Sistema de gestió de vídeo *off-line* per una *smart-room*

Albert Gil Moreno
Abril 2007

Director de projecte
Josep Ramon Casas

A vosaltres, Irenes i Albert, per tot l'amor que mai ens deixarem de donar.
A tu, per somriure'm cada matí.

I a vosaltres, Elena i Fernando, per què lluny o a prop, sempre hi sou.

Moltes gràcies.

El meu projecte, el vostre.

Amb aquest projecte sembla acabar-se un llarg camí universitari...
Però no és així, només en comença un de nou.

És per això que vull agrair al Josep Ramon Casas la confiança dipositada en mi des del primer dia i l'oportunitat brindada de poder continuar treballant plegats.

També vull agrair el suport inestimable d'en Joachim Neumann, qui m'ha donat tota mena de consells vitals al llarg d'aquests mesos, i d'en Jordi Salvador, qui amb les seves reconfortants valoracions sempre m'ha animat a millorar.

I als companys de Viladecavalls, a qui dec molt més del que es pensen. I als de la Facultat de Matemàtiques, per totes les tardes de llibres.

Però, sense cap mena de dubte, no puc estar de reconèixer aquí, davant d'aquest projecte, que aquest trajecte universitari hauria estat igual sense en Jorge i l'Eduard, companys aquí, i on sigui.

Aquest projecte és, també, de tots vosaltres.

La llum ens guia,
l'amor arriba,
els anys i els dies,
la vida,
un munt d'instants.

Sopa de Cabra
.

Índex

Índex ..... **vii**

Índex de figures ..... **ix**

Sobre el document ..... **xi**

Capítol 1. Introducció **1**

1.1. Context del projecte 1
1.1.1. El projecte CHIL: Computers in Human Interface Loop *1*
1.1.2. Les CHIL smart-rooms *2*

1.2. Motivació principal: sistemes incompatibles 3

1.3. Objectius del projecte 5

Capítol 2. Característiques de l'entorn del projecte **9**

2.1. La *smart-room* de la UPC 9
2.1.1. Sensors *10*
2.1.2. Equipament i configuració *10*

2.2. Què és i per què utilitzem *Smart Flow*? 11

Capítol 3. Descripció del nou sistema *off-line* **15**

3.1. Les parts del nou sistema *off-line* 15

3.2. Funcionament de la *Recording-GUI* 17

3.3. Funcionament del sistema utilitzant *Smart Flow* 21
3.3.1. Configuració individual dels clients *22*
3.3.2. Configuració col·lectiva de clients en mapes *24*

Capítol 4. Desenvolupament i implementació del sistema **29**

4.1. Història del projecte 29

4.2. Característiques rellevants de la implementació de *Smart Flow* 31
4.2.1. Fitxers de configuració de Smart Flow *31*
4.2.2. Gestió dels clients *32*
4.2.3. Mecanisme intern de comunicació i transport de flows *33*
4.2.4. Limitacions del wrapper UPC-Smart Flow *34*

4.3. La *Recording-Lib* 35
4.3.1. La Recording-Lib com a llibreria de suport pels clients *36*
4.3.2. Com a llibreria de control de clients *36*
4.3.3. Llibreria pstream *37*

4.4. Compatibilitat entre *Smart Flow* i la *Recording-Lib* 37

4.5. Sincronisme 39
4.5.1. Sistema de sincronisme en una gravació *41*
4.5.2. Sistema de sincronisme per la lectura *48*

4.6. Altres 51
4.6.1. Eines de desenvolupament de software *51*
4.6.2. Escalabilitat del sistema *52*

4.6.3. Drivers per Linux ... 53
4.6.4. Sistema de fitxers ... 53
4.6.5. Orientació a objectes ... 54
4.6.6. Nou format estàndard per gravacions ... 54
4.6.7. QT com a llibreria gràfica ... 54
4.6.8. XML com a llenguatge de configuració ... 55

Capítol 5. Resultats ... 57
5.1. Objectius aconseguits i limitacions del sistema ... 57
5.2. Tests ... 58
5.3. Estat final del sistema ... 59

Capítol 6. Valoracions i millores ... 61
6.1. Valoració d'objectius ... 61
6.2. Possibles millores i ampliacions del sistema ... 62

APÈNDIX A. Sistema de pre-compilació i generació de *Recording-Clients* ... 65

APÈNDIX B. Post processat d'una gravació. Algoritme i resultats pràctics. ... 67

APÈNDIX C. Documentació de la *Recording-Lib* ... 75

APÈNDIX D. Estructura i format estàndard CHIL de gravacions ... 81

APÈNDIX E. Fitxers de configuració ... 83

Referències ... 93

Índex de figures

Figura 1. Sessions de treball habituals (esquerra) i amb l'ús de tecnologies CHIL (dreta). (Font: [26]) 1

Figura 2. Esquema genèric d'una CHIL *smart-room*. (Font: [7]) 2

Figura 3. *Stack* de software genèric per una CHIL *smart-room*. (Font: [8]) 3

Figura 4. Il·lustració del funcionament bàsic del sistema *off-line*. 4

Figura 5. Il·lustració del funcionament bàsic del sistema *on-line*. 4

Figura 6. Esquema funcional dels sistemes d'escriptura i lectura. 6

Figura 7. Esquema funcional del sistema *on-line*. 7

Figura 8. Esquema del sistema on-line utilitzant un sistema de lectura enlloc del de captura. 7

Figura 9. Il·lustració de la smart-room de la UPC i el laboratori de televisió (Font: [27]) 9

Figura 10. Il·lustració de la *smart-room* de la UPC amb els seus sensors. 10

Figura 11. Il·lustració del funcionament bàsic de *Smart Flow*. (Font: [4]) 12

Figura 12. Captura de pantalla de la interfície gràfica de *Smart Flow*. 13

Figura 13. *Stack* funcional de software del nou sistema *off-line*. 16

Figura 14. Mapa de *Smart Flow* creat automàticament.per treballar amb el sistema *off-line*. 17

Figura 15. Captura de la *Recording-GUI*. Les tres pestanyes estan disponibles a la zona superior. 18

Figura 16. Quadre de controls de la pestanya principal de la *Recording-GUI*. Si l'usuari activa *Smart Flow mode* l'opció de *read* estarà disponible, en cas contrari estarà deshabilitada l'opció de *read*. 18

Figura 17. Diagrama de blocs dels passos a realitzar en una gravació/calibració. 19

Figura 18. La pestanya de configuració de la *Recording-GUI*. 20

Figura 19. Captura de la pestanya de configuració avançada de la *Recording-GUI*. 21

Figura 20. Exemple de fitxer de configuració per *clients*. 22

Figura 21. Exemple de llista de servidors on es capturen les imatges. 24

Figura 22. Llista de *clients* de *Smart Flow* necessaris per realitzar gravacions. 24

Figura 23. Exemple de configuració d'un mapa de *Smart Flow* per realitzar gravacions. 25

Figura 24. Exemples de mapes de lectura, un de lectura a temps real (esquerra) i l'altre de lectura controlada (dreta). El mapa de lectura controlada ha estat creat per l'estudiant de doctorat Jordi Salvador en la seva participació en les Avaluacions 2007 del projecte CHIL. 26

Figura 25. Fitxer de configuració per un mapa de lectura. 27

Figura 26. Resultat de l'execució d'un *client* de *Smart Flow* amb el paràmetre `-sf-dump-args`. 32

Figura 27. Comanda utilitzada per *Smart Flow* per executar clients 32

Figura 28. Diagrama de blocs funcionals per la transmissió de fluxos de dades a *Smart Flow*. 33

Figura 29. Il·lustració dels diferents blocs que intervenen en el transport de dades en *Smart Flow*. (Font: [5]) 34

Figura 30. *Stack* de connexions de software del nou sistema *off-line*. 36

Figura 31. Diagrama UML del control remot de client mitjançant la llibreria pstream. 38

Figura 32. Esquema de generació de clients amb la interfície de Smart Flow o amb la Recording-Lib. 39

Figura 33. Esquema de gravació totalment sincronitzada. 40

Figura 34. Esquemes dels possibles errors de sincronisme pràctics en una gravació multicàmera. Diferent freqüència de captura (esquerra), inici diferent de la captura (centre) i *dropped-frames* (dreta). 40

Figura 35. Esquema de sincronisme en la captura i l'assignació de timestamps. 42

Figura 36. Esquema d'un error de sincronisme provocat per una mala assignació de timestamps. 42

Figura 37. Esquema de sincronisme en temps real. Aquest sistema s'ha descartat. 44

Figura 38. Esquema general del sistema de sincronisme utilitzat. 44

Figura 39. Gravació amb problemes de sincronisme que ha de ser processada. 46

Figura 40. Creació de la càmera *master*. 47

Figura 41. Clipping de la càmera *master*. 47

Figura 42. Sincronització de la càmera 1. 47

Figura 43. Sincronització de la càmera 2. 47

Figura 44. Sincronització de la càmera 3 48

Figura 45. Esquema del sistema de sincronisme de lectura. 49

Figura 46. Esquema de lectura síncrona amb l'ús del *trigger* global. 50

Figura 47. Esquema de lectura síncrona a temps real, és a dir, seguint els *clients* de lectura. 51

Figura 48. Fitxer de configuració de *SCons* que genera diferents executables d'un mateix fitxer font. 53

Figura 49. Modificació realitzada al *wrapper v4l* de *Video For Linux*. 53

Figura 50. Esquema global de dualitat (escriptura/lectura) i compatibilitat (captura/lectura). 58

Sobre el document

1. Estructura del document

Aquest document s'ha estructurat en els següents capítols:

- **Capítol 1: Introducció.** Aquest primer capítol explica breument els objectius del projecte CHIL: *Computers in Human interface Loop*, així com el funcionament bàsic d'una *smart-room*. Un cop situat el context general del projecte s'explica la situació actual de les arquitectures de software presents a la *smart-room* de la UPC i s'exposen els objectius concrets d'aquest projecte.
- **Capítol 2: Característiques de l'entorn del projecte.** En aquest capítol es descriuen amb més detall els aspectes pràctics que determinen el sistema desenvolupat. Per tal de poder entendre millor com funciona aquest sistema, cal conèixer amb més detall les característiques físiques i tècniques de la *smart-room* de la UPC, així com el funcionament bàsic de la tecnologia de *Smart Flow*, al voltant de la qual girarà tot el projecte.
- **Capítol 3: Descripció del nou sistema *off-line*.** En aquest capítol es presentarà el nou sistema *off-line* i s'explicarà el seu funcionament. Tot i que aquest capítol no és un manual d'usuari, al seu final el lector ha de ser capaç de treballar amb el nou sistema sense masses dificultats.
- **Capítol 4: Desenvolupament i implementació del sistema.** En aquest capítol es mostren els detalls més rellevants de la implementació del nou sistema *off-line*. Per facilitar-ne la lectura el lector trobarà molts esquemes descriptius i poc codi. El lector més interessat per la codificació del sistema pot trobar les parts més destacades del codi desenvolupat en els diferents annexos. En aquest capítol s'explicaran entre d'altres detalls:
    - Les característiques de la implementació de *Smart Flow* que més han marcat el projecte.
    - L'arquitectura de software desenvolupada i la solució als diferents problemes trobats..
    - La informació completa del sistema de sincronisme global.
- **Capítol 5: Resultats.** En aquest capítol es mostren els resultats dels tests del sistema global realitzats i es repassen els objectius plantejats a l'inici.
- **Capítol 6: Valoracions i millores.** Per últim, en aquest capítol l'autor fa una valoració del nou sistema i de la feina feta durant el seu desenvolupament.

Tot seguit el lector també trobarà diferents annexos amb exemples, documentació del software desenvolupat i informació ampliada.

2. Referències

Les referències utilitzades en aquest projecte es llisten al final del document. Al llarg del text s'utilitza la notació [número de referència] per indicar el lector on s'ha consultat aquesta informació o bé on pot trobar-ne més.

Degut a què la majoria de documentació utilitzada ha estat en línia o en format electrònic, i per tal de facilitar-ne l'accés al lector, s'han seguit les recomanacions publicades per la Bibliotècnica de la UPC per documents electrònics i audiovisuals.

3. Idioma

Aquest projecte està redactat en català. No obstant, tot el software generat en el desenvolupament del projecte, així com la seva documentació tècnica, s'ha desenvolupat íntegrament en anglès.

Per una millor harmonia entre aquest redactat i el sistema desenvolupat s'ha decidit no traduir aquells termes anglesos que fan referència als aspectes més tècnics, ni tampoc aquells que són àmpliament utilitzats en el context del projecte. En aquests casos s'utilitzarà el format *italic* o cursiu.

Tot seguit es llisten aquests termes i d'altres i se'n fa un breu descripció de referència..

4. Definicions i abreviatures

Terme	Descripció
API	*Application User Interface*
background	En software fa referència a un procés que s'executa en segon pla.
background	En vídeo fa referència al fons estàtic de les imatges.
bug	En software fa referència a un error en el codi font.
C/C++ language	Llenguatges de programació. C++ és un super-conjunt de C que permet *OOP*..
CHIL	*Computers in Human Interface Loop.* Projecte europeu descrit en el document.
clipping	En vídeo s'entén com al procés de retallar temporalment una seqüència.
CORBA	*Common Object Request Broker Architecture*. Arquitectura de programació distribuïda.
daemon	Concepte de software. Procés que s'executa en segon pla.
debug	Concepte de software. Depuració i correcció d'errors en una aplicació.
driver	En software fa referència a les aplicacions que controlen als dispositius físics.
dropped-frame	En adquisició de vídeo fa referència a una imatge que no s'ha capturat.
file-system	Sistema de fitxers.
flag	En software fa referència a variables que es poden activar o desactivar.
flow	En software fa referència a un flux de dades que es transmet entre aplicacions.
frame	En vídeo fa referència a una imatge d'una seqüència.
frame rate (fps)	Número d'imatges per unitat de temps (*frames per second*).
GUI	*Graphic User Interface*
html	*Hyper Text Markup Language.* Llenguatge àmpliament utilitzat a Internet.
LAN	*Local Area Network.* Xarxa d'abast local.
Linux	Sistema operatiu de codi obert basat en *UNIX*.
listener	Concepte de software. Procés que es manté inactiu esperant events per processar-los.
log	En software fa referència a l'informe que genera una aplicació sobre el seu propi estat.
MOTU	*Mark of the Unicorn*. Fabricant de l'equip de sincronisme *MOTU Digital Timepiece A/V*
middleware	Concepte de software. Capa que connecta el S.O. i la xarxa amb les aplicacions.
NAS	*Network Attached Storage.* Dispositiu d'alta capacitat d'emmagatzematge en xarxa.
NIST	*National Institute of Standards and Technology*
NTP	*Network time protocol.*
OOP	*Object Oriented Programming*
PC	*Personal Computer*
pipes	En software fa referència a estructures que permeten comunicar processos.
PTZ	Càmera *Pan-Tilt-Zoom*
QT	Entorn de desenvolupament d'aplicacions gràfiques desenvolupat per Trolltech.
rack	Mobiliari específicament dissenyat per contenir equips informàtics.
recording	En aquest projecte fa referència al conjunt de fitxers que formen una sola gravació.
Smart Flow	Software de NIST. Permet la creació de *flows* entre aplicacions a alta velocitat.
smart-room	Sala intel·ligent, que respon al que hi passa i presta serveis avançats.
socket	Concepte de software. Interfície de comunicació entre aplicacions.
splitter	Dispositiu que permet distribuir un mateix senyal d'entrada a diferents sortides
SSH	*Secure Shell Protocol.* Protocol que permet l'ús segur de terminals remots.
stack	En software fa referència a la seva organització jeràrquica, modular o funcional.
SVN	*Subversion software.* Software del control multiversió.
switch	Dispositiu que permet seleccionar un senyal de sortida entre diferents entrades.
timestamp	Marca que indica un instant de temps.
trigger	Senyal de dispar que generalment marca l'inici d'una tasca.
UPC	Universitat Politècnica de Catalunya
wiki	Tecnologia *web* que permet editar pagines *web on-line* per varis usuaris.
Windows	Sistema operatiu de Microsoft.
wrapper	En software es refereix a quan una capa de software que encapsula a una altra.
XML	*Extensible Markup Language.* Llenguatge descriptiu de dades.

Taula 1. Definicions i abreviatures.

Capítol 1. Introducció

Aquest projecte tracta sobre el desenvolupament d'un sistema de software per ser utilitzat en la *smart-room* de la UPC. Aquesta *smart-room* s'ha desenvolupat principalment en el marc del projecte europeu CHIL: *Computers in Human Interface Loop*.

En aquest capítol s'expliquen breument els objectius del projecte CHIL i els d'una *smart-room* en general. Finalment donarem un cop d'ull a la situació dels sistemes software existents a la *smart-room* de la UPC i es detallaran els objectius concrets d'aquest projecte.

1.1. Context del projecte

1.1.1. El projecte CHIL: Computers in Human Interface Loop

El projecte CHIL [9] és un projecte integrat (*IP*) en el *Sixth Framework Programme* de la Comissió Europea amb data d'inici al Gener de 2004 i una durada de 40 mesos.

L'objectiu general del projecte CHIL és el de crear entorns i sistemes on les computadores ofereixin els seus serveis d'una manera més "humana". Es busca la millor manera d'aconseguir que les computadores no trenquin els procediments de treball naturals entre persones, sinó que comprenguin aquests procediments i utilitzin aquesta informació per oferir els seus serveis de la millor manera possible.

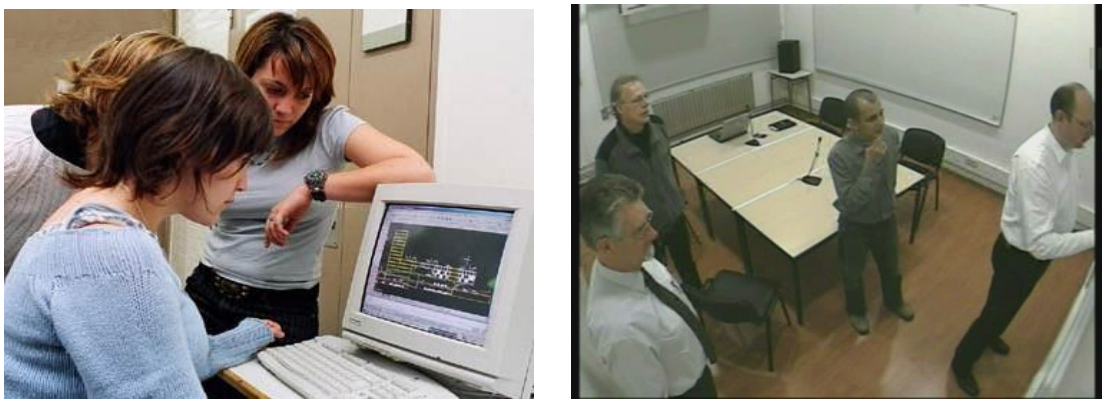


Figura 1. Sessions de treball habituals (esquerra) i amb l'ús de tecnologies CHIL (dreta). (Font: [26])

A la Figura 1 podem veure la típica escena de treball en grup, on la computadora és el centre d'atenció de les persones obligant-los a utilitzar constantment tecles i botons i centrant la vista en un petit monitor. L'objectiu del projecte CHIL és canviar aquesta manera habitual, però poc natural de treballar per la de la dreta, on les persones treballen i es comuniquen d'una manera més natural mentre la computadora queda en un segon pla analitzant la situació i les activitats dels usuaris i oferint els seus serveis de la millor manera possible.

