poder veure una evolució).

Cons: els objectes amb moviment lent tardarien més temps en ser detectats, perquè el seu color tindrà una variància més amplia que la del fons.

Mètode Wren [3]

En primer lloc, s'ha de fer constància que aquest mètode té una orientació i va ser dissenyat més concretament per la identificació de persones. Bàsicament aquest mètode presenta tres parts: primera, modelarà a l'objecte; segona, modelarà el fons; i , tercera, modelarem el moviment d'aquest objecte. Així doncs:

- Modelació de la persona. Els *clusters* amb punts 2D, tenen mitjanes de l'espai 2D (μ) i matrius de la covariància (K). Per descriure les estadístiques del objecte a l'espai utilitzarem un model Gaussià tal que:

$$\Pr(\mathbf{O}) = \frac{\exp\left[-\frac{1}{2}(\mathbf{O}-\mu)^T \mathbf{K}^{-1}(\mathbf{O}-\mu)\right]}{(2\pi)^{\frac{m}{2}} \left|\mathbf{K}\right|^{\frac{1}{2}}}$$

D'altra banda, píxel per píxel aplicarem el següent filtre de suport. Definim el filtre $s(x,y) = \text{argmax}_k(d_k(x,y))$ pel blob* k**.** Els seus resultats els tractarem tal que:

$$s_k(x,y) = \begin{cases} 1 & (x,y) \in k \\ 0 & \text{otherwise} \end{cases}$$

L'aplicació d'aquest filtre suposarà la separació dels canals per colors.

* Blob: (del anglès) bombolla. En aquest entorn s'utilitza per anomenar als objectes definits en una imatge *binària* (blanc i negre). Més endavant aprofundirem en aquest tema.

- Modelació de l'escena. Calcularem aplicant un valor mitjà del color (μ_0) per cada punt de la textura de l'objecte. La distribució (K_0) de cada píxel serà modelada mitjançant la matriu gaussiana de covariàncies. Així doncs per cada frame calcularem de manera recursiva les estadístiques pels píxels visibles amb el filtre:

$$\mu_t = \alpha \mathbf{y} + (1-\alpha)\mu_{t-1}$$

D'aquesta manera obtindrem el blob principal (o més d'un).

- Càlcul del moviment. El primer pas serà actualitzar el model espacial per cada blob usant el seu model dinàmic per la imatge actual:

$$\hat{\mathbf{X}}_{[n|n]} = \hat{\mathbf{X}}_{[n|n-1]} + \hat{\mathbf{G}}_{[n]}\left\{\hat{\mathbf{Y}}_{[n]} - \hat{\mathbf{X}}_{[n|n-1]}\right\}$$

On X és l'estat estimat del vector, posició i velocitat, $\hat{Y}$ les coordenades mitjanes del la posició i G la matriu Kalman del guany que assumeix una dinàmica Newtoniana.

A continuació calcularem les probabilitats de cada píxel de pertànyer a un blob o al fons. *y* serà el vector (x, y, Y, U, V) i *k* nombre d frame. Per tal d'eliminar les ombres farem $U^* = U/Y$, i $V^* = V/Y$, per la normalització, tal que:

$$d_k = -\frac{1}{2}\left(\mathbf{y}^* - \boldsymbol{\mu}_{\mathbf{k}}^*\right)^T \mathbf{K}_k^{*-1}\left(\mathbf{y}^* - \boldsymbol{\mu}_k^*\right) - \frac{1}{2}\ln\left|\mathbf{K}_k^*\right| - \frac{m}{2}\ln(2\pi)$$

- Finalment, per cada frame k estimarem un nou model:

$$\hat{\boldsymbol{\mu}}_k = E\left[\left(\mathbf{y} - \boldsymbol{\mu}_k\right)\left(\mathbf{y} - \boldsymbol{\mu}_k\right)^T\right]$$

i una nova distribució:

$$\hat{\mathbf{K}}_k = E\left[\left(\mathbf{y} - \boldsymbol{\mu}_k\right)\left(\mathbf{y} - \boldsymbol{\mu}_k\right)^T\right]$$

Pros: és un mètode força consistent, preparat per afrontar canvis d'iluminació i moviment ràpid.

Cons: el major inconvenient és el fet d'estar dissenyat per detecció de persones. A més, presenta tendència a no borrar elements que hagin estat en contacte o valors similars al blob principal.

Mètode Prati [4]

En aquesta funció s'ha treballat sobre la idea d'aconseguir gran precisió detectant objectes en moviment, a temps real. Per això, limitarem els falsos negatius, els píxels no detectats dels objectes. I a més, detallarem amb multa cura els píxels de l'objecte a extreure, evadint: objectes secundaris en moviment, ombres, renou, entre altres.

En aquesta explicació utilitzarem una nomenclatura específica:

· Objecte visual en moviment (*MVO*): conjunt de punts que pertanyen a l'objecte amb moviment no nul.

· Known object (*KO*): Objecte a conèixer.

· Background (*B*): fons.

· Ghost (*G*): conjunt de punts detectats en un moviment però que no corresponen al moviment d'un objecte real.

· Shadow: punts connectats de *B* modificats per una ombra projectada. Les ombres poden es poden classificar per: MVO_{sh} ombra relacionada amb *MVO*, per tant, també es mourà; G_{sh} ombra no connectada amb el moviment d'un objecte real.

* Comentari: Les ombres estàtiques se inclouran a *B*.

Procediment matemàtic:

- Donat un punt *p* a un temps *t* d'un frame, $I^t(p)$, obtindrem el vector del seu color (R,G,B).

- Entenem com *KO* a un moment t tal que:

$$\mathbf{KO}^{\mathbf{t}} = \{\mathbf{MVO}^{\mathbf{t}}\} \cup \{\mathbf{MVO}_{\mathbf{SH}}^{\mathbf{t}}\} \cup \{\mathbf{G}^{\mathbf{t}}\} \cup \{\mathbf{G}_{\mathbf{SH}}^{\mathbf{t}}\}$$

- A continuació avaluarem el punt p per associar-ho a *KO* o al *B*. Els primers valors es prediuen amb la informació estadística ($\mathbf{B_s^{t+\Delta t}}(p)$) del següent grup (*S*) d'elements:

$$S = \{\mathbf{I^t}(p), \mathbf{I^{t-\Delta t}}(p), \ldots, \mathbf{I^{t-n\Delta t}}(p)\} \cup w_b\{\mathbf{B^t}(p)\}$$

Aquest grup també inclou un valor obtingut dels *n* frames exemple amb valors de backgrounds passats amb un pes, ω_b. Aleshores dels *n* frames es traurà una submostra de l'original, un cada Δt.

- Així doncs el model de les dades estadístiques de *B* es calcularà segons:

$$\mathbf{B_s^{t+\Delta t}}(p) = \underset{i=1,\ldots,k}{arg\,min} \sum_{j=1}^{k} Distance(\mathbf{x_i}, \mathbf{x_j}) \quad \mathbf{x_i}, \mathbf{x_j} \in S,$$

On la distància serà *L-inf distance* de l'espai R,G,B:

$$Distance(\mathbf{x_i}, \mathbf{x_j}) = max(|x_i.c - x_j.c|) \quad \text{with} \quad \mathrm{c} = \mathrm{R}, \mathrm{G}, \mathrm{B}.$$

- D'altra banda, l'aprenentatge del background es farà mitjançant:

$$\mathbf{B_k^{t+\Delta t}}(\mathrm{p}) = \begin{cases} \mathbf{B^t}(\mathrm{p}) & if \;\; p \in \mathbf{O}, \mathbf{O} \text{ in } \{\mathbf{MVO^t}\} \cup \{\mathbf{MVO_{SH}^t}\} \\ \mathbf{B_s^{t+\Delta t}}(\mathrm{p}) & if\; p \in \mathbf{O}, \mathbf{O} \text{ in } \{\mathbf{G^t}\} \cup \{\mathbf{G_{SH}^t}\}. \end{cases}$$

De manera que l'actualització selectiva del background seguirà una selecció punt per punt tal que:

$$\mathbf{B^{t+\Delta t}}(p) = \begin{cases} \mathbf{B_s^{t+\Delta t}}(p) & if \;\nexists\, \mathbf{O} \in \mathbf{KO^t} : \; p \in \mathbf{O} \\ \mathbf{B_k^{t+\Delta t}}(p) & otherwise. \end{cases}$$

- Finalment, per millorar la detecció, la subtracció de *B* se calcularà prenent les dades cromàtiques en un punt de brill com a:

$$\mathbf{DB^t}(p) = Distance(\mathbf{I^t}(p), \mathbf{B^t}(p))$$

Cons: Aquesta metodologia presenta un punt crític al mesurar la distància. Si un objecte en moviment fa una pausa, prou llarga, s'integrarà amb el fons. Si després continua es detectarà com fantasma per la resta de futurs frames. De manera que evitaria que aquella àrea s'actualitzés tant pel *KO* com pel *B*... causant així un deadlock.

Pros: Una gran avantatge és que el model del background no es corromprà pels objectes en moviment. Això és gràcies a poder usar intervals Δt curts i *n* petites.

Mètode Zivkovic [5]

Seguint la trajectòria dels darrers mètodes, Zivkovic treballarà avaluant la una funció sobre la densitat per cada píxel individualment. Mitjançant un GMM (*Gaussian Mixture Model*) pel background substraction i

unes eficients equacions per l'actualització. És una remodelació de l'algoritme de Stauffer i Grimson, 1999.

A nivell matemàtic:
- Primer, el píxel s'avaluarà:

$$\frac{p(\mathrm{BG}|\vec{x}^{(t)})}{p(\mathrm{FG}|\vec{x}^{(t)})} = \frac{p(\vec{x}^{(t)}|\mathrm{BG})p(\mathrm{BG})}{p(\vec{x}^{(t)}|\mathrm{FG})p(\mathrm{FG})} \quad \begin{cases} \text{si > 1 fons (}BG\text{ - background)} \\ \text{si < 1 objecte (}FG\text{ - }foregorund\text{)} \end{cases}$$

- Segon, partint de la base que podrem tenir una referència aproximada de l'objecte a conèixer $p(\vec{x}^{(t)}|\mathrm{FG})$, podrem avaluar si un píxel és del fons tal que:

$$p(\vec{x}^{(t)}|\mathrm{BG}) > c_{\mathrm{thr}} (= p(\vec{x}^{(t)}|\mathrm{FG})p(\mathrm{FG})/p(\mathrm{BG}))$$

On c_{thr} és una constant threshold. Utilitzem threshold per referir-nos a un "filtre" que definirà què és cada píxel. A continuació, per aconseguir major adaptabilitat als canvis calcularem una mitjana Gaussiana de la densitat durant un període *T*. Així doncs, usarem una GMM amb *M* elements:

$$\hat{p}(\vec{x}|\mathcal{X}_T, \mathrm{BG}+\mathrm{FG}) = \sum_{m=1}^{M} \hat{\pi}_m \mathcal{N}(\vec{x}; \hat{\vec{\mu}}_m, \hat{\sigma}_m^2 I)$$

$\hat{\vec{\mu}}_1, \ldots, \hat{\vec{\mu}}_M$ estimacions de les mitjanes.
$\hat{\sigma}_1^2, \ldots, \hat{\sigma}_M^2$ estimacions de les variàncies del GM.
$\hat{\pi}_m$ estimació del pes mitjà => sempre >1

- Tercer, per una nova entrada $\vec{x}^{(t)}$ en un moment *t*, les equacions per l'actualització seran:

$$\begin{aligned} \hat{\pi}_m &\leftarrow \hat{\pi}_m + \alpha(o_m^{(t)} - \hat{\pi}_m), \\ \hat{\vec{\mu}}_m &\leftarrow \hat{\vec{\mu}}_m + o_m^{(t)}(\alpha/\hat{\pi}_m)\vec{\delta}_m, \\ \hat{\sigma}_m^2 &\leftarrow \hat{\sigma}_m^2 + o_m^{(t)}(\alpha/\hat{\pi}_m)(\vec{\delta}_m^T\vec{\delta}_m - \hat{\sigma}_m^2) \end{aligned} \qquad \vec{\delta}_m = \vec{x}^{(t)} - \hat{\vec{\mu}}_m$$

On $\alpha = 1/T$ és una constant que cada període *T* deixarà de tenir en compte els valors antics.

