



# VISIÓ PER COMPUTADOR

# EN UNA PLATAFORMA BASADA EN UN DSP

Memòria del Projecte Fi de Carrera
d'Enginyeria en Informàtica
realitzat per

**Ricard Royo Tort**

i dirigit per

**Miquel Àngel Senar Rosell**

Bellaterra, 14 de Juny del 2007

# Índex de continguts

**TAULA DE FIGURES ..... 5**

**1. INTRODUCCIÓ ..... 6**

1.1 INTRODUCCIÓ I MOTIVACIÓ DEL PROJECTE ..... 6

1.2 OBJECTIUS DEL PROJECTE ..... 7

1.3 PLANIFICACIÓ DEL PROJECTE ..... 8

1.4 ORGANITZACIÓ DE LA MEMÒRIA ..... 9

**2. AXIS 242S IV VÍDEO SERVER ..... 11**

2.1 QUÈ ÉS UN VÍDEO SERVER ? ..... 11

2.2 AXIS 242S IV VÍDEO SERVER ..... 11

2.3 ARQUITECTURA DE LA PCB ..... 12

2.4 AXIS ETRAX100LX ..... 13

2.5 AXIS ARPTEC-2 ..... 14

2.6 TEXAS INSTRUMENTS TMS320DM642 ..... 15

2.7 EL FLUX DE DADES ..... 16

2.8 AXIS SDK ..... 17

2.9 MÈTODE DE DESENVOLUPAMENT PER AL DSP ..... 17

2.10 MÈTODE DE DESENVOLUPAMENT PER A L'ETRAX100LX ..... 20

**3. TEXAS INSTRUMENTS TMS320DM642 ..... 21**

3.1 LA FAMÍLIA DAVINCI™ DE TEXAS INSTRUMENTS ..... 21

3.2 ARQUITECTURA DEL TMS320DM642 ..... 21

3.3 ARQUITECTURA DEL NUCLI C64X™ ....22

3.4 EL PORT DE VÍDEO ....24

3.5 STACK SOFTWARE....26

3.6 APIS D'AXIS COMMUNICATIONS ....27

**4. IMPLEMENTACIÓ DE L'ALGORISME DE BACKGROUND SUBTRACTION....31**

4.1 CONSIDERACIONS D'US....31

4.2 FONAMENTS MATEMÀTICS DEL UNIMODAL I EL MULTIMODAL ....33

4.3 IMPLEMENTACIÓ DEL UNIMODAL ....37

4.4 ESTAT DE L'ART I PROBLEMES TROBATS EN L'UNIMODAL ....42

4.5 CARACTERÍSTIQUES DE L'ALGORISME FINALMENT APLICAT....45

4.6 RESTRICCIONS DE LA PLATAFORMA D'AXIS ....46

4.7 OPTIMITZACIONS EFECTUADES....49

4.8 PROBLEMES TROBATS ....50

4.9 TESTS DE L'ALGORISME ....52

4.10 BENCHMARKS ....53

**5. DESENVOLUPAMENT DEL COMPTADOR ZENITAL DE PERSONES....55**

5.1 ESPECIFICACIONS DEL COMPTADOR....55

5.2 ALGORISME DE COMPTATGE DE PERSONES....57

5.3 PROVES EN MATLAB ....61

5.4 CONCLUSIONS OBTINGUDES....65

5.5 IMPLEMENTACIÓ EN EL DSP ....66

5.6 ARQUITECTURA DE L'APLICACIÓ ..... 67

5.7 INTERFÍCIE GRÀFICA ..... 69

5.8 RESTRICCIONS DE LA PLATAFORMA D'AXIS ..... 71

6.9 PROBLEMES TROBATS ..... 73

5.10 TESTS DE L'APLICACIÓ ..... 74

**6. CONCLUSIONS I LÍNIES FUTURES DE TREBALL ..... 75**

6.1 CONCLUSIONS ..... 75

6.2 LÍNIES FUTURES DE TREBALL ..... 77

**BIBLIOGRAFIA ..... 78**

**ABSTRACT ..... 80**

# Taula de Figures

FIGURA 1- 1 – DIAGRAMA DE GANTT DEL PROJECTE....8

FIGURA 2- 1 - EXEMPLE ARQUITECTURA AMB VÍDEO SERVERS....11
FIGURA 2- 2 - AXIS 242S IV VIDEO SERVER....12
FIGURA 2- 3 - VISTA FRONTAL I POSTERIOR DEL VÍDEO SERVER....12
FIGURA 2- 4 - FOTOGRAFIA DEL PLC DEL VIDEO SERVER....13
FIGURA 2- 5 - AXIS ETRAX100LX....14
FIGURA 2- 6 - AXIS ARPTEC-2....15
FIGURA 2- 7 - TEXAS INSTRUMENTS TMS320DM642....15
FIGURA 2- 8 – FLUX DE DADES DEL PLC....17
FIGURA 2- 9 – ESQUEMA DE CONNEXIONAT DEL EMULADOR JTAG....18
FIGURA 2- 10 – ESQUEMA DE PROGRAMACIÓ PER AL DSP....19

FIGURA 3- 1 – DIAGRAMA DE BLOCS DEL TMS320DM642....22
FIGURA 3- 2 – ARQUITECTURA DEL NUCLI C64X™....24
FIGURA 3- 3 – DIAGRAMA DE BLOCS DEL PORT DE VÍDEO....25
FIGURA 3- 4 – ESQUEMA DE CAPES SOFTWARE DEL TMS320DM642....27
FIGURA 3- 5 – DIAGRAMA DE BLOCS DEL SDK D'AXIS....30

FIGURA 4- 1– IMATGES PROBLEMÀTIQUES DE CÀMERES DE SEGURETAT....32
FIGURA 4- 2 - SEQÜÈNCIA ORIGINAL DE VÍDEO....33
FIGURA 4- 3 - GRÀFICA DE VALORS DEL PÍXEL CENTRAL EN EL TEMPS....34
FIGURA 4- 4 - FREQÜÈNCIA D'APARICIÓ DE VALORS EN EL PÍXEL CENTRAL....34
FIGURA 4- 5 - GRÀFIQUES DE LA DISTRIBUCIÓ NORMAL CENTRADES EN LA MITJANA DE FREQÜÈNCIES.....36
FIGURA 4- 6 - GRÀFIQUES DE DUES GAUSSIANES PER APROXIMAR LES FREQÜÈNCIES.....37
FIGURA 4- 7 - MITJAN A PONDERADA DE LA SEQÜÈNCIA ORIGINAL....38
FIGURA 4- 8 - DESVIACIÓ TÍPICA DE LA SEQÜÈNCIA ORIGINAL....39
FIGURA 4- 9 - RESULTATS DE L'ALGORISME....41
FIGURA 4- 10 - EXEMPLE DE SEQÜÈNCIA AMB FONS MÒBIL.....42
FIGURA 4- 11 - EXEMPLE D'ERRORS DE CLASSIFICACIÓ.....44
FIGURA 4- 12 - EXEMPLE DE L'ESPAI DE COLORS YCBCR....47
FIGURA 4- 13 - SAMPLEJAT 4:2:2....48
FIGURA 4- 14 - RESULTAT COMPARATIU ENTRE LA VERSIÓ PC I LA VERSIÓ DSP....53

FIGURA 5- 1 – ESQUEMA INSTAL·LACIÓ CÀMERA ZENITAL....55
FIGURA 5- 2 –SEQÜÈNCIA ZENITAL ORIGINAL....57
FIGURA 5- 3 – SEQÜÈNCIA DE DIFERÈNCIA DE FRAMES....58
FIGURA 5- 4 – SEQÜÈNCIA BINARITZADA....58
FIGURA 5- 5 – NÚMERO DE PÍXELS DINS DEL REQUADRE D'INTERÈS....59
FIGURA 5- 6 – RESULTAT DELS TESTS PROPORCIONALITAT DE COMPTATGE....62
FIGURA 5- 7 – GRÀFIC DE FIABILITAT EN FALSOS POSITIUS....64
FIGURA 5- 8 – GRÀFIC DE FIABILITAT EN FALSOS NEGATIUS....64
FIGURA 5- 9 – GRÀFIC DE FIABILITAT TOTAL....64
FIGURA 5- 10 – DIAGRAMA DE BLOCS DEL COMPTADOR DE PERSONES....68
FIGURA 5- 11 – CAPTURES DE PANTALLA DEL COMPTADOR DE PERSONES....69
FIGURA 5- 12 – ARQUITECTURA DE LA INTERFÍCIE GRÀFICA DEL COMPTADOR DE PERSONES....71
FIGURA 5- 13 – FOTOGRAFIA DE LA INSTAL·LACIÓ REAL DEL COMPTADOR.....74

# 1. Introducció

## 1.1 Introducció i Motivació del Projecte

DAVANTIS Technologies S.L. és una empresa innovadora creada el 2004 per un equip d'emprenedors multidisciplinari amb seu central al Centre de Visió per Computador de la Universitat Autònoma de Barcelona. Des de la seva creació s'ha centrat en convertir-se en el líder en disseny i fabricació de software intel·ligent per a videovigilància, oferint als seus clients productes per al control de perímetres, detecció d'intrusions i anàlisis comportamental dels clients.

DAVIEW Perimeters és el producte estrella de la casa per al control de perímetres i la detecció d'intrusos. Aquest integra en un sol producte la vigilància intel·ligent i monitoratge de les càmeres, esdevenint així un producte complert en si mateix. Apart de detectar les possibles intrusions, l'anàlisi intel·ligent diferencia entre persones, vehicles i altres objectes per tal de poder fer un filtratge adequat segons les necessitats del sistema. A més, DAVIEW Perimeters disposa de totes les funcionalitats freqüentment trobades en altres productes de monitoratge no intel·ligent com la gravació d'esdeveniments, establiment d'horaris de vigilància, nivells d'alarmes, alertes remotes, monitoratge a distància de les càmeres, etc.

Durant els últims anys hem pogut veure un gran increment en la capacitat d'integració dels fabricants de hardware que s'ha traduït en una millora en el poder de processament dels microprocessadors. Aquest increment ha vingut guiat per la coneguda llei de Moore que marca un increment de la capacitat d'integració del 200% cada 18 mesos. Donat aquests fets, el preu i la capacitat de processament dels microprocessadors i DSP per a sistemes encastats fa que a dia d'avui aquests es puguin veure com una solució força viable per al tractament d'imatges multimèdia.

A DAVANTIS Technologies s'han realitzat diversos estudis pels quals es veu una forta tendència a integrar l'anàlisi intel·ligent de les imatges en les pròpies càmeres de seguretat. D'aquesta manera s'aconsegueix acostar a la càmera tota la intel·ligència donada per els algorismes de visió, permetent que aquesta reaccioni als estímuls externs procedents de la imatge que està gravant. Apart d'això, el fet de no haver de disposar d'un servidor específic per a l'anàlisi de les imatges, fa que sigui possible tenir sistemes del tipus thin-client[1] molt més robustos i escalables que la clàssica arquitectura client-servidor. Tot això fa que els sistemes encastats siguin cada cop la solució predominant per a la videovigilància del futur.

## 1.2 Objectius del Projecte

La feina feta durant la realització d'aquest projecte ha girat en tot moment al voltant del *242S IV Video Server*, el primer servidor de vídeo de la casa Axis Communications que incorpora de sèrie un DSP de Texas Instruments especialment orientat a la programació d'algorismes de vídeo intel·ligents. Els detalls d'aquest, així com les consideracions tècniques que s'han tingut en compte es parlaran amb més detall en propers capítols però es podria dir que els objectius d'aquest projecte s'han centrat principalment en dos branques ben diferenciades:

- Implementar l'algorisme de *Background Subtraction* usat en el DAVIEW Perimeters al DSP TMS320DM642 de Texas Instruments, avaluar-ne el rendiment, comparar-lo amb la versió per a PC i estudiar l'adequació d'aquest processador com a nucli de processament per a migrar-hi la part servidora de la plataforma.

- Desenvolupar un comptador de persones per càmera zenital en el servidor de vídeo 242s iv d'Axis Communications i avaluar aquesta plataforma per a desenvolupar una solució complerta que integri tots els seus components.

Cal remarcar que en el desenvolupament del comptador no s'ha donat tanta importància en el fet de trobar un algorisme adequat per al comptatge de persones com en el

[1] Sistemes en les que el client realitza poca o cap part del processament de les imatges.

fet d'obtenir una solució complerta que integri totes les parts del sistema. A més, Pol Cunill Rafael, company nostre a DAVANTIS Technologies, ha estat realitzant el seu PFC paral·lelament al meu avaluant diferents algorismes per al comptatge de persones en un entorn PC. És intenció de l'empresa migrar en un futur proper algun d'aquests algorismes a la plataforma en la que jo he implementat el meu comptador i per tant integrar-la amb la solució complerta que jo he realitzat en aquest projecte.

## 1.3 Planificació del projecte

La realització d'aquest projecte va començar el Juliol del 2006 a DAVANTIS Technologies i ha tingut una duració de pràcticament un any. Durant el transcurs d'aquest temps, s'han realitzat diversos canvis de rumb que han permès avançar en el coneixement del problema i que, fins i tot, han portat a la creació de noves línies de productes dins de l'empresa. Cal remarcar que no hi havia cap tipus d'experiència en programació d'aquest tipus de sistemes a l'empresa i en el moment en el que es va acceptar el projecte hi va haver una gran fase de documentació sobre el problema.

La figura 1-1 mostra el diagrama de Gantt que s'ha seguit durant la realització del projecte. Aquest diagrama mostra de forma resumida quins han estat els principals passos a l'hora de desenvolupar el projecte.

| Id. | Nombre de tarea | Comienzo | Fin | Duración |
|---|---|---|---|---|
| 1 | Documentació sobre el DSP | 03/07/2006 | 15/09/2006 | 55d |
| 2 | Primers Testos amb el Emulador | 01/08/2006 | 31/08/2006 | 23d |
| 3 | Entrega d'un Servidor de Vídeo 242S IV | 04/09/2006 | 04/09/2006 | 1d |
| 4 | Implementació del Unimodal en el DSP | 18/09/2006 | 09/11/2006 | 39d |
| 5 | Medició de Benchmarks i comparació de resultats amb la versió per PC | 10/11/2006 | 18/12/2006 | 27d |
| 6 | Testeig d'algorismes de comptatge de persones en Matlab | 08/01/2007 | 23/01/2007 | 12d |
| 7 | Documentació sobre el Servidor de Vídeo | 24/01/2007 | 21/03/2007 | 41d |
| 8 | Desenvolupament del l'algorisme de comptatge de persones en el DSP | 24/01/2007 | 08/02/2007 | 12d |
| 9 | Migració de l'algorisme de Background Subtraction en el DSP | 12/02/2007 | 22/03/2007 | 29d |
| 10 | Medició de Benchmarks i comparació de resultats amb la versió per PC | 23/03/2007 | 16/04/2007 | 17d |
| 11 | Desenvolupament de l'interfície gràfica del comptador | 16/04/2007 | 14/05/2007 | 21d |
| 12 | Desenvolupament de la part de configuració del comptador | 30/04/2007 | 25/05/2007 | 20d |
| 13 | Escriptura de la memòria del projecte | 01/05/2007 | 11/06/2007 | 30d |

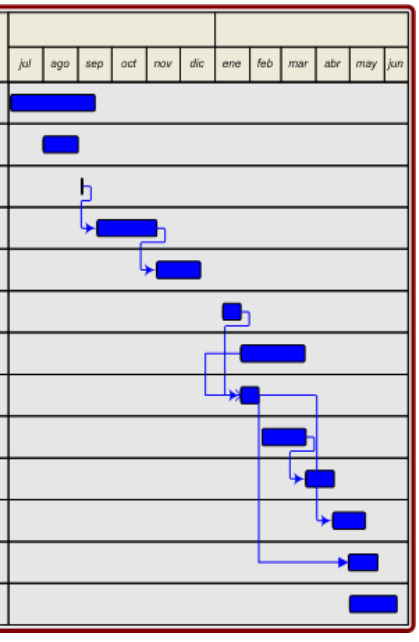


Figura 1- 1 – Diagrama de Gantt del projecte

## 1.4 Organització de la Memòria

La memòria d'aquest projecte està organitzada en 5 capítols de la següent forma:

- El capítol 2: Axis 242S IV Vídeo Server. Farà un recorregut per el Vídeo Server d'Axis explicant-ne la seva arquitectura, els seus components principals i la manera que tenen aquests de cooperar per a realitzar una determinada tasca. A més, en aquest capítol també s'explicarà el mètode de desenvolupament que s'ha fet servir per a programar el DSP TMS320DM64 i el SoC[2] Etrax100LX.

- El capítol 3: Texas Instruments TMS320DM642. Explica les principals característiques del DSP de TI. Entre aquestes, es veurà l'arquitectura del nucli del DSP i el seu controlador de vídeo. En aquest capítol, a més, s'explicarà quin és el stack de software que ens proporciona Texas Instruments per a simplificar la tasca de programació del seu DSP.

- El capítol 4:Implementació de l'algorisme de *Background Subtraction*. S'expliquen els passos que s'han seguit per a la implementació de l'algorisme de *Background Subtraction* a la nostra plataforma. En primer lloc, es fa un recorregut per els fonaments matemàtics en els que es basen els algorismes d'aquest tipus. Tot seguit, s'explica com funciona l'algorisme Unimodal i se'n fa una crítica tot veient quin és l'estat de l'art. Finalment es recullen els passos que s'han seguit per a desenvolupar l'algorisme, les optimitzacions efectuades, els problemes trobats i els tests i benchmarks que s'han obtingut.

- El capítol 5: Desenvolupament del Comptador Zenital de Persones. Fa un recull dels passos que s'han seguit per al desenvolupament del comptador de persones. En primer lloc, s'explica com funciona l'algorisme i es detallen el conjunt de tests que s'han realitzat en Matlab per a determinar la fiabilitat del mateix, tot deixant-ne veure les conclusions obtingudes. Tot seguit s'explica l'arquitectura de l'aplicació sencera, els diferents components que la formen i

[2] System on chip

la manera que tenen d'interaccionar entre ells. Finalment es detallen els problemes trobats, i els tests als quals s'ha sotmès l'aplicació.

- El capítol 6: Conclusions i línies futures de treball. Fa un recull de les conclusions obtingudes tant en la implementació de l'algorisme de *Background Subtraction* com en el desenvolupament del comptador zenital de persones. També es proposen properes línies de treball que caldria seguir en les dues branques d'aquest projecte.

# 2. Axis 242S IV Vídeo Server

## 2.1 Què és un Vídeo Server ?

Es coneix un vídeo server com una màquina dedicada a servir vídeo a través d'una xarxa. Aquest s'encarrega d'enviar un *streaming*[3] de les imatges que captura d'una altra font per que puguin ser visualitzades des de qualsevol màquina connectada a la xarxa.

En aplicacions de vigilància aquestes imatges són finalment monitoritzades en una aplicació que captura totes les seqüències enviades des dels diferents vídeo servers connectats a la xarxa. D'aquesta manera s'aconsegueix l'aprofitament màxim dels recursos de la xarxa i una gran escalabilitat del sistema. En la Figura 2-1 podem veure un exemple d'arquitectura usual en aquest tipus d'instal·lacions.

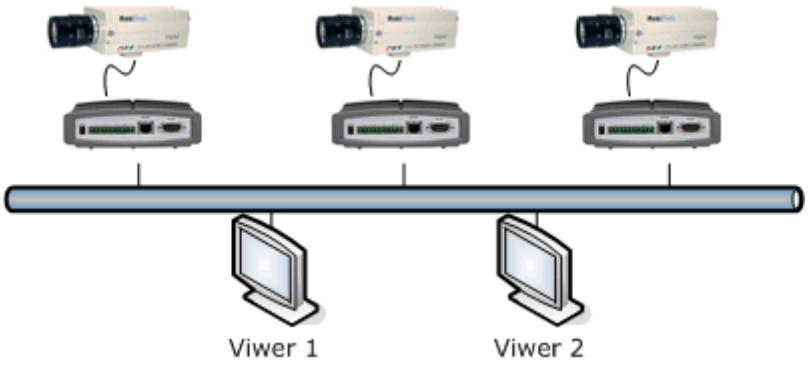


**Figura 2- 1 - Exemple arquitectura amb vídeo servers**

## 2.2 Axis 242S IV Vídeo Server

*Axis Comunications* es el líder mundial en productes de vídeo en xarxa com càmeres-IP vídeo servers. L'any 2004 Axis va llençar al mercat el 242S IV, el primer model de la casa que incorpora de fàbrica un DSP de Texas Instruments específicament orientat al tractament de vídeo. Tal i com el seu nom indica aquest model pretén cobrir la demanda creixent de vídeo intel·ligent integrant-ho tot en un sol dispositiu. En la figura 2-2 podem veure una fotografia de l'aspecte extern del 242S IV.

[3] Mètode per enviar dades en temps real "tou" a través de la xarxa.

Figura 2- 2 - Axis 242S IV Video Server

## 2.3 Arquitectura de la PCB[4]

A la figura 2-3 podem veure les parts anteriors i posteriors respectivament del vídeo server. Tal i com es pot apreciar, aquest model disposa de dues entrades analògiques del tipus coaxial i *s-vídeo*[5]; una sortida analògica del tipus coaxial; dues entrades d'àudio; un port ethernet; un *terminal block*; un port *d-sub*[6]; un connector per l'alimentació i diversos leds indicadors d'estat.


Figura 2- 3 - Vista frontal i posterior del vídeo server

Si desmuntem la coberta exterior amb una clau TOR[7] podem accedir al circuit imprès on hi podrem apreciar els diferents components presents a la placa. En la figura 2-4 podem veure marcats els tres circuits integrats principals del nostre PLC. El número 1 és el DSP de Texas Instruments TM320DM642 del qual parlarem amb més detall en el capítol 2 de la memòria. El circuit marcat amb un 2 es l'Etrax100LX, un microprocessador desenvolupat per Axis dissenyat per simplificar el desenvolupament d'aplicacions amb un Linux encastat.

[4] Princted Circuit Board
[5] Standard per a la transmissió de vídeo analògic a través d'un cable
[6] Port sèrie asíncron
[7] Tipus de clau mecànica per al muntatge industrial

El número 3 és l'Arptec-2 un ASIC[8] també desenvolupat per Axis pensat per ser inclòs en totes les càmeres-IP i vídeo servers per a tasques de compressió de vídeo.