Per tal que això sigui possible CHIL pretén avançar en un seguit de tecnologies d'anàlisi perceptiu i proposa alguns possibles serveis utilitzant-les:

- *Connector*. Servei que pretén ajudar a connectar dues persones distants en el millor moment possible i pel millor mitjà possible basant-se en l'activitat i preocupacions de cada una d'elles.
- *Memory Jog*. Servei que pretén donar informació i recordatoris útils a l'usuari en funció de l'entorn present. Reconeix la gent amb qui s'està en cada moment, l'activitat que s'està fent i mostra la informació més rellevant en cada moment.
- Espais de treball cooperatiu. Servei que pretén ser un assistent per reunions realitzant resums i minutes així com donant informació de trobades prèvies o relacionades.
- *Attention Cockpit*. Assistent que informa sobre l'actitud d'una audiència enfront un orador informant-lo sobre quins temes centren més l'atenció i quin interessen menys.

1.1.2. Les CHIL smart-rooms

En el nostre context definim una *smart-room* com un espai físic limitat que detecta l'activitat que s'hi produeix i presta diferents serveis avançats als seus usuaris en funció d'aquesta activitat.

Al projecte CHIL es defineixen el que s'anomenen com a CHIL *smart-rooms*, plantejant dos possibles escenaris pràctics:

- Aula docent
- Sala de reunions

El projecte CHIL també defineix algunes de les informacions que les CHIL *smart-rooms* han de detectar:

- Nombre de persones en la sala.
- Localització, identitat i orientació de cada una.
- Reconeixement d'accions, activitats i situacions.
- Localització i classificació d'events acústics.
- Detecció, localització i reconeixement de la parla.

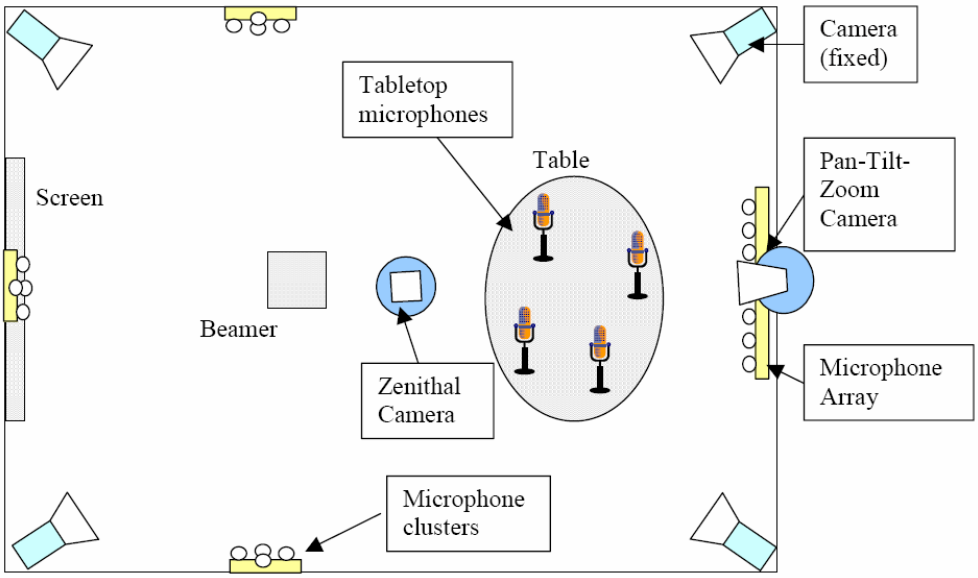


Figura 2. Esquema genèric d'una CHIL *smart-room*. (Font: [7])

Perquè tot això sigui possible primer de tot cal que una *smart-room* monitoritzi l'activitat que s'hi produeix mitjançant diferents sensors audiovisuals. En el context del projecte CHIL s'especifiquen els requeriments bàsics d'aquest equipament [7]:

- Quatre càmeres fixes a cada una de les cantonades de la sala.
- Una càmera activa *pan-tilt-zoom* (*PTZ*) per reconeixement de cares.
- Una càmera panoràmica o zenital al sostre de la sala.
- Un *array* de micròfons.
- Tres clústers de micròfons.

- Quatre micròfons de sobre taula.
- Un micròfon (*close-talking*) per cada participant.

Apart dels sensors audiovisuals, una *smart-room* ha de disposar de l'equipament informàtic i el software necessari per poder transportar, processar i emmagatzemar les dades capturades pels sensors, així com un software o sistema d'alt nivell que ofereixi els serveis als usuaris. El projecte CHIL dona l'arquitectura de sistema de la Figura 3 com a referència.

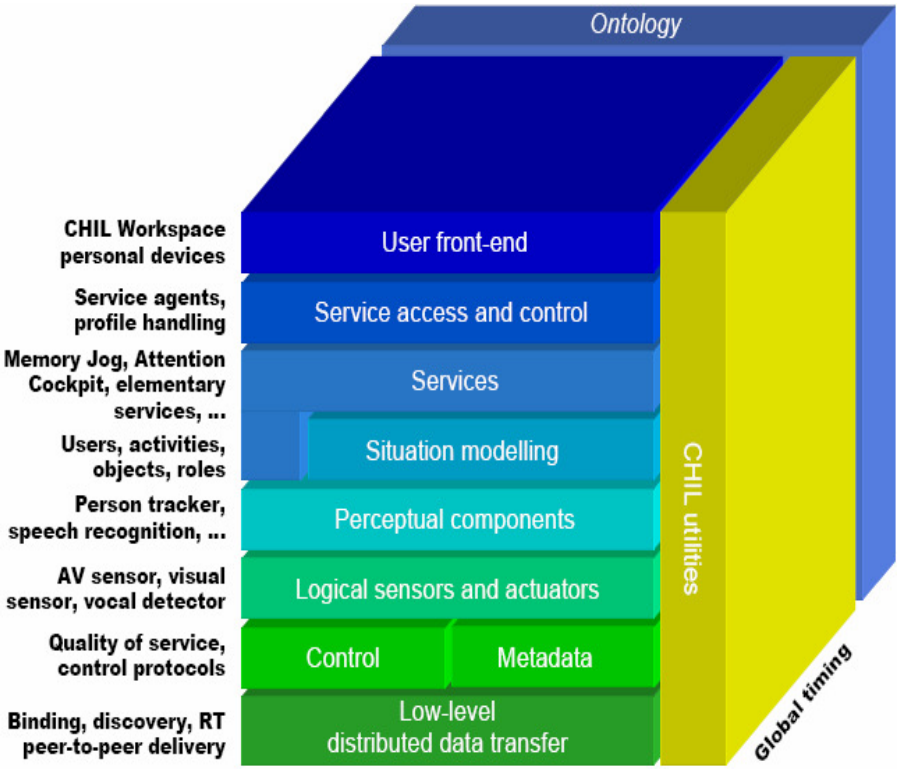


Figura 3. *Stack* de software genèric per una CHIL *smart-room*. (Font: [8])

Amb aquest projecte es pretén donar un servei específic a la ***smart-room* de la UPC**[1] **dins del context del projecte CHIL**, i centrant-nos bàsicament en la **gestió de dades de vídeo**.

1.2. Motivació principal: sistemes incompatibles

Podem definir dos grans sistemes de software a la *smart-room* de la UPC. Cada un amb funcions diferents:

- Sistema ***off-line***

  És el sistema encarregat de generar i emmagatzemar dades audiovisuals capturades a la *smart-room*, així com de la calibració de totes les càmeres.

[1] Més endavant farem una descripció detallada de les especificacions concretes de la *smart-room* de la UPC.

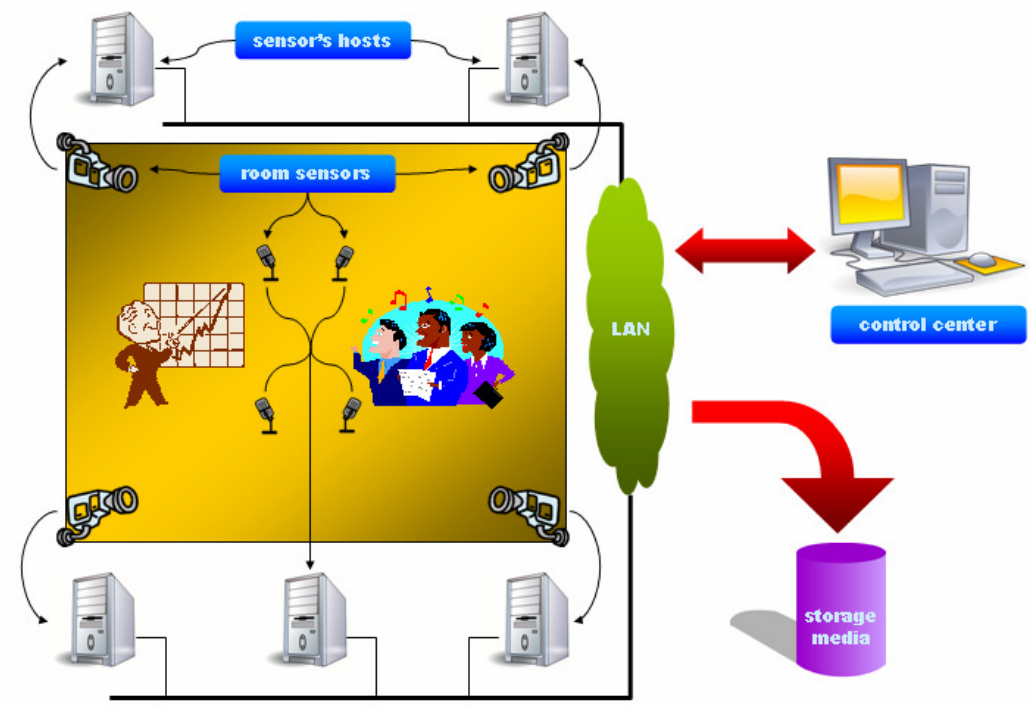


Figura 4. Il·lustració del funcionament bàsic del sistema *off-line*.

L'objectiu és que aquestes dades siguin utilitzades posteriorment com a dades de test per les diferents tecnologies en desenvolupament. És per això que el sistema *off-line* també ha de donar les eines per poder llegir i reproduir aquestes gravacions.

- Sistema ***on-line***.

És l'encarregat de realitzar demostracions públiques i en temps real de les capacitats de la *smart-room*. Es basa en el processat i anàlisis en temps real de les dades capturades, la detecció de l'activitat existent i un sistema de nivell superior que gestiona els diferents serveis de la sala basant-se en aquesta activitat.

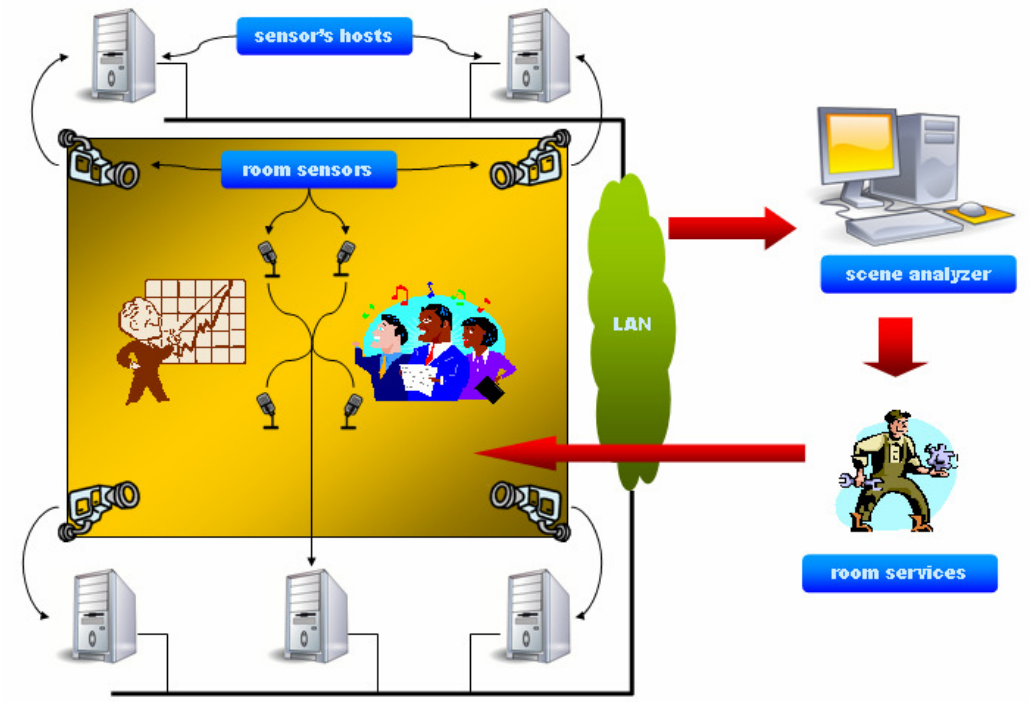


Figura 5. Il·lustració del funcionament bàsic del sistema *on-line*.

Per la part de les **tecnologies de vídeo**, la pròpia evolució del projecte CHIL i de la *smart-room* de la UPC, han desembocat en una implementació de software completament diferent per cada un d'aquests dos sistemes.

Pel que fa el sistema *off-line* tenim al que es pot considerar com el precedent immediat d'aquest projecte, el projecte d'en Jordi Cenzano: **Adquisició de senyal de vídeo multicàmera per a una *smart-room*** [11]. Aquest projecte consistia en un sistema hardware de captura de vídeo síncrona, i un software de control basat en una aplicació per *Windows*[1].

Per la seva banda, el sistema *on-line* està basat en la tecnologia de ***Smart Flow*** proposada per *National Institute of Standards and Technology (NIST)*[2]. Aquesta tecnologia defineix una arquitectura de transport de dades que permet executar aplicacions en diferents computadors d'una mateixa xarxa de tal manera que poden intercanviar grans quantitats de dades a alta velocitat. *Smart Flow* funciona sobre Linux i elimina la necessitat d'implementar un sistema client/servidor.

Per tant, en el moment d'iniciar aquest projecte existeixen dues arquitectures de software de processat de vídeo incompatibles a la *smart-room* de la UPC:

- Sistema *off-line*: Funciona sobre ***Windows***. Està basat en una arquitectura client/servidor i comunicació per *sockets*.
- Sistema *on-line*: Funciona sobre **Linux**. Està basat en la tecnologia de ***Smart Flow***.

És d'aquí, d'aquesta dualitat incompatible que neix la motivació inicial i principal d'aquest projecte: **homogeneïtzar els dos sistemes presents a la *smart-room* de la UPC desenvolupant un nou sistema *off-line*, compatible amb l'actual sistema *on-line***.

1.3. Objectius del projecte

Més enllà de la motivació i l'objectiu principal del projecte plantejats, es fixen també els següents objectius i es deixen oberts alguns altres en funció de l'evolució del propi projecte i del seu context.

(a) Estudi de la viabilitat del sistema off-line sobre Linux / Smart Flow

Com hem comentat anteriorment, la motivació inicial d'aquest projecte és la fusió dels dos entorns de software presents a la *smart-room*: els sistemes *off-line* i *on-line*. Lògicament cada sistema té els seus requeriments tècnics específics que determinen l'ús d'una arquitectura de software determinada o una altra.

Els requeriments més importants per tal que les gravacions realitzades amb el nou sistema *off-line* siguin de la major qualitat possible són:

- Nombre d'imatges perdudes, o *dropped-frames* mínim.
- Assegurar la sincronització de les diferents càmeres[3].

Al sistema *on-line*, el nombre de *dropped-frames* no és tan important ja que el propi requeriment processat en temps real obliga sovint a ignorar algunes imatges.

Per tan, com a primer objectiu del projecte tenim el d'estudiar la viabilitat de la fusió dels dos entorns de software en un de sol. Cal estudiar el comportament de l'actual arquitectura *on-line* pel que fa als requeriments dels serveis *off-line*. Concretament cal **estudiar l'arquitectura de *Smart Flow* per** tal de

[1] El fet que els resultats d'aquest projecte s'hagin estat utilitzant durant dos anys a la *smart-room* de la UPC, ha fet que aquest precedent hagi estat de gran utilitat a l'hora de definir les funcionalitats desitjades, també, pel nou projecte.
[2] En capítols posteriors s'explicarà la tecnologia de *Smart Flow* àmpliament.
[3] En l'apartat 4.5 s'explicaran els diferents problemes de sincronisme del sistema. Els *dropped-frames* en són també part.

comprovar que es pot **garantir el sincronisme entre les càmeres**, així com per comprovar que el **número de *dropped-frames* és mínim**. En cas contrari caldrà buscar-hi les corresponents solucions.

(b) Estandardització de les dades generades per l'entorn off-line.

Una de les activitats relacionades amb el projecte CHIL des del seu inicis l'any 2004 és la producció de dades de diferents *smart-rooms* pel seu posterior ús com a dades de test per diferents tecnologies. Aquestes dades consten de fitxers d'àudio i vídeo sincronitzats, així com de fitxers descriptors i de meta-dades[1].

Al llarg del projecte CHIL s'ha anat definint una norma comuna entre els seus participants pel format d'aquestes gravacions [6]. Tot i així, les pròpies dimensions del projecte CHIL han portat actualment a tenir gravacions en diferents formats.

Tenint en compte que la generació d'aquestes gravacions és responsabilitat de l'entorn *off-line*, un altre dels objectius principals del projecte serà el de **definir un format estàndard per les futures gravacions** realitzades a la *smart-room* de la UPC, més enllà del projecte CHIL.

Paral·lelament també es volen **desenvolupar les eines que permetin actualitzar al nou format totes les dades recollides al llarg del projecte CHIL**.

(c) Implementació d'un sistema dual i compatible

Dins del sistema *off-line* distingim tres grans funcionalitats:

- Realització de gravacions. Controlat pel sistema d'escriptura**.**
- Reproducció de gravacions. Controlat pel sistema de lectura.
- Calibració de càmeres.

Una de les funcionalitats més importants que es volen afegir al nou sistema *off-line* és que sigui un sistema **dual: escriptura** / **lectura**. El sistema *off-line* ha de permetre, no només generar gravacions, sinó també ha de proporcionar les eines necessàries per poder reproduir-les posteriorment.