Establirem la propietat $o_m^{(t)}$ a 1 als components "propers" i la resta a 0. Considerarem que la mostra és propera al component si la distància Mahalanobis és ,per exemple, menys que tres. Ho calcularem:

$$D_m^2(\vec{x}^{(t)}) = \vec{\delta}_m^T \vec{\delta}_m / \hat{\sigma}_m^2$$

En cas de no trobar cap "proper" , generarem un nou component segons: $\hat{\pi}_{M+1} = \alpha$, $\hat{\vec{\mu}}_{M+1} = \vec{x}^{(t)}$ i $\hat{\sigma}_{M+1} = \sigma_0$, on σ_0 serà una variància inicial.

Pros: permet seleccionar automàticament el nombre necessari de components per píxel, optimització de la primera versió, el qual ens permetrà adaptar-ho perfectament a situacions estàtiques.

Cons: els canvis d'iluminació sobtada, poden fer aparèixer nous elements "fantasmes" i fer desaparèixer elements reals. A més a més, al igual que Grimson, a nivell de disseny és molt extens i lent.

2.4 – Morfologia i blobs

Un vegada hem assolit aquest punt, ja hem comprés que a partir d'una imatge, un frame, aplicant els anteriors algoritmes obtindrem una imatge resultant. A continuació (Fig. 2.5), podem veure un exemple obtingut amb un dels mètodes anteriors.

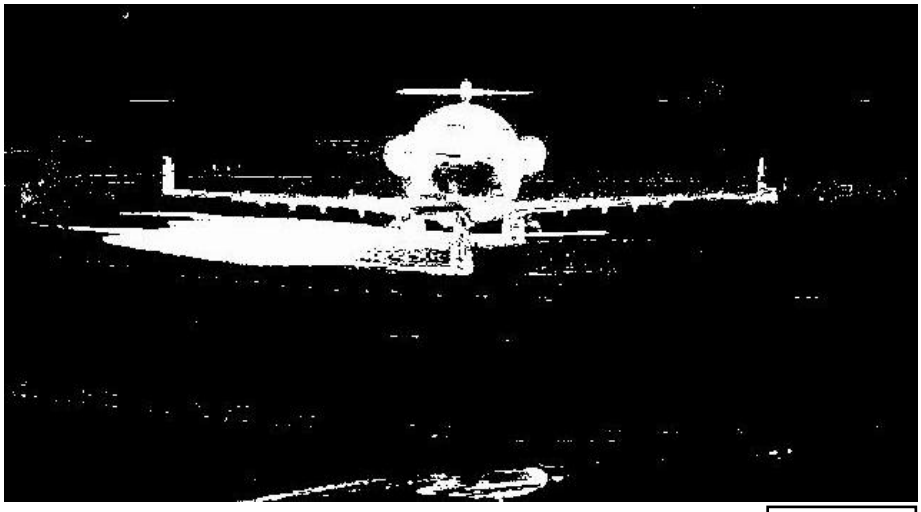


Fig.2.5

És fàcil comprendre que aquesta imatge encara necessita ser processada donat que apareixen elements que no són l'avió i l'avió no surt definit al 100%. L'avaluació píxel per píxel fa que els canvis de llum, reflexes, ombres entre altres... com ja hem donat a entendre, surtin plasmades com "objecte en moviment", *foreground*. I de la mateixa manera hi ha píxels que tot i estar en moviment no es detecten (a la imatge anterior la punta davantera no apareix).

Així doncs, aquest processament necessari el podem dividir en dues parts fonamentals: morfologia i blobs. Com observarem al codi, s'ha de dur a terme en aquest ordre donat que morfologia potenciarà l'efectivitat dels blobs.

Morfologia

Com la pròpia paraula indica, estudiarem cada frame per aplicar canvis de forma al resultat parcial obtingut fins al moment. La idea bàsica és molt senzilla: donat que busquem un avió, un element gran de la imatge, tots els elements massa petits (soroll) els esborrarem; i un cop fet això, als elements grans els reomplirem en cas que tinguin "forats".

Blobs

Aquesta eina ve donada per una llibreria addicional. Cal assenyalar que gairebé totes les aplicacions relacionades amb el background substraction fan ús d'aquesta. La funció permet trobar l'element que desitgem dins una imatge a partir de certs paràmetres, com ara "l'objecte més gran" o "l'objecte que es troba a una posició".

Exemple

A continuació per una major comprensió, mostrarem pas a pas com evoluciona la imatge tractada. Tot i així anunciarem quins mètodes de la llibreria OpenCv [6] hem utilitzat.

- cvCreateStructuring: assigna i omple un objecte *IplConvKernel** per a que pugui ser emprat com estructura d'operacions morfològiques. Cal assenyalar que per l'aplicació de diferents filtre haurem d'utilitzar diferent estructures, donat que no és igual erosionar que dilatar.
- cvErose: mitjançat una estructura, erosionarem una imatge seguint una tècnica per proximitat dels píxels veïns. És a dir, podrem eliminar els petits píxels blancs que estiguessin rodejats de píxels negres i al contrari. Utilitzada al nostre codi.
- cvDilate: (per optimitzacions finals no utilitzat) mitjançat una estructura, ampliarem seguint un criteri zones i els píxels veïns. És a dir, si trobem un conjunt de píxels que s'ajusti al nostre criteri de tamany, rodejats de píxels del color contrari ho canviem.
- cvMorphologyEx: eina per poder realitzar transformacions morfològiques millorades a partir de les eines bàsiques erosió i dilatació. Aquesta funció té diferents funcionalitats:
  - Opening (obertura): en primer lloc aplicaríem una dilatació i llavors una erosió.
  - Clossing (clausura): en primer lloc la dilatació i en segon l'erosió. Aquesta és usada al nostre codi per "omplir forats".

La següent imatge (Fig. 2.6) és la imatge que hem elegit per analitzar. Hem establert agafar-la a falta de 16 segons del vídeo A320.avi i aplicant el mètode Grimson.

Fig.2.6

Com podem observar hi ha zones que no apareixen marcades "en moviment" (píxels blancs), tot i que si apareix la ombra i un cotxe. A continuació, a la Fig 2.7 li apliquem el filtre de morfologia:

Fig.2.7

Tot i el soroll, i altres elements, que havien aparegut inicialment podem comprovar com amb els filtres hem fet desaparèixer una gran part. No obstant, el filtre de blobs si mostra (Fig. 2.8) més canvis:

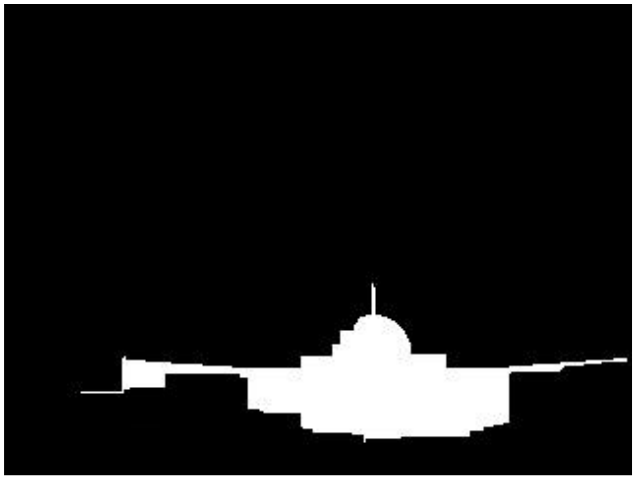


Fig.2.8

Clarament observem com s'han unit totes les parts de l'avió, de forma general. Cal aclarir que la imatge final (Fig. 2.9) serà un retall rectangular i no pas amb el model del blob. En aquest codi, amb el blobs només indiquem on és l'element més gran. I, a partir d'això, quins són els punts espacials (x,y) on tallar.

Fig.2.9

2.5 – Requeriments funcionals

A continuació, el següent diagrama d'ús (Fig. 2.6) mostra quines són les opcions que ofereix el nostre codi a l'usuari:

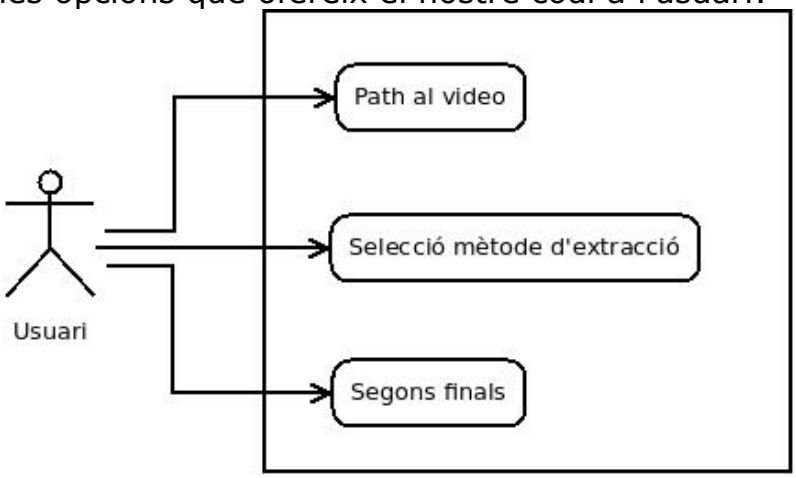


Fig.2.10

Casos d'ús:

· Introducció del path del vídeo.

Per paràmetre haurem d'introduir a quina adreça es troba el vídeo que volem analitzar. Serà responsabilitat de l'usuari que el vídeo estigui en el format adequat (.avi).

· Selecció de mètode d'extracció.

Per paràmetre haurem de seleccionar un dels diferents mètodes pel BS. Les opcions, per odre, són:

1 - Eigen
2 - Wren

3 - Adaptive
4 - Grimson
5 - Zivkovic
6 - Prati
7 - Mean

· Segons finals

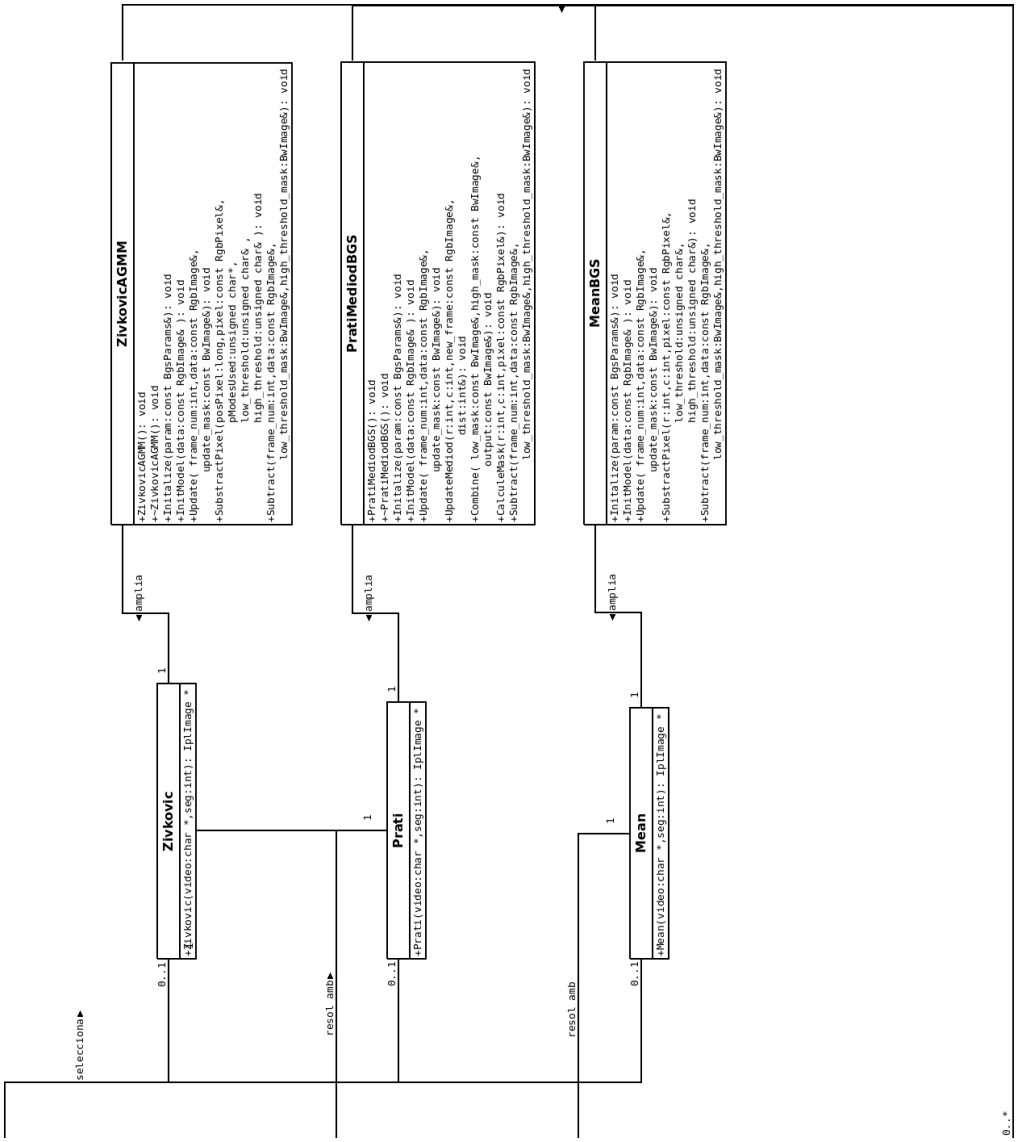
En darrer lloc, introduirem a falta de quan temps volem que es faci la *foto final*. La selecció de la imatge final pot variar, si ho fem molt aviat a un vídeo curt pot ser encara no s'hauria fet suficient aprenentatge o si ho fem excessivament tard l'avió ja estarà quiet i es consideraria background per falta de moviment.