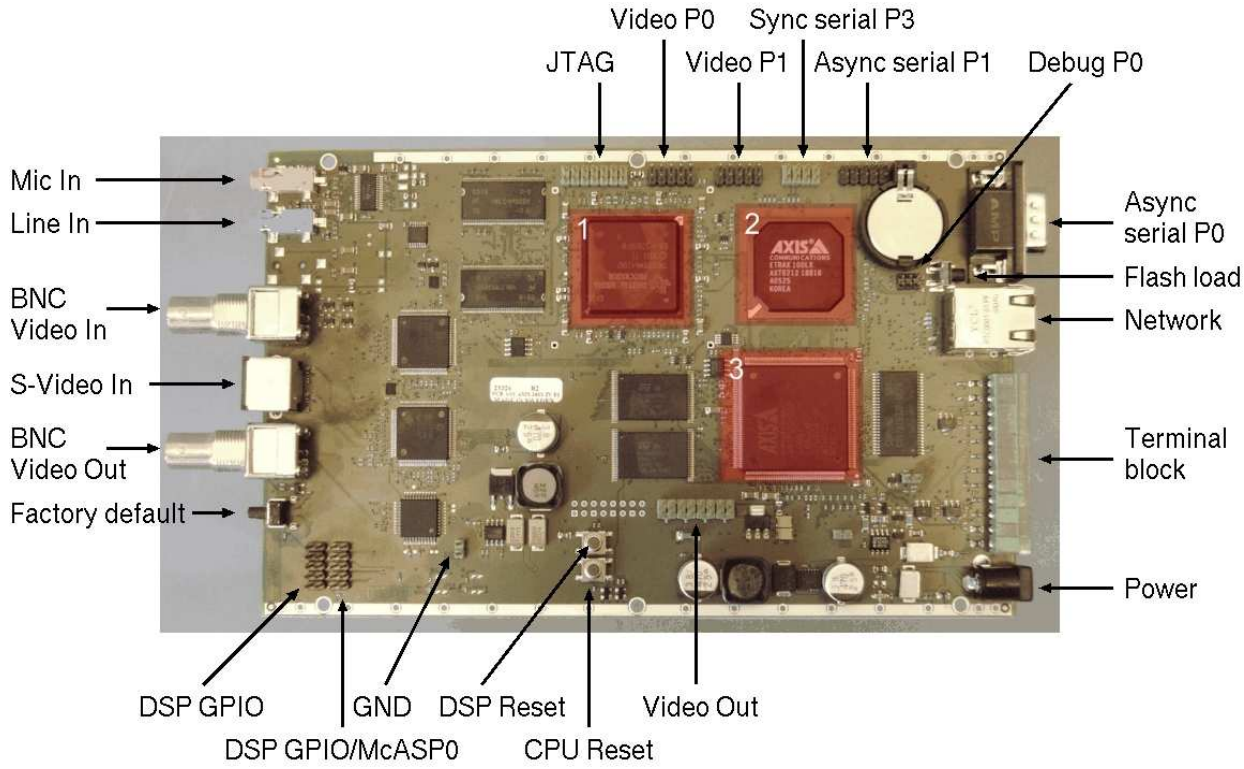


Figura 2- 4 - Fotografia del PLC del video server

## 2.4 Axis ETRAX100LX

l'Axis ETRAX100LX és un SoC[9] basat en un microprocessador de 32 bits RISC[10] pensat per al desenvolupament d'aplicacions en xarxa encastades en servidors de vídeo, d'impressió i càmeres-IP. Està especialment orientat per a l'ús d'un kernel de Linux sense modificacions en sistema de gestió de memòria, ja que incorpora una *MMU*[11] específica per a tal fi. A més l'ETRAX100LX incorpora un controlador Ethernet 10/100MBit, 4 ports sèrie asíncrons, 2 ports sèrie síncrons, 2 ports USB[12], 2 ports paral·lels, 4 ports ATA[13], 2 ports SCSI[14], suport per a SDRAM[15], Flash i EEPROM[16], així com una cache unificada pera dades i instruccions de 8 Kbytes.

[8] Aplication Specific Integrated Circuit
[9] System on Chip
[10] *Reduced Instruction Set Computer*
[11] Memory Managment Unit
[12] Universal Serial Bus
[13] *Advanced Technology Attachment*
[14] *Small Computer System Interface*
[15] *Synchronous Dynamic Random Access Memory*


**Figura 2- 5 - Axis ETRAX100LX**

Dins el nostre PCB, el microprocessador s'encarrega de fer de màster del sistema, controlant totes les parts reactives de l'aplicació així com un petit servidor web per al control de la configuració del servidor de vídeo. A més, per al desenvolupament de la interfície web del nostre comptador de persones hem hagut de fer us del compilador creuat que proporciona Axis per al desenvolupament de scripts CGI[17] per a la gestió dels logs del sistema.

## 2.5 Axis ARPTEC-2

l'ARPTEC-2, també produït per Axis Communications, és un *ASIC*[6] específicament dissenyat per a ser incrustat en totes les càmeres i servidors de vídeo de la marca. Conté un microprocessador de 32 bits RISC[8] que s'encarrega d'efectuar les decissions de codificació a nivell de bloc. A més, per tal d'accelerar el processament de les imatges, s'ha inclòs hardware específic i canals dedicats de DMA per a la transmissió de dades des de la memòria SRAM interna a la memòria SDRAM externa.

[16] *Electrically-erasable programmable read-only memory*
[17] Common Gateway Interface


Figura 2- 6 - Axis ARPTEC-2

Per a la nostra aplicació no em fet us d'aquest chip, ja que, tal i com es veurà en la següent secció, el flux de dades que passa per al DSP no és el mateix que el flux de dades que arriba al ARPTEC-2. Tot i així, cal dir que les imatges que nosaltres processem en el DSP són també enviades al ARPTEC-2, sense la nostra intervenció, per a ser comprimides i finalment associades a la informació generada per al DSP en format SVG[18].

## 2.6 Texas Instruments TMS320DM642

El TMS320DM642 és un DSP de la família DaVinci™ de Texas Instruments especialment optimitzat per al processament de dades multimèdia per a aplicacions a temps real. Està basat en l'arquitectura VelociTI.2™ VLIW[19] amb un clock rate de 600MHz i dos nivells de cache.


Figura 2- 7 - Texas Instruments TMS320DM642

Aquest ha estat el processador en el qual hem centrat més els nostres esforços. Tant l'algorisme d'extracció de fons com el comptador de persones han estat implementats per

[18] Scalar Vector Grafics. Estàndard per a gràfics vectorials en format XML.
[19] Very Long Instruction Word

aconseguir una optimització de rendiment per a aquest DSP. Cal dir però que bona part de la feina ens ha sigut facilitada gràcies a que el SDK[18] de Axis ens donava el Sistema Operatiu i els drivers específicament dissenyats per a l'arquitectura de la PCB. En el capítol 3 es parlarà amb més detall de la seva arquitectura i de les capes de software que s'han fet servir per al desenvolupament.

## 2.7 El flux de dades

Per tal d'entendre be el funcionament de l'aplicació cal tenir en compte l'arquitectura de la placa desenvolupada per Axis. Aquesta, incorpora una sèrie de restriccions que no permeten que determinades funcionalitats siguin fàcilment desenvolupades. Tot i així, cal mencionar que aquestes restriccions tenen sentint dins el context en el que Axis a posat a disposició dels seus socis un SDK[20] específicament lligat a aquesta arquitectura.

Tal i com es pot veure en la figura 2-8, el flux de vídeo que prové de l'entrada analògica es divideix en dos per a ser mostrejat i quantitzat pels conversors analògic-digital. Un cop digitalitzat, el primer flux és enviat al ARTPEC-2 on s'hi processaran les imatges per a la seva compressió, escalat, rotació i *overlay*[21] segons sigui el cas. L'altre flux és enviat al DSP de Texas Instruments per al seu tractament on s'hi aplicaran tots els algorismes de visió segons l'aplicació que es desitgi fer. Posteriorment els dos fluxos s'uneixen un altre cop el ETRAX100LX per a ser enviats per a la xarxa als diferents clients. Cal dir que, tal i com està dissenyada l'arquitectura i l'SDK[18], no està previst que un flux estrictament de vídeo sigui enviat a l'ETRAX100LX des del DSP, sinó més aviat la informació extreta a partir de l'anàlisi que s'hagi fet amb els corresponents algorismes de visió. Per a tal fet, s'ha previst en el SDK[18] un mètode per enviar dades posicionals en format SVG[16]. Apart d'això, s'ha afegit un conversor digital-analògic a la sortida de vídeo del DSP.

Aquesta darrera sortida l'hem connectat nosaltres a una capturadora de vídeo analògica instal·lada en l'estació de treball per a poder realitzar una depuració. Per a poder processar i veure correctament la senyal de vídeo s'ha fet servir una aplicació força coneguda per a aquestes tasques, el Virtual Dub.

[20] Software Development Kit.
[21] Procés en el qual se superposa una imatge estàtica (ie un logotip) sobre la seqüència de vídeo.

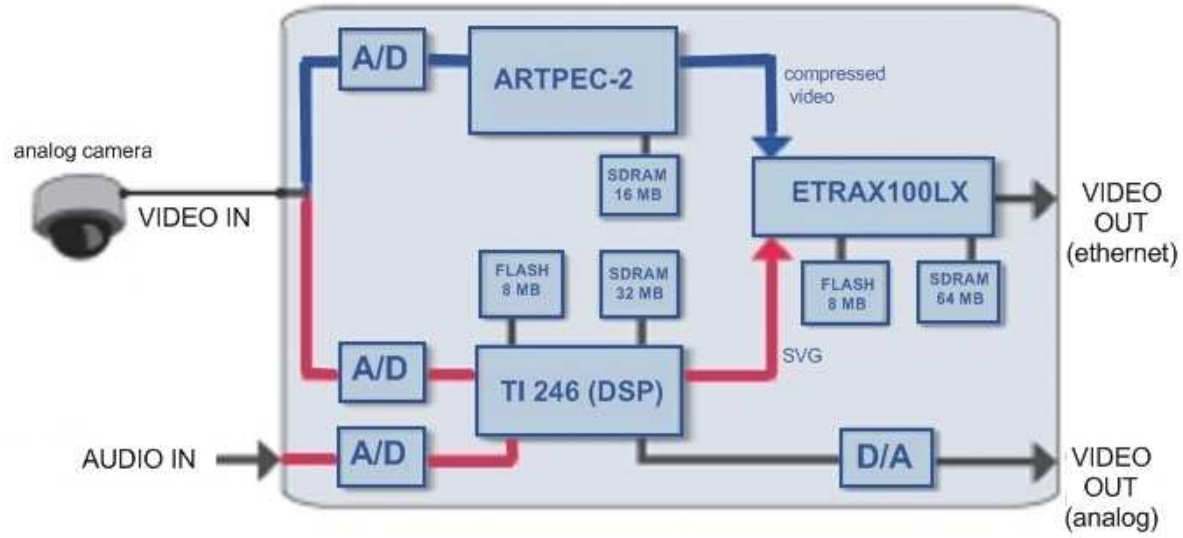


Figura 2- 8 – Flux de dades del PLC

## 2.8 Axis SDK

Gràcies al contracte de col·laboració entre DAVANTIS Technologies S.L. i Axis Communications hem pogut tenir a la nostra disposició una sèrie de SDK per al desenvolupament d'aplicacions incrustades en el DSP i el Linux incrustat en el ETRAX100LX. Aquests SDK han estat crucials en el desenvolupament, no tan sols per la simplificació a l'aportació de software que suposen, sinó per a la integració de les nostres aplicacions dins el marc de software que Axis ja inclou en els seus vídeo servers.

Apart d'això, Axis també ens ha proporcionat assistència tècnica telefònica i per e-mail per a resoldre tots els problemes que ens hem trobat durant el desenvolupament de la nostra aplicació. En els capítols successius es parlarà amb més detall de les funcionalitats específiques que ha aportat cada SDK i quines d'aquestes funcionalitats hem utilitzat per al nostre propòsit.

## 2.9 Mètode de desenvolupament per al DSP

Per al desenvolupament del l'algorisme d'extracció de fons i el comptador de persones hem agut de generar codi per a dos processadors diferents: el TMS320DM642 de TI i l'ETRAX100LX. En els dos casos les tecnologies necessàries per a produir el codi han estat

diferents, entre altres coses per a la necessitat de compiladors específics per a cada processador.

Tal i com s'ha pogut veure en la figura 2-4, la nostra placa incorpora un connector del tipus JTAG[22]. Aquest connector s'utilitza normalment per a programar sistemes encastats amb l'ajuda d'un programador específic segons el microprocessador. En el nostre cas, Texas Instruments i algunes empreses associades proporcionen un emulador/analitzador JTAG[20]. Aquest està estretament lligat al Code Composer Studio, un IDE també desenvolupat per TI que ens permet realitzar de forma senzilla la generació, emulació, volcat i depuració del nostre codi. En la figura 2-9 podem veure una fotografia d'un d'aquests emuladors i l'esquema de connexionat típic per a la nostra placa.

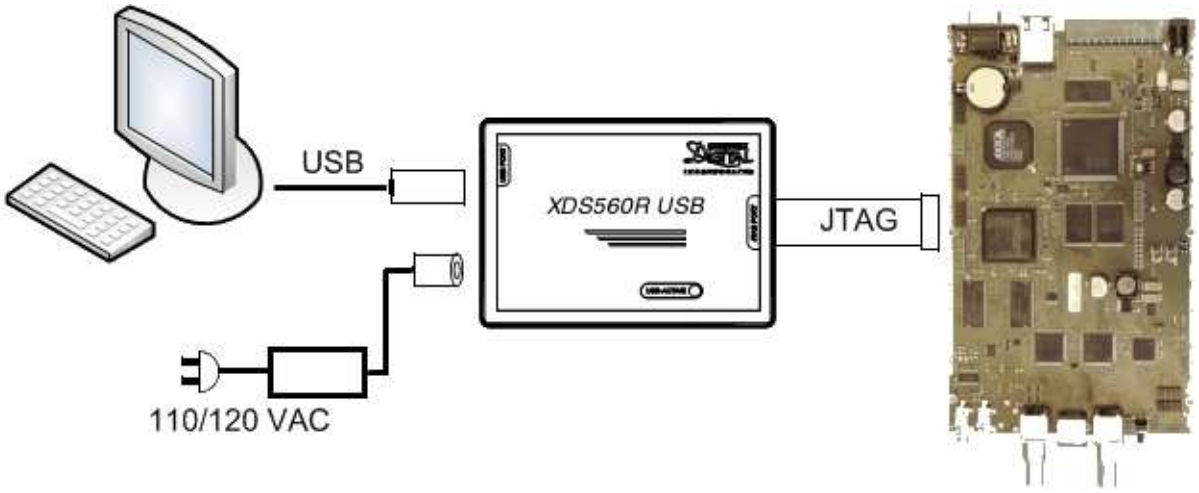


**Figura 2- 9 – Esquema de connexionat del emulador JTAG**

Malauradament, degut al limitat pressupost destinat al projecte i el fet que els emuladors JTAG per a aquesta arquitectura acostumen a ser força cars, hem hagut de fer servir una altra tècnica de desenvolupament que, tot i no ser tant pràctica com la mencionada més amunt, ens ha servit perfectament per als nostres propòsits. Aquesta tècnica fa us del SDK que ens ha proporcionat Axis per a enviar traces de l'execució de codi al ETRAX100LX. Allà, un procés desenvolupat per Axis util·litza la biblioteca syslog per enregistrar totes les traces provinents del DSP en els logs del sistema operatiu linux. D'aquesta manera, és possible poder fer un depurat del codi força bàsic fent crides a la

[22] Joint Test Action Group

l'API[23] proporcionada per el SDK d'Axis. En propers capítols es parlarà amb més detall d'aquesta API[21].

A més a més, hem fet servir la interfície web desenvolupada per Axis incrustada en el vídeo server per a poder volcar el nostre codi a la memòria Flash del DSP. Aquesta interfície permet carregar i descarregar mòduls IV[24] prèviament formatjats segons les especificacions d'Axis i TI d'una forma força senzilla. A més, per tal de poder veure de forma visual com tracta les imatges el nostre DSP, hem utilitzat la sortida analògica connectada a una capturadora de vídeo instal·lada en la nostra estació de treball depurant la part visual dels nostres algorismes. Per tal de fer això, hem fet us del Virtualdub, una aplicació open source [25]molt freqüentment usada per a la captura, transformació i compressió de seqüències de vídeo. En la figura 2-10 podem veure l'esquema de programació que finalment hem fet servir per a desenvolupar el projecte.

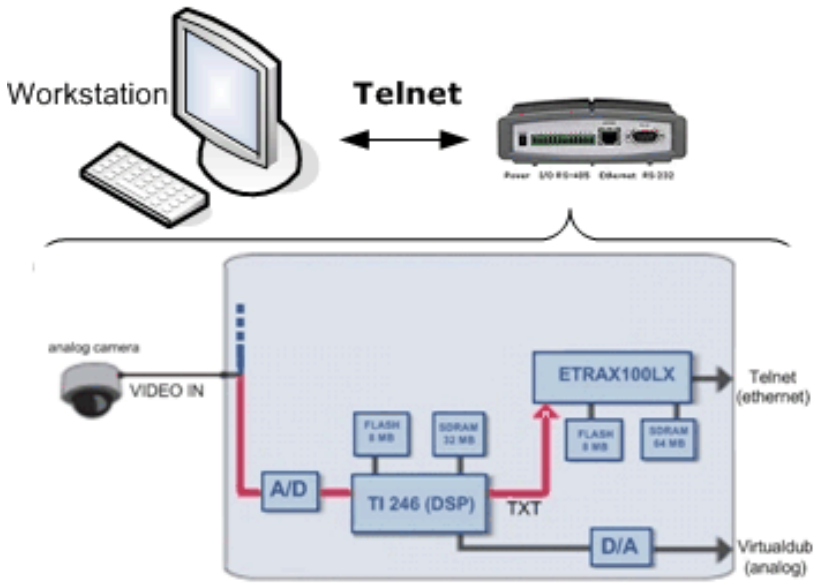


**Figura 2- 10 – Esquema de programació per al DSP**

[23] Aplication Programming Interface
[24] Terme usat per Axis per anomenar el firmware que es pot carregar en el DSP
[25] Terme normalment usat per a aquelles aplicacions llicenciàdes sota la GPL (General Public License)

## 2.10 Mètode de desenvolupament per a l'ETRAX100LX

Per tal de desenvolupar codi per a l'ETRAX100LX, hem hagut de fer us d'un altre mètode totalment diferent al fet servir per al DSP. Axis Technologies proporciona un altre SDK per a la compilació de codi específic per a processadors ETRAX. Aquest SDK, instal·lat en una Workstation Linux, ens ha servit per a programar el scripts CGI necessaris pel processament de les logs del comptador de persones.

El procés de desenvolupament que hem seguit per al ETRAX100LX ha suposat, en primer lloc, fer un compilat per al Linux de la Workstation on hem fet un primer testejat dels algorismes amb dades simulades. Seguidament, hem compilat els CGIs amb el compilador creuat i els hem volcat al ETRAX fent us d'una connexió FTP, i finalment hem fet un depurat i testejat de l'aplicació amb les dades reals generades per al DSP.

# 3. Texas Instruments TMS320DM642

## 3.1 La família DaVinci™ de Texas Instruments

Texas Instruments ofereix una gran gama de DSPs per a cobrir les diferents necessitats que el mercat demanada actualment. Una de les famílies que s'està potenciant més últimament és la de solucions basades en DSP DaVinci™. Aquesta família de SoCs[26] està específicament lligada al tractament de vídeo i àudio a temps real combinant en un sol chip un *nucli*[27] DSP amb diferents perifèrics i hardware d'acceleració per al processament d'imatges. A més, els nous models de la família, TMS320DM644x, incorporen en el propi chip un microprocessador ARM926 per a suportar les parts més reactives del desenvolupament.

Cal mencionar que tots els SoCs[1] de la familia DaVinci™ estan basats en els *nuclis* de DSPs C64x™ i C64x+™ . Aquests son compatibles a nivell de codi amb tots els nuclis de la família C6000, present en diverses famílies de DSPs de Texas instruments, i per tant permeten una reutilització força gran en cas de canvi a un sistema diferent.

## 3.2 Arquitectura del TMS320DM642

El TMS320DM642 és un dels primers SoCs[1] de la família DaVinci™ que Texas Instruments va llençar la mercat. Aquest està basat en un nucli DSP C64x™, disposa d'arquitectura tipus Harvard[28] i conté dos nivells de cache. El primer nivell disposa de 16 Kbytes del tipus *direct mapped*[29] per a instruccions i 16 Kbytes *two-way set-associative*[30] per a les dades. El segon nivell, en canvi , és comú per a les dades i les instruccions, i disposa de 256 Kbytes configurables en memòria mapejable o cache.

Apart de tot això, el propi chip incorpora un conjunt divers de perifèrics integrats que permeten que el chip pugui ser fet servir per a diverses aplicacions. La majoria d'aquests no

[26] System on Chip. Integra en un sol chip el que abans es soldava en una PCB.
[27] Microprocessador integrat en un system on chip
[28] Arquitectura en la que la cache de dades i d'instruccions estan separades. En aquest cas només el primer nivell de cache està separat, el segon nivell es comú.
[29] Tipus de cache en la que cada entrada de la memòria pot se col·locada en només una posició dins la cache
[30] Tipus de cache en la que cada entrada de la memòria pot ser col·locada en 2 posicions dins la cache

es fan servir dins la placa d'Axis, ja que aquesta disposa d'un altre microprocessador, l'ETRAX100LX, que conté un sistema operatiu Linux fent de host del sistema. Cal remarcar, però, que el DM642 si que controla la entrada i sortida analògica del sistema i, per tant, fa us dels seus 3 ports de vídeo, tal i com hem vist en la figura 2-9 del capítol anterior. A més, el sistema també fa us del port HPI[31] per a la comunicació amb l'ETRAX100LX, cosa que fa gràcies al SDK[32] que Axis ha proporcionat.