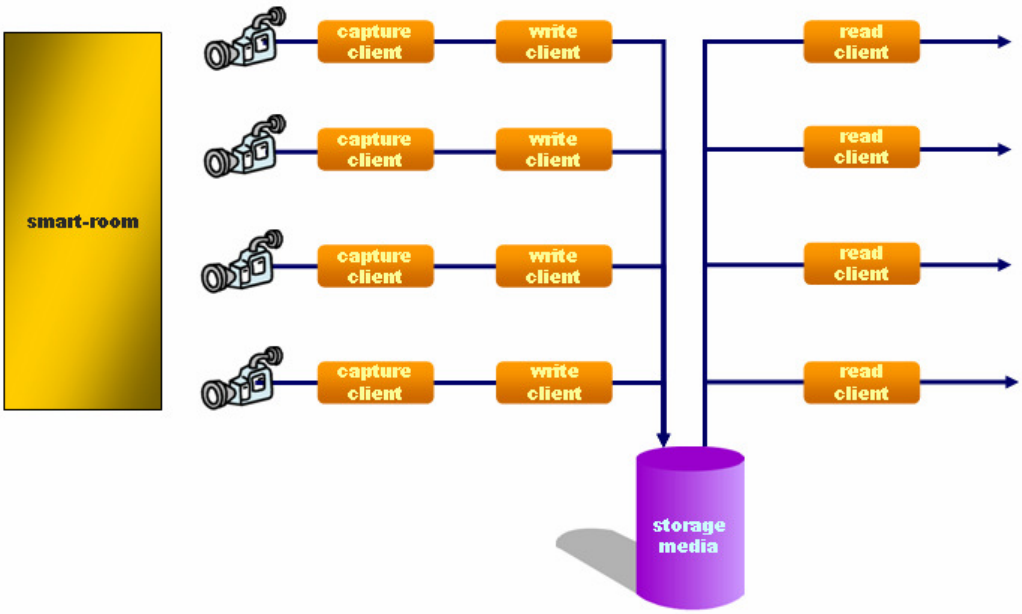


Figura 6. Esquema funcional dels sistemes d'escriptura i lectura.

[1] En el context del projecte CHIL s'anomena *seminari* a cada conjunt d'aquests fitxers, mentre que en aquest nostre projecte utilitzarem generalment el terme *recording* o *gravació* per fer-ne referència.

Per altra banda l'entorn *on-line* sempre processa les dades capturades en temps real. La Figura 7 il·lustra com les dades capturades s'analitzen en diferents etapes de processat audiovisual i finalment s'envien els resultats al control de serveis de la sala per tal que aquesta reaccioni.

Això implica que en el moment d'iniciar aquest projecte, per poder provar el correcte funcionament de les aplicacions d'anàlisi i serveis *on-line* és sempre necessària la participació de persones, o usuaris, a la *smart-room*.

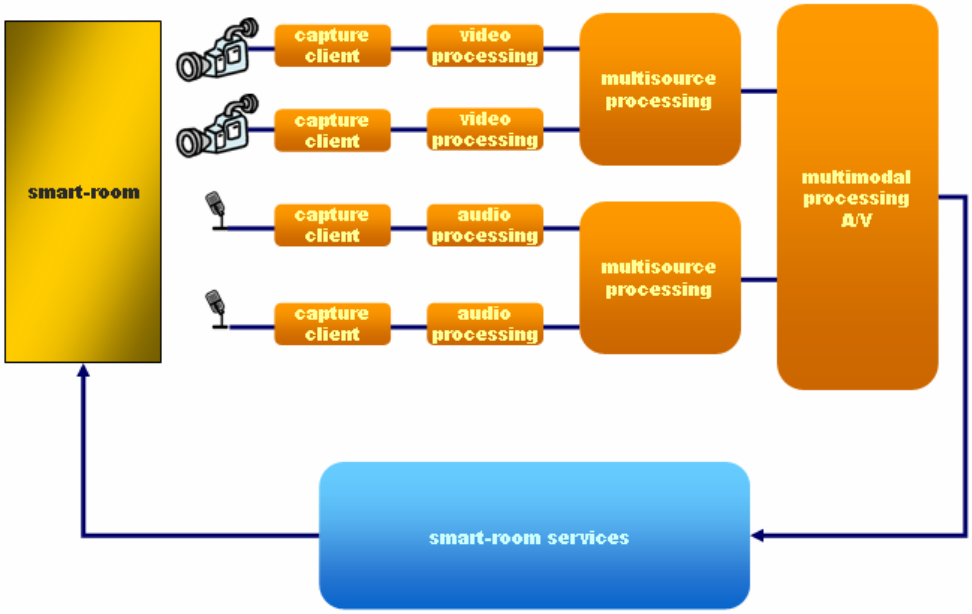


Figura 7. Esquema funcional del sistema *on-line*.

És per això que, si s'ajunten l'objectiu de dualitat amb l'objectiu general d'unificar i fer compatibles els entorns *on-line* i *off-line* obtenim com a resultat, i alhora també com a nou objectiu, la **compatibilitat entre els sistemes de captura i lectura**.

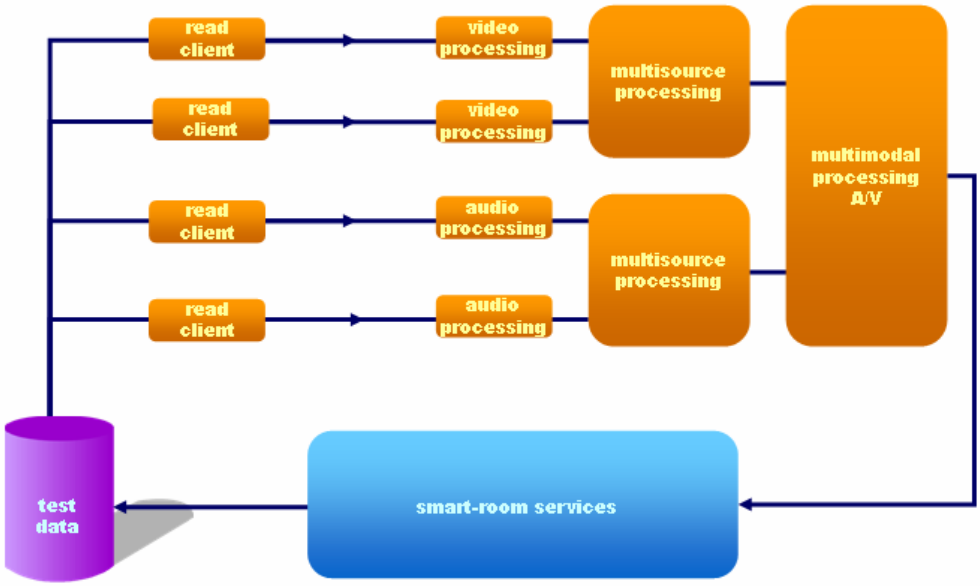


Figura 8. Esquema del sistema on-line utilitzant un sistema de lectura enlloc del de captura.

Comparant la Figura 7 amb la Figura 8 veiem que aquesta compatibilitat ha de permetre provar les aplicacions de demostració i presentació amb dades de test gravades prèviament, enlloc de fer-ho obligatòriament en temps real.

(d) Base de dades de meta-dades

Com s'ha comentat anteriorment una de les activitats relacionades amb el projecte CHIL és la producció de dades pel seu posterior ús com a dades de test de noves tecnologies. A mesura que el nombre de seminaris i altres gravacions s'ha anat incrementant s'ha detectat la necessitat de disposar d'una base de dades on s'hi llistin totes les gravacions disponibles així com les meta-dades que les defineixen.

Tot i que en el moment d'iniciar el projecte aquesta base de dades no està dissenyada, forma part dels objectius d'aquest projecte el d'**incloure aquesta base de dades** com a interfície de control per emmagatzemar noves gravacions, així com també per seleccionar i reproduir gravacions prèviament guardades.

(e) Interfície gràfica integrada

Com en tot sistema de gestió resulta fonamental l'existència d'una interfície que permeti realitzar totes les operacions disponibles.

És per això que, com a objectiu final del projecte, es vol desenvolupar una interfície gràfica integrada que gestioni el nou sistema *off-line*. **L'objectiu és disposar d'una interfície gràfica** amb les funcionalitats de l'actual aplicació per *Windows* i que sigui **compatible amb *Smart Flow*.**

Capítol 2. Característiques de l'entorn del projecte

Els objectius d'aquest projecte són clarament pràctics: s'han de desenvolupar unes eines de software que formin un nou sistema *off-line* per funcionar a la *smart-room* de la UPC

Justament aquesta naturalesa pràctica fa que l'entorn en què s'ha desenvolupat aquest projecte hagi estat determinant. És per això, i per entendre millor el seu funcionament, que abans de presentar el nou sistema cal donar un cop d'ull més detallat a la *smart-room* de la UPC, al seu hardware i al software en què es basa el nou sistema: *Smart Flow*.

2.1. La *smart-room* de la UPC

La *smart-room* de la UPC està ubicada al mòdul D5-006 del Campus Nord de la UPC. La sala està unida al laboratori de televisió mitjançant una paret plegable tal i com es mostra a la Figura 9.

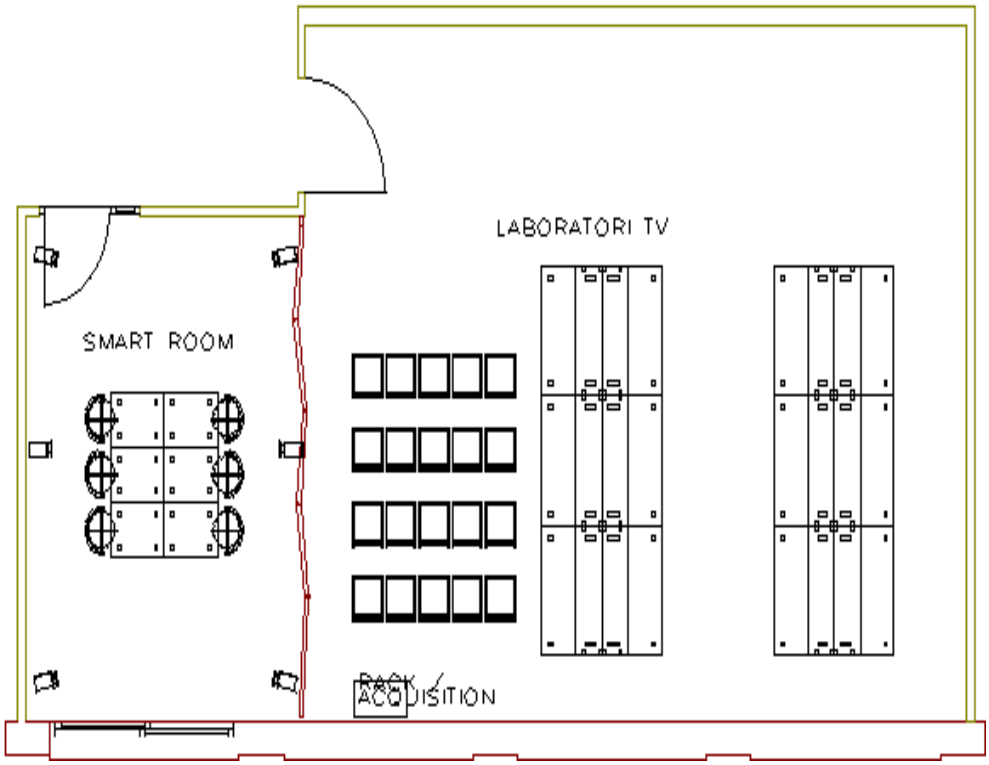


Figura 9. Il·lustració de la smart-room de la UPC i el laboratori de televisió (Font: [27])

Com hem comentat en el capítol anterior, per tal que una *smart-room* funcioni correctament calen un conjunt de sensors que capturin dades audiovisuals de la sala, diferents equips de computació, sincronisme i emmagatzematge de dades, i una arquitectura de software que ho gestioni.

2.1.1. Sensors

Com es mostra a la Figura 10, la sala disposa dels següents equips sensors:

- Un *array* de 68 micròfons *MkIII*.
- Tres clústers de quatre micròfons *Hammer* cada un.
- Sis càmeres JVC TK-C1481BEG [11] enfocant al centre de la sala.
- Una càmera zenital al centre de la sala.
- Una càmera *PTZ* que enfoca exclusivament a la porta d'entrada.

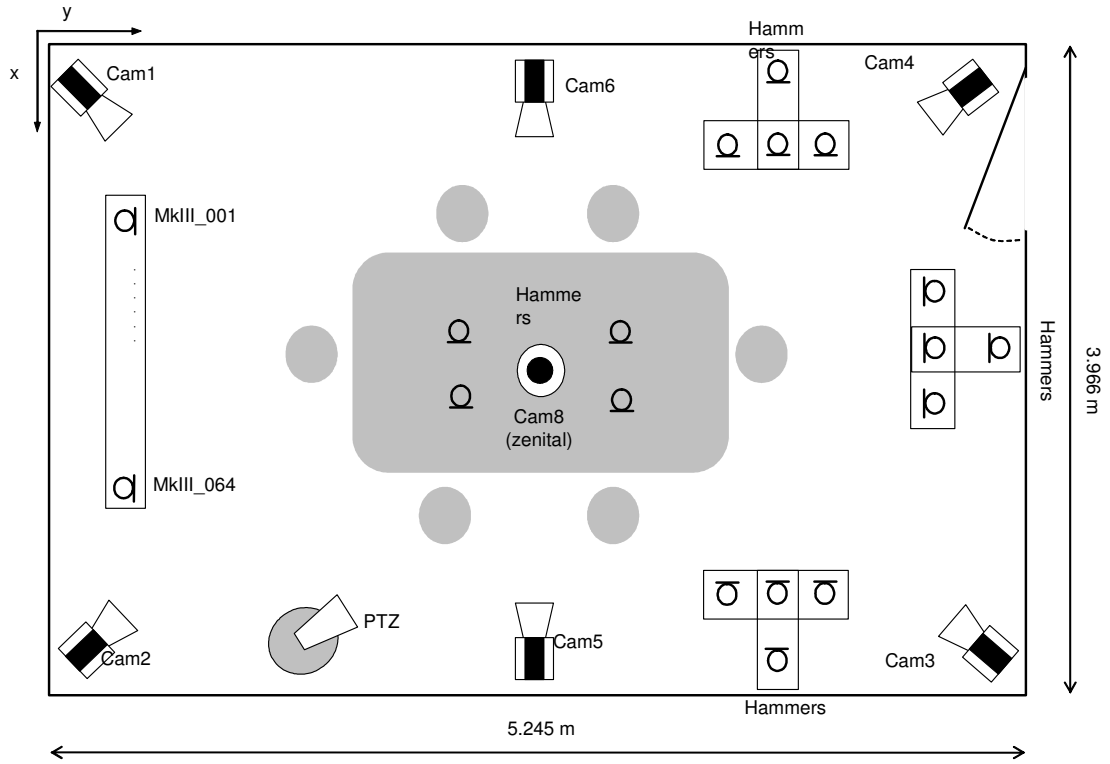


Figura 10. Il·lustració de la *smart-room* de la UPC amb els seus sensors.

2.1.2. Equipament i configuració

La Figura 9 mostra que la *smart-room* disposa de dos *racks* ubicats a la sala contigua (laboratori de televisió). En aquests *racks* hi podem trobar el següent equipament:

- *PCs* amb capturadores d'àudio i vídeo (*Osprey 210* de *ViewCast Corporation* [24]).
- Diferents *splitters* i *switches*.
- Equips de sincronisme: *MOTU Digital Timepiece.*
- Equips d'emmagatzematge d'informació: *NAS* [25]
- Consoles per controlar els diferents equips[1].

Tots els *PCs* funcionen amb *Linux* com a sistema operatiu[2], excepte un. Aquesta excepció es deu, entre d'altres motius, a la necessitat de mantenir *Windows* en una de les màquines per poder utilitzar l'actual

[1] Només hi ha dos terminals físics, un a l'interior de la sala i l'altre l'exterior. Tots els equips que disposen de port de consola són accessibles mitjançant un *terminal-switch.* Per tal que hi pugui accedir més gent simultàniament, es disposa d'un software de terminal virtual anomenat *NoMachine* [20] que s'ha utilitzat abastament en el desenvolupament del projecte.
[2] Tot i que no tots tenen la mateixa versió, la gran majoria utilitza Kubuntu [29].

sistema d'adquisició de vídeo. Un dels objectius de la sala a nivell informàtic mantenir la compatibilitat i uniformitat entre sistemes operatius, utilitzant codi obert sempre que sigui possible.

Els *PCs* amb capturadora disposen d'una partició de disc en format *ext2* on s'ho escriuen les dades capturades.

Tots els *PCs* juntament amb el *NAS* estan organitzats en una *LAN*. Com s'explicarà més endavant aquesta xarxa ha de complir amb els següents requisits:

- Sincronització[1]: El rellotge intern de cada *PC* ha d'estar sincronitzat amb la resta. Per fer-ho s'utilitza el protocol ***NTP*** ([12], [13]).
- Administració d'usuaris: Els usuaris de *Smart Flow* han d'estar registrats a tots els *PCs* amb el mateix identificador d'usuari de Linux.
- Protocol ***SSH***: Els usuaris de *Smart Flow* han d'estar autenticats pels servidors *SSH* de la xarxa [14].

Com a política d'administració s'ha configurat un únic usuari per treballar amb l'entorn *off-line*, l'**usuari *recording***. Aquest usuari compleix amb tots els requisits necessaris per treballar amb *Smart Flow* i el nou sistema *off-line*.

2.2. Què és i per què utilitzem *Smart Flow*?

En una *smart-room* cal gestionar les dades audiovisuals capturades pels sensors i enviant-les cap als equips de computació on seran processades en temps real. Per això cal crear una arquitectura de software que permeti transportar grans quantitats de dades entre els equips d'adquisició i els equips de computació per fer-les arribar als diferents algoritmes que les analitzaran. També cal que aquests algoritmes puguin rebre i/o enviar els seus resultats cap a d'altres algoritmes i/o d'altres equips de computació.

És amb aquest objectiu que el *National Institute of Standards and Technology* dels Estats Units (*NIST)* ha desenvolupat *Smart Flow* [3]. Aquesta tecnologia defineix una arquitectura de transport de dades basada en *Linux* que permet executar diferents aplicacions en diferents computadors d'una mateixa xarxa de tal manera que poden intercanviar grans quantitats de dades estructurades entre elles.

Aquestes aplicacions reben el nom de *Smart Flow Clients*, els computadors que formen la xarxa s'anomenen *Smart Flow Servers*, i les estructures de dades que s'intercanvien *Smart Flow Flows*. No obstant, per comoditat, ens hi referirem respectivament com a *clients, servers* i *flows*.

Tot i que aquesta nomenclatura pot portar a creure el contrari, *Smart Flow* elimina la necessitat d'implementar una arquitectura client/servidor, permetent concentrar tots els esforços en el desenvolupament de l'algorísmica dels *clients*, que són els que processen les dades audiovisuals.

La imatge de la Figura 11 mostra el mecanisme bàsic de *Smart Flow*. Hi podem veure com cada aplicació (*client*) està unida a una computadora (*server*) i envia i/o rep dades d'altres *clients* mitjançant *flows*.

Smart Flow permet gestionar aquesta arquitectura mitjançant una interfície gràfica. Aquesta interfície permet configurar quins *clients* s'executen i en quins *servers*, així com també permet connectar els diferents *flows* de cada *client* entre si.

[1] En l'apartat 4.5 s'explica el sistema global de sincronisme. Veurem que el sincronisme de la xarxa és especialment important.