3 - Disseny

En aquest capítol de la memòria tractarem i mostrarem la nostra aplicació a nivell d'implementació. Per això hem decidit separar-ho en diferents seccions: diagrama de classes, arxius que conformen el projecte, diagrama de seqüència, exemple d'ús i codi de suport Test avions.

3.1 Diagrama de classes

Com es pot observar al diagrama només es fan referència als arxius cpp. Al següent apartat s'explica amb detall les raons d'aquest fet.



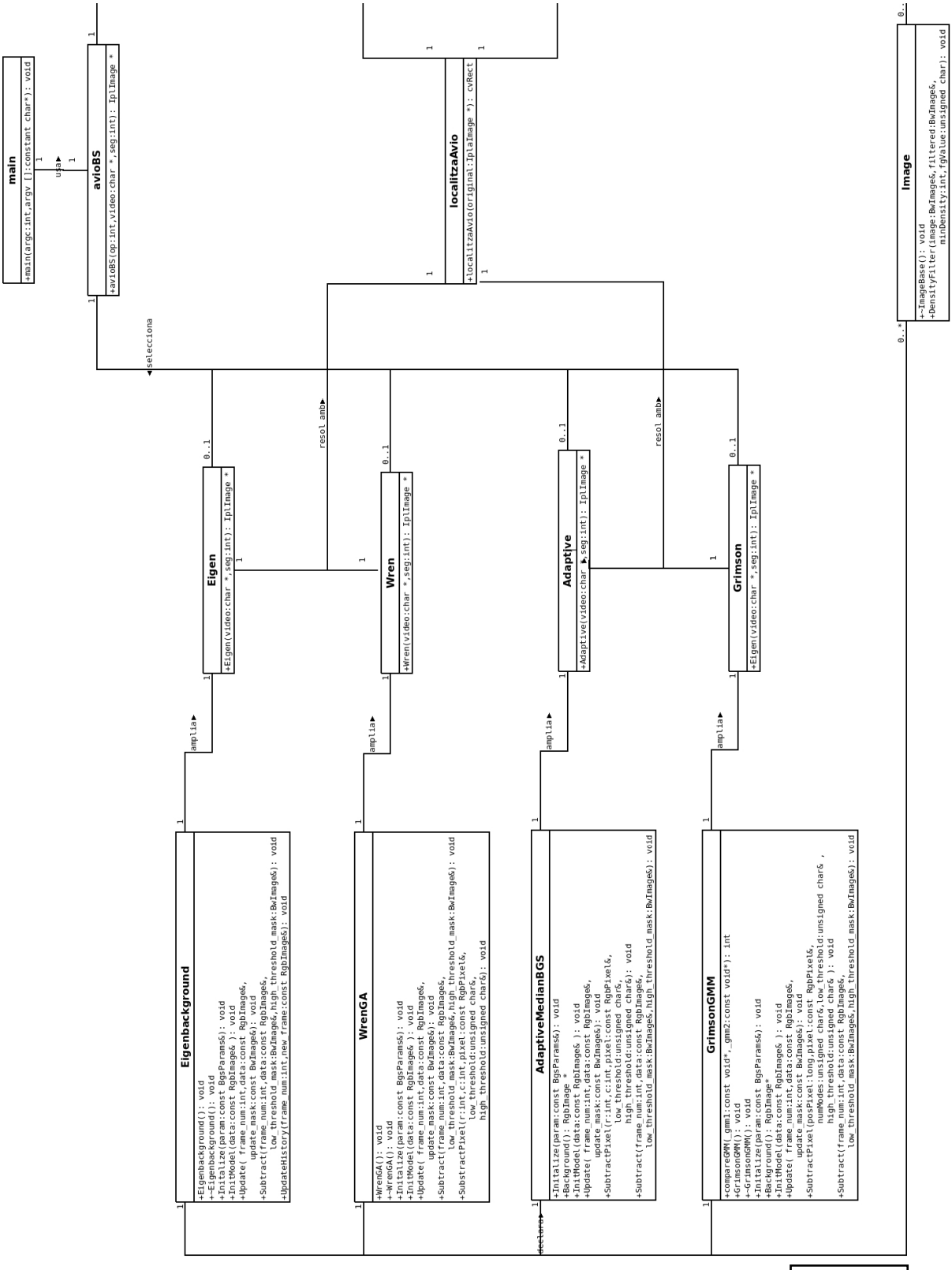


Fig.3.1

Per continuar, volem aclarir el diagrama de classes per facilitar la comprensió d'aquest. Explicarem com és el curs general d'una execució.

La classe *main* usarà *avioBS* per que aquest l'indexi al mètode demanat. D'aquesta manera, seleccionem l'arxiu principal d'un del mètodes: *Eigen*, *Wren*, ...

Els seus mètodes essencials per una subtracció es troben a *Eigenbackground.cpp*, *WrenGA.cpp*, ... aquests seran l'ampliació del principal.

Des de l'ampliació, abans anomenada, del nostre mètode declarem tants structs *Image* com necessitem.

Un cop arribats a aquest punt, quan haguem triat la imatge definitiva la tractarem i resoldrem des de *localitzaAvio*.

Finalment, retornarem al *main* un retall de la imatge processada.

3.2 Tipus d'arxius

A continuació, llistarem i farem una breu explicació de cada arxiu del projecte. D'aquesta manera, es podrà aclarir la funcionalitat de cada una i com treballen entre elles.

Però abans farem una diferenciació sobre les diferents extensions que es presenten. Com ja hem dit a l'apartat anterior, aquest projecte està realitzat en C++, un llenguatge orientat a objectes. D'aquesta manera trobarem 2 extensions principals: *.cpp i *.hpp (és de suposar, que la resta d'arxius dins de la carpeta del codi seran referents al projecte del Visual Studio).

3.2.1 hpp

Són aquells fitxers que treballen com una capçalera per inicialitzar algunes variables i mètodes de manera pública o privades dins un name space, i així condicionar i indexar l'accés als diferents recursos.

A continuació, llistarem els diferents arxius:

- Bgs.hpp: funció que defineix de manera *virtual* mètodes que pretenem redefinir en altres classes derivades. En aquest cas, definim els mètodes comuns a tots els algorismes (als 7). Aquests són: Initialize, InitModel, Substract, Update i Background.

- BgsParams.hpp: permet inicialitzar els paràmetre comuns a tots els algorismes BGS. A més dels "gets" de height, width i size; s'ofereix el mètode setFrameSize.

- Image.hpp: defineix diferents classes, que són: ImageIterator, RgbPixel, RgbPixelFloat i ImageBase. A més, declara la constant NUM_CHANNELS igual a 3 (RGB) i la funció DensityFilter, desenvolupada a Image.cpp.

A partir d'aquest punt comentarem l'arxiu que està associat a cada un dels set algorismes. Així doncs, prèviament, volem aclarir l'estructura d'aquests fitxers.

En primer lloc, es definiran els paràmetres. La part etiquetada com *public* seran els getters mentre que la *private* serà la inicialització d'aquests paràmetres.

En segon lloc, definirem les capçaleres dels mètodes. Com a públiques, a més de les especificades a Bgs.hpp, declararem els constructors si aquell mètode ho requereix. I privates, altres mètodes específics d'aquell algorisme.

*Comentari: unes variables molt importants seran els low_threshold i high_threshold, presents a tots els algorismes. Aquests paràmetres serveixen per classificar un píxel com *foreground* o *background*. Així doncs, una manera senzilla d'entendre-ho, segons la variància i colors d'un píxel aquest serà considerat l'objecte o el fons segons si està o no dins el rang del *llindar* (thresholds).

- Eingenbackground.hpp: els paràmetres específics són HistorySize, nombre de frames emprats per crear l'eigenspace; i EmbeddedDim, dimensió de l'eigenspace. De banda dels mètodes públics, Eigen presenta dos constructors, Eigenbackground i ~ Eigenbackground. Com privats: UpdateHistory, per actualitzar el background segons les últimes dades obtingudes al Substract.

- WrenGA.hpp: presenta els paràmetres específics: alpha, constant que serveix per l'actualització de l'objecte en moviment; learning_frames, nombre de frames per modelar inicialment el background. A les funcions públiques es presenta el constructor com ~WrenGA; a més té SubstractPixel com a mètode privat, per aplicar el rang de thresholds a nivell individual.

- AdaptiveMedianBGS.hpp: dos variables específiques: SamplingRate, variable que condiciona cada quan s'actualitzarà el fons; i learning_frames. El seu constructor és ~AdaptiveMedianBGS, i del costat privat: SubstratPixel.

- GrimsonGMM.hpp: presenta dos paràmetres addicionals als comuns: Alpha i MaxModes, valor que determina el nombre de píxel. Aquesta classe, a més a més, instància un struct GMMGausian, amb valors de derivació estàndard, pes, entre altres... Crearem aquesta "matriu de dades" per cada píxel. Té dos constructors: GrimsonGMM i ~GrimsonGMM. En darrer lloc, presenta una funció privada SubstractPixel.

- ZivkovicAGMM.hpp: té exactament les mateixes característiques que GrimsonGMM. Tot i que el struct (GMM) conté característiques diferents.

- PratiMediodBGS.hpp: en aquest cas, hi ha tres variables exclusives: Weight, pes del píxel; SamplingRate, idèntica que a AdaptiveMedianBGS; i HistorySize, idèntica que a Eigenbackground.

- MeanBGS.hpp: els seus paràmetres especials són: Alpha i LearningFrames. Un sol constructor, ~MeanBGS, i SubstractPixel, com a mètode privat.

Un cop explicades totes les diferències entre les capçaleres és obvi que tots es basen en paràmetres i funcions semblants. Però cap mètode és idèntic a altre.

3.2.1 cpp

Aquests són els arxius que contenen, la implementació del codi, i seran els encarregat de dur a terme les funcionalitats. Com es pot observar al diagrama de classes (Fig. 3.1).

Abans de continuar explicant cada un dels arxius, aclarirem un detall sobre el projecte. Com ja he s'explicava a la introducció aquest projecte és només la meitat del total. Per aquesta raó hem dissenyat el codi per oferir-ho com a utilitat/llibreria d'un aplicatiu per identificar models d'avió. Es vol aclarir aquest punt, donat que al codi presentat s'han d'introduir les variables (path, selecció de mètode i els segons) manualment al codi ja que donat que utilitzem Visual Studio no ho podem passar per paràmetre. El meu company Francesc Muñoz Romero va dissenyar una interfície gràfica per una millor interacció amb l'usuari i oferir la funcionalitat completa d'una manera intuïtiva i senzilla.

A continuació, llistarem els diferents arxius:

- main.cpp: introduïm les dades: enter op, opció de mètode op∈ [1,7] (referència de cada valor: 2.5 Requeriments funcionals); enter seg, segons abans del final moment de la selecció del frame; char * vídeo, path del vídeo a analitzar. Passarem aquest tres valors a avioBS i ens retornarà la imatge original en blanc i negre retallada i centrada a l'avió. Aquesta imatge seria el return per l'extensió de l'aplicació del meu company.