En la figura 3-1 podem veure un diagrama de blocs on es poden veure les diferents parts remarcades en aquest apartat:

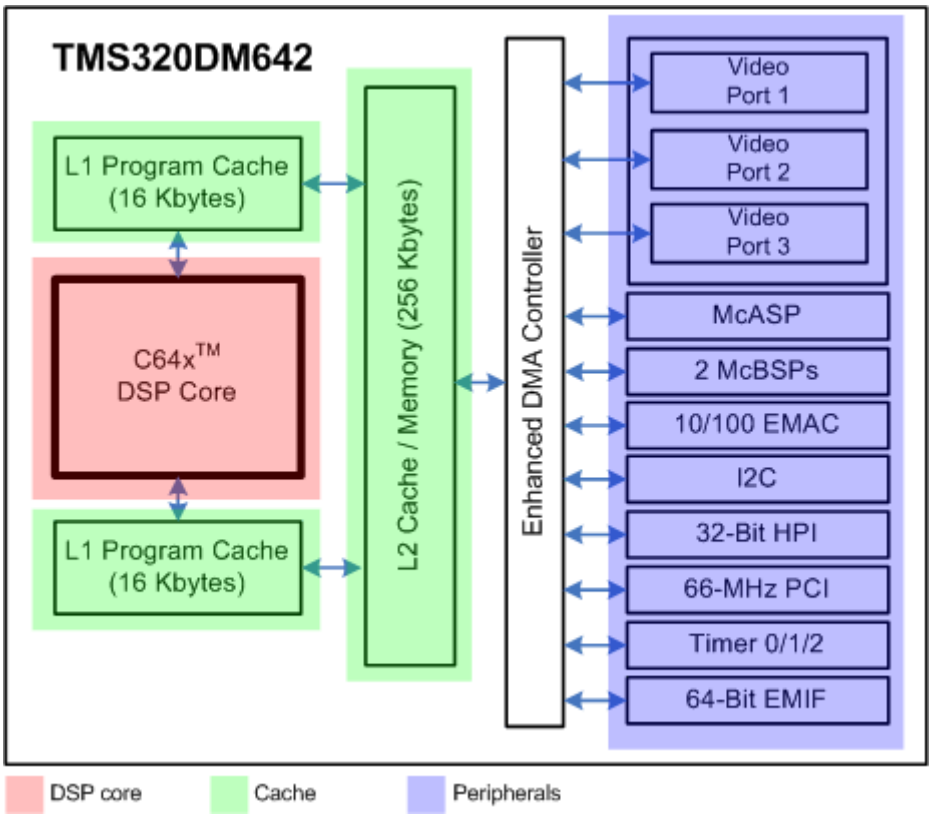


Figura 3- 1 – Diagrama de blocs del TMS320DM642

## 3.3 Arquitectura del nucli C64x™

Per tal d'entendre la idoneïtat del DM642 per a la nostra aplicació, així com algunes de les decisions preses durant el desenvolupament del projecte, cal entrar una mica en detall

[31] Host Port Interface. Port paral·lel present en alguns microprocessadors usat per a que un altre microprocessador pugui accedir a la seva memòria principal.
[32] Software Development Kit

en l'arquitectura amb la que està basada el nucli DSP del nostre sistema. Tal i com hem mencionant anteriorment, aquest nucli està basat en el nucli C64x™ de Texas Instruments. Aquesta, present en molts altres SoCs[1] de la fàbrica, permet que el codi implementat sigui compatible amb el codi present en totes les arquitectures del tipus C6000.

Una de les característiques fonamentals del nucli és del tipus VLIW[33] amb un amplada de paquet d'instruccions de 256bits. Aquest es caracteritza per permetre una gran capacitat de paral·lelisme a costa de deixar l'aprofitament d'aquest paral·lelisme a mans del compilador. En altres paraules, cada paquet d'instruccions especifica quines unitats d'execució del microprocessador han de fer-se servir en el proper cicle. Per tant, això fa que sigui el compilador el que hagi de decidir quines operacions s'han d'executar en paral·lel, cosa que n'augmenta la complexitat i el marge d'acció que es pugui assolir al optimitzar el codi. A més, en aquest cas, el compilador donarà més opcions per optimitzar el procés de compilació que un compilador per a una arquitectura *superescalar*[34].

En segon lloc, l'altre característica important que cal tenir en compte és el fet que es tracta d'un nucli DSP sense unitat de coma flotat. Això, tot i que no limita el fet que es puguin realitzar operacions en coma flotant en el codi del nostre DSP, fa que aquestes requereixin de diversos cicles del processador per a executar-se. Aquest fenomen, ha tingut una rellevància fonamental en el desenvolupament del projecte, ja que el nostre algorisme de *background subtraction* requereix de multiplicacions i divisions en coma flotant per al seu correcte funcionament i, per tant, ha fet falta realitzar algunes optimitzacions per a millorar el rendiment d'aquestes operacions en el DSP.

Finalment, caldria mencionar que el nucli C64x™ disposa de dos bancs de 32 registres de propòsit general, 8 unitats funcionals (.L1, .L2, .S1, .S2, .M1, .M2, .D1 i .D2), dos camins de càrrega des de memòria (LD1 i LD2), dos camins de guardar a memòria (ST1 i ST2), dos camins d'adreces (DA1 i DA2) i dos camins creuats entre els blocs de dades (1X i 2X). En la figura 3-2 podem veure una representació de la nostra arquitectura.

[33] Very Long Instruction Words
[34] Arquitectura en la que és el hardware del microprocessador el que decideix quines operacions s'executen en paral·lel

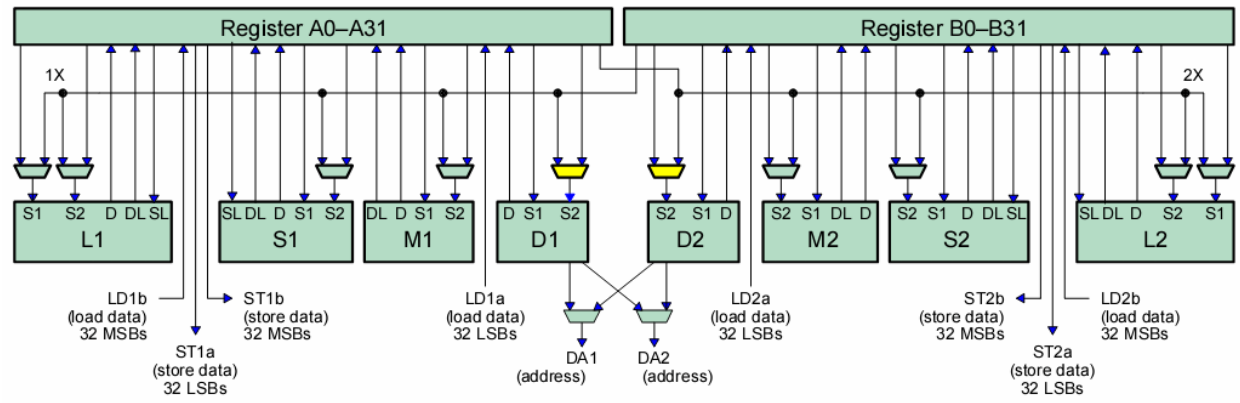


Figura 3- 2 – Arquitectura del nucli C64x™

La unitats .M s'encarreguen d'operacions de multiplicació i multiplicació-acumulació de fins a 16bits; les .L s'encarreguen d'operacions aritmètiques i comparacions de fins a 40 bits; les .S s'encarreguen d'operacions aritmètiques i binaries de fins a 40 bits; finalment les .D s'encarreguen d'operacions necessàries pel càlcul d'adreces. Com es pot veure, el conjunt d'unitats funcionals i la gran ortogonalitat del nucli permeten un grau de paral·lelisme molt elevat.

## 3.4 El port de vídeo

El port de vídeo que incorpora el DM642 pot funcionar tant com a port de captura com d'enviament o com a Transport Stream Interface[35] (TSI). Donat el fet que aquesta última opció no es la que nosaltres farem servir, ens concentrarem en els modes de captura i enviament.

En mode de captura de vídeo, el nostre port ens ofereix dos canals de 8/10 bits a fins a 80 MHZ de velocitat. Aquesta captura pot venir directament d'una càmera digital o be d'una càmera analògica fent servir un descodificador. Cal mencionar que les entrades de dades venen en l'espai de colors YCbCr[36] 4:2:2 seguin la recomanació ITU-R BT.656[37]. Això serà un factor determinant a l'hora d'implementar el nostre algorisme d'extracció de fons ja

[35] Protocol de comunicació definit en el MPEG-2 per a transmetre fluxes de audio i vídeo.
[36] Espai de colors en usat en sistemes de video digital on la lluminància (Y) i la cromància (Cb i Cr) es separen.
[37] Recomanació feta per la ITU (International Telecomunication Union) per al streming degital de vídeo PAL i NTSC en definició estàndard.

que aquest treballa en l'espai de colors RGB i per tant ens caldrà fer una conversió. A més, el port també incorpora un canal Y/C[38] de 16/20 bits, dos canal de vídeo raw, un convertidor 4:2:2 a 4:2:0 i un escalador d'imatges per al mode 4:2:2 de 8 bits.

En el mode d'enviament de vídeo ens permet un sol canal 8/10 bits de fins a 110MHZ, un canal Y/C[13] 16/20, un canal de vídeo raw i un escalador per al mode 4:2:2 de 8 bits. Cal dir que en aquest cas també enviarem les dades en l'espai YCbCr[11] 4:2:2 seguint la recomanació ITU-R BT.656[12].

Cal mencionar a més que el port de vídeo incorpora un buffer de 2560bytes compartit tant per la captura com l'enviament de vídeo. Aquest buffer està connectat directament amb una interfície DMA[39] que ens servirà per mantenir lliure el processador durant els processos de captura. En la figura 3-3 es pot veure un diagrama on s'hi identifiquen els diferents pipelenes de captura i enviament així com el buffer del que s'ha parlat anteriorment. Els fluxos de vídeo es poden veure dibuixats en vermelll

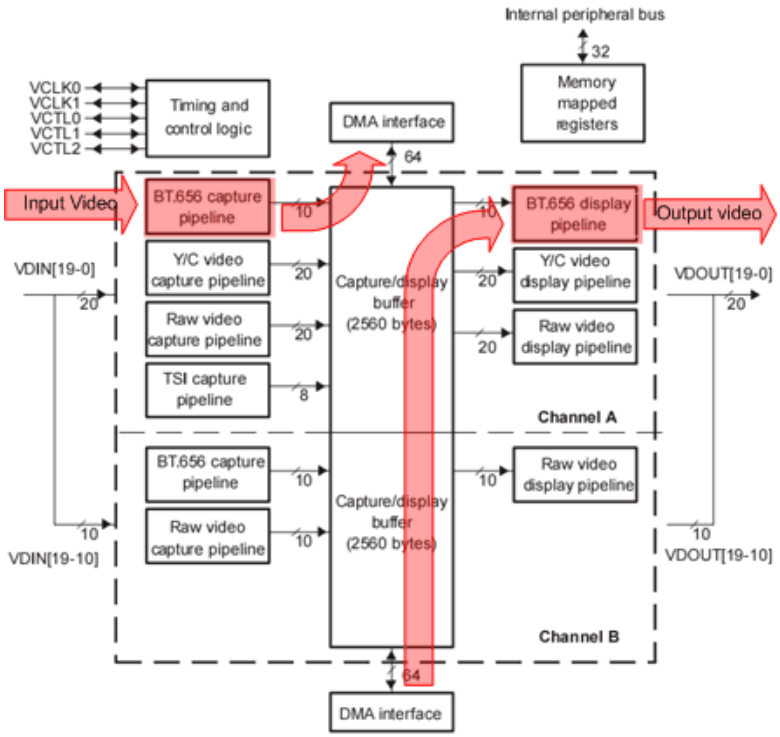


Figura 3- 3 – Diagrama de blocs del port de vídeo

[38] També conegut com S-Vídeo. Senyal analògica de vídeo que transporta la lluminància i la cromància per dos senyals de vídeo separades.
[39] Direct Memory Access.

## 3.5 Stack software

Texas Instruments proporciona una stack[40] software per als DSP de la gama C6000. Aquest està principalment basat en el kernel DSP/BIOS per al desenvolupament d'aplicacions a temps real i multi tasca. Aquest kernel es configura mitjançant una interfície gràfica de configuració que permet especificar en temps de disseny tots aquells components que caldrà incloure en el DSP/BIOS final. Gràcies a aquest procés de configuració estàtic es pot reduir considerablement el footprint de memòria eliminant totes aquelles funcions del kernel que no es facin servir del codi final de l'aplicació. A més, donat el fet que es tracta d'una configuració que es fa abans de la compilació, ens permet crear un munt d'objectes estàtics que d'altra manera hauríem de crear de forma dinàmica en temps d'execució, augmentant conseqüentment el cost computacional de l'aplicació.

Dins el nostre projecte, Axis communications ha inclòs en el seu SDK un Kernel DSP/BIOS i el conjunt de drivers necessaris per a fer funcionar la nostra aplicació. Aquest DSP/BIOS inclou interfícies i APIs per al tractament d'interrupcions hardware i software, threads prioritzats, funcions periòdiques, streams, pipes, clock, semàfors i mailboxes. Totes aquestes APIs simplifíquen de forma dràstica el procés de desenvolupament d'aplicacions per als DSPs de Texas Instruments degut al fet que no és necessari desenvolupar codi d'aquest tipus cada cop.

Apart, Axis també proporciona un sèrie d'APIs específicament dissenyades per a integrar-se al màxim amb les seva plataforma. Cal destacar que aquesta API està a un nivell per sobre de la DSP/BIOS i els drivers del sistema, fent-ne us i afegint una capa d'abstracció més sobre aquestes. De forma gràfica la figura 3-4 mostra l'esquema de capes que s'executa en el nostre DSP. Els mòduls marcats em vermell indiquen aquelles capes que venen distribuïts dins el SDK de Axis i que, tot i que en molts casos no estan completament desenvolupats per ells, si que els han adaptat o optimitzat per a integrar-los en el servidor de vídeo.

[40] Pila. Stack software s'usa per a descriure les diferents capes de software que aniran en un microprocesador.

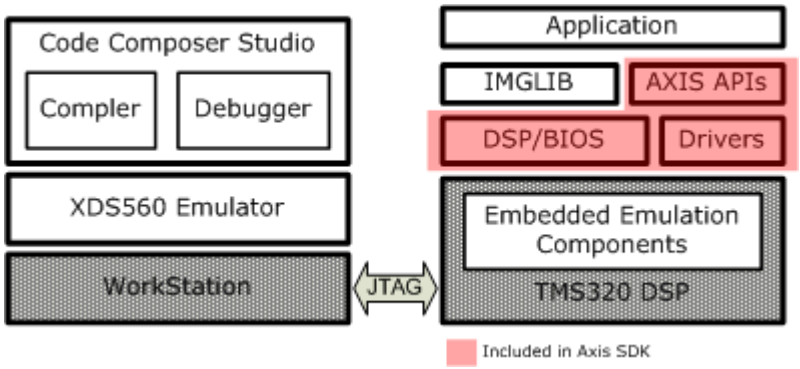


Figura 3- 4 – Esquema de capes software del TMS320DM642

En aquest diagrama també podem apreciar una capa de software anomenada IMGLIB. Aquesta és una llibreria proporcionada per Texas Instruments específicament dissenyada per al tractament d'imatges. Dins d'aquesta hi podem trobar convolucions, correlacions i d'altres funcions usades comunament en la visió per computador totalment optimitzades per al nostre DSP.

## 3.6 APIs d'Axis Communications

Tal i com em mencionat en l'apartat anterior, dins el SDK d'Axis Communications s'ha inclòs una capa més d'abstracció per a efectuar operacions integrades en la seva plataforma. Aquestes APIs estan estretament lligades a la resta de software present al servidor de vídeo així com a l'arquitectura de tots els vídeo servers i càmeres ip d'Axis.

L'eix fonamental sobre el que es mouen totes les APIs d'Axis és segurament el port HPI. Aquest, tal i com hem mencionat anteriorment, és un port de comunicació paral·lel que permet que l'ETRAX100LX tingui accés a tot l'espai d'adreçament del DSP amb excepció dels registres de configuració de la cache L2, els registres de selecció d'interrupcions i els registres de lògica d'emulació. Tota la comunicació es fa sempre a través de l'EDMA[41] de Texas Instruments, cosa que fa que sigui sempre asíncron al processador.

[41] Enhanced Direct Memory Access. DMA desenvolupat per TI inclòs en els cores C64™

Es pot dir doncs que la primera API que Axis inclou dins el seu SDK és precisament de control del HPI. Aquesta permet portar a terme la comunicació amb el ETRAX a través del HPI, mantenir una llista de funcions call-back[42] segons un registre d'aquestes efectuat anteriorment i controlar les dimensions i funcionament buffers de lectura i escriptura usats en la comunicació a través de l'HPI.

Per sobre d'aquesta capa, Axis ha afegit una sèrie de APIs que fan us d'aquesta primera capa i afegeixen una vegada més un nivell d'abstracció al sistema. Seguidament detallaré quines són i quin us se'n fa dins la nostra aplicació:

- *Debug API:* Aqueta api permet enviar traces al ETRAX on són enregistrades per un dimoni dins del syslog[43] del sistema. Gràcies a aquesta api hem pogut depurar el nostre codi en el DSP enviant traces que podíem veure fent un Telenet al ETRAX

- *SVG[44] API:* Aquesta api permet enviar imatges associades al frame actual en format SVG[19]. Un cop enviades a través d'HPI, aquestes imatges s'envien conjuntament amb el MJPEG[45] a través d'HTTP on qualsevol aplicació externa pot rebre les dades i fer-ne us.

- *Param API:* Proporciona una interfície per a escriure i llegir paràmetres del sistema. El vídeo server manté una base de dades de paràmetres de configuració del sistema per a controlar totes les funcionalitats del vídeo server. Aquesta interfície permet que n'afegeixis de nous o be que es notifiqui el DSP quan un paràmetre es canvia.

[42] Funcions que són cridades quan succeeix algun event.
[43] Protocol per enviar logs sistemes tipus UNIX a través d'una xarxa o canal de comunicació.
[44] Scalar Vector Graphics. Estàndard d'imatges vectorials basat en XML.
[45] Motion JPG. Estàndard de vídeo basat en JPG

- *Event API:* Mitjançant aquesta API es pot notificar al ETRAX quan ocorre algun esdeveniment en el sistema. D'aquesta manera es pot, per exemple, canviar l'estat de les sortides digitals per a activar algun mecanisme extern al vídeo server (ie. un relé)

- *File API:* Aquesta api permet llegir i escriure fitxers del sistema de fitxers de l'ETRAX. Per a realitzar aquesta esc

- *LED API:* Permet controlar l'estat dels leds del panell frontal del vídeo server. En la nostra aplicació el fem servir per indicar quan el sistema es troba en funcionament.

- *License API:* Permet que els mòduls IV implementin un sistema de protecció de còpia associat a una llicència. De moment, no hem fet servir aquest mòdul per desenvolupar un sistema de llicències, però està previst que es faci servir en un futur quan l'aplicació estigui més evolucionada.

- *Vídeo API:* Aquesta API s'empra per a la captura i l'enviament de vídeo a través de les entrades i sortides analògiques. Fa us del driver proporcionat per Texas Instruments per a la interfície amb el port de vídeo del DSP, afegint-hi sobre aquest una capa d'abstracció més per a simplificar el seu funcionament.

De forma gràfica, doncs, podem veure en la figura 3-5 com interaccions les diferents APIs proporcionades per Axis. El quadre marcat amb vermell indica l'aplicació desenvolupada en aquest projecte. També es pot apreciar com l'API de vídeo no fa us de HPI per a la seva comunicació sinó que interacciona directament amb el driver de vídeo.

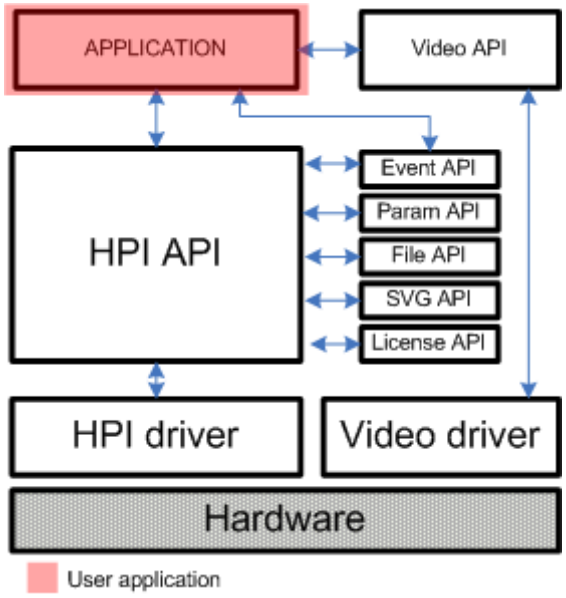


Figura 3- 5 – Diagrama de blocs del SDK d'Axis

De forma gràfica, doncs, podem veure en la figura 3-5 com interaccions les diferents APIs proporcionades.

# 4. Implementació de l'algorisme de Background Subtraction

## 4.1 Consideracions d'us

Per a la implementació de l'algorisme de *background Subtraction* cal tenir en compte una sèrie de consideracions prèvies que afectaran a la complexitat de l'algorisme. Totes aquestes restriccions ens venen donades tant per a l'aplicació de les diverses capes posteriors a l'extracció de fons com per les característiques de les imatges sobre les quals ha de funcionar aquest algorisme.

En primer lloc, cal remarcar que les imatges preses per càmeres de vigilància són imatges que tenen un grau de dificultat molt elevat. Aquestes imatges estan grabades normalment per càmeres de seguretat analògiques de baixa qualitat, amb un grau elevat de soroll i no sempre instal·lades en localitzacions òptimes per a efectuar una bona extracció del fons. Tot això efecte en gran mesura les característiques de la seqüència de vídeo que cal processar i a nivell general podem dir que el nostre algorisme haurà de suportar les següents condicions:

- Filmacions de dia o de nit (tant en interior com en exterior).
- Canvis de llum sobtats donats, per exemple, per una llum o fanal que es troba al mig de la imatge.
- Condicions climatològiques adverses: pluja, vent, ...
- Fons de la imatge amb un cert nivell de moviment donat, per exemple, per un arbre o arbust que es mou.
- Ombres o focus de llum deguts als fars dels vehicles o a la pròpia il·luminació ambiental.

En la figura 4-1 podem veure uns quants exemples de seqüències de vídeo reals que han donat problemes de detecció. La imatge 1 mostra una escena en la que les persones

només ocupen un pocs pixels; en la 2 la llum d'una fanal del carrer provoca una distorsió en la imatge; en la imatge 3 el vent fa moure els matolls del fons fent que el background es comporti de manera totalment dinàmica; finalment, en la 4 es pot veure una aranya al mig de l'escena. Com a curiositat, les aranyes són un gran problema per a la vídeo vigilància intel·ligent ja que tendeixen a construir les seves teranyines en les cobertes de protecció de les càmeres.