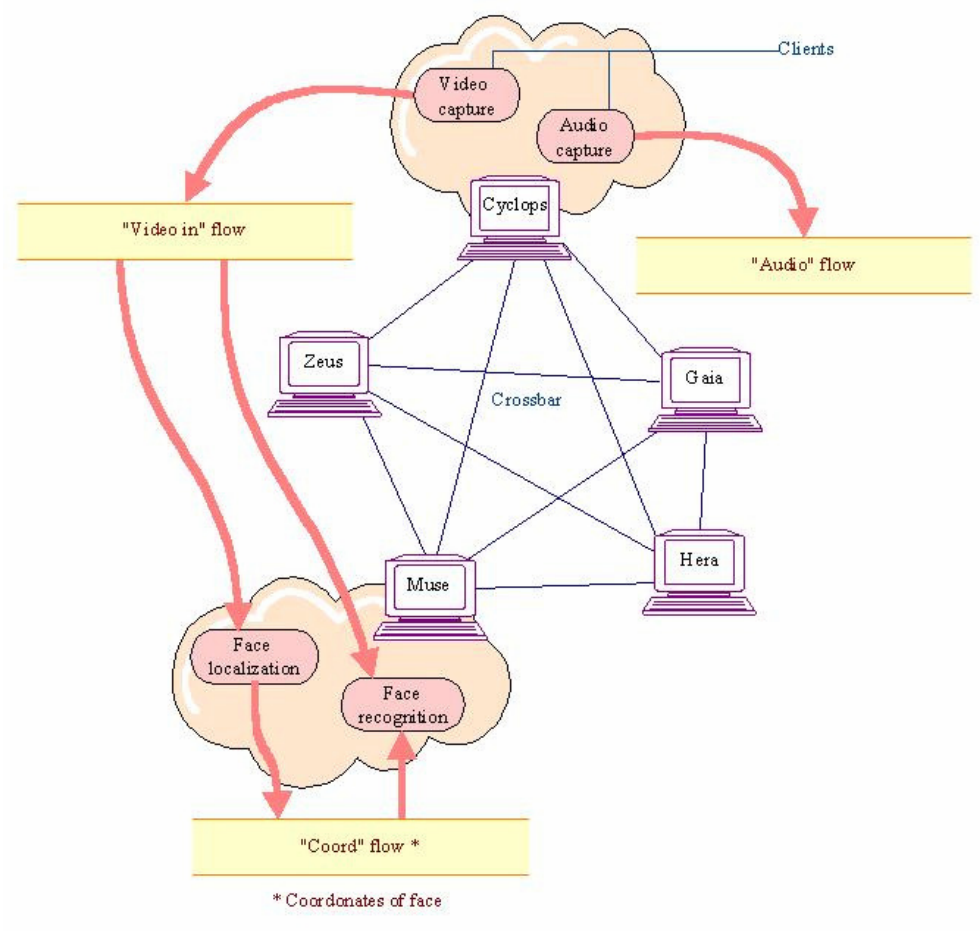


Figura 11. Il·lustració del funcionament bàsic de *Smart Flow*. (Font: [4])

En la Figura 12 s'observen les principals finestres de configuració de *Smart Flow*:

(1) Menú principal de l'aplicació. Permet carregar fitxers de configuració i mapes.
(2) Finestra de *logs*. Monitoritza la sortida estàndard de cada un dels *clients*.
(3) Llista de *clients* i *servers* disponibles. Aquí es poden afegir els nous clients o nous *PCs* a la xarxa.
(4) Mapa. És la principal interfície gràfica de configuració de *Smart Flow*. Permet escollir quins *clients* s'executen, en quin *server* i com es connecten els *flows*. En la imatge s'hi distingeixen:
   a. *Clients*: Representats per quadrats. Són de color groc si estan aturats i blau si s'estan executant. La informació del servidor que tenen assignat es pot veure a sota cada *client*.
   b. *Flows*: Representats per línies blaves que van d'un client a d'altres. Els *flows* actius es mostren en un blau més clar.
(5) Visor de *flows*. Si el contingut d'un *flow* és vídeo *Smart Flow* permet veure'n el contingut en aquest visor.

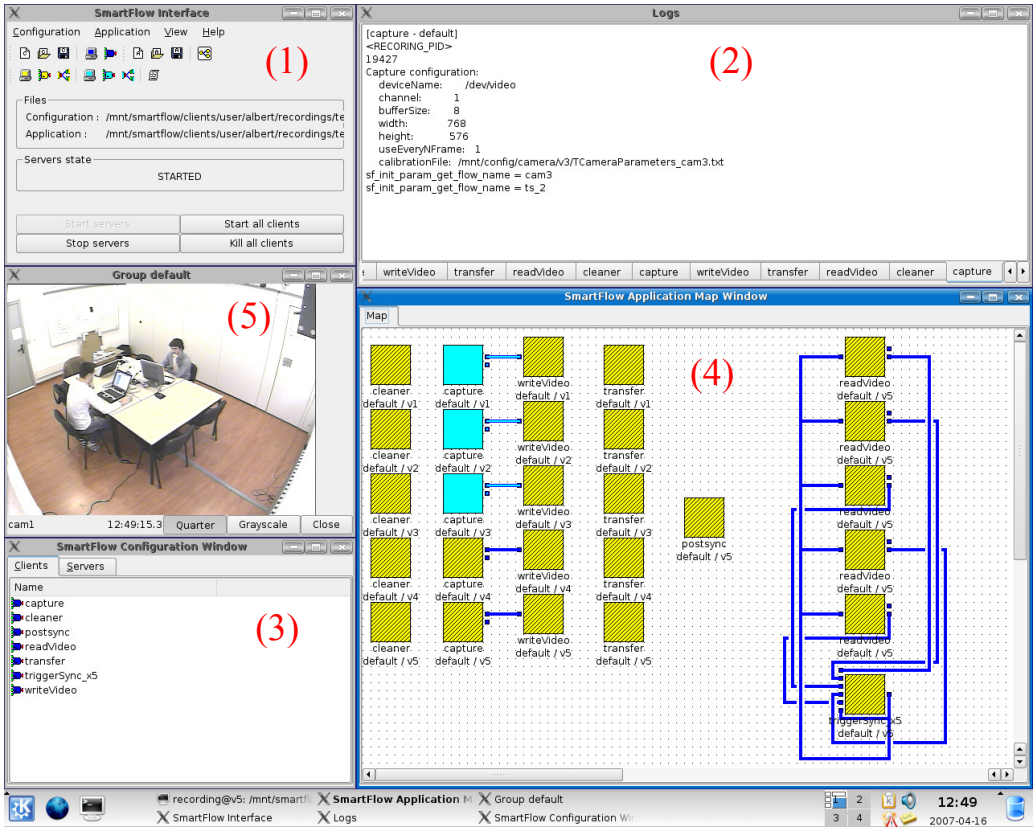


Figura 12. Captura de pantalla de la interfície gràfica de *Smart Flow*.

Smart Flow és un sistema públic, de codi obert i escrit en C. La segona versió del sistema s'ha publicat al febrer del 2007.

Gràcies a la llicència pública de *Smart Flow*, la UPC ha pogut encapsular la seva *API* nativa en C en una jerarquia d'objectes C++ anomenada **UPC-*Smart Flow***. Aquesta encapsulació, o *wrapper,* fa més fàcil la programació dels *clients*, defineix i estructura els tipus de *flows* permesos, així com també obliga a seguir unes polítiques a tots els desenvolupadors de *clients*[1]. És per aquest motiu que es continua utilitzant encara la primera versió de *Smart Flow*.

[1] Com veurem en l'apartat 4.2 l'ús de la llibreria UPC-*Smart Flow* retalla en certa manera la flexibilitat de *l'API* nativa en C de *NIST*. Això fa que en alguns casos molt especials sigui necessària la utilització d'aquesta *API* nativa.

Capítol 3. Descripció del nou sistema *off-line*

El nou sistema *off-line* no es basa en una sola i complexa aplicació, sinó en varies aplicacions petites que implementen diferents funcionalitats. En el context del projecte ens referim constantment a aquestes aplicacions com a *clients*.

Si bé aquests *clients* poden executar-se individualment com qualsevol altra aplicació, per tal de treure'n tot el profit cal que s'executin distribuïdament a la xarxa de la *smart-room* i amb una configuració comuna determinada.

Per configurar-los i executar-los correcta i còmodament s'ha creat una interfície gràfica específica anomenada *Recording-GUI*. Aquesta interfície permet realitzar totes les tasques del nou sistema *off-line* de manera còmoda i senzilla.

Per altra banda, gràcies a què aquests mateixos *clients* també poden executar-se com a *clients* de *Smart Flow,* és possible realitzar les mateixes tasques que fa la *Recording-GUI* des de la pròpia interfície gràfica de *Smart Flow*. Aquesta compatibilitat també permet utilitzar aquests *clients* per altres funcionalitats o en altres sistemes desenvolupats sobre *Smart Flow*.

En aquest apartat s'expliquen les funcionalitats del nou sistema *off-line*, els *clients* que el composen i com s'integren en la *Recording-GUI*. També s'exposa breument la manera de configurar aquests *clients* individualment i com utilitzar-los des de *Smart Flow*[1].

3.1. Les parts del nou sistema *off-line*

De manera abstracta podem dir que el nou sistema *off-line* consta dels subsistemes que es mostren en la Figura 13.

La implementació d'aquests sistemes ha donat com a resultat un conjunt de *clients*. L'estructura bàsica del nou sistema *off-line* es basa en els següents *clients*:

- *capture* : Captura imatges i assigna *timestamps*[2].
- *writeVideo* : Guarda (escriu) imatges al disc.
- *readVideo* : Llegeix imatges prèviament guardades a disc.
- *processRecording* : Post processa les imatges guardades a disc[3].
- *triggerSync* : S'utilitza per garantir la lectura síncrona de disc.
- *showRoom*[4] : Permet visualitzar l'exactitud d'una calibració.

[1] Si el lector vol utilitzar la *Recording-GUI* per treballar amb el nou sistema *off-line* aquest capítol li aporta la informació introductòria necessària, però si vol utilitzar *Smart Flow* cal que el lector estigui familiaritzat prèviament amb la seva interfície [3].
[2] A l'apartat 4.5 s'expliquen els detalls sobre el sincronisme del sistema i l'assignació de *timestamps*.
[3] Com es veurà en l'apartat 4.5 l'objectiu principal d'aquest post processat és l'alineament temporal de les imatges i garantir-ne la sincronització.
[4] Aquest client ha estat desenvolupat pel grup CHIL-UPC amb el suport especial d'en Joachim Neumann.

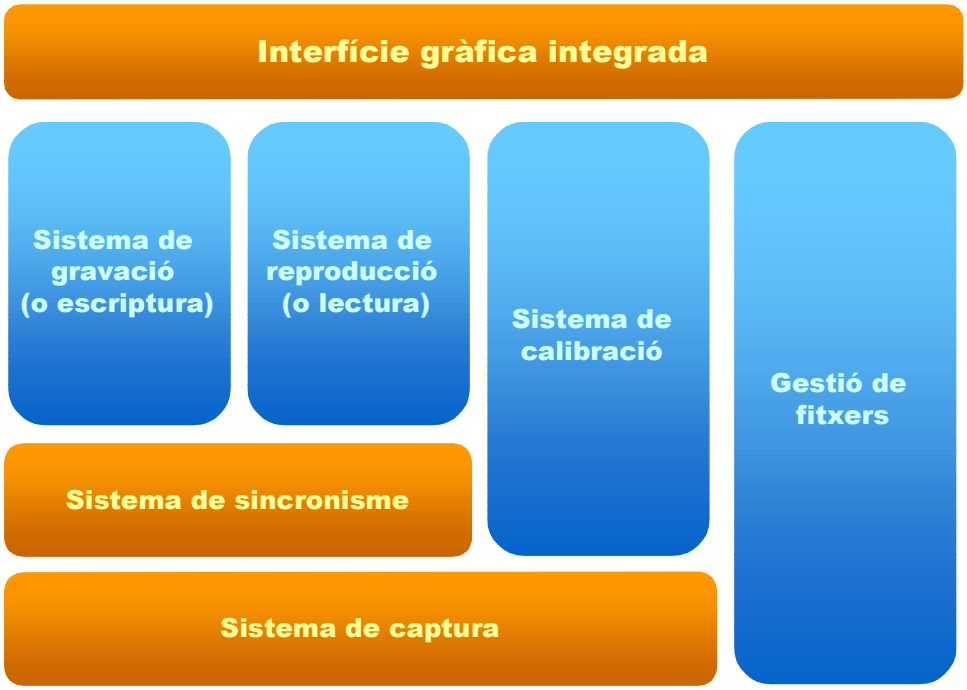


Figura 13. *Stack* funcional de software del nou sistema *off-line*.

Per tal de completar les funcionalitats desitjades i d'automatitzar diferents tasques s'han desenvolupat també les eines següents:

- *cleanRecording* : *Client* que elimina gravacions temporals.
- *transferRecording* : *Client* que transfereix gravacions guardades localment al seu repositori.
- *Recording Update* : Sistema d'actualització de gravacions al nou format.
- *Data base link* : Mòdul que permet afegir dades de noves gravacions a la base de dades.
- *Smart Flow files creator*: Mòdul que automatitza la configuració de *Smart Flow*.

Per gestionar tots aquests *clients* és possible utilitzar dues interfícies gràfiques diferents:

- La *Recording-GUI* : Interfície gràfica senzilla i específica del nou sistema off-line.
- La *GUI* de *Smart Flow* : Interfície genèrica que permet major flexibilitat, afegint complexitat.

L'eina de creació de fitxers de configuració de Smart Flow nomenada (*Smart Flow files creator*) s'ha desenvolupat justament per automatitzar la configuració de *Smart Flow* i reduir-ne la complexitat. Aquesta eina genera mapes i fitxers de configuració de *Smart Flow* com el que es mostra a la Figura 14. També genera automàticament fitxers de configuració vàlids per tots els *clients* de nou sistema *off-line*.

En els següents apartats s'expliquen el funcionament i configuració d'aquests *clients* i la manera d'utilitzar les diferents interfícies gràfiques.

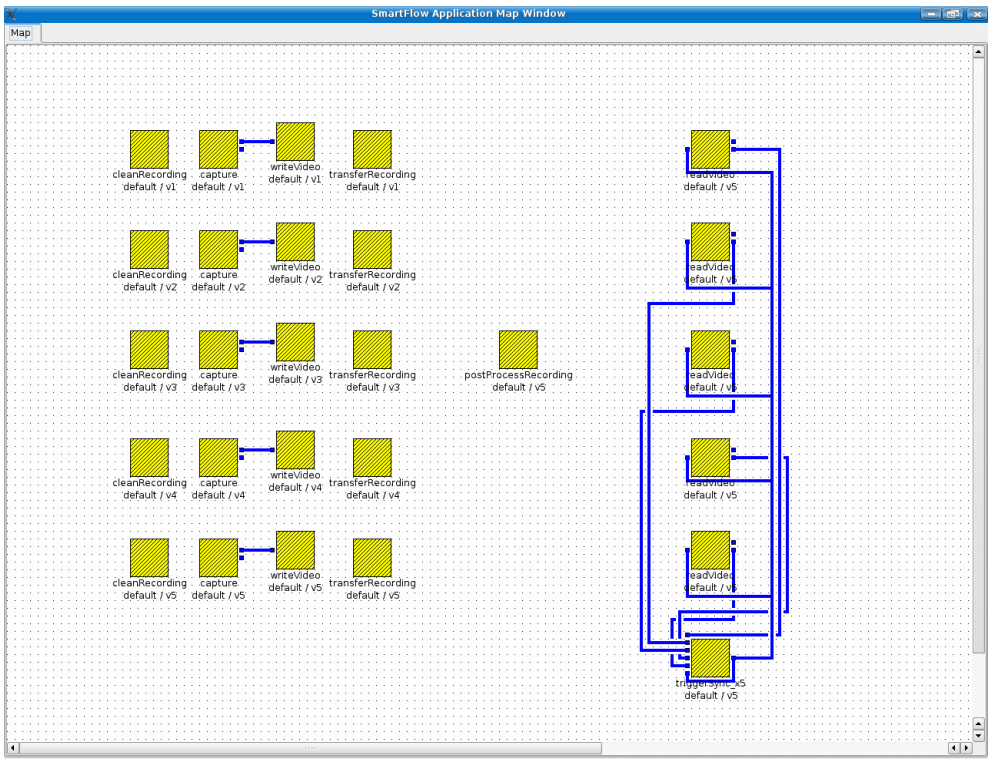


Figura 14. Mapa de *Smart Flow* creat automàticament.per treballar amb el sistema *off-line*.

3.2. Funcionament de la *Recording-GUI*

La *Recording-GUI* és la interfície gràfica integrada i específica del nou sistema *off-line*. El seu objectiu és automatitzar i simplificar al màxim la realització de les tasques corresponents al sistema *off-line*. Les seves funcionalitats principals són:

- Automatitzar el sistema de calibració de les càmeres.
- Realitzar i emmagatzemar gravacions multicàmera i sincronitzades.
- Afegir les meta-dades de les gravacions a la base de dades.
- Comprovar el sincronisme de la xarxa de la *smart-room*[1]
- Comprovar l'estat de les calibracions de les càmeres.
- Crear fitxers de configuració de *Smart Flow*.
- Controlar l'execució de *clients* de *Smart Flow*.

La *Recording-GUI* està organitzada en només tres pestanyes:

- *Main* : Pestanya principal on es controla l'execució dels *clients* i es visualitzen els resultats.
- *Configuration* : Pestanya on es configuren les càmeres que es volen utilitzar.
- *Advanced* : Pestanya de configuració avançada.

[1] Com veurem a l'apartat 4.5 per tal que el nou sistema funcioni correctament és necessari que els rellotges interns dels *PCs* connectats a les càmeres estiguin sincronitzats, per això s'utilitzarà *NTP*. La *Recording-GUI* permet monitoritzar l'estat de *NTP*.