- avioBS.cpp: part encarregada de fer la redirecció a cada mètode de subtracció. Únicament és un switch.

* Comentari: tots els mètodes estan dividits en dos arxius. El primer té el nom del mètode (Eigen.cpp, Adaptive.cpp, ...) i el segon té el nom idèntic al esmentat als hpp (Eigenbackground.cpp,

AdaptiveMedianBGS.cpp, ...). Des de avioBS cridarem als del primer tipus. Tots aquests presenten la mateixa estructura, exceptuant la inicialització de les variables específiques. Així doncs, explicarem amb detall el funcionament pel mètode Eigen, i de la resta només els valors exclusius.

- Eigen.cpp: en primer lloc, s'inclouen les llibreries: *BlobResult.h* (2.4 Morfologia i blobs); *cv.h* , *cxcore.h* i *highgui.h* aquestes ofereixen les eines necessàries per treballar amb el vídeo, venen incloses al conjunt conegut com OpenCV. També inclourem una crida a la llibreria per indexar als metodes genèrics: *Eigenbackground.hpp*. A més, inclourem la crida a *localitzaAvio*, passem una imatge en format IplImage * i ens retornarà un rectangle (*cvRect*) amb els punts exactes d'on s'ha localitzat l'avió.

A continuació, declarem les variables que necessitarem: dues imatges, *IplImage * final* i *frameResize*, i un valor, *escala*, per redimensionar la mida de la imatge amb la que treballem (per defecte 2). Aquest últim valor és per una optimització del codi, ens ajudarà ja que si fem que les imatges siguin la meitat de grans el temps de còmput es millora notablement.

El següent punt és llegir la informació del vídeo passat per paràmetre, i assegurar que s'obri correctament. Línees més abaix crearem un buffer per anar emmagatzemant.

Amb les dades obtingudes del vídeo, inicialitzem quatre variables: *width* (reescalat), altura; *height* (reescalat), amplada; *fps*, frames per segons; i *num_frames*, nombre total de frames al vídeo. També inicialitzarem *frameResize* amb aquestes dades. Per acabar, creem els thresholds i sentenciem que llegirem el vídeo frame per frame.

A continuació, crearem els dos thresholds (*low_threshold_mask* i *low_threshold_mask*). Seran imatges en blanc i negre, d'un sol canal i amb el mateix tamany que el frame original.

El següent punt és exclusiu per Eigen. Com ja hem explicat amb anterioritat aquest apartat serà exclusiu per cada mètode.

Cream un struct *params* on inicialitzarem les variables esmentades a Eiginbackground.hpp. Les constants són:

LowThreshold: 15*15
HighThreshold: 2**LowThreshold*
HistorySize: 100
EmbeddedDim: 20

Finalment, abans de passar a la lògica de la substracció, creem un struct *bgs* que ens permetrà cridar a tots els mètodes de Eigenbackground.cpp, mitjançant Eigenbackground.hpp. I inicialitzem (*Initialize*) amb els paràmetres anteriors.

Arribats a aquest punt, comença la substracció.

Per cada frame, anirem comprovant l'existència i ,si és així, ho anirem desant a la variable *frame*. A continuació, segons havíem comentat sobre l'optimització, canviarem la mida del frame. I mitjançant *frame_resize* desarem les dades modificades a *frame_data* i les dades originals a *frame*.

Amb el primer frame iniciarem el model de substracció (*InitModel*). Seguidament, aplicarem el filtre de substracció (*Substract*), i girarem la imatge (durant el càlcul pateix un gir).

Al següent punt, comprovem (*i=num_frames-seg*fps*) si és el moment en el tractem la imatge desitjada segons el valor *seg* passat al *main*.

Si és així, fem la crida a la classe localitzaAvions, que ens retornarà un rectangle emmarcant la imatge de l'avió. Finalment la redimensionem proporcionalment al tamany original.

Finalment, setejam les dades les dades del low_threshold i tornem a començar fins arribar a la imatge desitjada. Que serà el retorn.

- Wren.cpp: pel struct *params,* exclusiu d'aquest mètode, inicialitzarem les variables tal que:

  LowThreshold: 3.5*3.5
  HighThreshold: 2**LowThreshold*
  Alpha: 0.005
  LearningFrames: 30

- Adaptive.cpp:

  LowThreshold: 40
  HighThreshold: 2**LowThreshold*
  SamplingRate: 7
  LearningFrames: 30

- Grimmson.cpp:

  LowThreshold: 3.0*3.0
  HighThreshold: 2*LowThreshold
  Alpha: 0.001
  MaxModes: 3

- Zivcovik.cpp:

  LowThreshold: 5.0*5.0
  HighThreshold: 2*LowThreshold
  Alpha: 0.001
  MaxModes: 3

- Prati.cpp:

  LowThreshold: 30
  HighThreshold: 2*LowThreshold
  SamplingRate: 5
  HistorySize: 16

Weight: 5

- Mean.cpp:
  LowThreshold: 3*30*30
  HighThreshold: 2*LowThreshold
  Alpha: 1^{-6}
  LearningFrames: 30

- Eigenbackground.cpp: de la mateixa manera, en primer lloc inclou la llibreria a stdio.h.

Dins de la pròpia classe, trobem la implementació de tots els mètodes anomenats a la capçalera d'Eigenbackground.hpp. Així doncs, els dos primers mètodes que trobem són els constructors. *Eigenbackground*, inicialitza les variables a NULL; ~ *Eigenbackground* les desarà com a matrius. Això serà idèntic per tots els algorismes.

D'altra banda, *Initalize* rep la variable *param* amb ella crearà una nova imatge pel fons de mateix tamany (*width* & *height*).

InitModel per totes les variables que siguin diferents de NULL las inicialitzem amb una matriu, concretament per *m_pcaData* crearem una de tamany *HistorySize* x *Size*; per acabar iniciarà a buit la imatge creada prèviament a *Initalize*.

Tot i tenir la capçalera del mètode *Update*, aquest no està implementat, està buit.

El següent mètode és *Substract*. El podríem dividir en tres parts fonamentals:

· si *frame_num* < *HistorySize* : encara no hi ha prou informació així doncs que seguim al moment d'aprenentatge, cada píxel l'atribuïm al fons, *BACKGROUND*.

· si *frame_num* == *HistorySize* : iniciarà les variables necessàries i crearà així un nou eigenspace.

· si *frame_num* >= *HistorySize* : aquest apartat és pròpiament la substracció del fons. Per fer-ho, projectem la nova imatge a l´eigenspace.

Abans d'acabar, el mètode *Subtract* fa una crida a *UpdateHistory.*

Aquest últim mètode només tindrà servei pels frames amb els que fem l'aprenentatge (*num_frame*<*HistorySize*).

- WrenGA.cpp: en primer lloc, els seus constructors només fan referència a la variable *m_gaussian*, a *WrenGA* ho iniciem a NULL mentre que ~*WrenGA* ho buida.

Initalize creara una matriu gaussiana per cada píxel. A més, crea una imatge pel background (*m_background*).

A continuació, el mètode *InitModel* inicialitza el model de background mitjançant la informació del primer frame, *data*, i el valor de *m_variance* anterior. Ambdós s'introduiran a l'array *m_gaussian* per cada canal.

Update un cop superada la fase d'aprenentatge, actualitzarà la nostra imatge amb la nova informació.

Com ja havíem esmentat, Wren compte amb ambdós parts per la substracció. *SubstractPíxel*: primer calcularà la distància entre el model i el píxel, i llavors, comprovarà si s'ajusta al model del fons o de l'objecte. Aquest mètode serà invocat des de *Substract* per cada píxel.

- AdaptiveMedianBGs.cpp: la primera funció que trobem és *Initalize*, que inicialitza mitjançant la variable *params* rebuda per paràmetre. A més crearem una imatge, *m_median*. D'altra banda, *Background* és un mètode que retorna *m_median*.

Al igual que al cas anterior (WrenGA), el InitModel inicialitzarà la imatge *m_median* amb les dades del primer frame.

Update, cada cert nombre de frames, si hem superat la fase d'aprenentatge, actualitzarà el nostre model.

També comprovem com té dos mètodes per la substracció. *SubstractPixel* calcularà si el píxel és o no part del fons segons la diferència, per cada canal, entre el nou frame i els thresholds. *Substract* el cridarà per cada píxel i actualitzarà els tresholds abans de sortir.

- GrimsonGMM.cpp: presenta dos constructors. Tant *GrimsonGMM* com *~GrimsonGMM* fan referència a *m_modes*, el primer ho iguala a NULL mentre que el segon elimina els valors de la taula.

Initalize a més de settejar amb els paràmetres recollits a Grimson.cpp, inicialitza: *m_bg_threshold* = 0.75; *m_variance* = 36.0; una matriu gaussiana per cada píxel; i finalment dues imatges: *m_modes_per_pixel* , d'un sol canal, i *m_background*, amb tres. El mètode *Background*, retornarà aquesta última imatge.

InitModel tot i rebre la informació del primer frame (*data*) settejarem tots els valors de la GMM de cada píxel , *m_ modes*, a 0. Aquests valors són: *weight*, *variance, muR*, *muG, muB* i *significants*.

Update, no està implementat.

SubstractPixel és un mètode molt extens que presenta varies "etapes". Primer calcularà el nombre de Gaussians inclosos al model de fons, segon actualitzem totes les distribucions i comprovem si existeix alguna coincident, tercer renormalitza els pesos, quart ordenem de forma descendent i , cinquè, si al segon pas no hem trobat coincidència crearem un nou mode. Abans d'acabar es reordenarà, altra vegada, per si hi ha hagut canvis. I finalment, es farà la diferenciació de si el píxel pertany o no al fons. *Substract* cridarà el mètode anterior per cada píxel d'un frame.

- ZivkovicAGMM.cpp: també presenta dos constructors. *ZivkovicAGMM* inicialitza NULL les variables: *m_modes* i

m_modes_per_pixel; *~ZivkovicAGMM* eliminarà els valors de ambdós arrays.

Initalize després de desar *params*, trebut per paràmetre, iguala les següents variables: *m_num_bands* = 3, *m_bg_threshold* = 0.75, *m_variance* = 36.0, *m_complexity_prior* = 0.05. A més, crearem una matriu per cada píxel, inicialitzem els modes pels píxels, i, finalment, creem la imatge *m_background.*

InitModel tot i rebre la informació del primer frame (*data*) settejarem tots els valors de cada píxel: *m_ modes_per_pixel, weight*, *sigma, muR*, *muG,* i *muB*; tots ells inicialitzats a 0.

Update també està buit.

Tot i a la diferent implementació i matemàtica emprada, tant *SubstractPixel* i *Substract* presenten la mateixa tècnica i estructuració.

– PratiMediodGBS: dos constructors. *PratiMediodBGS* inicialitza *m_median_buffer* a NULL i *~ PratiMediodBGS* buida el contingut.

Initalize, primer desa el contingut de *params*, acte seguit crearem una IplImage nova per cada threshold (high i low) i per *m_background*. Per últim, en *m_median_buffer* cream un nou struct MEDIAN_BUFFER.

Update, després de calcular les noves imatges de substracció, modificarà el buffer de la imatge amb l'ajuda de UpdateMediod.

En aquest cas, *Substract* compta amb dos mètodes d'ajuda *CalculateMask* i *Combine.* Mentre estem a la etapa de aprenentatge buidarem el contingut dels thresholds. A partir d'aquest moment, per cada píxel calcularem la mascara a la que pertany i llavors combinem ambdós thresholds.

El primer dels dos mètodes, bàsicament calcularà la distancia entre un píxel nou i la mitjana d'aquell al model; després compararà si és part del fons o l'objecte i ho assignarà al threshold. El segon, *Combine*, mesclarà el dos thresholds segons el nombre de píxels que s'han aprés prèviament.

- MeanBGS.cpp: El primer mètode que observem és Initalize. Aquest desarà *params*, i crearà dos imatges noves: *m_mean* i *m_background*.

InitModel actualitzarà tots els píxels de *m_mean* amb les dades del primer frame.

A continuació, *Update*, actualitza el model del background.