**Figura 4- 1– Imatges problemàtiques de càmeres de seguretat**

Apart de tots els factors donats per les característiques de les imatges, hi ha una altra restricció que ens vindrà donada per les característiques de les capes de *tracking*[46] i *classificació*[47]. La implementació de l'algorisme que funciona per a PC s'ha aconseguit optimitzar perquè funcioni a una velocitat de 17 fps en un Intel Pentium D a 3 GHz i 2GBytes de RAM. Aquesta velocitat de processament disminueix notablement quan s'hi

[46] Capa en la que s'intenta trobar una relació entre els objectes detectats en frames diferents per a seguir la trajectòria d'un objecte. Veure capítol 1.
[47] Capa en al que s'intenta classificar els objectes que hi a l'escena per a determinar si es tracta d'una persona, un vehicle o un altre objecte. Veure capítol 1.

afegeixen les capes posteriors, cosa que fa que s'hagi agut de limitar l'algorisme, per a que funcioni a 6 fps. Per tant, per un funcionament òptim del nostre algorisme caldria avaluar fins a quin punt seria possible mantenir el mínim de 6 fps o fins i tot millorar-lo per a aconseguir incloure més capes de software en el servidor de vídeo.

## 4.2 Fonaments matemàtics del Unimodal i el Multimodal

Per entendre una mica les bases dels algorismes de *Background Subtraction* ens hem basat en els algorismes proposats en [1] i [2]. En aquests dos casos es modelen les variacions en els píxels d'una imatge amb una o varies gaussianes definides per la seva mitjana i la seva variància. En aquest apartat farem un recorregut per aquests dos models per aclarir una mica millor quin és l'estat de l'art en aquest tipus d'algorismes i entendre una mica quins problemes tenen aquests models.

En primer lloc, suposem que tenim una seqüència de vídeo en la qual volem separar el fons de la imatge, dels diferents objectes que s'hi mouen. Per simplificar, en aquest primer exemple farem servir la seqüència controlada de la figura 4-2 en la que una persona creua la pantalla d'esquerra a dreta per davant d'un fons totalment uniforme.

**Figura 4- 2 - Seqüència original de vídeo**

Si ens fixem en el píxel central de la imatge i en fem un estudi al llarg del temps, podrem observar que la majoria del temps aquest píxel està mostrant el color beix de la paret. En el moment en el que la persona creua, aquest píxel canvia i mostra el color vermell de la camisa. Finalment, quan la persona marxa de la imatge, aquest píxel torna a mostrar el

color beix de la paret. La Figura 4-3 mostra una gràfica en la que es pot veure aquesta evolució i apreciar com, en el frame 40 aproximadament, la persona creua la imatge.

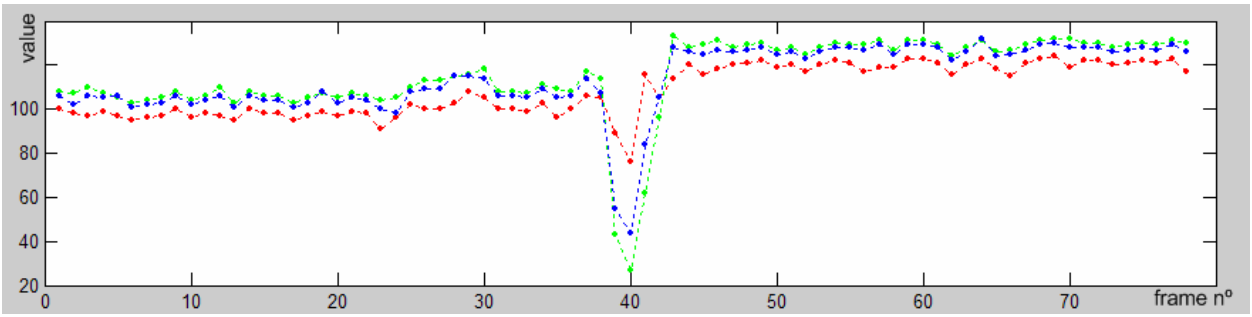


Figura 4- 3 - Gràfica de valors del píxel central en el temps

Si ara fem una acumulació del nombre de cops que un valor concret ha aparegut en el mateix píxel i ho representem en una gràfica, podem veure la distribució que han tingut els valors d'aquest píxel. La figura 4-4 mostra aquesta distribució en forma de freqüència d'aparició d'un valor per als tres components del RGB.

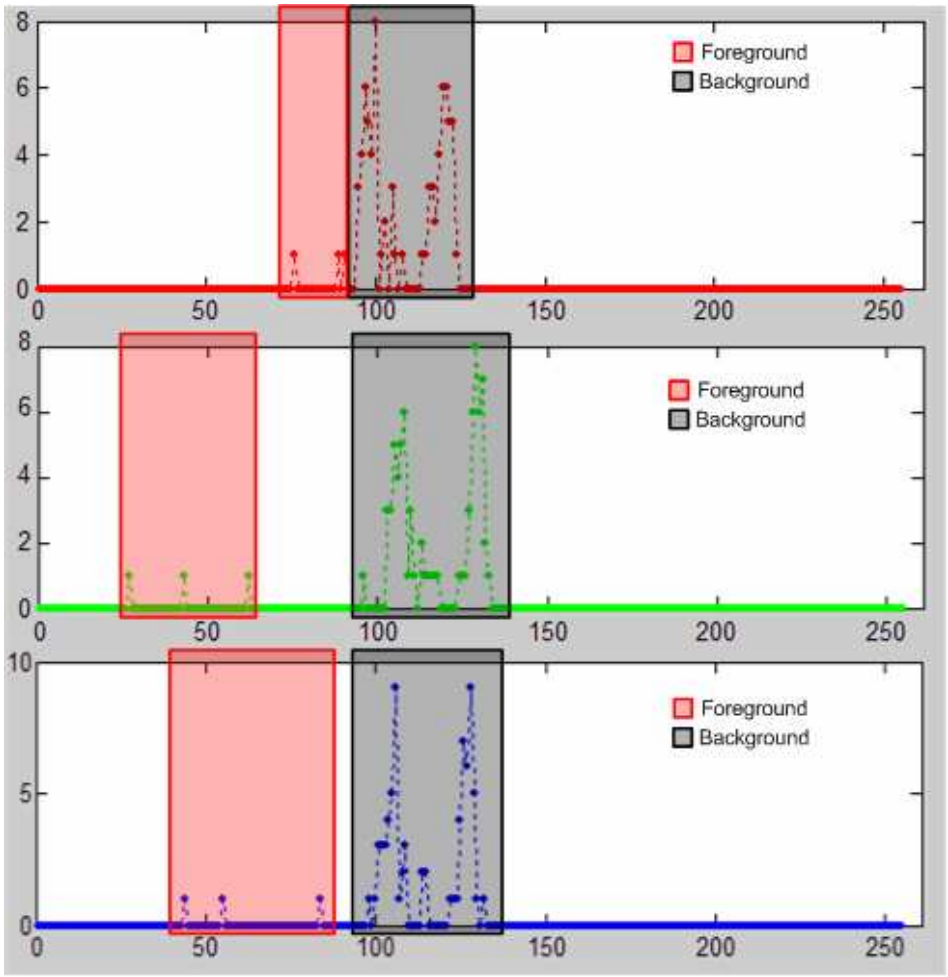


Figura 4- 4 - Freqüència d'aparició de valors en el píxel central

Com es pot veure, les regions marcades en gris són valors del píxel que pertanyen al fons de la imatge. Per altra banda, els valors marcats amb vermell mostren aquells valors que pertanyen a la persona.

El model de *background unimodal* assumeix que la distribució dels colors que cauen en un sol píxel de la imatge es pot representar amb una gaussiana. Aquesta gaussiana ve definida completament per la mitjana dels valors del píxel i la seva desviació típica. Hem procedit a calcular la mitjana i la desviació típica de tots els valors observats en el píxel central de la imatge fent servir les següents fórmules:

$$\bar{x} = \frac{\sum_{i=1}^{n} a_i}{n} = \frac{a_1 + \cdots + a_n}{n}$$

$$s^2 = \frac{\sum_{i=1}^{n} (x_i - \overline{x})^2}{n}$$

Seguidament hem representat una gaussiana definida per els valors prèviament calculats i els l'hem sobreposat a les dades de freqüència que hem pogut veure a la figura 4-4. El resultat el podem veure en la figura 4-5. La gaussiana s'ha situat en els tres casos sobre els valors que representen el fons de la imatge, deixant fora tots aquells valors que pertanyen a la persona.

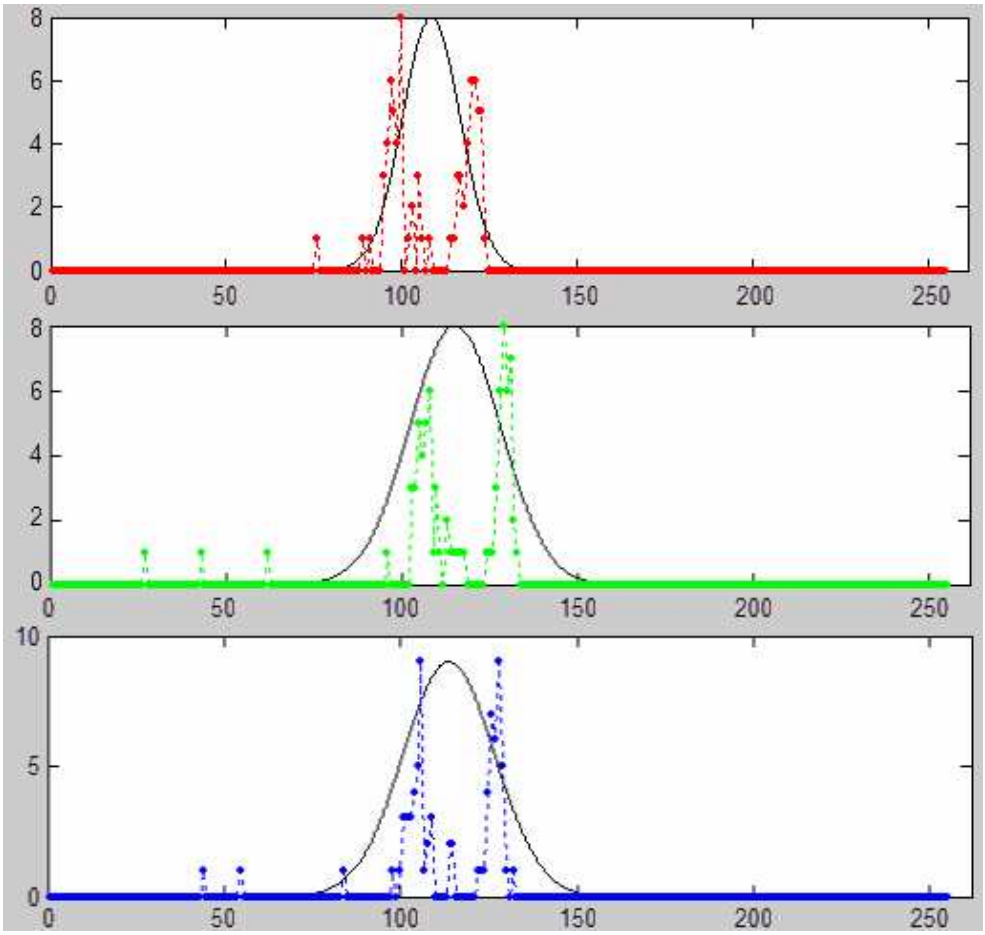


**Figura 4- 5 - Gràfiques de la distribució normal centrades en la mitjana de freqüències.**

Vistos els resultats de la figura 4-5, es podria modelar el fons de la nostra imatge com tots aquells valors que caiguessin al centre de la distribució normal (ie. la campana de gauss) i classificar tots aquells valors que caiguessin fora d'aquesta com a *foreground*.

Tot i que en aquest cas sembla ser que una sola campana gaussiana seria suficient per a modelar el nostre sistema, la pràctica ens ha demostrat que no n'hi ha prou. De fet, amb una simple observació de les gràfiques de la figura 4-5 i 4-5 es pot veure com el nostre fons oscil·la entre dos rangs de colors. Això es degut a l'efecte del soroll i variacions en la il·luminació que afecten a la nostra imatge. És per aquest motiu que algorismes com els descrits en [2] basen els seus models matemàtics en múltiples gaussianes.

Per a representar aquest fenomen, hem agrupat els nostres punts usant l'algorisme *K-means* amb valor K = 2. Un cop agrupats, hem calculat la mitjana i la desviació típica per a cada clúster. Això ens ha definit, per a cada component, dues gaussianes centrades en els valors de màxima freqüència que ens modelitzen molt millor que abans el fons de a imatge. En la figura 4-6 podem veure el resultat de totes aquestes operacions. Tal i com es pot apreciar, cadascuna de les dues gaussianes ens modelitza un dels valors sobre els que ha anat oscil·lant el píxel. A més, en aquest cas les dues gaussianes exclouen molt millor els píxels del *foreground* que en cas d'una sola gaussiana.

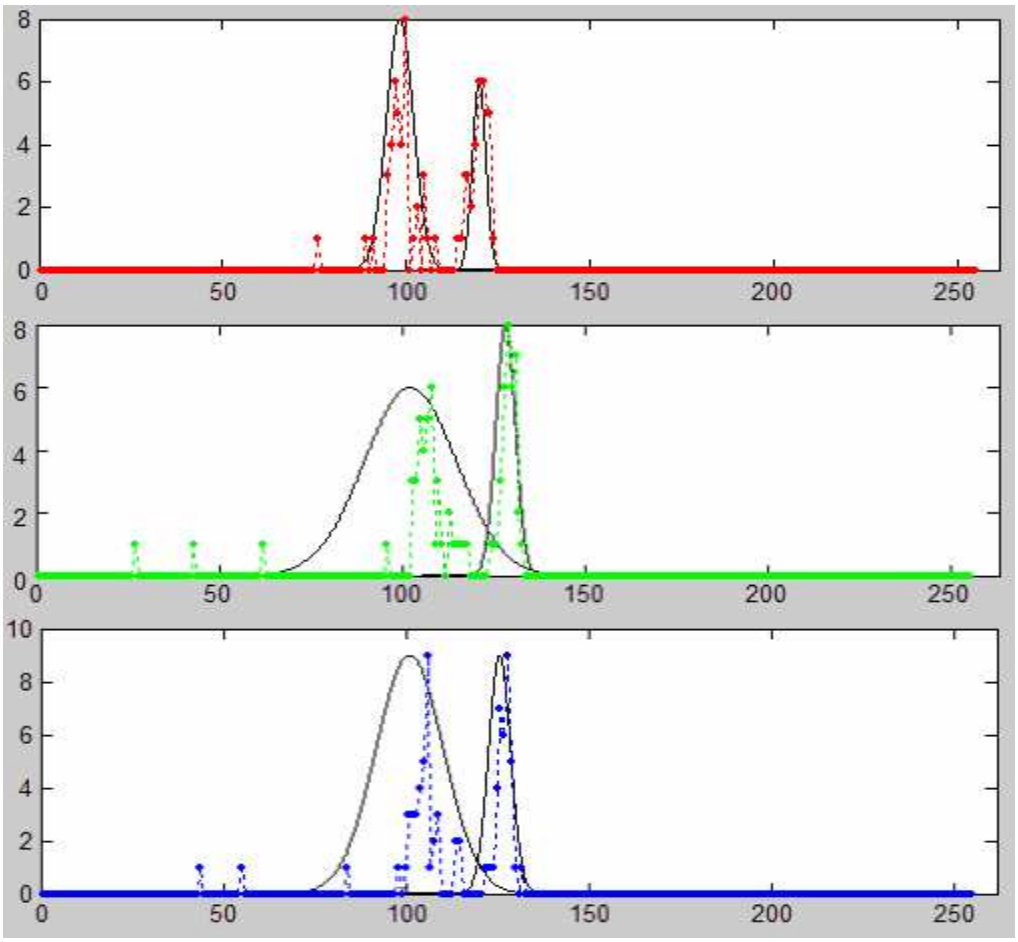


Figura 4- 6 - Gràfiques de dues gaussianes per aproximar les freqüències.

## 4.3 Implementació del Unimodal

Per a la implementació de l'algorisme de *background Subtraction* varem desenvolupar en primer lloc una aproximació senzilla fent servir el model explicat a [1].

Aquesta, tal i com hem mencionat en l'apartat anterior, es basa en modelar la distribució dels valors que pren un píxel amb una sola gaussiana. Tot i la simplicitat del model i la poca fiabilitat que aquest dona, hem cregut oportú incloure aquí un detall del mateix per tenir una idea clara de com es fa servir.

En primer lloc, partirem de la seqüència de vídeo que hem vist a la figura 4-2. La gaussiana que modela un píxel ve totalment definida per la mitjana dels valors que aquest ha anat prenent al llarg del temps i la desviació típica d'aquests. Ara be, per calcular la mitjana dels valors cal que tinguem la seqüència d'aquests en la seva totalitat. Per tant, aproximarem aquesta mitjana amb una ponderació basada en un paràmetre $\alpha \in [0, 1]$. La fórmula del *running average (1)* ens permet calcular la mitjana dels valors d'un píxel tenint en compte la mitjana anterior $\mu_{t-1}$ i el valor actual del píxel $x_t$.

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

Així doncs, si apliquem aquesta fórmula a tots els píxels de la nostra seqüència, el que aconseguim és una altra seqüència en la que cada frame mostra la mitjana ponderada dels valors previs a aquest. La figura 4-7 la seqüència resultant per a diferents valors de $\alpha$.

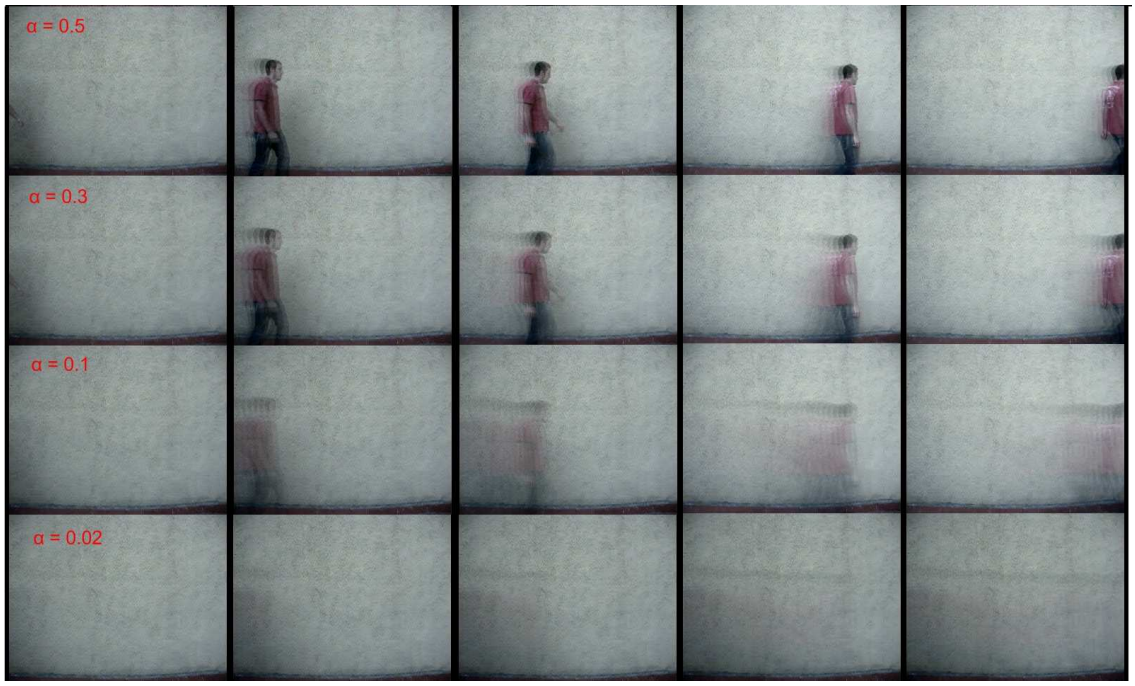


Figura 4- 7 - Mitjan a ponderada de la seqüència original

Com es pot apreciar, com més petit sigui el valor de α menys pes se li dona al frame actual i per tant més lentament aprèn el nostre algorisme. Normalment treballarem sempre amb valors de α molt petits ja que d'aquesta manera es modelitza molt millor el fons de la imatge (veure α = 0.02).

Tot seguit, ens caldrà calcular la desviació de la nostre imatge. Tal i com ens passava amb la mitjana, ens caldrà una manera de calcular-la sense tenir la seqüència de valors completa. Farem servir la mateixa estratègia que abans i donarem un pes a la desviació del valor actual fent servir el paràmetre $\beta \in [0, 1]$. La fórmula (2) ens mostra el càlcul de la variància $\sigma_t^2$ (desviació típica al quadrat) fent servir la mitjana prèviament calculada $\mu_t$.

$$(2)\ \sigma_t^2 = \beta\,(\mu_t - x_t)^2 + (1 - \beta)\sigma_{t-1}^2$$

El resultat d'aplicar aquesta operació en tots les píxels el podem veure en la figura 4-8. Hem fixat el paràmetre α = 0.02 i hem provat valors diferents de β per veure els resultats.