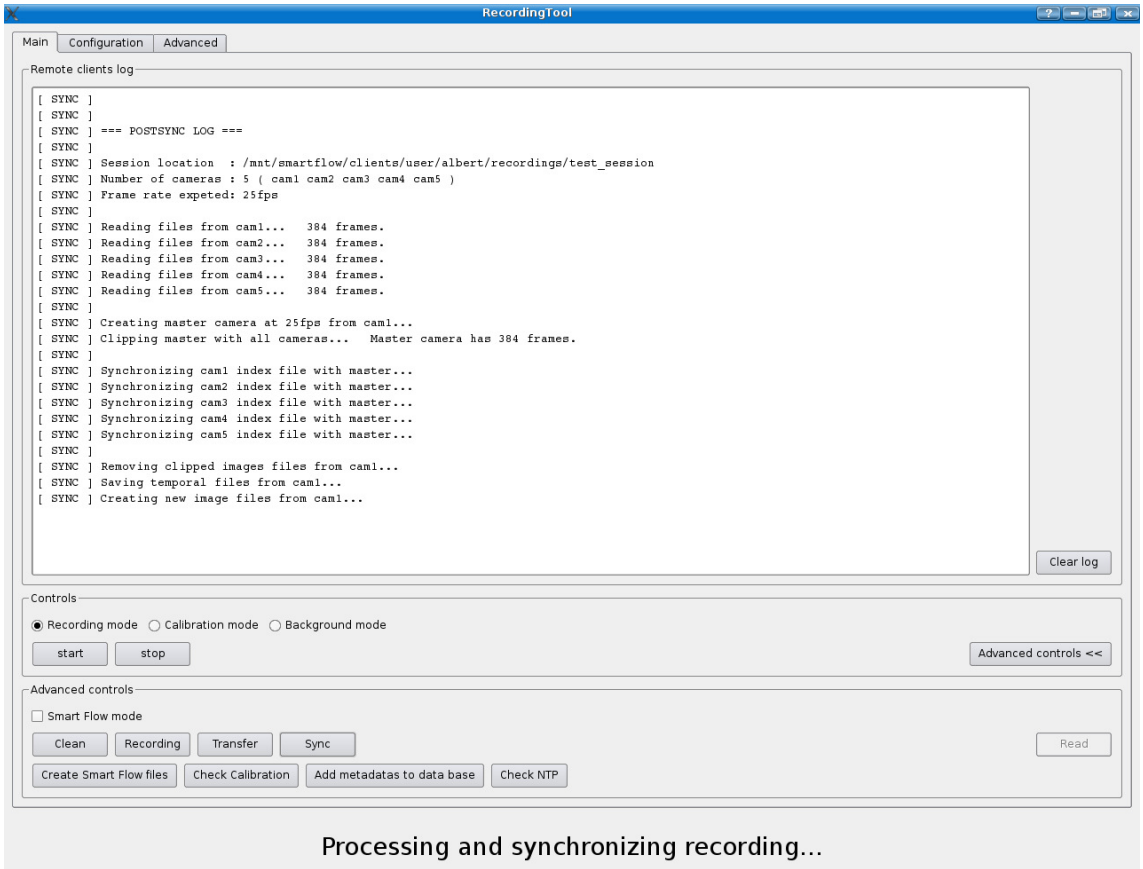


Figura 15. Captura de la *Recording-GUI*. Les tres pestanyes estan disponibles a la zona superior.

La pestanya principal, o *Main,* està formada per:

- La finestra de *logs* on es visualitzaran els resultats dels diferents *clients* que s'executaran remotament.
- Controls bàsics.
- Controls avançats ocultables.
- Barra d'estat.

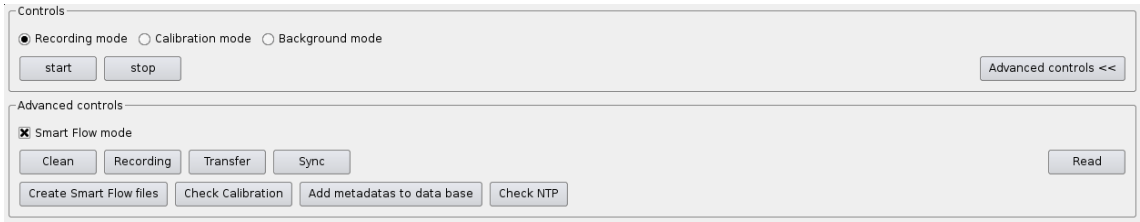


Figura 16. Quadre de controls de la pestanya principal de la *Recording-GUI*. Si l'usuari activa *Smart Flow mode* l'opció de *read* estarà disponible, en cas contrari estarà deshabilitada l'opció de *read*.

Com podem veure a la Figura 16 l'aplicació té tres modes de funcionament:

- *Recording mode* : Per realitzar una gravació normal.
- *Calibration mode* : Per calibrar càmeres una a una.
- *Background mode* : Per guardar imatges de *background*.

Un cop escollit el mode de funcionament, amb el botó *start* s'inicia un assistent que guia a l'usuari a través dels diferents passos que s'han de seguir per tal de realitzar la funcionalitat desitjada. L'assistent dona informació de l'estat actual del procés i del següent pas a donar.

En el moment que s'inicia la captura i escriptura de les imatges, l'usuari podrà aturar-la quan ho desitgi amb el botó de *stop*.

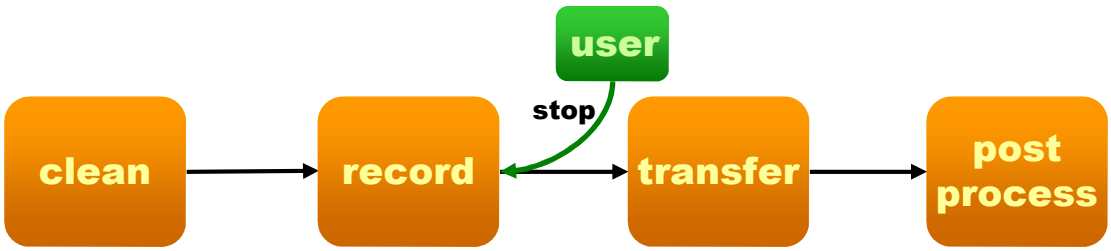


Figura 17. Diagrama de blocs dels passos a realitzar en una gravació/calibració.

Com s'il·lustra a la Figura 17, els passos que segueix l'assistent són els següents:

- *Clean* : S'eliminen les gravacions temporals en els *servers* on es capturaran i guardaran localment les noves imatges.
- *Record* : S'inicia la captura i escriptura de les imatges als diferents *servers*. L'usuari pot aturar la gravació quan ho desitgi.
- *Transfer* : Es transfereixen els fitxers guardats localment al seu destí final o repositori.
- *Post Process* : Es post processa la gravació/calibració per tal de sincronitzar-la/calcular les matrius de calibració. En *background mode* no es realitza aquest pas.

La Figura 16 també ens mostra els controls avançats de la pestanya principal. Aquests controls ens permeten realitzar les següents funcionalitats:

- Treballar amb els *clients* de *Smart Flow*[1].
- El mode de treball amb *Smart Flow* habilita la possibilitat d'iniciar la lectura d'una gravació (*read*).
- Executar pas a pas una gravació/calibració.
- Crear els fitxers de configuració de *Smart Flow*.
- Afegir una gravació a la base de dades.
- Comprovar la calibració actual de les càmeres.
- Comprovar l'estat de la sincronització per *NTP* de la xarxa.

La Figura 18 ens il·lustra com la pestanya de configuració permet :

- Afegir o eliminar càmeres al sistema. S'especifica el *server* on està connectada i el fitxer de calibració.
- Configurar la captura d'imatges.
- Escollir el format de gravació.
- Seleccionar el directori temporal on cada càmera guardarà les seves imatges localment[2].
- Seleccionar el directori final on es transferiran, processaran i guardaran tots els fitxers.

[1] Només serà possible treballar amb *clients* de *Smart Flow* si prèviament s'han activat els *servers* necessaris des de la *GUI* de *Smart Flow*. Per més informació sobre els *severs* de *Smart Flow* consultar l'apartat 4.2.
[2] Per assegurar la qualitat de les gravacions cal que el disc local on es guarden utilitzi el *file-system ext2*. Per això tots els servers tenen una partició anomenada `/CHILrecording`.

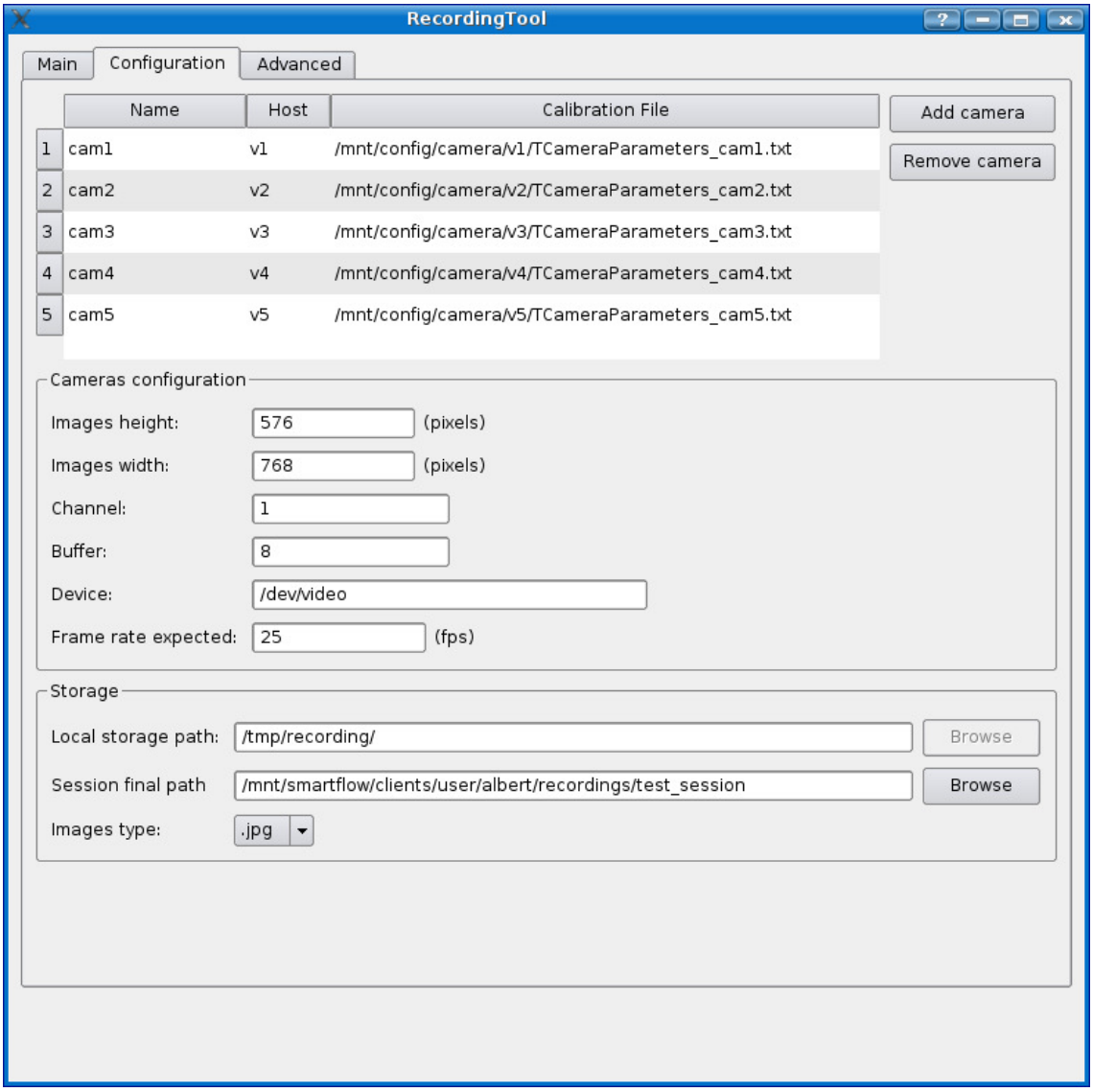


Figura 18. La pestanya de configuració de la *Recording-GUI*.

Com il·lustra la Figura 19 la pestanya de configuració avançada permet:

- Escollir quins *clients* s'utilitzaran.
- Configurar el nombre d'imatges de calibració que es guardaran.
- Configurar el nombre d'imatges de *background* que es guardaran.
- Escollir des de quin *PC* es post processarà la gravació.

El lector es pot fer una idea que la realització de gravacions utilitzant la *Recording-GUI* és una tasca força senzilla. L'usuari ha d'escollir les càmeres que vol utilitzar i iniciar l'assistent. Aquest el guiarà en tot el procés, al final del qual la gravació estarà emmagatzemada i **preparada per poder ser llegida amb el sistema de lectura**. En aquest sentit cal dir que en el procés de gravació es generen els fitxers de configuració de *Smart Flow* que permetran llegir-ne el contingut.

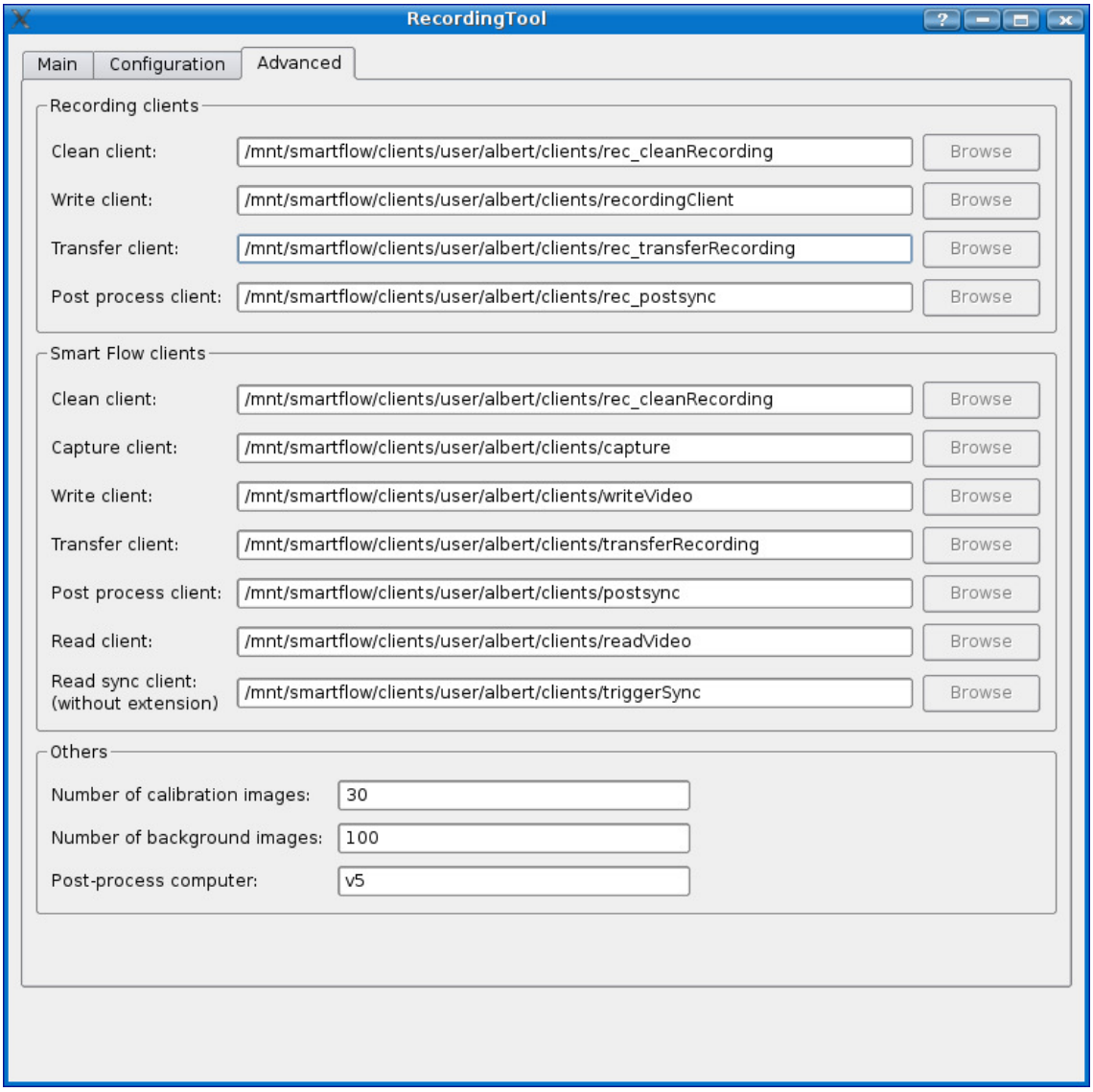


Figura 19. Captura de la pestanya de configuració avançada de la *Recording-GUI*.

Com veurem en el següent apartat, si es volen realitzar gravacions directament des de *Smart Flow* sense utilitzar la *Recording-GUI* la tasca es complica i s'allarga força. No obstant, la pròpia *Recording-GUI* incorpora un mòdul que genera tots els fitxers de configuració necessaris per poder utilitzar *Smart Flow* còmodament.

És a dir, és possible utilitzar la *Recording-GUI* només per configurar *Smart Flow* i posteriorment fer servir la seva interfície per controlar l'execució dels *clients*.

3.3. Funcionament del sistema utilitzant *Smart Flow*

El nou sistema *off-line* està disponible des de la *Recording-GUI*, explicada a l'apartat anterior, i des de la GUI de *Smart Flow*. Si bé en el cas d'utilitzar la *Recording-GUI* la configuració interna dels *clients* és automàtica, en el cas d'utilitzar només la interfície de *Smart Flow* la configuració haurà de ser manual[1]. També necessitarem configurar manualment aquests *clients* si es volen utilitzar en altres sistemes o tasques.

En aquest apartat s'explicaran els passos a seguir per configurar *clients* individual i col·lectivament per poder realitzar les tasques del sistema *off-line* des de *Smart Flow*. Com veurem, el procés és força més llarg si no s'utilitzen les eines d'automatització integrades a la *Recording-GUI*.

[1] Es recorda al lector que la *Recording-GUI* incorpora un mòdul que permet configurar automàticament *Smart Flow*. La configuració serà manual si tampoc s'utilitza aquest mòdul.

3.3.1. *Configuració individual dels* clients

En general podem dir que tots els *clients* es configuren mitjançant uns paràmetres de la línia de comandes i mitjançant fitxers de configuració.

Client	Paràmetres	Descripció
cleanRecording	configFile	Neteja una gravació del directori temporal.
capture	configFile calibrationFile	Captura les imatges i assigna timestamps.
writeVideo	configFile cameraName	Guarda imatges a disc i genera els fitxers de metadades.
transferRecording	configFile	Transfereix gravacions entre directoris.
processRecording	configFile	Processa una gravació per sincronitzar-la.
readVideo	configFile cameraName	Llegeix una gravació i crea un flow de vídeo
triggerSync	configFile	Sincronitza la lectura emetent *triggers* i rebent *timestamps*.
showRoom	room.xml	Visualitza la imatge amb la quadrícula de la sala..

Taula 2. Els *clients* utilitzats pel sistema *off-line* i els seus paràmetres.

Com s'observa a la Taula 2 la majoria dels clients reben com a paràmetre un fitxer de configuració. Tot i que en molts casos només calen un parell de paràmetres, s'ha seguit l'estratègia d'utilitzar fitxers de configuració per tal facilitar la configuració des de Smart Flow[1]. D'aquesta manera, si configurem diferents *clients* perquè llegeixin el mateix fitxer de configuració, modificant-lo estarem canviant la configuració de tots els *clients* alhora i no haurem de canviar els paràmetres de cada client un per un[2].

```
<?xml version="1.0" encoding="UTF-8" ?>
<config>
  <CaptureConfig>
    <imageHeight>576</imageHeight>
    <imageWidth>768</imageWidth>
    <channel>1</channel>
    <buffer>8</buffer>
    <useEveryNFrame>1</useEveryNFrame>
    <deviceName>/dev/video</deviceName>
  </CaptureConfig>

  <TriggerSync>
    <fpsExpected>25</fpsExpected>
    <checkSync>1</checkSync>
  </TriggerSync>


  <RecordConfig>
    <recordingLocalStorage>/tmp/recording/</recordingLocalStorage>
    <sessionFinalPath>/mnt/smartflow/clients/user/albert/recordings/example</sessionFinalPath>
    <outputExtension>.jpg</outputExtension>

    <writeBackgroundMode>0</writeBackgroundMode>
    <writeCalibrationMode>0</writeCalibrationMode>
    <readBackground>0</readBackground>
    <backgroundImages>100</backgroundImages>
    <calibrationImages>30</calibrationImages>
    <firstFrame>0</firstFrame>

    <fpsExpected>25</fpsExpected>
    <camNames>cam1 cam2 cam3 cam4 cam5</camNames>
  </RecordConfig>
</config>
```

Figura 20. Exemple de fitxer de configuració per *clients*.