Com a alguns casos anteriors, presenta dos mètodes per la substracció. *Substract* cridarà per cada píxel a *SubstractPíxel*, qui mitjançant les distàncies actualitzarà els thresholds, i finalment desarem aquestes actualitzacions.

3.3 Test avions

L'última part de codi que ens queda per comentar l'hem anomenat: Test avions. Aquesta aplicació és petita i senzilla.

Bàsicament és un sol arxiu que treballa amb imatges obtingudes del codi anterior. Concretament les imatges obtingudes després d'aplicar l'etapa de morfologia i blobs. Aquest codi les unirà i comparà quin percentatge de perfecció poden ser dos mètodes treballant en parella. Al següent tema, (4 - Resultats) podrem observar exemples i resultats obtinguts.

En aquest cas, totes les tasques s'executen des d'un sol arxiu: main.cpp.

- main.cpp: inclou quatre llibreries: *istream.h, cv.h, cxcore.h,* i *highgui.h*.

En primer lloc, llegirà les tres adreces per agafar: *imgPerfect*, imatge retallada a mà per tenir un cas perfecte; *imgMetode*, per la imatge d'un dels mètodes; i *imgMetode2*, per l'altre. A més crearem dos imatges noves: una per desar el resultant de la intersecció (*ImgInt*)i l'altre pel de la unió (*ImgUnio*)

Utilitzant les eines cvAnd i cvOr , per aquest ordre, aplicarem la intersecció i la unió.

- cvAnd: aplica pixel(I) = ImgMetode(I) & ImgMetode2(I). És a dir, si els dos píxels són blancs (o negres) es quedarà blanc (o negre). Si són diferent no hi ha canvi.
- CvOr: aplica pixel(I) = ImgMetode(I) || ImgMetode2(I). Aquest només necessita que un dels dos píxels sigui de manera diferent per canviar-li el color, si són idèntics no hi ha variació.

4 - Resultats

En aquest apartat, anirem utilitzant cada un dels mètodes implementats per diferents situacions i amb diferents paràmetres d'entrada. Com és obvi, n'hi haurà que funcionaran millor en situacions de llum diürna, d'altres treballaran millor amb vídeos més curts, i així per totes les simulacions possibles.

Així doncs, hem establert que els punts més important a analitzar seran: dia/nit , és a dir l'hora del dia en que s'ha gravat aquell vídeo; i temps que triga en l'execució.

Hem decidit que aquests dos paràmetres són els més rellevants. D'una banda és imprescindible conèixer quan triga un mètode en avaluar, i si aquest fet dóna millors resultats de dia o de nit.

Volem esclarir que no estem determinant que altres factors com el segon quan s'agafa la imatge, *seg*, o el tipus d'avió; no siguin importants. Però sigui com sigui serà igual per tots els mètodes, i en principi, a tots els afectaria igual. En canvi, quan temps triga un mètode en executar o com treballa de dia/nit són fets més particulars de cada mètode.

A més, per que el camp d'estudi sigui major hem agafat tots els vídeos d'avions diferents.

Altre punt molt important a esclarir és que en determinats vídeos, pel tipus de format, hi ha mètodes que no poden executat o només executen parcialment. Per aquesta raó, és molt positiu poder comptar amb varietat de mètodes.

Dels dos millors resultats obtinguts en cada cas, buscarem quina parella de mètodes és més eficient per cada situació gràcies a Test avions.

4.1 – Resultats per vídeos diürns

Els vídeos diürns presenten dificultats diferents que la resta de hores del dia. La llum més clara fa que hi hagi molts més reflexes i que elements secundaris (cotxes, persones, ocells, entre altres...) també siguin captades per la càmera.

- Les dades referents al primer objecte d'estudi són:

Model avió: Airbus 320
Segons finals: 16 segons (480frames)
Duració vídeo: 44segons (1320frames)
Temps execució:
- Eigen: error
- Adaptive: 46s
- Wren: 51s
- Grimson: 1m20s
- Zivkovic: 52s

Prati: 1m48s (agafa de 5 e 5)
Mean:54s

A continuació (Fig. 4.1), veurem les seves imatges finals, les que retornaríem.

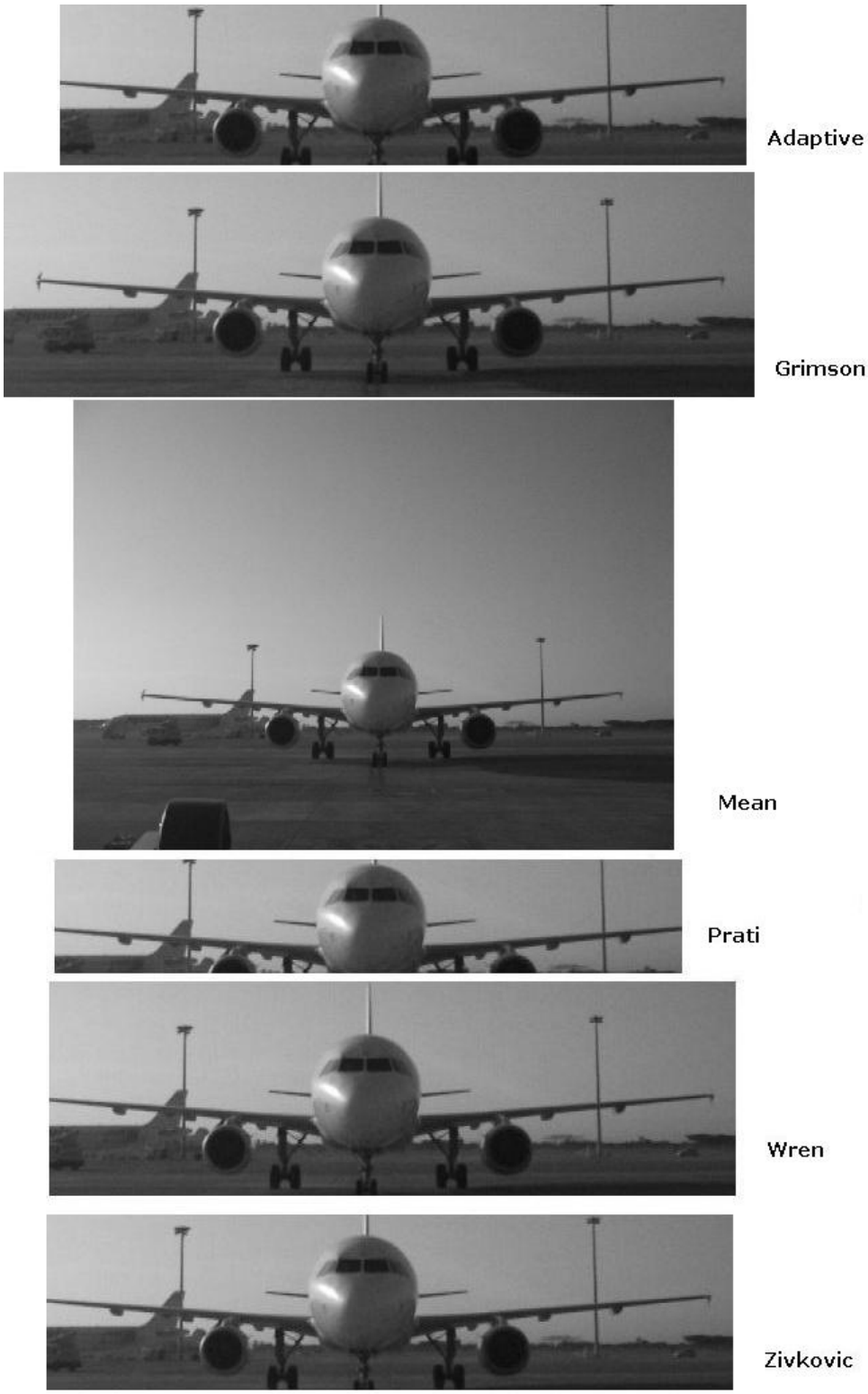


Fig.4.1

Començarem comentant les imatges, trobant similituts, i després observant en general quin mètode ha obtingut millors resultats.

Pot ser la imatge que crida més l'atenció és Mean. És evident que no ha ajustat el retall a la imatge de l'avió. Aquest fet es deu a que la seva etapa de substracció no ha tingut efecte, i al no detectar res ho retalla tot.

A simple vista, podriem assumir que Adaptive, Zivkovic i Prati han retallat en excés la imatge, donat que es perd part de les rodes i la cua. Aquest fet està causat per l'etapa de substracció. Com podem veure a la imatge Fig. 4.2:

Fig.4.2

Mentre que Grimson (imatge inferior) ha observat el moviment a les rodes i ombra de l'aeroplà, Adaptive (imatge superior) no. Per aquesta raó, tot i aplicar l'etapa morfològica i de blobs, el mètode no detectarà les rodes i tallarà des del punt amb moviment més inferior i més a l'esquerra.

De la mateixa manera és fàcil observar com ambdós mètodes no han pogut distingir un troç de l'ala dreta de l'avió (a l'esquerra de la imatge) però com amb l'etapa de morfologia aquell costat s'unirà a la resta de l'avió, sinó també podria aparèixer tallada. Com en alguns casos inferiors.

Així doncs, el dos mètodes amb millors resultats són Grimson i Wren, tant per la imatge final com pel temps d'obtenció.

- Segon vídeo d'estudi:

Model avió: Boeing 757
Segons finals: 16 (480frames)
Duració vídeo: 1m30segons (2820frames)

Temps execució:
Eigen: error
Adaptive: 2m1s
Wren: 2m14s
Grimson: 5m11s
Zivkovic: 2m30s
Prati: 4m10s (agafa de 5 e 5)
Mean:2m26s

A la següent imatge, Fig. 4.3, mostrem les imatges finals.

De nou, Mean no ha detectat moviment i per tant ha mostrat una imatge completa, sense retallar. Prati, totalment al contrari, només ha detectat moviment en una zona petita per aquesta raó mostra només un troç de la part de la cabina.

Adaptive, Wren i Zivkovic han retallat aproximadament bé. És normal considerar que si la cua es de color blanc es pugui confondre amb el cel en un dia tan clar. A més a més són els que tenen temps més baixos.

Finalment, sense cap dubte la millor imatge obtinguda d'aquest Airbus és aconseguida per Grimson però també és el que ha presentat el pitjor temps.

Adaptive

Grimson

Mean

Prati

Wren

Zivkovic

Fig.4.3

- Tercer, i últim, vídeo d'estudi diürn:

Model avió: CRJ200
Segons finals: 7 (210frames)
Duració vídeo: 39 segons (1170frames)

Temps execució:
Eigen: error
Wren: 55s
Adaptive: 26s
Grimson: 1m22s
Zivkovic: 49s
Prati: 1m50s (agafa de 5 e 5)
Mean:1m2s

A continuació (Fig. 4.4) mostrarem totes les imatges resultants.

Aquest cas, a simple vista tornem a observar la nul·litat de subtracció al mètode Mean.

Però si que apareixen altre dades important d'analitzar. Adaptive, Wren i Zivokovic han desplaçat, considerablement, el retall de la imatge cap a la dreta. Si ens fixem es degut al pas d'un vehicle pel fons, la Fig. 4.4 mostra l'etapa prèvia a morfologia i blobs d'Adaptive.

Fig.4.4

En aquesta imatge es pot apreciar més clarament com ha agafat el cotxe com a part de la substracció. Tot i així, metre Grimson a fet un retall molt ample, sobra molta imatge a ambdós costats de les ales, Adaptive, Wren i Zivkovic ho han ajustat molt més. Per aquesta raó i pels temps d'execució, considerem millors aquests 3 mètodes.

Adaptive

Grimson

Mean

Prati

Wren

Zivkovic

Fig.4.5

4.2 – Resultats per vídeos nocturns

Per la seva banda, els vídeos gravats de nit tenen diferencies substancials amb els vídeos obtingut de dia. La llum en general serà menor, el cel i tot el fons serà més fosc, i per tant serà més difícil detectar moviment. A més a més, les llums que es detectin generalment seran focus que en cas d'apuntar a l'objectiu directament distorsionaran la figura a retallar.

Finalment, la majoria de càmeres tenen pitjor definició per gravar de nit, de manera que la qualitat de la imatge rebuda afectarà directament a la substracció.