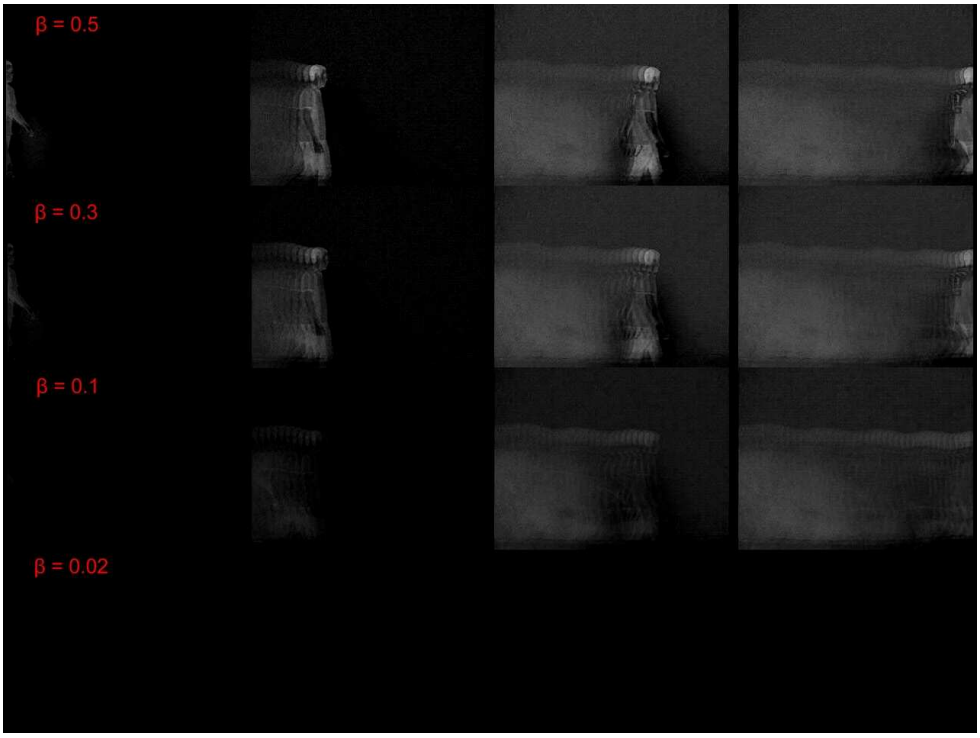


Figura 4- 8 - Desviació típica de la seqüència original

Al efectuar aquestes proves ens adonem clarament que hi ha un augment general de la desviació típica als últims frames de la nostra seqüència. Això probablement és degut al balanceig de blancs automàtic que moltes càmeres fan. En la pràctica ens interessarà detectar augments generals de la variància deguts a aquests fets o a canvis d'il·luminació a l'escena per a poder filtrar aquests millor. Segons les nostres proves, valors de $\beta = 0.1$ seran suficients donats una $\alpha = 0.02$. En altres paraules, en interessarà fer que aprengui més ràpidament la variància que té la nostra imatge que no pas la mitjana.

Si ara efectuem una diferència del frame actual amb la mitjana que hem obtingut en la figura 4-7 obtindrem la desviació d'aquest amb la mitjana. Ara podem fer servir la desviació típica calculada a la figura 4-8 com a llindar per a determinar quins píxels considerarem *background* i quins considerarem *foreground*. Així doncs, de manera simple, el nostre algorisme quedarà de la següent forma:

```
/*--Parametres de l'algorisme --*/
• = 0.02
• = 0.01
Threshold = 1.02
/*--Inicialització de l'algorisme --*/
Mitjana = LlegirPrimerFrame(video)
Desviacio = 0
/*--Bucle Principal --*/
Mentre(Frame = LlegirFrame(video))
     Mitjana = Frame * • + Mitjana * (1 – •)
     Distancia = Valor_Absolut(Mitjana - Frame)
     Desviacio = Distancia * • + Desviacio * (1 – •)
     Per cada pixel de Distancia
          Si Distancia[i] > (Desviacio[i] * Threshold)
               Frame[i] = FOREGROUND
          SiNO
               FRAME[i] = BACKGROUND
          Fi Si
     Fi Per
Fi Mentre
/*-- FI Bucle Principal --*/
```

Com es pot veure, en aquest últim pas hem afegit un paràmetre més al nostre algorisme, el threshold ϵ [1, 2]. Aquest ens indicarà quants cops més gran ha de ser la distància del frame actual a la desviació típica prèviament calculada. Així si, per exemple, el threshold = 1.02, la nostra distància ha de ser 1.02 cops més gran que no pas la desviació típica.

En la figura 4-9 es poden veure els resultats de l'algorisme agafant com a paràmetres $\alpha = 0.02$, $\beta = 0.1$ i Threshold = 1.02. La primera fila mostra la seqüència original; la segona, mostra la mitjana ponderada; la tercera, mostra la distància entre el frame actual i la mitjana ponderada; la quarta, mostra la desviació típica ; finalment, l'ultima fila ens mostra el resultat de l'algorisme en la que els píxels de color blanc són *foreground* i els de color negre són *background*.

**Figura 4- 9 - Resultats de l'algorisme**

## 4.4 Estat de l'art i problemes trobats en l'Unimodal

Un com hem vist una aproximació a l'algorisme de *Background Subtraction* Unimodal descrit en [1], passarem a fer uns quants comentaris sobre el mateix per veure una mica quins problemes té i quines solucions s'han proposat. A més, això ens servirà per entrar una mica per sobre en quin és l'estat de l'art dels algorismes de *BGS*[3]. Cal dir que tot i que l'algorisme que finalment hem migrat al nostre DSP no és aquest, si que l'hem fet servir en primera instància per a testejar la plataforma i tenir una base sobre la que anar treballant. Alguns dels problemes trobats en aquesta algorisme els hem extret dels mètodes per avaluar algorismes de *BGS*[48] a [3] i d'algunes consideracions prèvies extretes de [4].

El principal problema de modelar la variació dels valors d'un píxel amb una sola gaussiana és la motivació principal per a crear models multimodals. Tot i que a primera vista el fons d'una seqüència de vídeo pugui semblar estàtic, en molts casos aquest es comporta de manera dinàmica. Aquest dinamisme ve donat per canvis en la il·luminació, moviments de la vegetació deguts al vent, ondulacions de l'aigua, parpellejos de monitors, etc. Tot això fa que el fons de la nostra imatge sigui més un conjunt de moviments repetitius que no pas una imatge totalment estàtica. Un exemple d'això el podem veure en la figura 4-10. En aquesta imatge, l'aigua reflexa la llum solar provocant que el nostre model de *background* vagi canviant constantment.

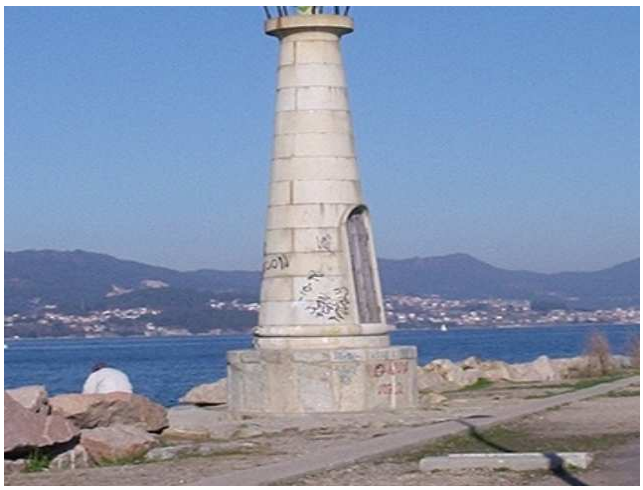

**Figura 4- 10 - Exemple de seqüència amb fons mòbil.**

[48] Background Subtraction.

És per aquest motiu que [2] proposa modelar el nostre sistema amb una combinació de gaussianes, tal i com hem vist en el primer apartat d'aquest mateix capítol. Aquest algorisme es basa en agrupar els valors que vagin prenent els diferents píxels en K clústers i modelar cadascun d'aquests amb una gaussiana. Per agrupar es fa servir un K-means[49] dinàmic que classifica els diferents valors de color en un dels clústers segons la seva proximitat al centroide[50] (ie. mitjana) prèviament calculada. Posteriorment, es modelitza el fons amb diverses gaussianes en la que cadascuna ve definida per la mitjana i la desviació típica dels punts del clúster en qüestió.

El segon problema que cal tenir en compte és el fet de la importància cromàtica que tenen els tres components del RGB. En el cas dels algorismes proposats a [1] i [2], es dona igual importància als tres components RGB. Aquest fet fa que l'algorisme sigui molt més sensible a canvis d'il·luminació i provoqui falsos positius en regions on hi ha un canvi sobtat en la il·luminació. Alguns algorismes de *BGS* intenten separar la luminància de la cromància per tal filtrar millor els canvis de llum, donant més pes als canvis cromàtics en la imatge que no pas els canvis d'il·luminació.

El tercer problema que hem detectat afecta a tots els algorismes basats en píxels. Aquest es basa en el fet que no es te en compte en cap moment la relació que hi ha entre els píxels d'una mateixa regió. A tall d'exemple podríem dir que si un píxel es classificat com a *background* i tots els píxels del seu voltant són classificats com a *foreground* és molt provable que aquest píxel formi part del mateix objecte que els del seu voltant i per tant, hauria de ser classificat com a *foreground* igualment. L'algorisme proposat a [4] intenta analitzar les diferents textures que presenten els píxels dins la imatge per a correlacionar els píxels entre si i fer una millor detecció. D'altra banda, [3] proposa incorporar al model de la gaussiana informació dels contorns per d'aquesta manera aconseguir millorar el percentatge de falsos negatius.

Un exemple d'aquests dos últims fenòmens el podem veure en la figura 4-11. Aquesta està extreta de l'algorisme descrit a l'apartat 4.3. Com es pot veure, el braç persona és

[49] En un conjunt de punts, el centroide és aquell punt de l'espai que minimitza les distàncies a tots els altres punts.
[50] El K-means és un algorisme freqüentment usat per a agrupar punts segons les seves característiques.

classificat com a *background* degut a la poca diferència cromàtica entre el braç i el color de la paret degut a la poca il·luminació de l'escena i al fet que no es tingui en compte la relació de proximitat que hi ha entre els píxels del braç els de la resta del cos.


**Figura 4- 11 - Exemple d'errors de classificació.**

En quart lloc, hem detectat un problema que ens sorgeix amb l'unimodal i el multimodal i que afecta al filtratge de les ombres i els focus d'il·luminació. Els algorismes proposats en [1] i [2] no discriminen entre *shadows*[51] i *highlights*[52]. Això fa que en posteriors etapes de l'aplicació sigui difícil discriminar aquells píxels que pertanyen a les ombres dels objectes d'aquells que pertanyen a l'objecte en si. Cal dir que en algunes aplicacions aquest problema s'intenta solucionar en posteriors etapes sense tenir en compte cap tipus d'informació donada per el model de *BGS*.

Un cinquè problema que hem pogut veure en [1] i [2] sorgeix quan un objecte en moviment es situa davant d'un altre objecte que també està en moviment o que encara no ha estat absorbit per el model de *background*. Aquest fenomen provoca que el primer objecte sigui invisible a ulls de l'algorisme. És per aquest motiu que el model proposat per [6] intenta efectuar una *BGS* basat en capes per tal de mantenir la informació d'aquells objectes que estan en moviment dins de l'escena.

[51] Ombres. Terme usat en els algorsimes BGS per a indicar que un objecte classificat com a foreground és més fosc que el backgroud
[52] Focus de llum. Terma usat en els algorismes de BGS per a indicar objectes classificats com a foreground però que són més lluminosos que el background.

Finalment cal mencionar la falta de control que es te sobre quant i com són absorbits els objectes que es queden parats al mig d'una escena. Aquest fenomen passa molt sovint en els vídeos que nosaltres haurem de tractar. Un exemple seria un cotxe que entre a l'escena i aparca. Aquest cotxe formaria part del *foreground* mentre estigués en moviment, però en el moment en el que el cotxe s'atura aniria essent absorbit progressivament en el *background* de l'escena. Això ens provoca un "trade-off", ja que, si el *background* absorbeix els objectes massa lentament, evita que es detectin els objectes que passen per sobre d'aquest, però si absorbeix massa ràpid pot ser que afecti a aquells objectes que es mouen molt a poc a poc. Apart d'això, la manera com aquests objectes són absorbits també és important. Per a posteriors capes de l'aplicació serà més desitjable que un objecte sigui absorbit de cop, que no que es vagi absorbint lentament o de forma difuminada.

## 4.5 Característiques de l'algorisme finalment aplicat

Tot i que per qüestions de petició de confidencialitat de DAVANTIS Technologies S.L. no se'm permet revelar les fons en les quals està basat l'algorisme que s'ha migrat a la plataforma d'Axis, si que en donaré les característiques externes i faré un repàs per els problemes que aquest algorisme ens soluciona. Esperem que el lector pugui entendre que en molts casos no s'entri en més detall en certes qüestions i només se'n esmenti la capa externa.

En primer lloc, cal dir que el model de colors que el nostre algorisme empre, dona molta més importància a la variància cromàtica que no pas a la lumínica. Aquest fet fa que l'algorisme respongui molt millor en situacions on hi ha poca lluminositat. Un exemple d'aquest cas ens el trobem quan una persona o un cotxe es mou per la ombra provocada per un edifici.

Seguidament, cal mencionar que l'algorisme te en compte la relació que hi ha entre un píxel i els píxels del seu entorn. Això en permet reduir al màxim el rati de falsos negatius, tal i com s'ha explicat en l'apartat anterior. A més, l'algorisme separa els píxels classificats com a *foreground* en píxels marcats com a *highlights* i píxels marcats com a *shadows*. Això ens permet filtrar en posteriors capes de la nostra aplicació les ombres provocades per els

objectes que es mouen en l'escena o els focus de llum provocats per els fars dels vehicles o les llums del mobiliari urbà, reduint de forma dràstica els falsos positius en la detecció.

Una altra característica important del nostre algorisme és la gran capacitat que te aquests per adaptar-se a canvis sobtats en la il·luminació. Aquest fet serà de gran importància quan intentem analitzar imatges gravades a l'exterior. Sovint les càmeres estan col·locades davant d'una llum que al encendre's o al apagar-se provoca un increment molt gran en la luminància de tota l'escena.

A més de tot això, l'algorisme ens permet controlar de forma molt nítida el temps i la forma en la que seran absorbits els objectes per el *background*. Aquest fet ens serà de gran utilitat en preses de càmera en la que hi intervinguin vehicles motoritats. En molts casos aquests s'aturaran al mig de l'escena durant molta estona i caldrà controlar com i quan deixen d'estar detectats per el sistema.

## 4.6 Restriccions de la plataforma d'Axis

Tot i que en els capítol 2, 3 i 4 ja comentem quines són les característiques que ens ofereix la plataforma d'Axis, en aquest apartat farem un recull de les restriccions que ens ha provocat per a la implementació del nostre algorisme, i esmentarem, en els casos que sigui possible donada la confidencialitat del mateix, les adaptacions que hem hagut de fer en l'algorisme.

La limitació principal a la que ens hem enfrontat a l'hora de realitzar la implementació ha estat el fet que el DSP de Texas Instruments no conta amb una unitat de coma flotant. Això, sumat al fet que el nostre algorisme es força depenent de les multiplicacions en coma flotant ens ha suposat en molts casos un decrement considerable en la velocitat de processament d'aquest tipus d'operacions. Tot i així, com es podrà veure en l'apartat d'optimitzacions, hem fet us de LUTs i algunes llibreries proporcionades per Texas Instruments per tal d'accelerar-les al màxim.

En segon lloc, tal i com hem pogut veure en l'apartat 3.4, el port de vídeo que incorpora el DSP segueix la recomanació ITU-R BT.656[53]. Aquesta recomanació engloba tant l'espai de colors YCbCr usat en l'enviament de les dades entre la càmera i el DSP, com l'esquema de codificat 4:2:2 que aquestes segueixen. Donat el fet que aquesta també ha suposat una limitació important i que el nostre algorisme funciona fent servir l'espai RGB, entrarem una mica més en detall a veure que és el que això realment implica.

L'espai de colors YCbCr s'usa freqüentment en sistemes de vídeo digital. Aquest és tot sovint confós amb l'espai YUV però, tot i que també separa la luminància de la cromància, fa servir unes altres fórmules per a la conversió. La raó de ser d'aquest espai de colors ve donada per les característiques cognitives de l'home. L'ull humà és molt més sensible als canvis en la luminància que no als canvis a la cromància. Aquest fenomen s'ha aprofitat en els sistemes digitals pera dedicar menys bits a la cromància que a la luminància, fent necessari un espai de colors en el que aquests dos component estiguin separats per tal de poder fer aquesta reducció de precisió.

A tall d'exemple, en la figura 4-12 s'ha inclòs una imatge amb els tres components Y (luminància), Cb i Cr (cromància) separats. Com es pot apreciar, la luminància no és més que la imatge en blanc i negre.

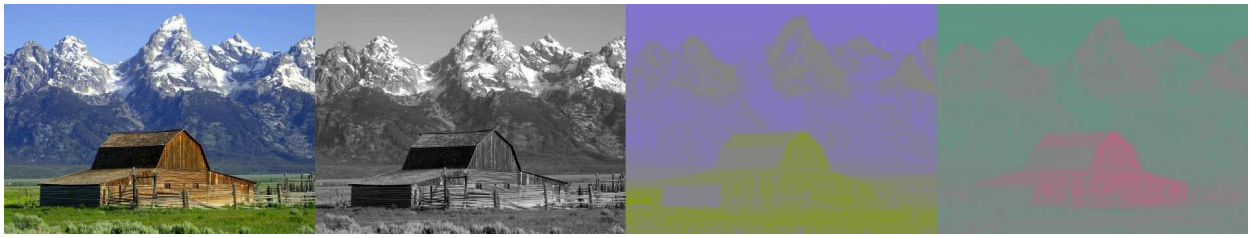

**Figura 4- 12 - Exemple de l'espai de colors YCbCr**

La recomanació ITU-R BT.656 també indica que s'ha de realitzar un sub-samplejat 4:2:2 de la ceromància per tal de reduir la quantitat de dades a transmetre. El 4:2:2 marca que els components de la cromància es mostregin a la meitat de freqüència que els components de la luminància. Aquesta reducció de freqüència de mostrejat disminueix en

[53] Recomanació feta per la ITU (International Telecomunication Union) per al streming degital de vídeo PAL i NTSC en definició estàndard.

una tercera part la quantitat d'informació a transmetre afectat de forma quasi imperceptible a la qualitat visual de la imatge.

En figura 4-13 podem veure un exemple del samplejat 4:2:2. Com es pot veure, només enviem els components Cb i Cr en píxels alternats. Un exemple de seqüència en la que això succeeix seria: YCbCrYYCbCrYYCbCrYYCbCr.

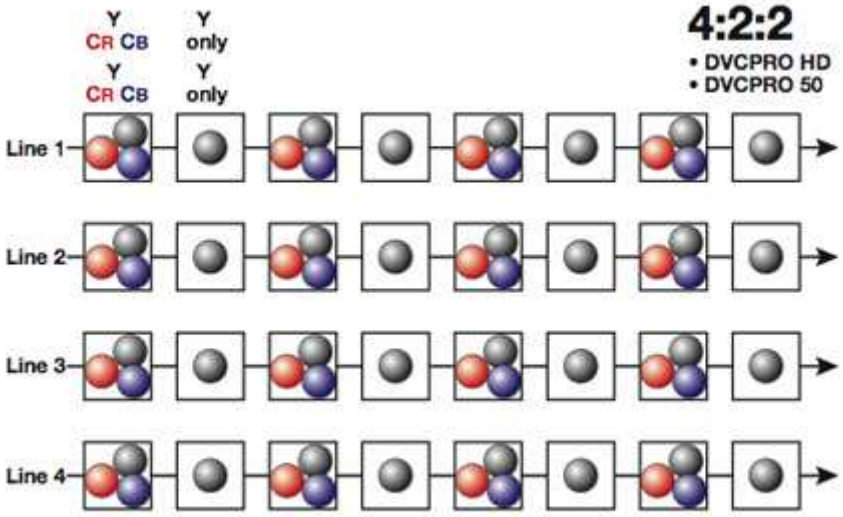


Figura 4- 13 - Samplejat 4:2:2

El fet que s'hagi seguit aquesta recomanació en el nostre DSP ha comportat que haguem hagut de fer algunes modificacions en el nostre algorisme. En primer lloc, s'ha analitzat la nostra imatge només tenint en compte aquells components que contenen informació cromàtica. Tot i que pot semblar que això pot afectar a la fiabilitat que tingui el nostre algorisme, no és el cas ja que, al comparar-lo amb la nostra versió per a PC, ens trobem en que també s'hi aplica aquesta característica en forma de tauler d'escacs. Apart d'aquest canvi, i donat el fet que el nostre algorisme funciona en l'espai RGB, s'ha hagut de fer una conversió de l'espai YCbCr a l'espai RGB. Aquesta conversió ha afectat al rendiment del algorisme en la plataforma d'Axis, però, gràcies algunes optimitzacions efectuades a posteriori, se n'ha minimitzat l'efecte al màxim.

## 4.7 Optimitzacions efectuades

Donat el requeriment inicial de mantenir la velocitat de processament del nostre algorisme a un mínim de 6fps, s'han hagut de fer algunes optimitzacions en el nostre codi per tal d'augmentar el rendiment fins aquest punt. Malgrat això, donada la limitada capacitat per a depurar el codi que ens dona el nostre sistema de desenvolupament basat en traces (veure apartat 2.9), considerem que encara es pot optimitzar molt més el codi, això si, fent us de l'emulador JTAG que ens proporciona Texas Instruments. Aquest emulador ens permet veure a temps real quines parts del nostre DSP es fan servir en cada instrucció, així com l'estat de les nostres memòries cache. Aquesta funció ens permet optimitzar el codi al màxim per a aprofitar el paral·lelisme que ens proporciona l'arquitectura VLIW i optimitzar l'us de les memòries cache.

Cal dir que, tal i com s'ha explicat en l'apartat 3.2, la memòria cache L2 del nostre DSP es pot configurar per a que sigui mapejada i feta servir com a memòria principal. Seguint les recomanacions del servei d'assistència tècnica d'Axis, varem decidir configurar aquesta memòria per a que funcionés com a memòria principal. D'altra banda, és precisament en aquesta regió del nostre mapa de memòria on hi hem guardat totes aquelles estructures de dades i variables de les quals hi fèiem un us més intensiu. D'aquesta manera, tot i que hi ha dades que no ens han cabut dins la memòria L2, hem aconseguit un accés a memòria força ràpid en moltes operacions.

En segon lloc, tal i com hem comentat en l'apartat anterior, el nostre algorisme de *BGS* depèn força de les operacions en coma flotant. A més, en la conversió de les nostres dades entre l'espai de colors YCbCr i l'espai RGB hem hagut d'efectuar també operacions en coma flotant. Per tal d'optimitzar aquest tipus d'operacions en la nostra plataforma, hem seguit dues estratègies ven diferenciades.

- En primer lloc, hem fet us de LUTs en totes aquelles operacions en les que s'hagués de realitzar una multiplicació entre una constant en coma flotant i numero de 8 bits. Això ens ha fet que haguem de guardar a la nostra memòria L2 una matriu de 256 posicions amb els resultats prèviament calculats d'aquestes

operacions. Com veurem en els *benchmarks* de l'apartat 4.9, aquesta optimització aplicada a la conversió entre l'espai YCbCr i l'espai RGB ens ha suposat un increment força elevat en el rendiment del sistema.