[1] L'experiència de treballar amb *Smart Flow* demostra que aquesta estratègia estalvia molta feina.
[2] Com veurem en l'apartat 4.3, això és possible gràcies a la llibreria de suport comuna anomenada Recording-Lib.

La Figura 20 mostra un exemple de fitxer de configuració vàlid per qualsevol *client*[1] i a la Taula 3 es descriuen tots els camps possibles d'aquests fitxers, el seu significat i quins clients l'usen.

Secció	Paràmetre	Clients afectats	Descripció
CaptureConfig	imageHeight	capture	Alçada de les imatges capturades
CaptureConfig	imageWidth	capture	Amplada de les imatges capturades
CaptureConfig	buffer	capture	Dimensions del *buffer* del *drivers* de captura (ex. `8`)
CaptureConfig	channel	capture	Canal del dispositiu de captura (ex. `1`)
CaptureConfig	device	capture	Dispositiu de captura (ex. `/dev/video/`)
CaptureConfig	useEveryNFrame	capture	Permet capturar només un de cada N *frames*.
RecordingConfig	sessionFinalPath	transferRecording processRecording readVideo	Directori de xarxa on es guardarà i processarà la gravació / calibració
RecordingConfig	recordingLocalStorage	writeVideo transferRecording	Directori local dels *PCs* amb capturadores on es guarden temporalment les imatges
RecordingConfig	outputExtension	writeVideo	Format de les imatges (ex. `.jpg`)
RecordingConfig	writeBackgroundMode	writeVideo	Selecciona guardar les imatges en mode background.
RecordingConfig	writeCalibrationMode	writeVideo	Selecciona guardar les imatges en mode calibració
RecordingConfig	camNames	processRecording	El nom de les càmeres a sincronitzar separades per espais.
RecordingConfig	readBackground	readVideo	Especifica si es volen llegir les imatges de background o no.
RecordingConfig	backgroundImages	writeVideo	Especifica el nombre d'imatges de background que s'escriuran.
RecordingConfig	calibrationImages	writeVideo	Especifica el nombre d'imatges de calibració que s'escriuran.
RecordingConfig	firstFrame	readVideo	Especifica per quin *frame* començarà la lectura.
RecordingConfig TriggerSync	fpsExpected	processRecording triggerSync	El numero de frames per segon que s'espera que tingui la gravació[2].

[1] Com s'explica al Capítol 4 s'ha escollit el format *XML* com a format de configuració de *clients* i descriptor de dades.
[2] En l'apartat 4.5 veurem que en el post processat es sincronitzen les càmeres amb una camera *master* que seguirà aquest *frame-rate*.

TriggerSync	checkSync	triggerSync	Habilita/deshabilita el *log* d'errors trobats en el sincronisme durant la lectura.

Taula 3. Descripció dels camps dels fitxers de configuració dels *clients*.

3.3.2. Configuració col·lectiva de clients *en mapes*[1]

En aquest cas ens cal distingir dos tipus de mapes o configuracions, d'escriptura i de lectura.

3.3.2.1. Configuració i execució de mapes d'escriptura. Realització de gravacions.

Per tal de poder configurar *Smart Flow* per realitzar gravacions cal seguir els passos següents:

1. Afegir els *PCs* on es capturen les imatges a la llista de servers:

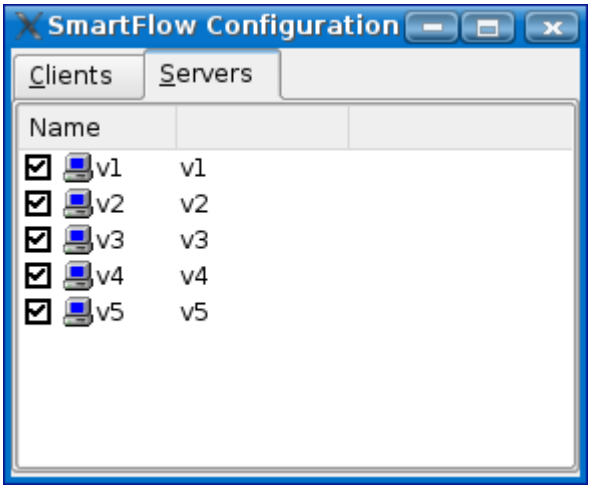


Figura 21. Exemple de llista de servidors on es capturen les imatges.

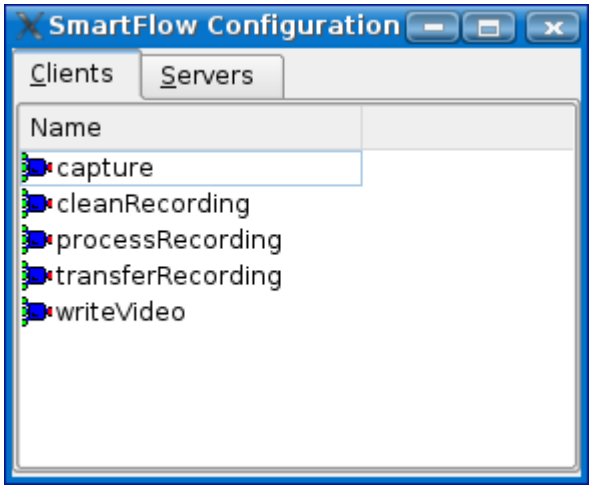


Figura 22. Llista de *clients* de *Smart Flow* necessaris per realitzar gravacions.

2. Afegir tots els *clients* que es mostren a la Figura 22
3. Configurar el mapa.

Com es mostra a la Figura 23 la configuració del mapa cal que compleixi les següents especificacions:

- Per cada càmera que es vulgui utilitzar s'ha d'afegir una còpia dels següents clients:
  a. *cleanRecording*
  b. *capture*
  c. *writeVideo*
  d. *transferRecording*
- A cada un d'aquest conjunt de quatre *clients* se'ls ha d'assignar un *server* diferent.
- Les parelles de clients *capture* / *writeVideo* que s'executen al mateix *server* han d'unir els seus *flows* de vídeo.
- El *client* de *processRecording* pot executar-se a qualsevol *server*. A ser possible el més ràpid.

[1] El lector que no tingui experiència en l'ús de la interfície de *Smart Flow* trobarà dificultats per seguir aquest apartat. Els objectius d'aquest apartat no són explicar com funciona aquesta interfície, sinó mostrar que és possible utilitzar-la per gestionar el sistema off-line i, sobretot, justificar la necessitat de desenvolupar una interfície que simplifiqui aquesta gestió. Al final de l'apartat el lector podrà comparar ambdues interfícies, la *Recording-GUI* explicada anteriorment i la de *Smart Flow* i veure el nivell d'automatització aconseguit per la primera. No obstant, pel lector interessat en seguir millor aquest apartat es recomanen les lectures [3] i [4].

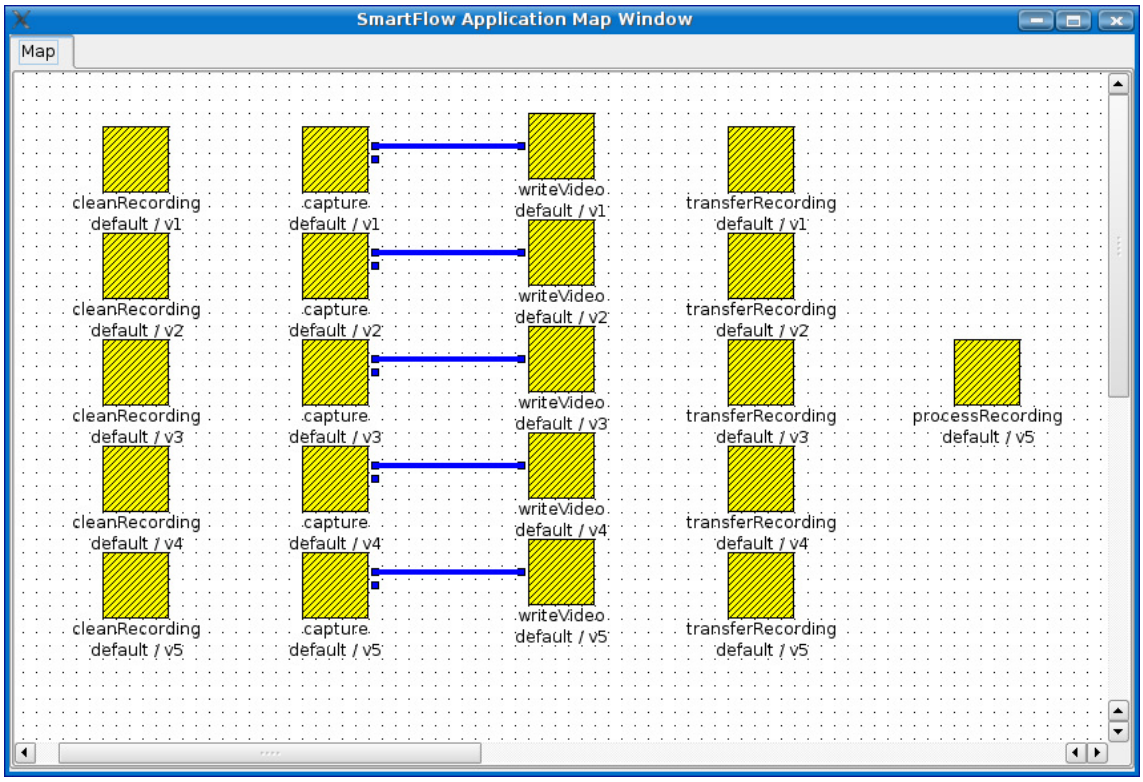


Figura 23. Exemple de configuració d'un mapa de *Smart Flow* per realitzar gravacions.

4. <u>Crear un fitxer de configuració comú</u>

Per crear el fitxer de configuració comú podem seguir la Taula 3 o utilitzar el de l'exemple de la Figura 20. Per tal que tots els *clients* que s'executen remotament tinguin accés a aquest fitxer de configuració és necessari guardar aquest fitxer en un directori de la xarxa accessible des de tots els *servers* que s'utilitzin.

5. <u>Definir els paràmetres per línia de comandes dels *clients*</u>

Per fer això s'han de seleccionar un per un els clients en el mapa i editar-ne els paràmetres.

- El segon paràmetre dels writeVideo, `cameraName`, ha de coincidir amb els noms que s'han donat en el fitxer de configuració en el camp `<cameraNames>`.
- Tots els clients, tenen com a primer paràmetre el fitxer de configuració creat en el punt 4.

6. <u>Iniciar el procés de gravació</u>

En aquest punt el *Smart Flow* ja està configurat per realitzar gravacions. La creació de fitxers de configuració automàtica que permet la *Recording-GUI* permet realitzar tota la configuració prèvia.

El procés d'execució d'una gravació amb *Smart Flow* consta de les mateixes etapes que amb la *Recording-GUI*:

(a) *clean* : Execució dels *cleanRecording* per assegurar que no hi ha fitxers temporals de gravacions prèvies.
(b) *record* : Execució de les parelles *capture* / *writeVideo* per capturar i escriure les imatges a disc.
(c) *stop* : Aturar la gravació parant les parelles *capture* / *writeVideo*
(d) *transfer* : Execució dels *transferRecording* per transferir els fitxers a un directori de xarxa comú.
(e) *post process*: Execució del processRecording per post processar i sincronitzar totes les càmeres.

3.3.2.2. Configuració i execució de mapes de lectura

La configuració i execució d'un mapa de lectura té molts punts en comú amb els passos explicats anteriorment. No obstant, les diferencies són notables i cal ressaltar-les.

En aquest cas de mapes de lectura cal:

1. Afegir el *server* on s'executaran tots els *clients* a la llista de *servers* (només cal un sol server).

2. Afegir els *clients readVideo* i *triggerSync_xN*, on N és el nombre de càmeres a llegir, a la llista de *clients*.

3. Configurar el mapa:

La lectura d'una gravació es pot fer en dos modes diferents:

(a) A temps real : Es llegeix al ritme al que es van capturar les imatges.
(b) Lectura controlada : Es llegeix al ritme al que es processen les imatges[1].

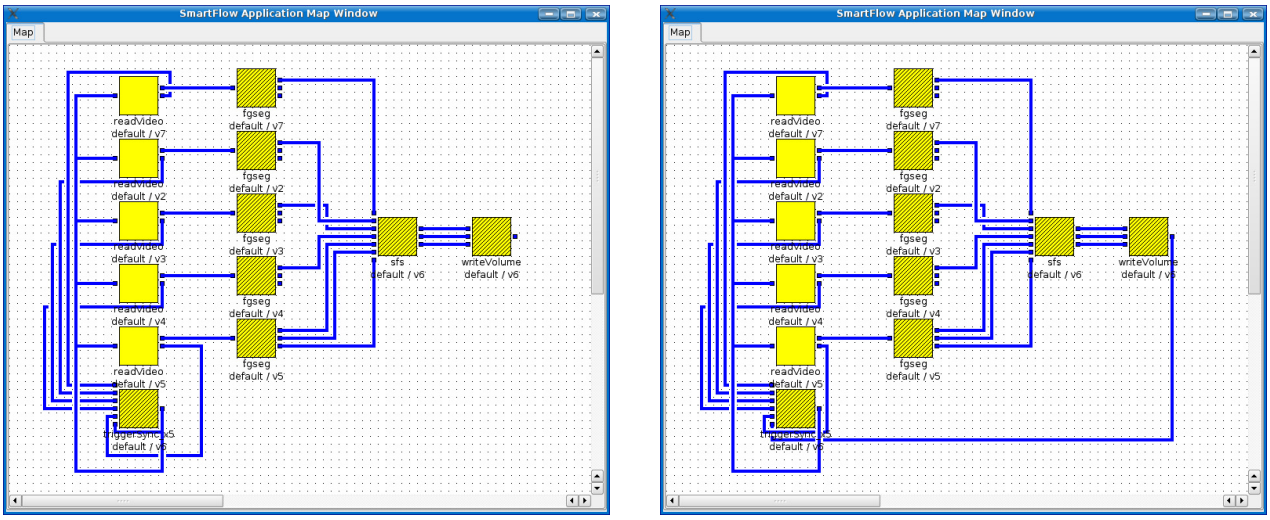


Figura 24. Exemples de mapes de lectura, un de lectura a temps real (esquerra) i l'altre de lectura controlada (dreta). El mapa de lectura controlada ha estat creat per l'estudiant de doctorat Jordi Salvador en la seva participació en les Avaluacions 2007 del projecte CHIL.

En tots dos casos la configuració del mapa té els següents punts en comú:

- Cal una còpia de *readVideo* per cada càmera.
- Només un *triggerSync_xN*.
- Els N *flows* de *timestamps* dels *readVideo* han d'unir-se a les entrades del *triggerSync*.

Com es mostra a la Figura 24, la diferència es troba en l'ús o no d'un *trigger* global:

- Amb *trigger* global : El *trigger* global s'ha de connectar a l'entrada del *triggerSync*.
- Sense *trigger* global : El *trigger* de sortida del *triggerSync* s'ha de connectar a la seva entrada.

4. Crear un fitxer de configuració comú

Igual que en el cas dels mapes d'escriptura cal crear el fitxer de configuració comú seguint la Taula 3. La Figura 25 ens mostra un exemple d'un fitxer de configuració pels *clients* de lectura.

[1] Per poder llegir al ritme en que les imatges són processades cal que l'últim *client* de processat emeti una senyal de *trigger*. Aquest *client* l'ha de desenvolupar l'usuari, no pertany al sistema *off-line*. Per més informació sobre creació de *clients* de *Smart Flow* i generació de *flows* de *trigger* es recomana la lectura [4].

```
<?xml version="1.0" encoding="UTF-8" ?>
<config>

  <TriggerSync>
    <fpsExpected>25</fpsExpected>
    <checkSync>1</checkSync>
  </TriggerSync>

  <RecordConfig>
    <sessionFinalPath>/mnt/smartflow/clients/user/albert/recordings/example</sessionFinalPath>
    <firstFrame>0</firstFrame>
  </RecordConfig>

</config>
```

Figura 25. Fitxer de configuració per un mapa de lectura.

5. Definir els paràmetres de línia de comandes dels clients al mapa de *Smart Flow*.

    - Com es mostra a la Taula 2 els *readVideo*, tenen com a primer paràmetre el fitxer de configuració creat en el punt 4.
    - El segon paràmetre de cada *readVideo*, `cameraName`, ha de coincidir amb un dels noms de les càmeres que hi ha a la gravació que es vol llegir.

6. Iniciar la lectura

En aquest moment el mapa de lectura està configurat per iniciar la lectura. L'usuari pot afegir normalment els seus *clients* de processat de vídeo i connectar-los als *clients* de lectura com ho faria amb els *clients* de captura.

Com en el cas de la realització de gravacions, la creació de fitxers de configuració automàtica de la *Recording-GUI* permet estalviar-se tots els passos anteriors.

Com s'ha mostrat, quan s'utilitza la interfície de *Smart Flow* l'usuari té la necessitat de crear manualment fitxers *XML* i de configurar tots els clients, un per un, en el mapa de configuració. Si pensem en l'escalablitat del sistema, ens adonem que si s'amplia el nombre de càmeres s'amplien el nombre *clients* i realitzar una gravació es transforma ràpidament en un procés llarg i tediós.

En aquest sentit ara podem apreciar millor el nivell d'automatització aconseguit per la *Recording-GUI*. Si bé l'ús de *Smart Flow* dóna una major flexibilitat en l'ús dels clients permetent-ne un ús diferent (ex. en altres i futurs sistemes), la *Recording-GUI* simplifica molt la realització de les tasques específiques per les que s'ha dissenyat: gestió de gravacions i calibracions.

Capítol 4. Desenvolupament i implementació del sistema

Aquest projecte s'ha desenvolupat, no com una única i gran aplicació amb moltes funcionalitats, sinó a base de múltiples i petites aplicacions creades per ser executades remotament. En el context del projecte, i degut a la seva vinculació amb *Smart Flow*, a aquestes petites aplicacions se les anomena genèricament com a *clients*.

Òbviament aquests *clients* comparteixen algunes funcionalitats comunes i necessiten mantenir una harmonia entre si. És per això que, com veurem en l'apartat 4.3, s'ha creat una llibreria de suport comuna anomenada *Recording-Lib*.

El desenvolupament del projecte ha fet que finalment aquests *clients* no siguin només executables mitjançant *Smart Flow*, sinó que s'ha creat un nou i senzill sistema de control remot de *clients* que funciona sobre la *Recording-Lib* i que, com veurem, és el que ha permès la creació de la *Recording-GUI*.