- Primer vídeo per l'estudi nocturn:

Model avió: B737
Segons finals: 13s (390frames)
Duració vídeo: 44s (1320frames)

Temps execució:
- Eigen: 59s
- Wren: 49s
- Adaptive: 23s
- Grimson: Error
- Zivkovic: Error
- Prati: 1m33s (agafa de 5 e 5)
- Mean:59s

La primera particularitat d'aquest vídeo és que sembla que hi més claredat. De la mateixa manera que anteriorment, Mean i Prati només han obtingut retalls d'una petita part. I tot i que Wren i Adaptive han obtingut resultats molt positius, sobretot en temps, la imatge millor centrada, i en un temps competitiu, és la d'Eigen.

Com ja hem especificat endalt Grimson i Zivkovik han donat un error, molt semblant al que ens ha passat al vídeo anterior.

Adaptive

Eigen

Mean

Prati

Wren

Fig.4.6

- Segon vídeo nocturn:

Model avió: B717
Segons finals: 9s (270frames)
Duració vídeo: 53s (1590frames)

Temps execució:
- Eigen:1m15s
- Wren: error
- Adaptive: 53s
- Grimson: error
- Zivkovic: 1m2s
- Prati: 2m21s (agafa de 5 e 5)
- Mean:1m13s

Com podem observar (Fig 4.6) pocs mètodes han aconseguit obtenir resultats.

El primer que ens crida l'atenció és que és el primer vídeo pel qual funciona Eigen. I el seu resultat és bastant encertat, ho ha ajustat exactament per l'ala esquerra.

Així com ja anunciàvem, cal tenir en compte que el mètode substract haurà considerat en tots els casos la llum del focus con element en moviment. Un exemple clar (Fig. 4.6) de la substracció d'Eigen:

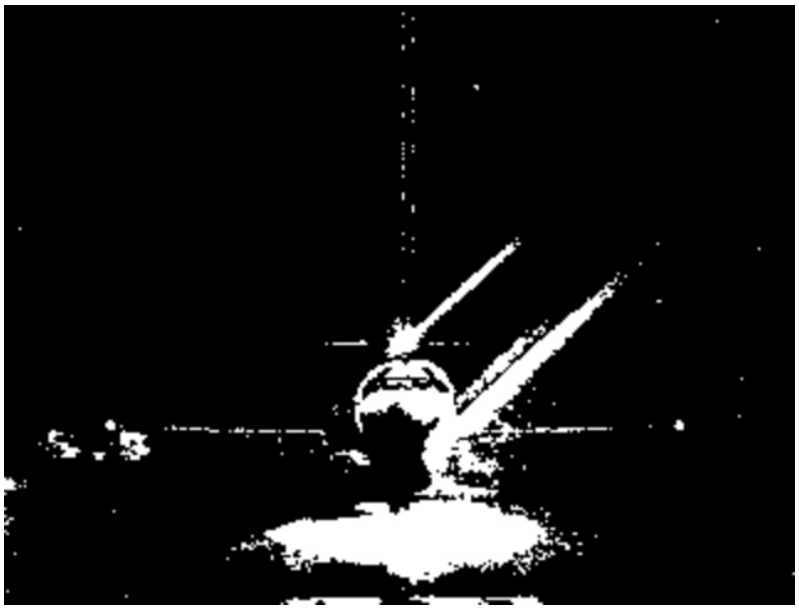


Fig.4.7

Altre punt curiós, és el primer vídeo on fallen Grimson i Wren. Tot i que començaven la substracció, al final donaven un error. Arribaven a un punt que no sabien interpretar.

Més enllà de les dades curioses, Prati ha seleccionat una petita part de la cabina de l'avió; Mean ha mostrat una petita taca (per

aquesta raó no es mostra però està desada a la carpeta de resultats); Zivkovic i Adaptive han tornat a obtenir els millors resultats.

Adaptive

Eigen

Prati

Zicvkovic

Fig.4.8

- Per últim, el tercer vídeo:

Model avió: E145
Segons finals: 6s (460frames)
Duració vídeo: 1m26s (2580frames)

Temps execució:
Eigen: error
Wren: 4m5s
Adaptive: 51s
Grimson: 6m20s
Zivkovic: 1m15s
Prati: 4m25s (agafa de 5 e 5)
Mean:3m12s

Abans de continuar amb les imatges resultants, volem fer una observació de com per aquest vídeo ,no molt més llarg que d'altres anteriors, però tot i així pràcticament tots els mètodes han pujat considerablement el seu temps de còmput. Adaptive segueix presentant el millor temps.

Les imatge finals són les mostrades a la Fig. 4.9.

Altre cop, Eigen és l'únic que ha donat un error, i Prati i Mean només retornen captures molt parcials.

Mentre Grimson, fa un retall molt ample, amb moltes parts sobrants; Adaptive, Zivkovic, i Wren han estat més precisos. I d'aquestes tres, només Adaptive i Wren tenen uns temps raonables.

Adaptive

Grimson

Mean

Prati

Wren

Zivkovic

Fig.4.9

4.3 – Resultats per vídeos del vespre

Hem decidit separar aquests vídeos i fer un nou grup d'estudi ja que la llum amb la caiguda del sol és molt específica i pot ajudar a matisar el treball d'alguns mètodes. Per desgràcia només tenim dos vídeos sobre els que fer les proves.

- El primer té les següents característiques:

Model avió: B767
Segons finals: 13s (390frames)
Duració vídeo: 1m40s (3030frames)

Temps execució:
Eigen: error
Wren: 2m27s
Adaptive: 32s
Grimson: 5m03s
Zivkovic: 2m9s
Prati: 5m (agafa de 5 e 5)
Mean: 2m46s

Imatges mostrades a Fig. 4.10.

Com podem apreciar en el conjunt d'imatges Fig. 4.10. Com a tants altres resultats, Mean no ha detectat cap moviment i mostra tota la imatge; i Prati ha percebut tant poc que el seu retall, és excessivament petit.

Adaptive tot i seguir sent el més ràpid ha eliminat tota la part de la cua i les rodes. I , tot i que Zivkovic i Wren han estat una mica més precisos, Grimson amb un temps molt superior de còmput ha obtingut la millor imatge final.

Adaptive

Grimson

Mean

Prati

Wren

Zivkovic

Fig. 4.10

- El segon, i últim vídeo de l'estudi:

Model avió: DH8
Segons finals: 7s (210frames)
Duració vídeo: 45s (1350frames)

Temps execució:
- Eigen: 49s
- Wren: Error
- Adaptive: 37s
- Grimson: Error
- Zivkovic: Error
- Prati: 1m15s(agafa de 5 e 5)
- Mean: 41s

Les imatges finals són:

Adaptive

Eigen

Mean

Prati

Fig. 4.11

Com podem comprovar, han fallat tres dels mètodes: Wren, Zivkovic i Grimson. No obstant Eigen si ha funcionat ara i ha obtingut una imatge descentrada, on falta un ala. Mean i Prati no han

aconseguit una bona substracció. Altra vegada, Adapative ha tingut el millor temps i la millor captura de l'avió.

4.4 – Comparació de resultats

Arribats a aquest punt compararem els resultats obtinguts a l'avaluació anterior, i mitjançant Test avions, buscarem si en els casos més desfavorables podríem obtenir una millora fent la unió entre dos mètodes. Cal esmentar que ara només valorarem la qualitat de la imatge obtinguda, i no pas el temps de còmput.

Casos estudiats de dia:

- A320: considerem que Grimson ha obtingut una imatge gairebé perfecta.
- B757: podem considerar que la millor imatge també l'ha aconseguit Grimson.
- CRJ200: en aquest cas, no hi una imatge realment centrada donat que cap dels mètodes ha aconseguit esborrar el vehicle del fons. Provarem quin percentatge d'encert tindrien Wren i Zivkovic. Les imatges oferides per Wren i Zivkovic (dalt i baix) són:

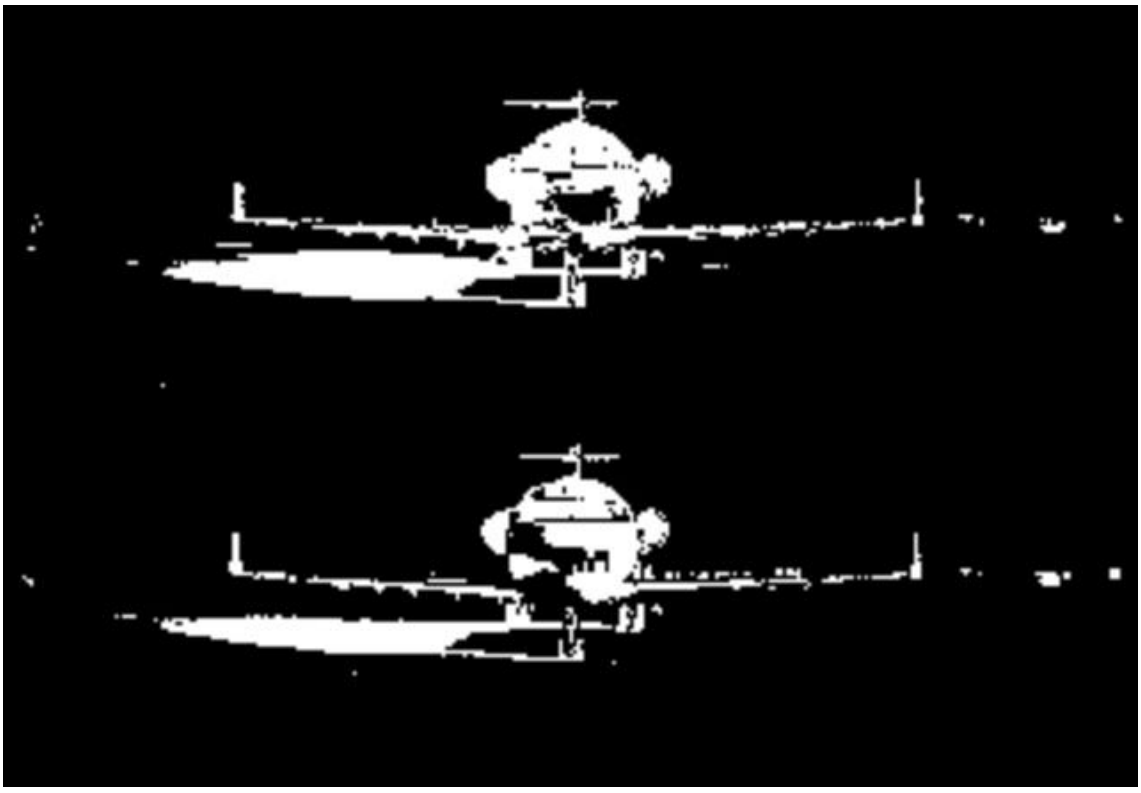


Fig. 4.12

I la imatge que utilitzem com avió perfecte:

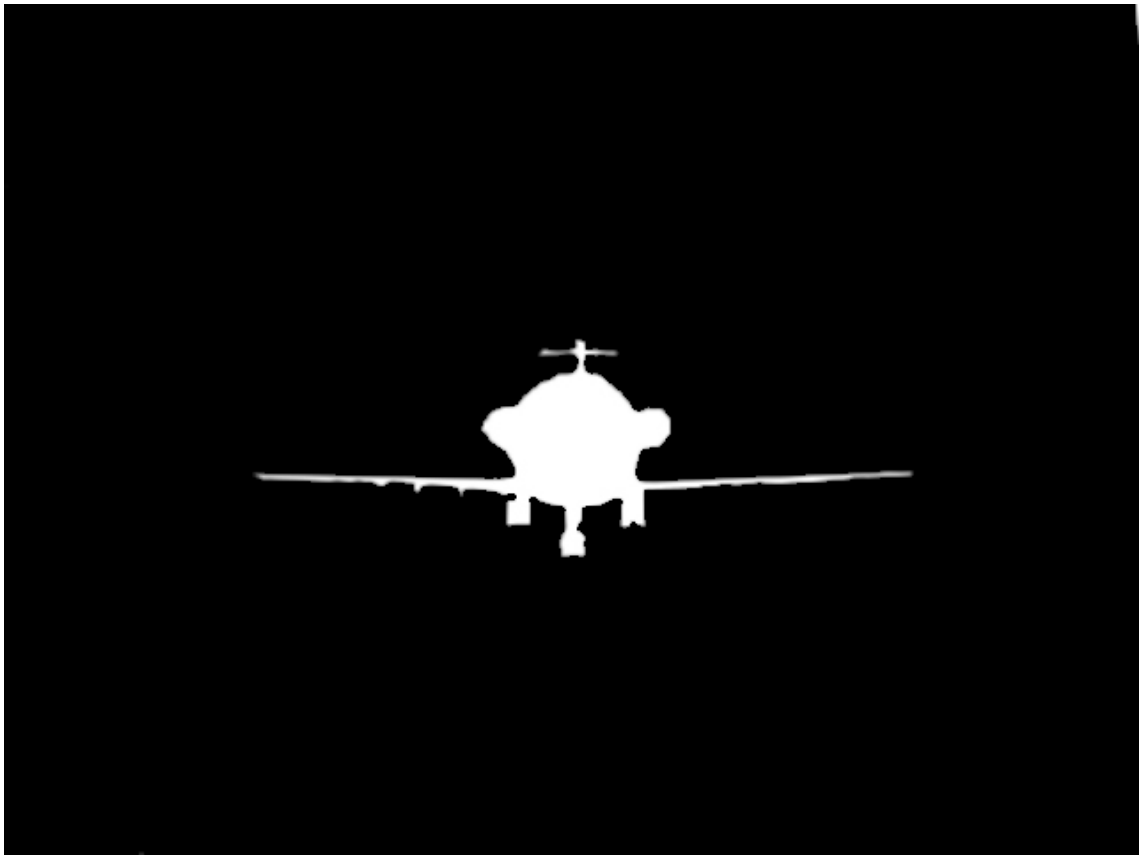


Fig. 4.13

Així doncs després d'executar el codi Test avions, hem calculat que la unió d'aquests dos mètodes seria d'un 34'9%% d'encert. Mentre que en separat seria, aproximadament, un 24'2% per Wren i un 25'1% per Zivkovic. S'ha de tenir present que aquests si consideren la ombra com un element en moviment, i són molts píxel "d'error".

Casos estudiats de nit:

- B737: En aquest cas es obvi que Eigen o Adaptive serien els mètodes amb una precisió major. Casi perfecte.
- E145: Aquest cas tot i que pràcticament tots havien obtingut un bon resultat, Adaptive seria la millor opció.
- B717: la imatge rebuda del vídeo dificulta l'obtenció d'un bon resultat, per la llum que es dirigeix frontalment a la càmera. Per aquesta raó hem volgut mesclar dos mètodes per tractar d'obtenir una millora.

Les imatges seleccionades són de Zivkovic i Adaptive (Fig. 4.14).

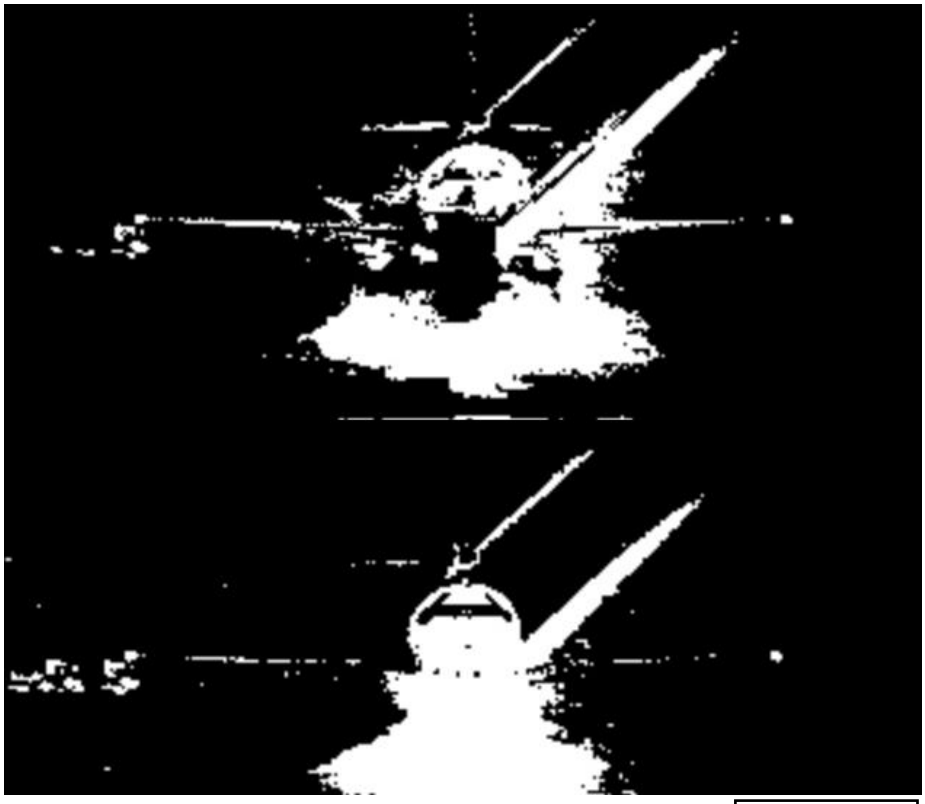


Fig. 4.14

I la imatge perfecta per fer de model (Fig. 4.15) és:

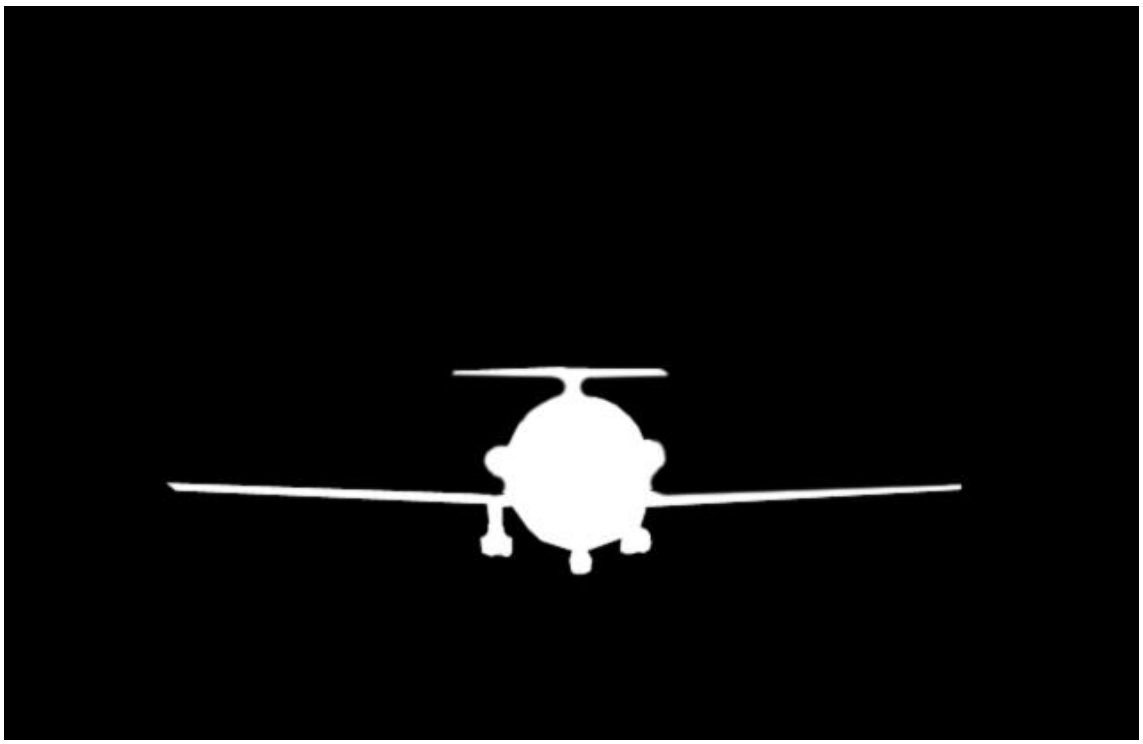


Fig. 4.15

Així doncs després d'executar el codi Test avions, hem calculat que la unió d'aquests dos mètodes seria d'un 22% d'encert. Mentre que en separat seria un 16'3% per Zivkovic i un 14'8% per Adaptive. No cal recordar que aquest cas seria especialment complicat.

Casos estudiats de vespre:

- DH8: Aquest cas tot i no haver aconseguit gaires resultats, només quatre i parcials, el d'Adaptive és molt bo.
- B767: Donat que Zivkovic i Wren eliminen coses diferents, el primer no detecta la cua i el segon no detecta les rodes hem decidit provar quina seria la seva combinació, tot i que Grimson tenia una bona aproximació.

Les dos imatges que utilitzaríem són, Zivkovic (dalt) i Wren (baix) (Fig. 4.16):

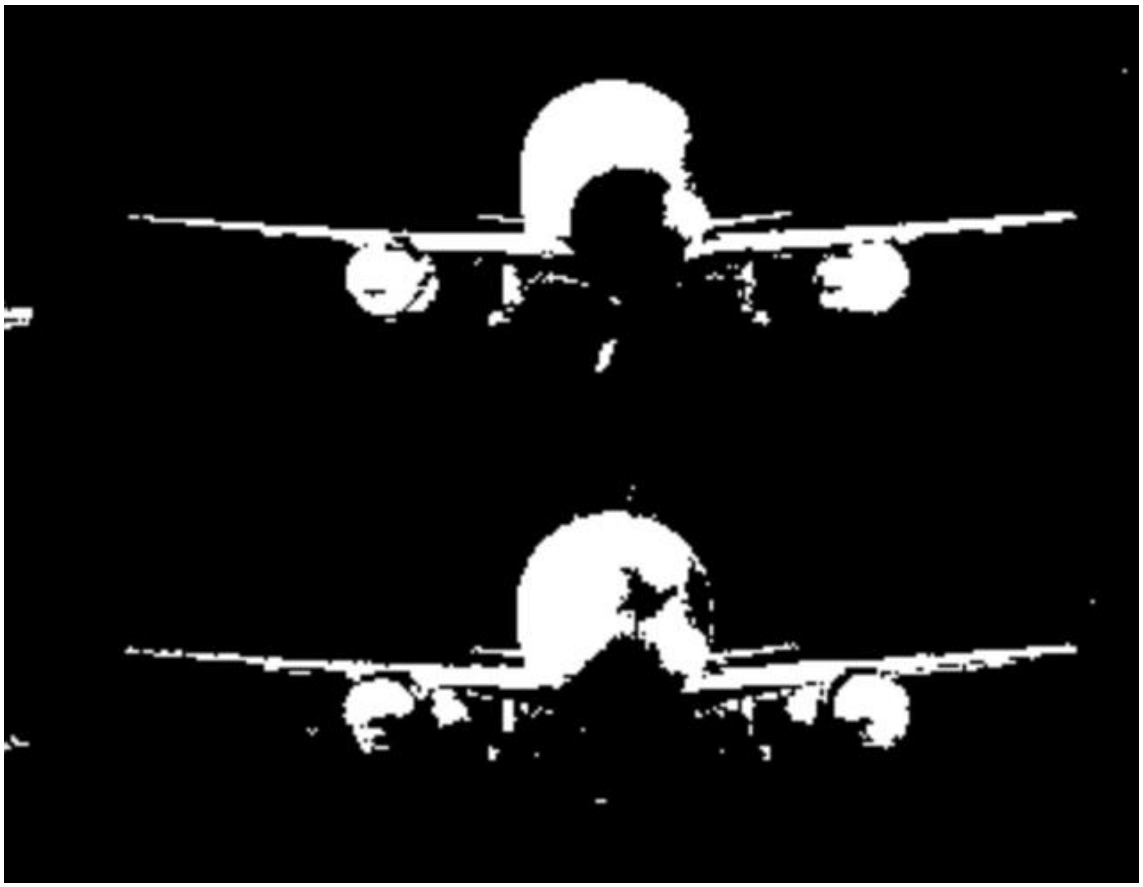


Fig. 4.16

I el model amb el que compararem (Fig. 4.17):