- En segon lloc, hem fet servir la llibreria proporcionada per Texas Instruments FastRTS per a efectuar totes aquelles operacions en coma flotant que no hagin pogut ser optimitzades amb l'us de les LUTs. Aquesta llibreria, proporciona una sèrie de funcions per a efectuar operacions bàsiques en coma flotant totalment optimitzades per als DSPs que no incorporen unitat de coma flotant.

Una altra optimització que hem pogut fer en el nostre del nostre desenvolupament ha estat en les operacions de divisió i multiplicació múltiples de 2. Aquestes operacions es poden simplificar fàcilment fent ús de *shifts* a l'esquerra, en el cas de les divisions, i *shifts* a la dreta en el cas de les multiplicacions. Tot i que molts compiladors ja fan aquest tipus de optimització per si sols, cal dir que en molts casos hem triat explícitament paràmetres de l'algorisme per tal de que fossin múltiples de 2 i així poder optimitzar les operacions amb aquests al màxim.

Finalment, cal mencionar que tot i que creiem que seria possible optimitzar el nostre codi per tal de fer un us més òptim de la cache L1, no hem tingut la necessitat ni el temps necessari per a fer-ho. Cal dir que, gràcies a les altres optimitzacions esmentades més amunt, ja hem pogut obtenir un rendiment satisfactori del nostre algorisme i, per tant, no hem dedicat més esforç en aquest sentit.

## 4.8 Problemes Trobats

Durant la implementació del nostre algorisme a la plataforma d'Axis ens hem trobat amb un munt de problemes, incloent totes les consideracions mencionades en l'apartat 4.6 i 4.7. Apart de totes aquestes, intentarem explicar de forma breu aquells punts que hagin estat més problemàtics en el moment del depurat.

El primer problema que ens ha sorgit ens ha vingut donat per una de les optimitzacions explicades en l'apartat anterior. Tal i com hem dit, hem optimitzat algunes multiplicacions i divisions per múltiples de 2 amb *shifts* a la dreta i a l'esquerra. Concretament en el cas de les divisions, varem tenir un problema força difícil de detectar degut a la pèrdua de precisió que aquest tipus d'operacions suposen. Per solucionar aquest problema varem augmentar la dimensió de les nostres variables en 8 bits i varem aplicar un *shift* de 8 posicions a l'esquerra. Solsament aquest fet ens dona 8 bits més de precisió en el moment d'efectuar una divisió optimitzada.

Un segon problema important que ens ha sorgit durant el desenvolupament ha estat en la conversió de l'espai YCbCr a l'espai RGB. Donades les limitades capacitats de depurat de les que disposàvem, ens ha estat força difícil comprovar que aquesta conversió era correcte. Per tant, varem realitzar un conjunt d'experiments empírics en els quals realitzàvem la mateixa conversió de colors en Matlab i en la nostra plataforma per a tenir un model de referència i poder estar segurs que els valors convertits eren equivalents.

També relacionat amb la conversió de colors, ens varem trobar amb un problema de pèrdua de precisió per fel d'usar LUTs en el nostre algorisme. Aquesta pèrdua de precisió era deguda al fet que les operacions prèviament calculades no s'havien guardat amb el suficient nombre de decimals i això suposava un decrement en la precisió de les operacions de multiplicació en coma flotant que feien us de les LUT.

Finalment, també ens varem topar amb un problema amb el funcionament del nostre model de background en l'algorisme finalment aplicat. Varem detectar aquest problema durant l'etapa de Test funcional de l'algorisme. No te sentit explicar aquí la resolució del problema que varem fer servir, donat el fet que no s'ha explicat en detall el funcionament de l'algorisme. Tot i així, com a curiositat, el símptoma de mal funcionament es produïa al cap de 2 hores d'execució ininterrompuda. Aquest anava absorbint de forma progressiva tots els objectes del *foreground* fins que a les 2 hores deixava de detectar cap tipus de moviment a l'escena. Cal dir que, després de diversos dies de depurat, varem trobar el problema i ara finalment està solucionat.

## 4.9 Tests de l'algorisme

A l'hora de testejar el nostre algorisme ens ha interessat detectar dos tipus de problemes que sorgeixen quan s'intenta fer una migració d'un algorisme de visió entre diferents plataformes.

Primerament, ens interessava detectar els problemes de funcionament del nostre algorisme que venen donats per errors de programació del nostre codi. Per fer-ho varem deixar el nostre algorisme corrent ininterrompudament durant diversos dies sencers. Tal i com hem vist en l'apartat 2.8, gràcies a aquest sistema de test, varem poder detectar els primers problemes funcionals.

En segon lloc, ens interessava comprovar que el resultat de l'algorisme era el correcte. En el cas dels algorismes de *BGS,* comprovar el funcionament correcte no és una tasca trivial. Tan és així que en [3] es proposen diversos mètodes per a avaluar el rendiment dels algorismes d'aquest tipus de manera que el resultat sigui comparable. En el nostre cas, per sort, ja disposàvem d'una versió prèviament implementada per a plataforma PC. Per això, n'hi va haver prou en fer córrer una mateixa seqüència de vídeo en les dues plataformes i comparar els resultats per a poder veure que són equivalents. La figura 4-14 mostra el resultat d'un d'aquests tests amb una seqüència vídeo freqüentment usada en sistemes de vídeo vigilància. La primera fila ens mostra la seqüència original; la segona ens mostra els resultats de l'algorisme en la plataforma basada en PC; finalment, la tercera fila ens mostra els resultats de l'algorisme en la nostra plataforma d'Axis. Com es pot apreciar, els resultats, tot i que presenten certes diferències degudes al soroll introduït per la conversió d'analògic a digital, són molt semblants i per tant es pot concloure que l'algorisme funciona correctament.

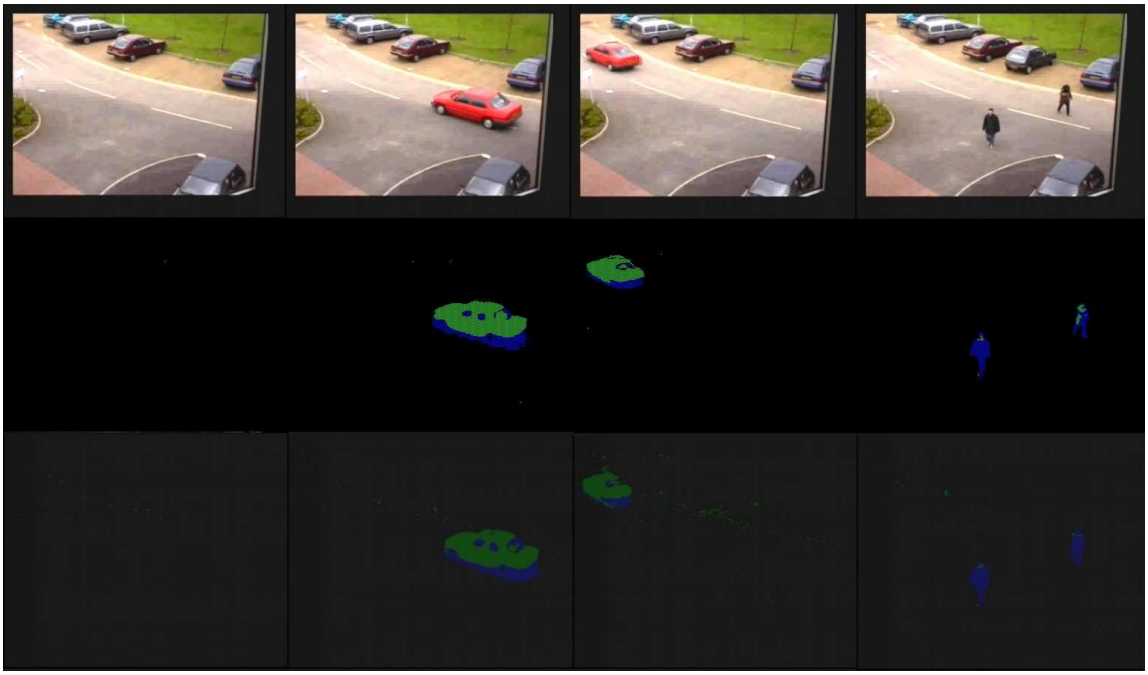

Figura 4- 14 - Resultat comparatiu entre la versió PC i la versió DSP

## 4.10 Benchmarks

Per acabar amb aquest capítol, explicarem una mica com varem estudiar la velocitat de processament que ens dona el nostre sistema basat en DSP per tal de comparar el rendiment d'aquest amb el de la nostra versió per a PC. Cal dir que aquests resultats, tot i no ser els més òptims, són suficients per a considerar la migració de part o tot l'anàlisi de visió del Daview a la nostra plataforma.

Per a realitzar aquests benchmarks, varem dividir els tests en 3 etapes de 1, 5 i 10 minuts per tal d'estudiar la tendència del nostre algorisme. Cadascuna d'aquestes etapes va esser testejada habilitant i deshabilitant l'us de les LUTs per a la conversió RGB a YCbCr. D'aquesta manera, el que aconseguíem era aïllar l'impacte d'aquesta conversió en el rendiment del nostre sistema. També varem aplicar les tres etapes de test aïllant la conversió YCbCr a RGB per si sola, també amb l'ús de LUTs i sense, per tenir una idea del cost computacional d'aquesta en el nostre sistema. Els resultats els podem veure en la taula 4-1. Com es pot apreciar, el rendiment del sistema amb l'ús de les LUT és de 7.406 fps, suficient tenint en consideració que en la plataforma PC n'hi ha prou amb un rendiment de 6 fps.

| | FrameRate / Time for 1 frame (1min) | FrameRate / Time for 1 frame (5min) | FrameRate / Time for 1 frame (10min) |
|---|---|---|---|
| CodeBook with LUT | 7.383fps / 135435us | 7.420fps / 134753us | 7.406fps / 135020us |
| CodeBook without LUT | 5.350fps / 186896us | 5.323fps / 187855us | 5.329fps / 187618us |
| YCbCr to RGB with LUT | 24.916fps / 40141us | 24.982fps / 40027us | 24.953fps / 40074us |
| YCbCr to RGB without LUT | 8.331fps / 120027us | 8.352fps / 119731us | 8.333fps / 119999us |

# 5. Desenvolupament del comptador zenital de persones

## 5.1 Especificacions del comptador

La segona part del projecte ha consistit en desenvolupar un comptador zenital de persones en la mateixa plataforma d'Axis. Això no tant sols ha servit per a obrir una nova línea de productes a DAVANTIS Technologies, sinó que també ha servit per a avaluar la plataforma com a suport per a la implementació de solucions empresarials complexes. Per aquest motiu, les especificacions del comptador han estat fortament influenciades tant per les necessitats dels clients, els quals prèviament ja havien demanat aquest tipus de productes a l'empresa, com per la necessitat d'integrar aquest producte en una plataforma amb fortes restriccions d'àmbit.

En primer lloc, cal tenir em compte la situació de la càmera en aquests tipus de comptadors. Aquesta, està situada normalment en un lloc elevat i enfocant en direcció al terra. Això permet tenir una vista zenital de les persones que passen per una determinada zona de la imatge. La distància entre el objectiu de la càmera i el terra dependrà de cada situació i per tant caldrà que l'algorisme, gràcies a la configuració prèvia que es faci en cada cas, s'adapti a la disposició específica de cada càmera. D'altre banda, l'angle focal de cada òptica i les diferents alçades de les persones faran que la imatge que finalment s'analitzi sigui diferent en cada cas. En la figura 5-1 podem veure un esquema de situació de la càmera en el que es poden apreciar les variables principals que influencien en les característiques de les imatges.

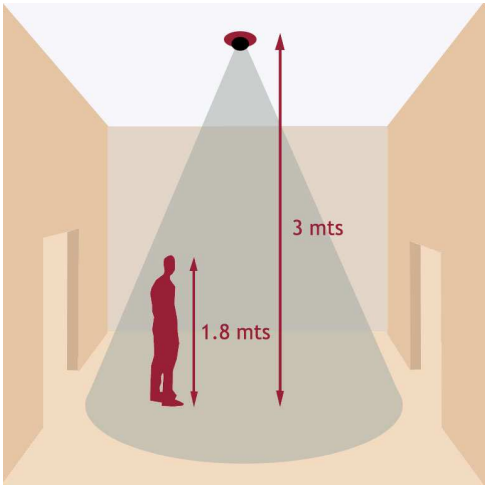


Figura 5- 1 – Esquema instal·lació càmera zenital

En segon terme, cal analitzar en quins llocs poden anar instal·lats aquests tipus de comptadors. Normalment, aquests aniran a l'entrada d'edificis en els que l'afluència de públic és elevada i un comptatge manual seria impracticable. Tot i que en la versió que s'ha implementat no s'ha tingut en compte la direccionalitat en la que es moguin les persones, s'entén que aquestes, al tenir llibertat de moviment, poden moure's en qualsevol direcció. Tot i així, per qüestions pràctiques, s`entendrà que cal comptar una persona quan aquesta creui una línea virtual que l'usuari hagi definit prèviament.

Un altre fet que cal considerar són aquelles restriccions que vinguin donades per la plataforma en si. Tal i com s'ha descrit en el capítol 2, Axis proporciona als seus socis una sèrie de SDKs per al desenvolupament de software per a la seva plataforma. Aquests SDK estan dissenyats, no tant sols per a facilitar el desenvolupament d'aplicacions per a la plataforma, sinó que també per que aquests s'integrin amb el software que ve de fàbrica en els servidors de vídeo. Malgrat aquest fet, en molts casos ens trobem amb certes limitacions d'àmbit a l'hora de desenvolupar la nostra aplicació i s'han de trobar solucions alternatives per evitar conflictes amb el software d'Axis. Per assegurar-se la compatibilitat, Axis ha desenvolupat una utilitat de test i un programa de certificació que assegura que els mòduls certificats funcionen correctament en la seva plataforma. Totes aquestes restriccions així com el programa de certificació venen recollides en [7]. A mesura que avancem en aquest capítol anirem detallant aquells casos en els que ens hagin suposat un problema.

Finalment, les especificacions més importants a les quals ens hem hagut de sotmetre ens han vingut per les necessitats dels clients i la situació del mercat. Donat el fet que actualment no existeix cap altre soci d'Axis a Espanya que hagi desenvolupat un comptador de persones per a la seva plataforma, varem haver de prioritzar les necessitats del nostres clients per tal de centrar-nos en primer lloc en aquelles funcionalitats que més demanada tinguessin. Així doncs, la prioritat de DAVANTIS ha estat desenvolupar un comptador amb les funcions bàsiques per a llençar-lo ràpidament al mercat i anar afegint noves funcionalitats a aquest a mesura que els clients les anessin demanant. Per tant, de forma ordenada, les funcionalitats mínimes que varen considerar que havia d'incloure són:

- Interfície basada en web.
- Estadístiques de comptatge de persones per hora i dia de la setmana
- Generació de gràfiques

- Exportació de les dades de comptatge a altres aplicacions per al seu posterior processament.

Cal aclarir que, donada la necessitat de desenvolupar ràpidament una solució, s'ha considerat menys important aconseguir una bona fiabilitat de comptatge o la capacitat de detectar la direccionalitat en la que es mouen les persones, que no pas oferir una solució complerta i útil. Tot i així, Pol Cunill Rafael, company nostre a DAVANTIS Technologies, ha estat realitzant el seu PFC paral·lelament al meu, avaluant diferents algorismes de comptatge de persones, per a migrar-los posteriorment a la plataforma d'Axis.

## 5.2 Algorisme de comptatge de persones

Tenint en compte que en el moment del desenvolupament de l'algorisme encara no s'havien obtingut resultats en l'avaluació de les diferents alternatives per al comptatge de persones, es va optar per a fer servir un algorisme molt senzill i provar-ne la seva fiabilitat sotmetent-lo a un conjunt de testos en Matlab. Aquest algorisme ha estat inspirat en anàlisis previs que s'havien fet sobre els productes oferts per la competència i per tant, a part de les proves efectuades en Matlab, està basat en una visió purament empírica del problema.

En primer lloc, tot i que com veurem més endavant no seran suficients, considerarem que les dades d'entrada que ens proporciona inicialment l'usuari són les coordenades de la línea virtual que haurà de creuar una persona per a que sigui comptada i les dimensions d'aquesta en número de píxels. Com exemple per a entendre una mica millor el seu funcionament, partirem de la seqüència de la figura 5-2. Per a que sigui mes visual, s'ha dibuixat de forma manual sobre les imatges la línea virtual prèviament esmentada.

**Figura 5- 2 –Seqüència zenital original**

Com es pot veure, en la seqüència apareix una persona que creua la imatge de dalt a baix. Si ara agafem cada píxel de la imatge i li restem el valor que tenia el mateix píxel en el frame anterior obtindrem una seqüència com la que apareix en la figura 5-3. Com es pot veure, només apareix la part de la persona que ha canviat entre un frame i el següent. Aquest efecte ens interessarà doncs, si una persona es queda aturada al mig de la imatge, aquesta desapareixerà a ulls del nostre algorisme.

Figura 5- 3 – Seqüència de diferència de frames

A nivell conceptual, podem entendre aquesta diferència de frames com un algorisme de background subtraction en el que el model del fons s'actualitza molt ràpidament. D'aquesta manera, si una persona no es mou, és absorbida per el model pràcticament a l'instant. Ara be, com que ara ens interessa determinar quantes persones hi ha a la imatge, procedirem a aplicar un llindar a tots els píxels per a determinar quants píxels ocupa la nostra persona. El resultat el podem veure en la figura 5-4. Com es pot apreciar, aquest llindar ens transforma la imatge en una imatge binaria en la que els píxels marcats en blanc són aquells píxels que han sofert un canvi des del frame anterior.

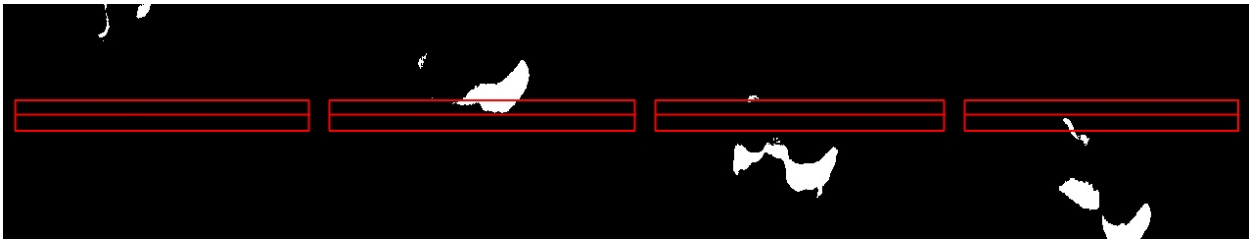

Figura 5- 4 – Seqüència binaritzada

Imaginem ara que ens fixem en una regió concreta de la imatge. Tal i com hem dibuixat en la figura 5-4, aquesta regió ens vindrà determinada per la línia virtual definida per el usuari i una altura *h* que també haurem d'incloure com a paràmetre. Si ara contem el nombre de píxels que cauen dins del requadre en cada frame de la nostra seqüència, es podrà apreciar clarament en quin moment la persona creua la imatge. La figura 5-5 mostra un gràfic en el que es pot veure aquest comptatge al llarg del temps. Com es pot apreciar en el frame 25 aproximadament, obtenim el màxim de píxels dins del nostre requadre, cosa que ens indica en quin moment la persona està creuant la línea.

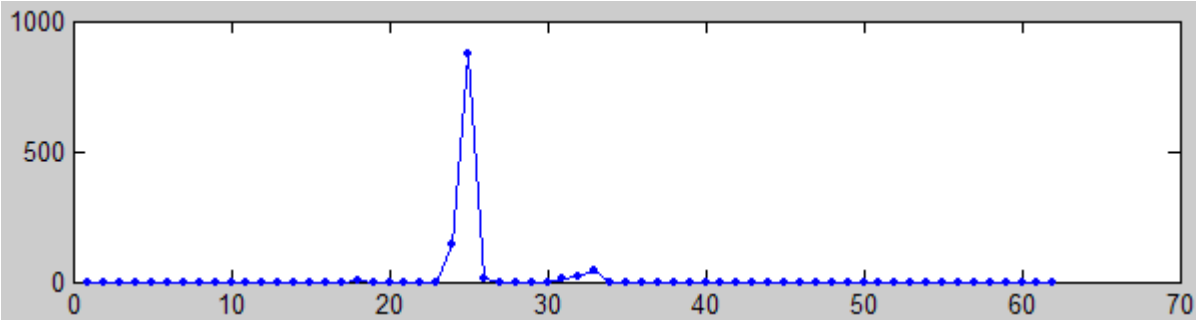


Figura 5- 5 – Número de píxels dins del requadre d'interès

Ara be, en un entorn real el nombre de persones que creuaran la línea alhora anirà variant al llarg del temps. A més, aquestes persones poden creuar la línea a diferents velocitats, cosa que pot provocar que, en comptes d'un sol màxim, obtinguem un conjunt de màxims seguits, indicant-nos que la persona s'ha mantingut dins del requadre per un temps determinat. Tot això provoca que haguem d'afegir una paràmetre més al nostre algorisme indicant-nos amb quina freqüència volem mirar el nombre de píxels que cauen dins del nostre requadre. Aquesta freqüència de mostreig es pot entendre com una mesura per indicar la velocitat mitja amb la que les persones creuen la nostra línea. A més d'això, el paràmetre que indica la dimensió que ocupa una persona en nombre de píxels ens servirà per a determinar el nombre de persones que hi ha en un moment determinat dins del nostre requadre.

Així doncs, per resumir, el nostre algorisme quedarà de la següent forma:

```
/*--Parametres de l'algorisme --*/

x1, y1, x2, y2  /* - Defineixen requadre d'interès. */
Threshold = 25;
MidaPersona = 700;
FrameRate = 2;

Comptador = 0; /*-- Variable del comptador --*/
NumFrames = 0; /*-- Variables auxiliars --*/

/*--Inicialització de l'algorisme --*/
FrameAntic = LlegirPrimerFrame(video)

/*--Bucle Principal --*/
Mentre(FrameNou = LlegirFrame(video))
     DiferenciaFrames = ValorAbsolut( FrameNou – FrameAntic )
     Per cada pixel DiferenciaFrames
          Si DiferenciaFrames[i] > Threshold
               FrameBinari = 1
          Si No
               FrameBinari = 0
          Fi Si
     Fi Per
     Suma = ComptarPixels(FrameBinari, x1, y1, x2, y2)
     NumPersones = Suma/MidaPersona;

     NumFrames = NumFrames +1

     Si NumFrames % FrameRate == 0
          Comptador = Comptador + NumPersones;
     Fi Si
Fi Mentre
/*-- FI Bucle Principal --*/
```

Com es pot veure, en el pseudocodi hem establert la variable *FrameRate* a 2. Això vol dir que el nostre algorisme funcionarà a una freqüència de mostreig de 12 fps. Ara be, la variable FrameRate només podrà tenir valors múltiples de 24, ja que sinó estaríem samplejant cada segon de la nostra seqüència en un frame diferent. Així doncs, depenent del

valor de la variable *FrameRate,* el nostre algorisme funcionarà a una freqüència de 24, 12, 8, 6, 4, 2, 1 fps.