Per altra banda el desenvolupament del sistema de sincronisme és també un dels punts més importants del sistema. Com veurem a l'apartat 4.5, el fet de post processar de les gravacions ha desembocat en un algoritme que deixa les gravacions en un estat òptim per ser llegides i analitzades per algoritmes de processat d'imatge.

En aquest apartat es farà un breu resum cronològic de l'evolució del projecte i a continuació es destacaran aquests i d'altres aspectes importants de la seva implementació.

4.1. Història del projecte

El projecte s'ha desenvolupat cronològicament en les següents etapes:

1. Documentació sobre *Smart Flow*, la *smart-room* i CHIL

Com tot projecte, aquest també ha necessitat d'una primera etapa de documentació. En aquesta etapa inicial s'han adquirit principalment coneixements sobre el funcionament de *Smart Flow* ([4], [5]) i sobre el funcionament actual dels sistemes *on-line* i *off-line* de la *smart-room* de la UPC ([7], [11], [12]), així com del seu entorn de desenvolupament ([15], [16]).

També s'han estudiat els diferents formats de dades i meta-dades del projecte CHIL ([6], [9]) per tal de definir el nou format estàndard.

2. Desenvolupament dels clients bàsics de *Smart Flow*

En aquesta etapa es desenvolupen els primers *clients* de *Smart Flow*:

- *capture*
- *writeVideo*
- *readVideo*

Aquests *clients* permeten fer les primeres validacions de *Smart Flow* com a *middleware* pel nou sistema *off-line*[1].

3. Desenvolupament dels sistemes de sincronisme

Analitzant els resultats obtinguts durant el desenvolupament dels primers *clients* s'arriba a les següents conclusions:

- El nombre de *dropped-frames* creix si s'utilitza un *client* de sincronització *on-line*.
- El nombre de *dropped-frames* és pràcticament nul, si la gravació es fa sense *client* de sincronització.

Amb aquests resultats es prenen les següents decisions sobre el sistema de sincronisme i s'implementen:

- El sistema d'escriptura requerirà d'un post processat per alinear temporalment totes les càmeres i corregir els *dropped-frames* i d'altres errors de sincronisme.
- El sistema de lectura implementarà un sistema de sincronisme mitjançant l'ús de *triggers*.

4. Adaptació de les gravacions prèvies al nou sistema

Es decideix el format estàndard de les gravacions i amb els nous sistemes d'escriptura i lectura en marxa, es desenvolupen les eines que permetran actualitzar les gravacions existents al nou format.

En aquesta etapa s'actualitzen totes les gravacions existents en el *NAS* utilitzant l'eina de post processat desenvolupada.

5. Desenvolupament d'una aplicació per línia de comandes per controlar *Smart Flow*

En aquest punt la funcionalitat i l'objectiu més bàsic del projecte s'han assolit. Ja es poden fer gravacions i calibracions amb la interfície de *Smart Flow*. Tot i així, el resultat no és un entorn còmode i senzill per l'usuari final. Per aquest motiu, i com a primera aproximació, es desenvolupa una aplicació per línia de comandes que controla els *clients* i la configuració de *Smart Flow*[2].

6. Documentació sobre control de processos remots

En aquest punt el projecte topa amb la dificultat de controlar els *servers* de *Smart Flow*. Es fa un estudi d'enginyeria inversa i s'observa que la complexitat és major de l'esperada. Finalment es desestima la implementació d'un mòdul propi per controlar aquests *servers*. En aquesta decisió es té en compte, no només la dificultat, sinó també la possibilitat que futures versions de *Smart Flow* canviïn aquest funcionament i el mòdul quedi ràpidament obsolet.

Es decideix buscar altres solucions i iniciar un nou procés de documentació. S'estudia l'opció d'utilitzar *CORBA* [10] com arquitectura de control de processos remots enlloc de *Smart Flow*. Les conclusions de l'estudi sobre *CORBA* són les següents:

- Ens permet fer tot allò que volem respecte al control remot d'aplicacions i processos.
- Tot i que les funcionalitats buscades són poques, el codi a afegir als *clients* és gran.
- Mantenir la compatibilitat amb *Smart Flow* es complica.

Finalment s'opta per la que es considera com la solució més senzilla: una solució pròpia i simple basada en la llibreria *pstream*[3] [19].

7. Desenvolupament de la *Recording-Lib*

En aquest moment les funcionalitats de l'aplicació per línia de comandes desenvolupada prèviament es modifiquen i s'inclouen en una llibreria donant origen a la *Recording-Lib*.

---

[1] Per aquests primers tests s'utilitza el mateix sistema de sincronització del sistema *on-line*.
[2] Aquesta aplicació no aconsegueix substituir totalment la *GUI* de *Smart Flow*. En l'apartat 4.2 s'expliquen els aspectes més rellevants de la implementació de *Smart Flow*, i el nivell control aconseguit.
[3] Aquesta llibreria gestiona l'execució d'aplicacions remotes i permet controlar-ne les entrades i sortides. En l'apartat 4.3.3 se'n detallarà l'ús i el funcionament.

Aquesta llibreria es desenvolupa amb una doble funcionalitat:

- Llibreria de suport per tots els *clients* per harmonitzar les seves funcionalitats comunes.
- Motor de control d'execució remota d'aquests mateixos *clients*.

Com ja s'ha dit, alguns aquests mètodes es desenvolupen basant-se en l'anterior aplicació per línia de comandes. Per això, en aquest punt del projecte la feina més gran recau en la creació d'una arquitectura de control de processos remots pròpia basada en la llibreria *pstream*. Aquesta arquitectura es fa pensant en el seu ús com a llibreria de suport per la futura *Recording-GUI*[1].

8. Desenvolupament de la *Recording-GUI*

Finalment es desenvolupa una senzilla *GUI* utilitzant la tecnologia de *QT* [21], l'anomenada *Recording-GUI*. El seu objectiu és integrar totes les funcionalitats del nou entorn *off-line* però tenint la mínima algorísmica possible. Es pretén deixar la implementació de tota la funcionalitat a la *Recording-Lib* desenvolupada anteriorment.

Aquesta separació es fa amb la intencionalitat que en el futur sigui fàcil poder canviar la interfície gràfica per qualsevol altra. Tot i així, el desenvolupament d'aquesta *GUI* és força costós ja que implica una nova documentació, en aquest cas sobre la tecnologia *QT*.

9. Test i *debugging*

Actualment el software que compon el nou sistema *off-line* es troba en un procés de test i validació. Si ve algunes de les parts ja han estat àmpliament testejades per diferents usuaris d'altres encara no són del tot estables.

En el capítol 0 es poden consultar els diferents resultats obtinguts així com l'estat del sistema en el moment de finalitzar el projecte.

4.2. Característiques rellevants de la implementació de *Smart Flow*

En l'apartat 2.2 hem vist el funcionament bàsic de *Smart Flow*. En aquest punt volem presentar-ne els aspectes interns que han estat més importants en el desenvolupament del projecte.

Alguns d'aquests aspectes no es troben en els seus manuals de *Smart Flow* ([4], [5]) i s'han estudiat directament del codi font mitjançant enginyeria inversa.

4.2.1. Fitxers de configuració de Smart Flow

Smart Flow es configura totalment mitjançant dos fitxers amb l'extensió *.rc* que es poden guardar o carregar des de la pròpia interfície gràfica de *Smart Flow*.

- Fitxer d'aplicació: Conté la llista de *clients* i *servers* disponibles així com les seves característiques.
- Fitxer de configuració o mapa: Conté la configuració del mapa, és a dir, quins *clients* s'executaran en quins *servers* i com es connecten els seus *flows*.

Per tal que la *Recording-Lib* sigui capaç de configurar automàticament *Smart Flow* per poder realitzar les funcions del sistema *off-line*, cal que sigui capaç de generar aquests fitxers[2].

Una de les característiques que ha facilitat la feina és que aquests fitxers, tot i tenir l'extensió *.rc*, són fitxers descriptors en format *XML*. S'han estudiat els seus camps i propietats i s'han incorporat els mètodes corresponents a la *Recording-Lib* per poder generar-los.

---

[1] A mode d'exemple podem dir que la *Recording-Lib* té mètodes de suport per treballar amb el *file-system* o amb *XML*, i alhora també té mètodes per iniciar els *clients* de captura i escriptura, parar-los, o per generar

[2] En APÈNDIX E. Fitxers de configuració s'hi poden trobar els detalls d'aquests fitxers així com alguns exemples.

Com s'explica al Capítol 3, la *Recording-GUI* permet generar aquests dos fitxers que, al obrir-se des de *Smart Flow*, el configuren per executar les funcionalitats de l'entorn *off-line*.

4.2.2. Gestió dels clients

Un *client* de *Smart Flow* és també una aplicació executable com qualsevol altra amb l'única peculiaritat que executa crides a l'API de *Smart Flow*. Per tant, també és possible executar aquests *clients* des de qualsevol *shell* de Linux.

Quan des de la *GUI* de *Smart Flow* s'afegeix un nou *client* a la llista, aquest és executat amb el paràmetre `--sf-dump-args`. Tal i com es mostra en l'exemple de la Figura 26, si el *client* està ben programat i fa un ús correcte de *l'API* de *Smart Flow*, el resultat de l'execució és un fitxer *XML* per la sortida estàndard que descriu les característiques del *client* i els seus *flows*[1]. Aquests fitxers s'utilitzen internament per configurar els mapes i les llistes de *clients*.

```
albert@v7:clients$ readVideo --sf-dump-args

<?xml version="1.0"?>
<client>
  <name>readVideo</name>
  <flows>
    <input>
      <data size="4" name="UPC::Trigger">
        <name>trigger</name>
        <group>default</group>
      </data>
    </input>
    <output>
      <video width="720" height="480">
        <name>video</name>
        <group>default</group>
      </video>
      <data size="4" name="UPC::Count">
        <name>ts</name>
        <group>default</group>
      </data>
    </output>
  </flows>
</client>
```

Figura 26. Resultat de l'execució d'un *client* de *Smart Flow* amb el paràmetre `-sf-dump-args`.

Igualment, quan des de la *GUI* de *Smart Flow* s'executa un *client* el mecanisme intern es basa en una comanda de *SSH* per *shell* [14]. Com es mostra en l'exemple de la Figura 27, en la comanda de SSH també s'hi afegeixen un seguit de paràmetres específics de *Smart Flow*, en aquest cas `--sf-flow`. Aquests paràmetres són utilitzats per configurar tots els *flows* del *client*.

Notis que per cada *flow* també s'especifica si és d'entrada o sortida i el seu número. En l'exemple hi ha un *trigger* d'entrada i dues sortides, una de vídeo i una de *timestamp*.

```
usage: ssh serverName clientName [--sf-flow i,inputFlowNumber,inputFlowName] [-
-sf-flow o,inputFlowNumber,inputFlowName] [clientParameters]

example: ssh v5 readVideo --sf-flow i,0,"trigger" --sf-flow o,0,"video" --sf-
flow o,1,"ts" /mnt/smartflow/config/readVideo.xml cam5
```

Figura 27. Comanda utilitzada per *Smart Flow* per executar clients remotament definint-ne els *flows*.

[1] Cal fer notar que aquest comportament fa que en la implementació d'un *Smart Flow Client* no es pugui mostrar res per la sortida estàndard fins que no s'hagin inicialitzat la part relacionada amb *Smart Flow*.

S'observa també a la Figura 27 que la comanda d'execució de *clients* també incorpora els paràmetres propis del *client*. En l'exemple hi veiem un fitxer de configuració i el nom d'una càmera.

El coneixement d'aquest funcionament intern de *Smart Flow* és el que ens permet configurar i executar *clients* remotament i poder connectar els diferents *flows*. La *Recording-Lib* incorpora uns mètodes que permeten llençar aquest tipus de comandes còmodament.

4.2.3. Mecanisme intern de comunicació i transport de flows

Es podria dir que *Smart Flow* s'executa de manera distribuïda. En concret s'executa un *daemon* a cada *server*[1] registrat. Aquests *daemons* són els que realment es comuniquen a través de la xarxa i finalment fan arribar els fluxos de dades als *clients*.

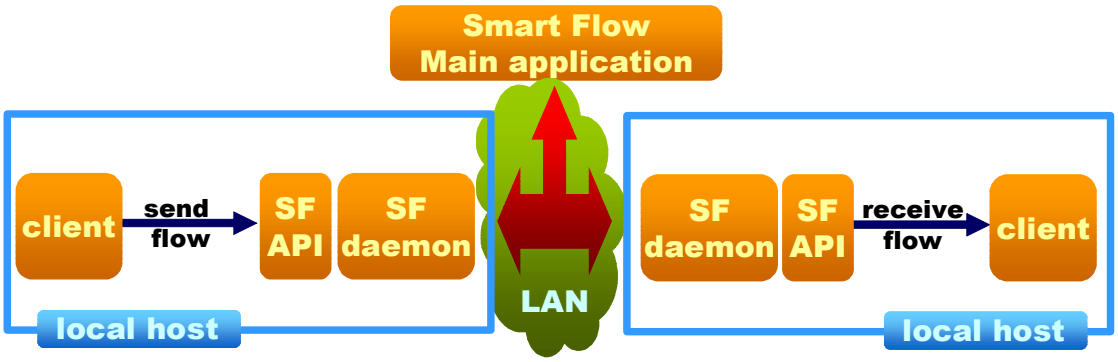


Figura 28. Diagrama de blocs funcionals per la transmissió de fluxos de dades a *Smart Flow*.

Mitjançant *l'API* de *Smart Flow* cada client defineix els *flows* que vol crear i, enviar o rebre de la xarxa. Quan un *client* s'executa en un *server*, el *daemon* que s'està executant en aquell mateix *server* captura aquestes peticions de *flows* i comença la comunicació amb la resta de *daemons* per tal de servir-les.

És així com *Smart Flow* aconsegueix que cada *client* rebi les dades que vol, en forma de *flow*, sense importar quin *client* les estigui enviant, ni a quin *server* s'estigui executant aquest altre *client*. També és així com permet enviar a cada *client* els seus resultats a la xarxa sense importar qui els acabi rebent, ni tan sols si algun *client* els rep o cap.

No obstant, tot i que aquests *daemons* s'executen en *background*, i podria semblar que només son *listeners* que serveixen peticions de l'*API*, en realitat estan en continua comunicació amb l'aplicació principal[2]. *Smart Flow* crea aquests *daemons*, els configura, s'hi comunica contínuament i, finalment, s'encarrega d'aturar-los al tancar-se.

Estudiant el codi font que controla aquests *daemons* s'ha comprovat que és una comunicació complexa i que no es disposa de mètodes públics que la simplifiquin. Això suposa que per poder controlar aquests *daemons* sense l'aplicació principal de *Smart Flow* és necessari un mòdul de control que emuli el propi de *Smart Flow*.

Aquesta opció és massa costosa i perillosa alhora. Costosa perquè el mecanisme és complex, i perillós perquè pot canviar totalment en futures versions de l'aplicació, fent que l'actualització del nostre sistema fos tan costosa com la pròpia implementació actual.

---

[1] En la documentació de *Smart Flow* s'utilitza el terme *local-server* per fer referència a aquests *daemons*.
[2] En la documentació de *Smart Flow* es coneix com *Control Center* a l'aplicació principal.

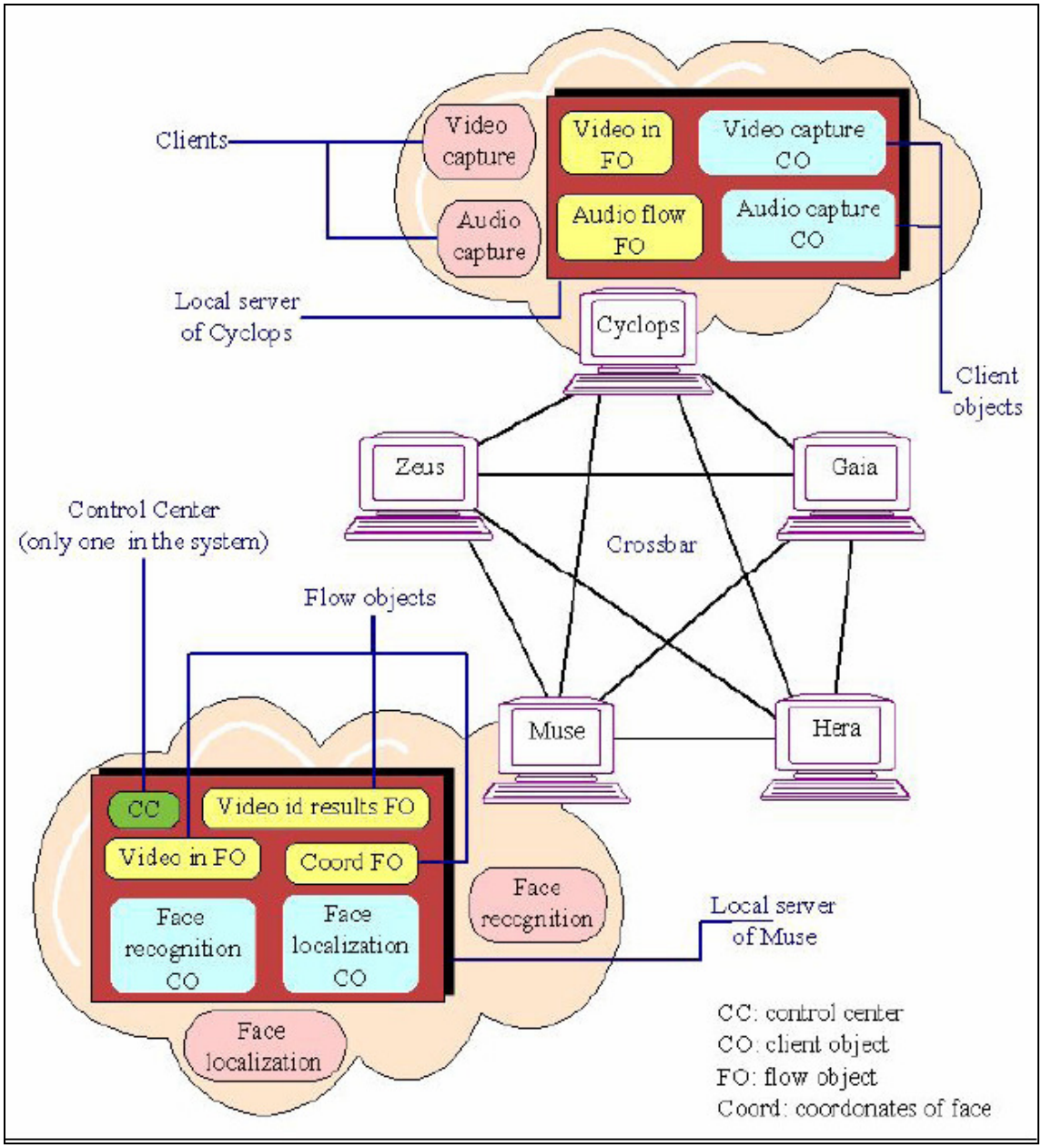


Figura 29. Il·lustració dels diferents blocs que intervenen en el transport de dades en *Smart Flow*. (Font: [5])

4.2.4. Limitacions del wrapper UPC-Smart Flow

Smart Flow proporciona una *API* escrita en el llenguatge de programació C. No obstant, gràcies a què es distribueix amb llicència de codi obert, la UPC ha pogut desenvolupar una capa de software escrita en C++ que l'encapsula, la *UPC-Smart Flow*.