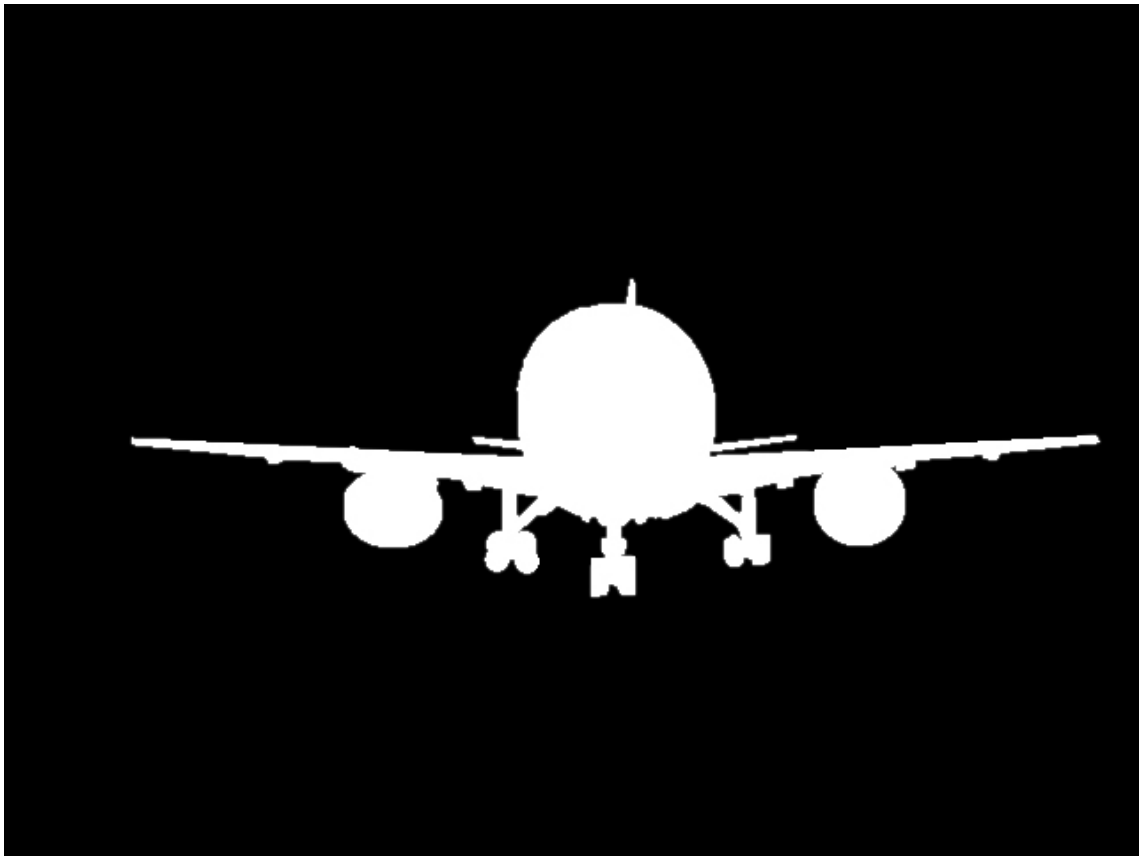


Fig. 4.17

Així doncs després d'executar el codi Test avions, hem calculat que la unió d'aquests dos mètodes seria d'un 68'8% d'encert. Mentre que en separat seria un per 61'3% Zivkovic i un 59'38% per Adaptive.

Per fer una síntesi de les dades obtingudes als resultats aquí presentem dos taules.

La primera (Fig. 4.18)sobre els temps (i també sobre si han obtingut o no resultat) i la segona (Fig. 4.19) sobre els percentatges de Test avions.

	Diurn			Nocturn			Vespre	
	A320	B757	CRJ200	B717	B737	E145	B767	DH8
Eigen	X	X	X	1m15s	59s	X	X	49s
Wren	46s	2m1s	55s	X	49s	4m5s	2m27s	X
Adaptive	51s	2m14s	26s	53s	23s	51s	32s	37s
Grimson	1m20s	5m11s	1m22s	X	X	6m20s	5m03s	X
Zivkovic	52s	2m30s	49s	1m2s	X	1m15s	2m9s	X
Prati	1m48s	4m10s	1m50s	2m21s	1m33s	4m25s	5m	1m15m
Mean	54s	2m26s	1m2s	1m13s	59s	3m12s	2m46s	41s

Fig. 4.18

	Diurn - CRJ200	Nocturn - B717	Vespre - 767
Wren	24'2%		
Zivkovic	21'5%	16'3%	61'3%
Adaptive		14'8%	59'38%
Total	34'9%	22%	68'8%

Fig. 4.19

5 - Conclusions

Amb aquest treball hem tractat de solucionar un problema real relacionat amb l'aeroport del Prat de Barcelona.

El projecte s'ha desenvolupat en un llenguatge c++ que, segons d'algorítmica i la implementació esmentades, és capaç de detectar el moviment d'avions al estacionar tractant imatges obtingudes arrel d'una entrada de vídeo.

A continuació avaluarem les dades obtingudes, resumides a les taules Fig. 4.18 i Fig. 4.19, per fer un balanç sobre cada mètode tant en el seu temps de còmput com a les situacions amb millor i pitjor rendiment.

A més, posteriorment proposarem un conjunt de millores i aspectes a perfeccionar.

5.1 – Avaluació dels mètodes

Eigen: mètode amb una algorítmica basada en principis diferents de tots els altres mètodes. La seva implementació és pot ser la més complexa de totes.

En qüestions de rendiment mostra aspectes molt diferents.

En primer lloc, el seu índex de funcionament és molt baix, només 3 vegades de 8 execucions totals. Com ja parlarem a 4.2 - Propostes de millora, pot ser el seu baix rendiment sigui producte dels canvis d'entrada.

D'altra banda, ha funcionat en 2 dels 3 vídeos nocturns i en un d'ells ha obtingut un bon resultat.

A priori, sense una optimització o compaginació amb altre mètode no seria el més adequat per implementar.

Wren: com ja comentàvem al analitzar-ho aquest mètode està dissenyat per identificació de persones. Tot i així els seus resultats han estat bons en una situació a l'aire lliure.

Ha funcionat per gairebé totes les proves. Els resultats han estat molt encertats sobretot amb llum diürna. Per aquesta raó, seria un mètode important que considerar en la implementació definitiva.

Adaptive: el primer que hem comentat sobre el mètode, al tema 2 – Anàlisi, és que era una optimització del mètodes Mean i Eigen. Sense cap dubte és el més competitiu.

En primer lloc, els seus temps de còmput són considerablement més baixos que els de la resta de mètodes. En la meitat de les proves a obtingut el millor temps. A més, no ha fallat cap cop en l'execució.

En segon lloc, ha obtingut varies imatges finals molt ben definides i perfilades.

Sense cap dubte aquest seria dels mètodes a implementar en una versió millorada.

Grimson: la seva algorítmica és molt semblant a la de Zivkovic. No obstant en execució presenten resultats molt diferents.

Primer, ha fallat a 3 de les 8 proves realitzades. A més, és el mètode que més triga en executar de tots. De manera contraposada, les seves imatges finals són de les millors perfilades.

Per tant, tot i la seva qualitat en detecció necessitaria una optimització en el temps de còmput. Com ja en parlarem al següent apartat (4.2 – Propostes de millora), pot ser només caldria afinar ce manera específica els seus Thresholds, entre altres variables.

Zivkovic: com ja esmentàvem al mètode anterior, sa seva algorítmica és molt semblant a la de Grimson.

En quan als resultats obtinguts, són molt positius. Tot i haver fallat en 2 execucions, els resultats que obté són molt detallats, en qualsevol àmbit. A més a més, realitza els càlculs en poc temps.

Sense cap dubte, aquest mètode ha ajudat en totes les optimitzacions a Test avions augmentat el percentatge d'encert. Per tant , hauríem de considerar-lo un mètode molt important a l'aplicació definitiva.

Prati: per com està plantejada l'algorítmica, aquesta funcionalitat no ha donat un rendiment molt alt. Ha executat en totes les ocasions, tot i que de nit el seu rendiment ha estat molt baix, també podria ser una necessitat d'afinar els thresholds.

Així doncs, si al nostre Test avions ho estructuréssim per passar varis mètodes, Prat sempre ens garantiria un resultat mínim, i això podria ser positiu. Però per individual és un mètode que encara s'hauria de perfilar una poc més.

Mean: com ja comentàvem la seva algorítmica és basa en dades estàtiques com molts altres mètodes.

Mentre que el seu grau percentatge d'actuació és del 100%, donat que no ha donat cap tipus d'error, les seves imatges finals no són gaire precises ja que en molts casos no detectava cap tipus de moviment i per tant retornava una imatge complet de liavió amb el fons sense delimitar la imatge. D'altra banda, ha detectat més moviment de nit però en zones realment petites, i per tant insuficients.

5.2 – Propostes de millores i solucions

A continuació volem tractar tots aquells aspectes que creiem que es podrien perfeccionar o canviar per obtenir un millor resultat.

En primer lloc, volem tractar un aspecte bastant negatiu de l'aplicació. Com ja hem comentat el codi està creat des d'una interfície Visual Studio. D'una banda facilita el fet de crear un projecte

en C++ i també el control sobre tots els arxius que el composen. Però hi ha actes que no haurien de suposar un problema, com ara agregar llibreries, però es compliquen excessivament fent que la dificultat no sigui la nostra aplicació sinó el manejament de la interfície.

D'altra banda, ara parlarem de quines millores ens hem plantejat per fer l'aplicació.

La primera millora hauria de tractar de millorar els resultats. Ja ens havíem plantejat obtenir una imatge amb un percentatge d'encert els més alt possible amb la unió (Test avions) dels resultats parcials de cada mètode. Això tindria dos conseqüències: en primer lloc, elevaria de forma insostenible el temps de còmput i, en segon lloc, faria més difícil l'execució d'aquest codi a temps real. És a dir, que si tots els mètodes han d'esperar a la finalització dels altres seria una solució que augmentaria el temps final de solució. Per tant, la nostra proposició seria la d'establir només uns mètodes per la feina amb una llum diürna, com ara Adaptive, Wren i Zivkovic, i d'altres pel torn nocturn, Adaptive, Grimson i Eigen.

I l'altre aspecte a comentar seria el fet de treballar amb un vídeo a temps real. Cal recordar que aquestes proves només han tractat de determinar el potencial de cada mètode en diferents situacions i que per tant el treball a temps real canviaria la situació notablement. Primer, hi hauria menys errors donat que l'entrada que reb cada mètode ja que tindria una entrada constant i, segon, ja podríem ajustar cada mètode (constants i thresholds, entre altres) pel reconeixement d'aquella entrada exclusivament. I d'aquesta manera, treballar sobre el camp per veure els punts forts de cada mètode basant-nos en les dades ja obtingudes.

5 - Referències

[1]. **Xiauoyu Wu, Yangsheng Wang and Jituo Li** – Video Background segmentation using adaptive background models.

[2]. **Chris Stauffer and W.E.L Grimson** - Adaptive background mixture models for real-time tracking

[3]. **Christopher Richard Wren, Ali Azarbayejani, Trevor Darrell, and Alex Paul Pentland** - Pfinder: Real-Time Tracking of the Human Body

[4]. **Rita Cucchiara, Costantino Grana, Massimo Piccardi, and Andrea Prati** - Detecting Moving Objects, Ghosts, and Shadows in Video Streams

[5]. **Zoran Zivkovic, Ferdinand van der Heijden** - Efficient adaptive density estimation per image pixel for the task of background subtraction

[6]. **OpenCV** .[Enlínea] http://opencv.willowgarage.com/wiki/.

5 - Apèndix

A continuació, declararem com estan estructurades les carpetes del CD adjunt:

Codi:

- Una carpeta comprimida anomenada Projecte avions, és un projecte de Visual Studio, desenvolupat en la versió 6.0. Aquesta conté tot el codi principal.
- Altra carpeta comprimida anomenada Test avions. És el codi de suport per fer proves.
- L'últim arxiu comprimit és cvblobslib_OpenCV_v8_3. La llibreria de blobs emprada en aquest desenvolupament.
- I finalment, OpenCV_1.1pre1a un arxiu executable amb la llibreria OpenCV amb les funcionalitats necessàries i una API per documentar-se.

Imatges, Gràfiques & Diagrames: carpeta amb tots les figures mostrades. A més hi ha una carpeta al seu interior anomenada Resultats, que conté totes les imatges obtingudes a cada estudi.

Mètodes: adjuntem totes les publicacions científiques de la bibliografia.

L'arxiu PDF de la memòria es troba a l'arrel principal.