## 5.3 Proves en Matlab

Tot i la senzillesa de l'algorisme de comptatge de persones, es va decidir desenvolupar una primera versió d'aquest en Matlab per a provar que l'algorisme realment funcionava i avaluar-ne la fiabilitat. També varem intentar trobar la relació d'alguns dels paràmetres de l'algorisme per tal de simplificar el procés de configuració de l'algorisme el màxim possible. Per a fer-ho, varem provar diferents configuracions sobre un conjunt de seqüències de vídeo gravades en diferents localitzacions i en varem mesurar els falsos positius, els falsos negatius i la fiabilitat de comptatge general.

Si analitzem el nostre algorisme, els paràmetres que influencien més en la seva fiabilitat de comptatge són el llindar de moviment, la mida de la persona, el framerate de mostreig i l'altura del requadre d'interès. Tot i així, en els nostre testos es va decidir fixar el llindar de moviment a un 10% de l'espai de colors per tal de simplificar els tests. Amb una capacitat de representació de 8 bits per a cadascun dels components de l'espai RGB, n'hi hauria prou amb que en algun d'aquests hi hagués una diferència de 25 nivells de color per a superar el llindar de moviment i classificar el píxel com a píxel de moviment.

Així doncs, deixant fixat aquest paràmetre, varem entrar a avaluar la relació que hi havia entre l'altura del requadre d'interès i el nombre de píxels que es comptaven en el nostre algorisme per cada frame. Per a fer això, varem sotmetre el nostre conjunt de test a una sèrie de proves amb diferents altures de requadre. Tal i com es pot intuir, varem descobrir una relació lineal entre el nombre de píxels que es comptaven dins del requadre i la mida d'aquest. En la figura 5-6 podem veure el resultat d'un d'aquests tests sobre una seqüència de vídeo en la que hi passen 5 persones. La línea negra mostra el resultat del comptatge amb una altura de quadre de 2 píxels, la verda mostra el resultat del comptatge amb una altura de 10 píxels i la blava mostra el resultat del comptatge amb una de 20 píxels.

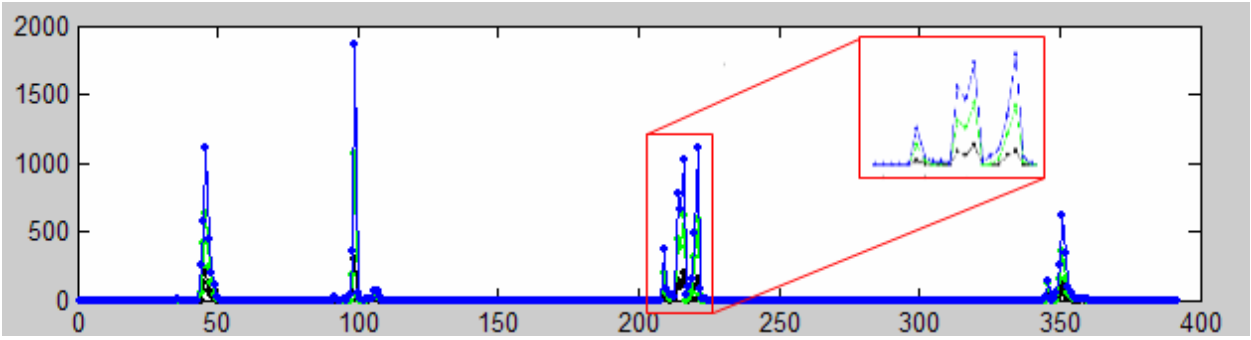


Figura 5- 6 – Resultat dels tests proporcionalitat de comptatge

Malgrat tot, aquesta gràfica no ens demostra que el increment en el nombre de píxels contats sigui directament proporcional a l'altura del quadre. Per comprovar-ho, varem calcular la diferència de comptatge entre les diferents proves efectuades amb diferents altures de comptatge i varem comprovar que, efectivament, per diferències d'altura de quadre equivalents, la diferència entre el nombre de píxels comptats era proporcional. Cal mencionar que aquesta relació de proporcionalitat només es compleix fins que l'altura de quadre ocupa una proporció de la pantalla determinada. En aquest punt la persona entra completament dins del nostre requadre i, per tant, manté el comptatge a un nivell constant.

Un cop vista aquesta relació de proporcionalitat varem assumir que, si la mida és prou petita, l'àrea del nostre requadre d'interès no afectarà en gran mesura a l'algorisme. Així doncs, varem fixar aquesta altura a 8 píxels i varem procedir a analitzar el framerate de mostreig i la mida de les persones. En el cas de la mida de les persones varem assumir que donat el fet que l'àrea del requadres és variable, és millor expressar aquesta mida en forma de percentatge d'ocupació d'aquest que no pas com a valor absolut. Donat el fet que el nombre de píxels contats és proporcional a l'àrea del requadre, el percentatge d'ocupació es indiferent a la mida d'aquest. Aquesta relació la podem veure en les següents fórmules. $N_{A1}$ i $N_{A2}$ són el nombre de píxels comptats en un instant concret donades unes àrea $A_1$ i $A_2$ determinades, $\alpha$ la relació de proporcionalitat entre aquests dos comptatges, $\beta$ el percentatge d'ocupació del requadre que ocupa una persona i $M_1$ i $M_2$ el nombre de píxels reals que ocupa una persona donades una $\beta$ i una area determinades.

$$N_{A1} = \alpha * N_{A2}$$

$$M_1 = \beta * A_1$$

$$M_2 = \beta * A_2$$

Ara be, en un moment determinat $N_{A1}$ i $N_{A2}$ seran proporcionals a $M_1$ i $M_2$ respectivament, per tant es pot veure que:

$$N_{A1} = \varphi_1 * M_1 \qquad N_{A2} = \varphi_2 * M_2$$

$$\varphi_1 * M_1 = \alpha * \varphi_2 * M_2$$

$$\varphi_1 * \beta * A_1 = \alpha * \varphi_2 * \beta * A_2$$

$$A_1 = \alpha * \varphi_2 / \varphi_1 * A_2 \rightarrow A_1 = K * A_2$$

Un cop vist això, varem procedir a sotmetre el nostre conjunt de test a diferents configuracions de framerate i percentatge d'ocupació i varem comparar els resultats amb els *ground truths*[54] que havíem efectuat manualment. Això ens va permetre mesurar la quantitat de falsos positius, la quantitat de falsos negatius i la fiabilitat total del nostre algorisme. Donat el fet que tenim 7 freqüències de mostreig diferents (24, 12, 8, 6, 4, 2, 1), varem sotmetre cada seqüència a 700 proves en les que combinàvem cadascuna de les freqüències amb 100 percentatges d'ocupació de quadre en intervals de 1%. El resultat va ser, com es d'imaginar, un llistat molt gran de fiabilitat, falsos positius i falsos negatius.

Per tal de que aquests resultats poguessin ser útils, es va optar per buscar una representació gràfica en la que es puguessin veure de manera clara en quins casos, i en quines configuracions el percentatge de falsos positius, falsos negatius i fiabilitat total és més òptim. Un exemple d'aquesta representació gràfica els podem veure en les figures 5-7, 5-8 i 5-9. Les columnes de la imatge delimiten les 7 freqüències de mostreig diferents, mentre que les files delimiten els diferents percentatges d'ocupació de quadre. El color blanc identifica aquelles configuracions que han obtingut uns resultats bons, mentre que el color negre identifica aquelles configuracions que han obtingut resultats dolents. Així doncs, la figura 5-7 mostra el gràfic de falsos positius, la figura 5-8 la de falsos negatius i la 5-9 la fiabilitat total.

[54] Taula de veritat del comptatge elaborada manualment

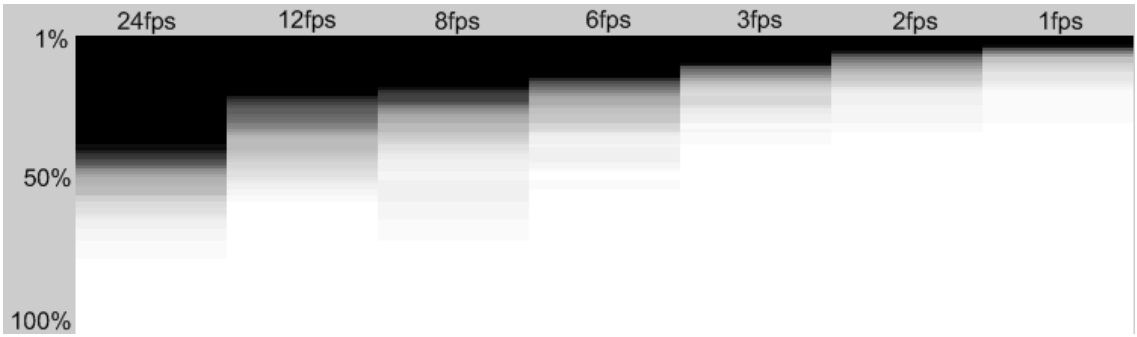


Figura 5- 7 – Gràfic de fiabilitat en falsos positius

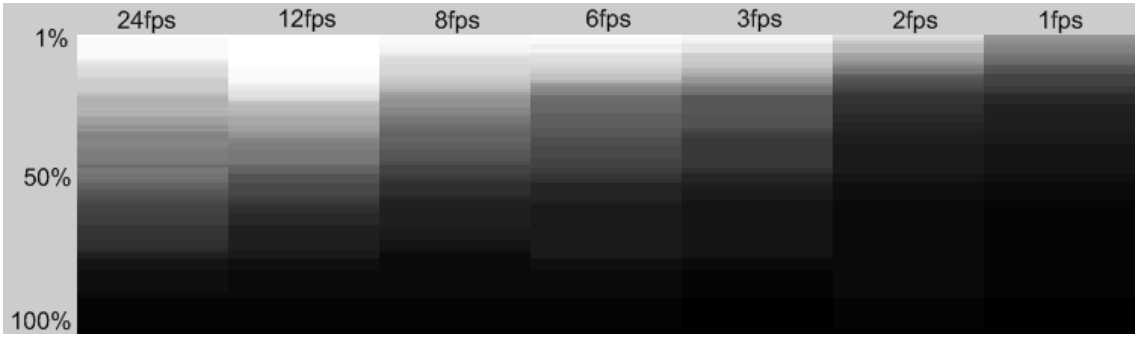


Figura 5- 8 – Gràfic de fiabilitat en falsos negatius

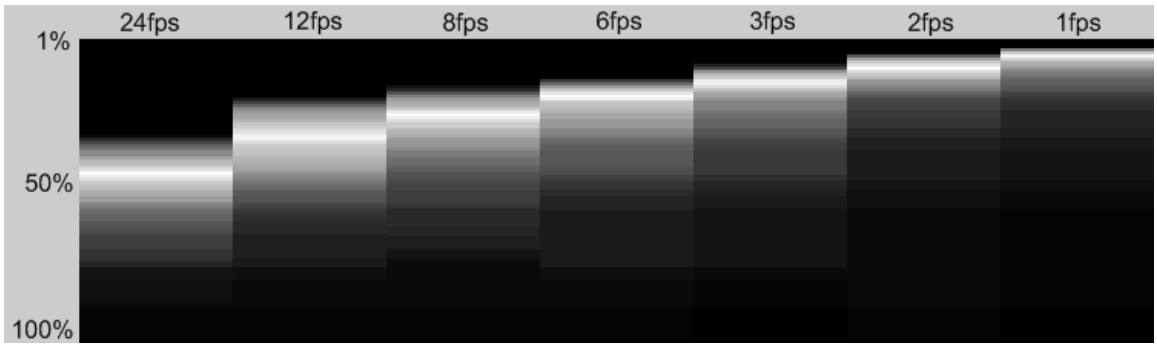


Figura 5- 9 – Gràfic de fiabilitat total

Com es pot veure, el gràfic de falsos positius (figura 5-7) mostra un gran percentatge de falsos positius per percentatges d'ocupació baixos i freqüències altes, mentre que el gràfic de falsos negatius (figura 5-8) mostra una baixa fiabilitat en percentatges d'ocupació alts i freqüències baixes. Finalment, el gràfic de fiabilitat total (figura 5-9) mostra les dades de falsos positius i falsos negatius combinades, identificant aquelles regions en les que la fiabilitat és màxima. El que cal remarcar d'aquesta última figura és que, en aquelles regions en les que la transició entre el color blanc de màxima fiabilitat i el color negre de mínima fiabilitat és més progressiva, es veuran configuracions més interessants, doncs permetran un ajustament del nostre algorisme més fi que no pas altre regions.

## 5.4 Conclusions Obtingudes

Un cop fetes les proves en Matlab, s'ha intentat extreure conclusions per a facilitar la configuració del nostre algorisme en entorns reals. La realitat és que, tot i que s'han obtingut fiabilitats per l'entorn del 90%, aquestes fiabilitats estan fortament condicionades a una bona configuració dels paràmetres de l'algorisme segons la situació. Tot i així es pot dir que, a nivell general, les conclusions obtingudes són les següents:

- L'àrea del requadre d'interès no és determinant per el bon funcionament de l'algorisme i per tant, mentre sigui més petita que el nombre total de píxels que ocupa una persona en la imatge, pot estar fixada per a tots els casos.

- La freqüència de mostreig i el percentatge d'ocupació del requadre tenen una relació directament proporcional i una influència molt més gran en el bon funcionament de l'algorisme que no pas l'àrea del requadre d'interès.

Per simplificar la configuració en entorns reals en els que s'hagin d'introduir els paràmetres de configuració de forma manual, s'ha elaborat una taula de paràmetres que dona, d'una manera totalment orientativa, la relació entre la freqüència de mostreig i el percentatge d'ocupació que cal fer servir.

| **Freqüència** | 24 fps | 12 fps | 8 fps | 6 fps | 3 fps | 2 fps | 1 fps |
|---|---|---|---|---|---|---|---|
| **Percentatge d'ocupació** | 45% - 70% | 35% - 65% | 25% - 55% | 20% - 55% | 15% - 50% | 10% - 40% | 5% - 30% |

Amb caràcter general, la conclusió principal que cal extreure de totes les proves que s'han efectuat sobre l'algorisme és, precisament, es que aquest és molt sensible a la idoneïtat dels paràmetres per a la situació específica de la càmera. Per tant, és totalment recomanable buscar una solució alternativa que tingui un grau de dependència inferior a la configuració i que, en conseqüència, sigui més adaptable als entorns reals.

## 5.5 Implementació en el DSP

Un cop avaluat l'algorisme que s'ha fet servir, passarem a explicar una mica per sobre com s'ha portat a terme la implementació en el DSP. Aquesta, tot i que no ha comportat grans problemes degut a la simplicitat de l'algorisme, ha hagut de tenir en compte una sèrie de factors per tal de que funcionés de la forma més òptima.

En primer lloc, per tal de que l'algorisme tingués un comportament exactament igual al que varem implementar per a fer les proves en Matlab, varem haver de realitzar la conversió de YCbCr a l'espai RGB. Aquesta conversió, tal i com s'ha explicat en l'apartat 4.6, suposa efectuar multiplicacions en coma flotant i, per tant, tal i com hem fet en el cas de l'algorisme de Background Subtraction, hem fet us de LUTs[55] per accelerar al màxim les operacions.

Una altra consideració que s'ha tingut en compte ha estat la forma en que s'ha calculat la diferència de frames. L'algorisme que s'ha explicat en l'apartat 5.2 especifica que s'ha de calcular la diferència per a tota la imatge. En realitat, per tal de simplificar els càlculs al màxim, només s'efectua la diferència d'aquells píxels que estan dins del nostre quadre d'interès. Això, tot i que pugui semblar trivial, estalvia una quantitat enorme de càlculs a l'algorisme accelerant d'aquesta manera el seu rendiment al màxim.

Finalment, cal esmentar que l'algorisme, ha diferència del de *background subtraction,* està integrat amb totes les altres parts software que formen la nostra aplicació. Això fa que l'algorisme hagi d'incorporar mecanismes de comunicació amb aquestes parts per tal de transmetre tant les dades de comptatge com els paràmetres de configuració. En el següent apartat explicarem de forma més detallada quins són aquests mecanismes i com s'utilitzen en la nostra aplicació.

[55] Look Up Tables. Taules emmagatzemades en memòria en les que les operacions estan prèviament calculades.

## 5.6 Arquitectura de l'aplicació

En aquest apartat intentarem donar una visió general dels components dels que consta el comptador de persones. Per combinar tots aquests components, s'han fet servir tant les llibreries proporcionades per Aixs, descrites en el apartat 3.6, com llibreries proporcionades per tercers que ens han estat de gran utilitat per a completar la nostra aplicació. Totes aquestes, no tant sols ens han estat necessàries per a simplificar la nostra feina, sinó que també ens proporcionen la interfície per a integrar l'aplicació amb els components software inclosos de sèrie en el servidor de vídeo.

Podem dividir l'aplicació en dues parts: L'algorisme de comptatge de persones, executat per el DSP, i d'interfície gràfica de l'aplicació, basada en entorn web i executada en el sistema Linux de l'ETRAX. Per a comunicar aquestes dues parts, farem servir diverses tècniques segons sigui el cas:

- Per als paràmetres de configuració de l'algorisme, s'ha fet servir la *Param API,* proporcionada per Axis i prèviament descrita en l'apartat 3.6. Aquesta API ens permet rebre una notificació en el DSP quant l'usuari canvia algun paràmetre del sistema des de d'interfície web. També permet emmagatzemar aquests paràmetres a la base de dades del sistema per mantenir la coherència amb la resta de paràmetres del servidor de vídeo.

- Per a l'enviament de les dades de comptatge a temps real i la resta de gràfics que es mostren sobreposades a les imatges de la càmera, s'ha fet servir la *SVG API*. Això ens ha permès generar dades SVG des del DSP que s'envien conjuntament amb cada frame i que posteriorment són dibuixades per un control ActiveX que s'executa a la màquina de l'usuari.

- Per a les estadístiques de comptatge s'ha fet servir la *File API*. Cada 15 minuts el DSP guarda en un fitxer del sistema el comptatge acumulat juntament amb una marca de temps que ens permetrà tenir, amb posterioritat, un log de tots els

comptatges que hagin succeït en el sistema. Posteriorment, aquestes dades seran processades per uns scripts CGI per a generar les dades estadístiques segons sigui el cas.

La figura 5-10 mostra un diagrama de blocs en el que es poden apreciar, de forma general, les diferents parts de la nostra aplicació i com es comuniquen entre si. Tal i com es pot veure, el gràfic també incorpora la part que corre en la màquina client de l'usuari final. Aquesta està composta per la interfície web i un control ActiveX que mostra les imatges que venen del servidor de vídeo i dibuixa a sobre d'aquestes els gràfics en SVG generats per el DSP.

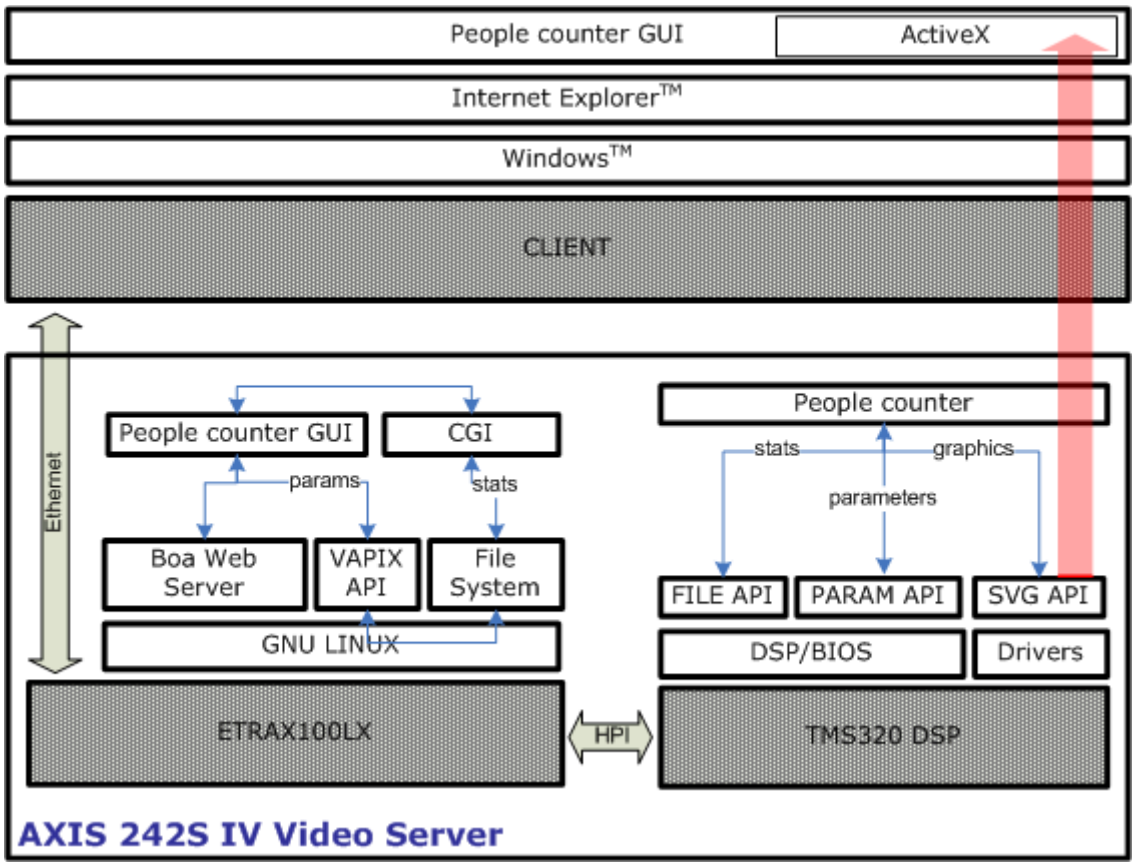


Figura 5- 10 – Diagrama de blocs del comptador de persones

## 5.7 Interfície gràfica

La interfície gràfica de la nostra aplicació ha estat dissenyada per a que sigui totalment intuïtiva i que, a la vegada, incorpori el conjunt de paràmetres de configuració que permetin ajustar l'algorisme de comptatge a cada situació. A més, com varem indicar en l'apartat d'especificacions, la nostra aplicació disposa d'un sistema de generació de gràfics per veure d'una forma visual el historial de comptatge del sistema. Aquesta sistema també cal que permetí exportar les dades generades per el comptador en format XML per a poder esser tractades per altres aplicacions, com el Microsoft Excel™.

De forma general, la nostra aplicació consta de dues planes web ben diferenciades. La plana del comptador mostra les imatges que provenen del servidor de vídeo en temps real, les dades de comptatge i els gràfics generats per el DSP. A més, aquesta plana ens permet realitzar consultes al sistema per veure en forma de gràfic de barres o en format XML l'historial del comptador. La plana de configuració, en canvi, mostra tots els paràmetres per a l'algorisme de comptatge així com els paràmetres que defineixen l'aspecte gràfic del comptador. En la figura 5-11 podem veure un parell de captures de pantalla de la interfície gràfica on es pot veure la pantalla del comptador i la generació de gràfics.

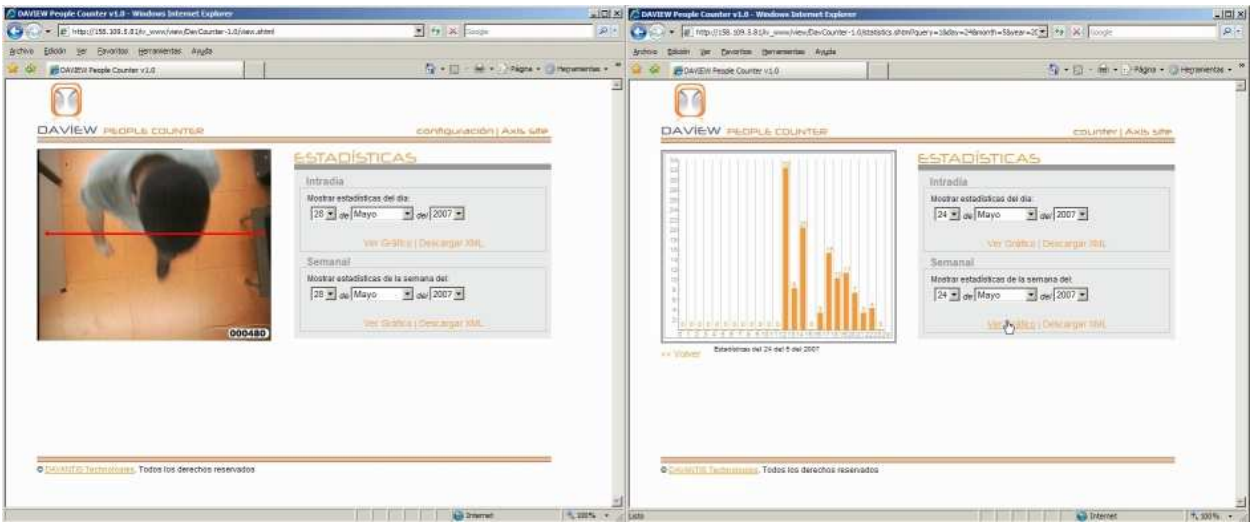


**Figura 5- 11 – Captures de pantalla del comptador de persones**

Per a desenvolupar la interfície web, hem fet us de tecnologies freqüentment fetes servir en aquests casos. En el cas de disseny gràfic, varem partir d'una maqueta realitzada en Adobe Photoshop i posteriorment la varem desenvolupar en XHTML[56] i fulles d'estil CSS[57]. A més, per a generar els gràfics de barres, varem fer servir la llibreria per a JavaScript wt_jsgraphics[8]. Aquesta llibreria, distribuïda sota la llicència LPGL[58], permet dibuixar formes simples en qualsevol navegador compatible, cosa que ens va proporcionar una base sobre la que varem dissenyar les nostres pròpies llibreries de generació de gràfics. Apart d'això, per a mostrar les imatges que provenen del servidor de vídeo, s'ha fet servir el Axis Media Control. Aquests control ActiveX permet incrustar les imatges de qualsevol producte d'Axis en una pàgina web per a integra-les amb una solució complerta. Aquest control també és l'encarregat de dibuixar sobre les imatges els gràfics en format SVG generats per el DSP.

D'altra banda, a la part servidora de la interfície, hem hagut de fer servir Server Side Includes[59] i CGI generats en C per tal de aconseguir el dinamisme de la web. Cal remarcar que la raó per la qual hem fet servir C compilat per a programar els scripts CGI en comptes d'usar llenguatges tipus Bash[60], ve donada per les limitacions d'àmbit que te la nostra plataforma i, per tant, es va haver de buscar una solució alternativa. En l'apartat 5.8, es parlarà amb més detall d'aquesta i d'altres qüestions.

Per compilar els scripts CGI desenvolupats en C s'ha hagut de fer servir un complidor creuat proporcionat per Axis Communications. Tot i així, en primera instància, es va haver de replicar l'estructura de directoris i fitxers del nostre servidor de vídeo en una màquina Linux per a dur a terme el depurat de l'aplicació. Un cop l'aplicació depurada, aquesta es va compilar per a l'ETRAX100LX i es va sotmetre a un altre testejat.

La figura 5-12 mostra, de forma resumida, l'arquitectura de d'interfície gràfica de la nostra aplicació. A la part client hi tenim el XHTML el CSS i el JavaScript, mentre que a la part servidora hi tenim el Servidor Web, els scripts CGI i els Server Side Includes.

[56] Extensible Hypertext Markup Language
[57] Cascade Style Sheets.
[58] Lesser General Public License
[59] Llenguatge de scripting molt simple que permet combinar diferents fitxers html per a generar una sola pàgina web.
[60] Bourne Again Shell

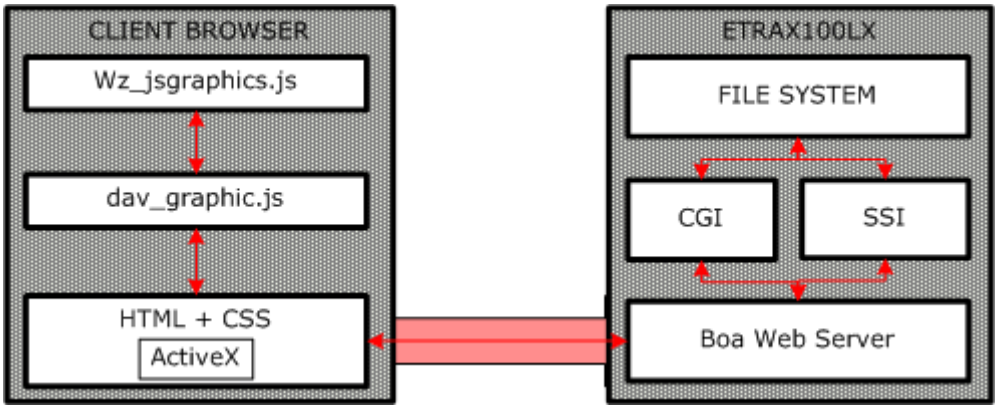


Figura 5- 12 – Arquitectura de la interfície gràfica del comptador de persones

## 5.8 Restriccions de la plataforma d'Axis

Durant el desenvolupament del comptador de persones ens hem trobat una sèrie de restriccions que ens han obligat a prendre decisions importants de disseny. Aquestes restriccions venen donades, be per les limitacions de la plataforma en si, be pel fet que s'hagi d'integrar la nostra aplicació amb el software present a la plataforma.

La primera limitació important ens ve donada per la memòria flash del sistema. Les memòries flash són un tipus de EEPROM[61] que permeten l'escriptura de dades en blocs grans de bytes. Aquest fet fa que el nombre de vegades que aquestes memòries poden ser escrites estigui limitat per el fabricant. Concretament en el nostre cas, el fabricant estima que les seves memòries començarien a fallar de mitjana a partir dels 100.000 cicles d'escriptura (MTTFF[62] = 100.000 write cycles). Per aquest motiu, Axis Communications recomana que el DSP no escrigui més de 4 cops cada hora en aquesta memòria flash. Aquesta limitació fa que el nostre comptador només guardi estadístiques de comptatge 4 cops per hora. Això limita una mica la robustesa del sistema en cas de fallada elèctrica, ja que les dades de l'últim quart d'hora es perdrien. Tot i així, s'ha considerat poc important aquesta limitació, doncs a nivell d'estadístiques n'hi ha prou amb mantenir un comptatge per hores.

[61] Electronically Eresable Programable Read Only Memory.
[62] Mean Time To First Failure

Una altra restricció amb la qual ens hem trobat ha vingut donada per el paquet BusyBox que incorpora el ETRAX. Aquest paquet engloba tot el conjunt de comandes típiques que trobaríem en un sistema Linux en un sol fitxer executable i està específicament dissenyat per a sistemes encastats. Algunes exemples de les comandes que hi podem trobar són el *awk, grep, cat, more* i moltes d'altres. El problema, en aquest cas, ve donat pel fet que BusyBox permet escollir en temps de compilació quines comandes s'inclouran en el fitxer final. Aixs Communications ha incorporat el paquet BusyBox amb un conjunt de comandes molt limitat dins del servidor de vídeo. Tant és així que, en el desenvolupament dels scripts CGI per a l'anàlisi dels logs del comptador, ens hem trobat amb problemes importants si fèiem servir una estratègia tipus bash. D'altra banda, no és possible, per limitacions d'àmbit de software, substituir el paquet BusyBox per un altre amb un conjunt de comandes més extens, doncs estaríem sobrepassant les limitacions imposades per Axis Communications en [7]. Per tant, es va optar per a desenvolupar els scripts CGI en C i compilar-los amb l'ajuda d'un compilador creuat també proporcionat per Axis.

A l'hora de desenvolupar el d'interfície web, també ens varem veure limitats per el software instal·lat en els sistema. El nostre servidor de vídeo incorpora un servidor web anomenat *boa*. Aquest, a més de ser mono-usuari, no està configurat per permetre l'ús de llenguatges de scripting externs. És per aquest motiu que, a l'hora de fer que les nostres pàgines web es construïssin de forma dinàmica, hem hagut de fer us dels Server Side Includes. Aquestes permeten una mica de condicionalitat a l'hora de construir les pàgines, fent us de inclusions de fitxes en fitxers. Això, tot i que no ha suposat un a gran limitació, si que ha fet que es trigués una mica més a desenvolupar l'aplicació, doncs ha calgut aprendre a fer servir aquest llenguatge.

Finalment, l'últim conjunt de restriccions ens han vingut per les especificacions imposades per Axis en [7]. Aquestes limiten fortament la nostra aplicació indicant-ne, tant l'espai màxim disponible per a la nostra aplicació, com els llocs on és permès llegir i escriure en el sistema de fitxers, així com aquelles funcionalitats del sistema de les que la nostra aplicació en pot fer us. Cal aclarir que, tot i que aquestes limitacions no són purament de la plataforma, si que evitarien que la nostra aplicació passés el programa de certificació d'Axis i, per tant, afectaria fortament a les ventes del comptador de persones.

## 6.9 Problemes trobats

Durant el desenvolupament del comptador de persones ens varem trobar amb diversos problemes importants. Tot i que de molts d'aquests problemes ja se n'ha parlat una mica en l'apartat anterior, aquí detallarem quins han estat concretament aquests problemes i quines solucions s'han pres en el desenvolupament.

En primer lloc, totes aquelles restriccions que s'han mencionat a l'apartat 6.8 han fet que haguem de mantenir una comunicació constant amb l'equip de desenvolupament de Axis. Això ha esta degut principalment a la falta de documentació específica sobre el tema i al conjunt de buits en les especificacions que aquesta empresa dona als seus socis. No obstant, un cop hem aconseguit un bon contacte amb el seu l'equip, hem obtingut un bon suport per part seva que s'ha traduït amb una aplicació més robusta.

Un altre problema que s'ha tingut que solucionar durant el desenvolupament és precisament la configuració del compilador d'Axis. Aquest no ve lliurement publicat en la web de l'empresa i l'hem hagut de demanar al suport tècnic. A més, la documentació inclosa està poc detallada i ha calgut també ajuda del servei de suport per solucionar certs problemes amb la plataforma.

Finalment, un dels problemes principals al qual ens hem hagut d'afrontar ha estat el fet de trobar una forma de testejar la fiabilitat de l'algorisme en entorns reals. Tal i com hem vist en l'apartat 5.4, l'algorisme de comptatge de persones és molt dependent de la idoneïtat dels paràmetres de configuració segons la situació. Això fa que sigui difícil mesurar fins a quin punt s'ha aconseguit una fiabilitat de comptatge prou bona per a ser instal·lat en entorn real. Aquest problema, de fet, segueix sense estar resolt, i tot i que hem obtingut resultats, tal i com hem vist en l'apartat 5.3, aquests són qüestionables com a mesura fiable de rendiment.

## 5.10 Tests de l'aplicació

Tant per al desenvolupament del comptador de persones com per a la fase de test, s'ha realitzat una instal·lació d'una càmera zenital per tal d'obtenir dades reals de comptatge en un entorn real. Aquesta càmera, s'ha situat en una zona de pas del Centre de Visió per Computador i s'ha deixat el comptador de persones funcionant durant més de 2 mesos. Això ens ha permès veure la robustesa de l'algorisme i de l'aplicació complerta en un entorn real. A més, les dades d'historial generades per el DSP s'han fet servir com a model per a desenvolupar el sistema de gràfics de barres del comptador i comprobar que funciona quan la quantitat de dades augmenta.

En la figura 5-13 podem veure una fotografia de la instal·lació. Cal remarcar que, tot i que en aquestes fotografies no es apreciar, la càmera feta servir és analògica i està connectada directament a un servidor de vídeo com el que hem pogut veure en el capítol 2.

**Figura 5- 13 – Fotografia de la instal·lació real del comptador.**

# 6. Conclusions i línies futures de treball

## 6.1 Conclusions

Tal i com s'ha explicat en la introducció d'aquesta memòria els objectius d'aquest projecte s'han dividit en dues branques ben diferenciades: En primer lloc, la implementació de l'algorisme de *background subtraction* en el TMS320DM642 de Texas Instruments com base per avaluar aquest mateix DSP per a la migració de tota la part servidora del DAVIEW Perimeters. I en segon lloc, el desenvolupament d'una solució complerta de comptatge de persones per a càmera zenital en el servidor de vídeo 242s iv d'Axis Communications. En aquest últim capítol detallarem quines són les conclusions que s'han extret en cadascuna d'aquestes dues branques i les possibles línies futures a seguir en cada cas.

En la primera part del projecte es va implementar l'algorisme de *background subtraction* al nostre DSP i s'hi van aplicar varies fases d'optimització de codi per a millorar-ne el rendiment (apartats 4.5, 4.6 i 4.7). Tot seguit es va comparar el funcionament de l'algorisme amb la versió per a PC i es va deixar en funcionant durant diversos dies per tal de comprovar que el seu funcionament era correcte (apartat 4.9). Finalment, varem sotmetre l'algorisme a un sèrie de tests per tal d'extreure'n benchmarks de rendiment comparables amb els de la plataforma PC (apartat 4.10). Així doncs, les conclusions que en podem extreure són:

- El rendiment dels algorismes desenvolupats per al DSP dependran en una gran mesura en les optimitzacions efectuades en el codi i en la capacitat del compilador per optimitzar l'ordre de les instruccions per a aprofitar el paral·lelisme que ofereix l'arquitectura VLIW. (Apartat 3.3).

- Les optimitzacions efectuades en el nostre algorisme, tot i que suficients per a assolir el rendiment requerit en aquest projecte, no serien suficients en el cas d'incloure altres capes de visió en el codi del DSP. (Apartat 4.7)

- Tot i que el DSP de Texas Instruments no és el més adient per a la implementació de l'algorisme de *background subtraction* deguda a la seva limitació per a efectuar operacions en coma flotant, es pot dir que és suficient per a mantenir el rendiment necessari equivalent al de la plataforma PC. (Apartat 4.10).

En la part del desenvolupament del comptador de persones per a càmera zenital es va procedir primer a testejar l'algorisme de comptatge de persones en Matlab per a comprovar el seu funcionament i extreure'n característiques que ens simplifiquessin el procés de configuració (apartat 5.3). Tot seguit, varem implementar l'algorisme en el nostre DSP on es va optimitzar lleugerament per a millorar-ne el rendiment (apartat 5.5). Un cop fet això, es va passar a implementar d'interfície gràfica començant per la pantalla de visionat de la càmera, seguit de la pantalla de configuració i la generació de gràfics de barres i dades XML (apartats 5.7 i 5.8). Finalment es va procedir a testejar l'aplicació complerta deixant-la en funcionament en un entorn real i efectuant-hi proves d'us (apartat 5.9). Així doncs les conclusions que podem extreure d'aquesta part són:

- L'algorisme de comptatge escollit és molt dependent de la idoneïtat de la configuració dels paràmetres segons la situació en la que es trobi la càmera. A més, aquesta dependència fa que sigui difícil avaluar la fiabilitat de l'algorisme en entorns reals (apartat 5.4)

- En general, la plataforma d'Axis és suficient per a la implementació d'una solució complerta sempre i quant les dades generades per el DSP no siguin seqüències de vídeo. En aquest cas, les limitacions arquitecturals de la placa fan que sigui impossible enviar-les al ARPTEC-2 i per tant. caldria trobar una solució alternativa per a la compressió d'aquest vídeo. (apartat 2.7)

Per tant, donat el fet que s'ha aconseguit el rendiment adequat en l'algorisme de *background subtraction* essent el seu funcionament equivalent a la versió per a PC i s'ha desenvolupat una solució complerta de comptatge de persones per a càmera zenital, es pot dir que, en termes generals, s'han assolit els objectius que es varen fixar al principi del projecte i aquests es poden considerar com a satisfactoris.

## 6.2 Línies futures de treball

Un cop vistes les conclusions procedirem a explicar quines són les línies futures de treball que caldria seguir en cadascuna de les dues branques d'aquest projecte.

En el cas de d'implementació de l'algorisme de *background subtraction* caldria veure quines optimitzacions més es poden aplicar a aquest per tal de millorar-ne el rendiment. D'aquesta manera aconseguiríem guanyar capacitat de processament per tal d'incloure altre capes de visió en el DSP. Tot i així, creiem que en el cas que es vulgui optimitzar més el codi, caldria fer us d'un emulador JTAG per a veure en quines regions d'aquest es produeixen més fallades de cache i *stalls* del processador.

Pel que fa al comptador de persones, tot i que la intenció de l'empresa és substituir l'algorisme de comptatge de persones per un dels algorismes avaluats per Pol Cunill Rafael en el seu PFC, creiem que caldria avaluar la fiabilitat de l'algorisme actual sotmetent-lo a més entorns reals per veure com reacciona aquest a situacions que en el laboratori no es produeixen. Això ens permetria tenir una base sobre la que anar comparant les diferents millores que s'hi poguessin aplicar en un futur.

# Bibliografia

[1] Wren, C., Azarbayguai, A., Darrel, T., Pentland, A., “Pfinder: Real-Time tracking of the human body. IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(7):780-785, 1997

[2] Grimson, E., Stauffer, C., “Adaptive background mixture models for real time tracking”. Computer Vision and Pattern Recognition Conference, 1999.

[3] J.Reno, N. Lazarevic-McManus, D. Makris and G.A. Jones, “Evaluating Motion Detection Algorithms: Issues and Results”. Digital Imaging Research Center, Kingston University, Penrhyn Road, Kingston upon Thames, Surrey, UK KT1 2EE

[4] Marko Heikkilä and Matti Pietikäinen, Senior Member, IEEE, “A Texture-Based Method for Modeling the Background and Detecting Moving Objects”

[5] Jabri, S., Duric, Z., Wechsler, H., Rosenfeld, A., “Detection and Location of People in Video Images using Adaptive Fusion of Color and Edge Information”. International Conference on Pattern Recognition (ICPR), 2000.

[6] Hironobu FUJIYOSHI and Takeo KANADE, “Layered Detection for Multiple Overlapping Objects” IEICE TRANS, INF & SYST, VOL.E87-D, NO, 12 DECEMBER 2004

[7] Axis Intelligent Video Mudule Compliance Test Tool v1.00. Axis Communications.

[8] Axis 242s IV Users Manual. Axis Communications

[9] Axis Etrax 100LX Data Sheet. Axic Communications

[10] Texas Instruments DSP Selection Guide

[11] Texas Instruments TMS320DM642 Video/Imagin Fixed Point Digital Signal Processor Data Sheet

[12] Texas Intruments C64x™ CPU reference guide (spru732c)

[13] Texas Instruments TMS320C64x DSP Video Port/VCXO Interpolated control (VIC) Port (spru629e)

[14] Texas Instruments DSP/BIOS Technical Overview (spra780)

[15] Texas Instruments TMS320C600 DSP Host Port Interface Reference Guide (spru578c)

# Abstract

En aquest projecte s'usa el servidor de vídeo d'Axis Communications 242s IV, basat en el DSP TMS320DM642 de Texas Instruments, com a plataforma per a la implementació d'un algorisme d'extracció de fons i pel desenvolupament d'una solució completa de comptatge de persones per a càmera zenital. En el primer cas, s'ha optimitzat i comparat el rendiment de l'algorisme amb el d'una versió per a PC per a avaluar el DSP com a processador per a la migració d'una aplicació completa de vídeo vigilància. En el segon cas s'han integrat tots els components del servidor en el desenvolupament del comptador per avaluar la plataforma com a base per a solucions completes.

En este proyecto se utiliza el servidor de vídeo de Axis Communications 242s IV, basado en el DSP TMS320DM642 de Texas Instruments, como plataforma para la implementación de un algoritmo de extracción de fondo y para el desarrollo de una solución completa de conteo de personas para cámara cenital. En el primer caso, se ha optimizado y comparado el rendimiento del algoritmo con el de una versión para PC para evaluar el DSP como procesador para la migración de una aplicación completa de vídeo vigilancia. En el segundo caso se han integrado todos los componentes del servidor en el desarrollo del contador para evaluar la plataforma como base para soluciones completas.

The 242s IV video server from Axis Communications has been used in this project. This server is based on Texas Instruments' DSP TMS320DM642 as a platform for the implementation of a background subtraction algorithm and the development of a full people counting solution for zenithal cameras. In the first case, the performance of the algorithm has been optimized and compared with a PC version in order to test the DSP as a base processor for the migration of a full video surveillance application. In the second case all components of the server have been integrated in the development of the counter in order to test the platform as a base for full solutions.