Aquesta capa, o *wrapper,* simplifica tot el procés d'inicialització i gestió de *flows* i, sobretot, defineix unes polítiques de meta-dades pels desenvolupadors de *clients* de la UPC. Això permet que els desenvolupadors puguin concentrar tot l'esforç en la programació dels seus algoritmes de reconeixement minimitzant l'esforç per incloure els seus algoritmes en un *client* i alhora que garanteix que tots els *clients* desenvolupats siguin compatibles entre si.

Per altra banda i com a contrapartida, aquest *wrapper* imposa unes determinades obligacions i limitacions. A continuació s'exposen les que han afectat més directament a la implementació d'alguns dels *clients* del nou sistema *off-line*.

4.2.4.1. Clients sense flows

El *wrapper UPC-Smart Flow* no permet crear *clients* sense *flows*. Evidentment, al crear aquest tipus de *clients* la utilització de *Smart Flow* perd sentit, doncs aquests *clients* són només aplicacions normals que no envien ni reben dades. No s'aprofiten les capacitats de transport de *Smart Flow* i per això el *wrapper* no té en compte aquest cas.

Però les funcionalitats desitjades per l'entorn *off-line* fan molt útil el desenvolupament d'aquest tipus de *clients*. Per exemple el *Transfer Recording*: aquest *client* no genera ni rep cap tipus de *flow* però s'ha d'executar remotament a cada un dels *PCs* on s'han capturat i guardat les imatges per transferir-les a la xarxa i ser emmagatzemades.

Aquestes utilitats es podien desenvolupar com a aplicacions normals i ser executades en una *shell* de Linux. D'aquesta manera, però, hauríem perdut la integració visual que suposa disposar de totes aquestes eines en la mateixa interfície gràfica de *Smart Flow* al costat dels *clients* de captura, escriptura o lectura i poder ser executades de la mateixa manera (gràfica), sense haver la necessitat de cap altra *shell*.

4.2.4.2. Clients realimentats i flows *de sortida previs als d'entrada*

Una de les polítiques que defineix el *wrapper* de *UPC-Smart Flow* és que cada *client* ha d'inicialitzar primer els *flows* d'entrada, després els de sortida i finalment enviar i rebre dades a través d'aquest *flows*. El procés d'inicialització de *flows* es basa en escriure o llegir les meta-dades que el defineixen. En concret, en la inicialització dels *flows* de sortida s'escriuen aquestes meta-dades, i en la dels *flows* d'entrada es llegeixen.

Aquesta diferència fa que, si bé la inicialització dels *flows* de sortida no és un procés de software bloquejant, la dels *flows* d'entrada sí l'és. Per poder llegir les meta-dades cal que primer el *client* que envia el *flow* les hagi escrit.

Això suposa dues limitacions:

- Cap *client* pot enviar cap dada fins que no ha inicialitzat totes les seves entrades. Això impedeix que un client que rebi algun *flow* i emeti, per exemple un *trigger*, pugui emetre res abans de rebre totes les entrades.
- No es pot realimentar un client. És a dir que no es pot configurar un client per rebre un *flow* que ell mateix genera ja que es quedarà bloquejat en la inicialització dels *flows* d'entrada.

Cal destacar que la implementació de *clients* amb aquestes necessitats és més complexa que en el cas anterior, el cas de *clients* sense *flows*. En implementar *clients* que s'han d'executar enmig d'una cadena realimentada de *flows* no es pot utilitzar *l'API* nativa de *Smart Flow* lliurement, s'han de seguir gran part de les estructures de dades i les polítiques definides pel *wrapper UPC-Smart Flow* per tal de garantir-ne la compatibilitat, però sense utilitzar els objectes que ofereix el *wrapper*. Això implica un estudi profund de *l'API* de *Smart Flow*, de *l'API* de *UPC-Smart Flow*.

Com a exemple podem dir que, tal i com veurem en l'apartat 4.5.2, el sincronisme del sistema de lectura es basa en un sistema realimentat de *clients* amb un generador de *triggers* i que, per tan, la implementació del *triggerSync* ha patit aquestes limitacions.

4.3. La *Recording-Lib*

La *Recording-Lib* és una llibreria que s'ha creat durant el desenvolupament del projecte i té una doble funcionalitat:

- Donar suport als diferents *clients* per tal d'harmonitzar-ne les definicions i funcionalitats comunes.
- Donar serveis a qualsevol *GUI* que vulgui gestionar les diferents operacions de l'entorn *off-line*, és a dir, que vulgui controlar aquests mateixos *clients*.

Aquesta llibreria està escrita en C++ i utilitza l'orientació a objectes per algunes funcionalitats, i l'estructura modular per d'altres[1].

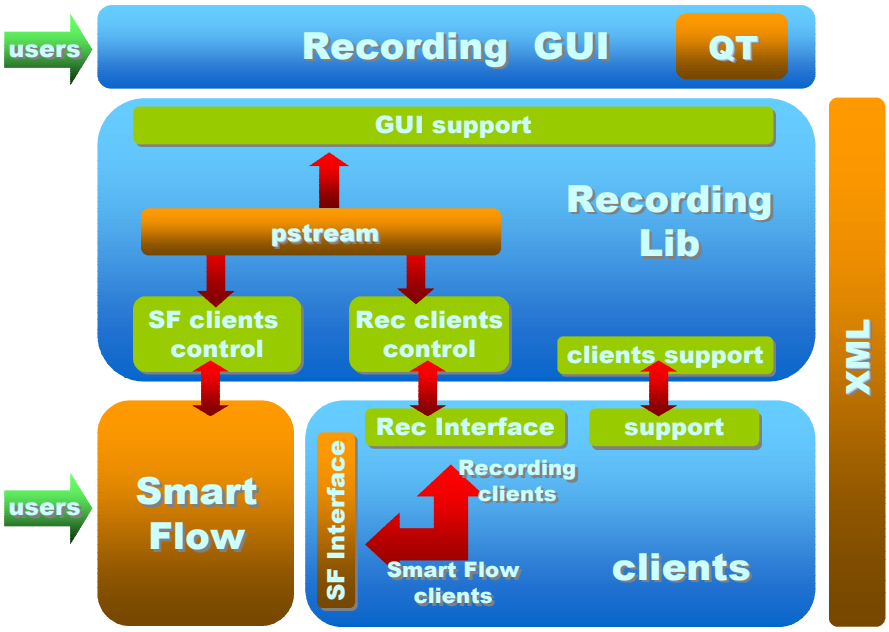


Figura 30. *Stack* de connexions de software del nou sistema *off-line*.

La Figura 30 ens mostra el que les diferents parts de la implementació del nou sistema *off-line* amb la *Recording-Lib* com a centre. Veiem que els usuaris poden accedir al sistema per dos punts diferents, la *Recording-GUI* i per *Smart Flow*, i que això implicarà l'ús de *clients* diferenciats, *Smart Flow Clients* o *Recording-Clients*[2]. La Figura 30 també ens separa el codi desenvolupat en aquest projecte (en blau) de les principals tecnologies públiques utilitzades (en marró): *XML*, *Smart Flow*, *QT* i la llibreria *pstream*.

4.3.1. La Recording-Lib com a llibreria de suport pels clients

Pel que fa al suport que la *Recording-Lib* dona als diferents *clients* destaquen les següents funcionalitats:

- Definició comuna de constants del sistema *off-line*.
- Encapsulació de la gestió de fitxers XML.
- Encapsulació de la gestió del *file-system* (permisos, *I/O*).
- Implementació de la interfície per crear un *Recording-Client*
- Sistema comú de *debugging*.

Aquesta part de la llibreria té una estructura bàsicament modular.

4.3.2. Com a llibreria de control de clients

[1] A l'0

Documentació de la *Recording-Lib* podem veure la documentació generada a través de *doxygen* d'aquesta llibreria.

[2] Més endavant s'explicarà en detall el que s'entén per *Recording-Client*, però aquí podem anticipar que bàsicament és el mateix que un *Smart Flow Client*, però amb una interfície per ser controlat diferent.

La *Recording-Lib* també inclou tota una sèrie de mètodes que permeten controlar els diferents *clients* del sistema *off-line*. Aquesta part de la llibreria permet les següents funcionalitats:

- Iniciar i aturar els *clients* de captura i escriptura en diferents *PCs*.
- Executar *clients* que eliminen, transfereixen i processen gravacions en diferents *PCs*.
- Creació de fitxers de configuració de *Smart Flow*.
- Obtenció del *log* generat pels diferents *clients*.

Aquesta part de la llibreria té una estructura bàsicament orientada a objectes i està dissenyada per ser utilitzada per qualsevol *GUI* que vulgui controlar les funcionalitats de l'entorn *off-line*.

4.3.3. Llibreria pstream

Tot i que l'objectiu inicial de la *Recording-Lib* és el de gestionar els *clients* i *serv*ers de *Smart Flow*, finalment només controla totalment els que podríem anomenar com a *Recording-Clients*[1]. El control de Smart Flow es limita a la configuració de sessions de gravació i reproducció i a l'execució de *clients*, però no als *servers*.

La implementació de la part de la *Recording-Lib* encarregada del control dels diferents *Recording-Clients* es basa en la llibreria *pstream* [19]. Aquesta és una llibreria escrita en C++ que gestiona l'execució d'aplicacions remotes i que, mitjançant la creació de *pipes* internes, permet controlar les entrades i sortides dels processos remots.

Com es mostra a la Figura 31, la *Recording-Lib* utilitza la llibreria *pstream* per executar els *Recording-Clients* i capturar-ne la seva sortida. Al inicialitzar-se utilitzant la *Recording-Lib*, els *Recording-Clients* queden registrats. Això permet que puguin ser aturats en qualsevol moment[2].

La *Recording-Lib* implementa tot el codi relacionat amb la llibreria *pstream*, encapsulant-ne el seu ús. També disposa de tots els mètodes necessaris per tal de poder visualitzar la sortida dels *Recording-Clients* des de qualsevol *GUI*.

4.4. Compatibilitat entre *Smart Flow* i la *Recording-Lib*

Un dels requisits del nou sistema *off-line* és el de la seva compatibilitat amb l'entorn *on-line*, i per tant amb *Smart Flow*. És per això que ja d'inici s'implementen totes les funcionalitats *off-line* desitjades desenvolupant només nous *clients* de *Smart Flow*.

Tot i que el fet de poder d'executar les funcionalitats de l'entorn *off-line* amb la *GUI* de *Smart Flow* és un dels objectius pràctics d'aquest projecte al que no s'ha volgut renunciar, la complexitat de configuració d'una gravació o calibració per un usuari amb aquesta interfície és més gran que la desitjada.

Això és degut a la pròpia flexibilitat i possibilitats de *Smart Flow.* Per aquest motiu que s'ha desenvolupat una nova interfície gràfica que integra d'una manera molt senzilla totes les funcionalitats de l'entorn *off-line*, l'anomenada *Recording-GUI*.

L'objectiu de la *Recording-GUI* és el de controlar els *Smart Flow Clients* prèviament desenvolupats, però degut a la dificultat del control dels *daemons* de *Smart Flow*[3], finalment s'ha creat una nova i senzilla arquitectura de control d'aplicacions remotes[4].

---

[1] A l'apartat 4.4 s'hi troba tota la informació referent als *Recording-Clients*. Però basti saber aquí que no hi ha diferència entre els *Smart Flow Clients* i els *Recording-Clients*, més enllà de la seva interfície de control.
[2] Al registrar-se tenim la informació completa del procés (amb el seu *pid*). Per tan podem comunicar-nos amb el procés remot, no només per la llibreria pstream, sinó també mitjançant senyals [1].
[3] Per més informació sobre el funcionament dels *daemons* de *Smart Flow*, consultar l'apartat 4.2.3.
[4] Per més informació sobre el nou sistema de control d'aplicacions remotes consultar l'apartat 4.3.

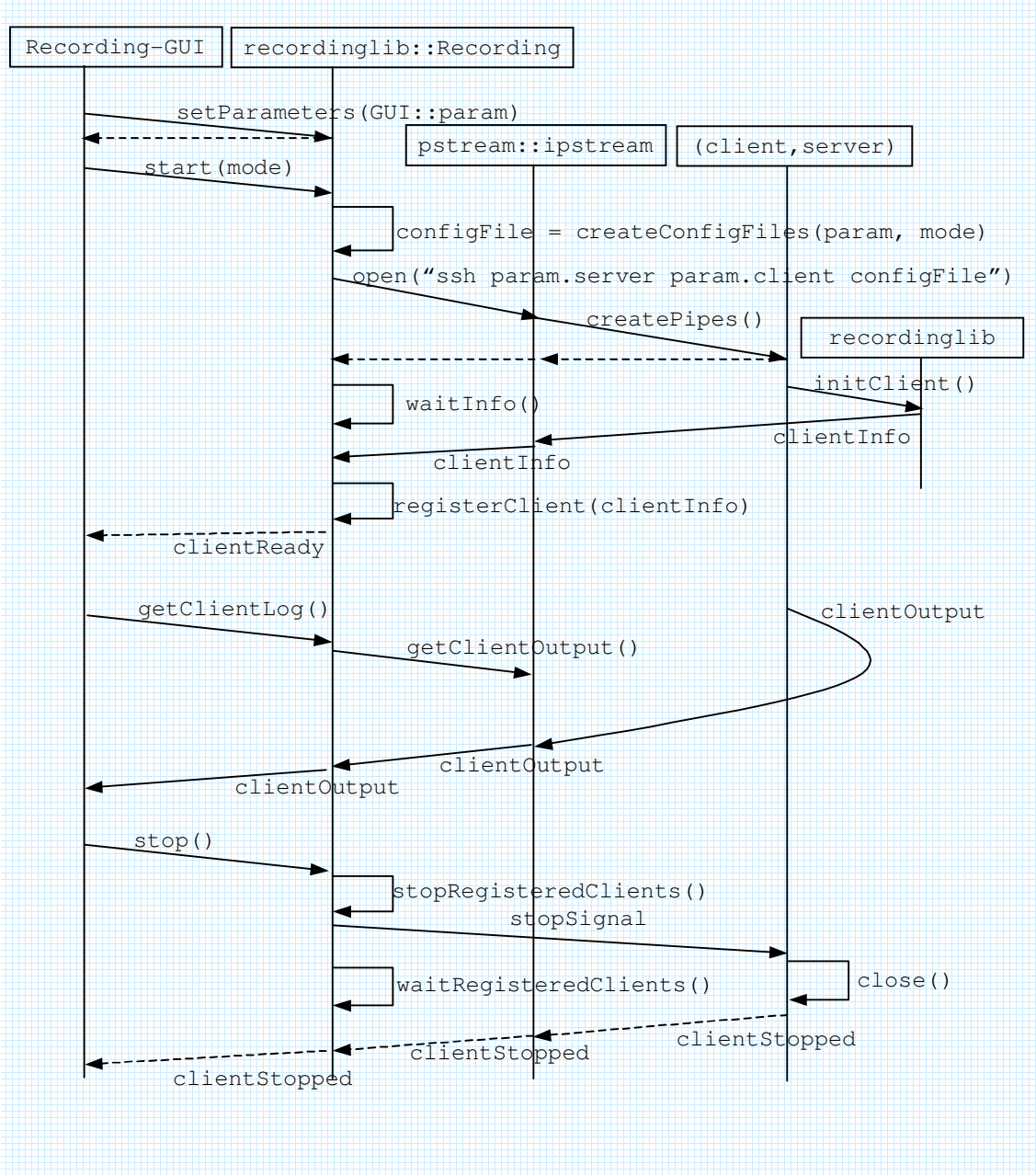


Figura 31. Diagrama UML del control remot de client mitjançant la llibreria pstream.

La nova arquitectura està implementada íntegrament dins la *Recording-Lib* i com s'ha dit anteriorment es basa en la llibreria pública *pstream* que permet executar processos remotament i capturar-ne les seves sortides.

Així doncs, aquesta nova arquitectura dóna origen un nou tipus de *clients*, els ***Recording-Clients***. Aquests *clients* són aplicacions executables iguals que els *Smart Flow clients*, a excepció de la seva interfície de control remot. Realitzen les mateixes funcionalitats i de la mateixa manera, però s'activen i es controlen amb una *API* diferent.

En funció de com es volen executar els *clients* del sistema *off-line* es pot utilitzar una interfície de control o una altra, uns *clients* o els altres. Per tant, aquesta nova arquitectura substitueix totalment la de *Smart Flow*.

Això podria fer pensar que perdem l'objectiu inicial de compatibilitat amb *Smart Flow*. Però això no és ben bé així. Si bé és cert que en el procés es generen dos *clients* executables diferents, i només un funciona sobre *Smart Flow*, també és cert que tots dos *clients* comparteixen el mateix codi font.

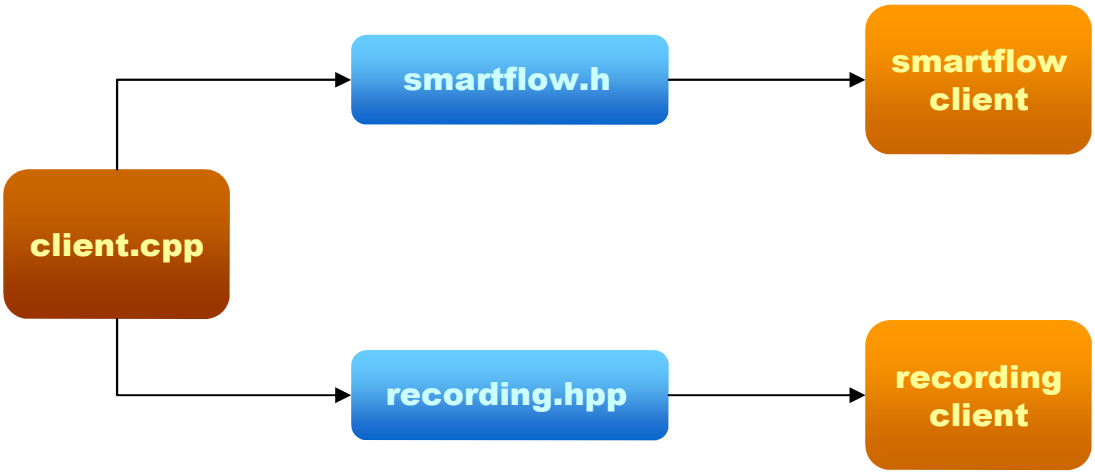


Figura 32. Esquema de generació de clients amb la interfície de Smart Flow o amb la Recording-Lib.

Per la implementació de l'esquema de la Figura 32 s'han utilitzat les instruccions de pre-compilació del llenguatge C/C++ de la manera més simple possible. Tots els *clients* que es volen utilitzar sota les dues arquitectures de control remot han d'utilitzar aquestes instruccions. D'aquesta manera per cada fitxer font d'un *client* és possible generar automàticament dos executables amb la mateixa funcionalitat, però amb diferents interfícies de control: una d'elles compatible amb *Smart Flow* i l'altra amb la *Recording-Lib*.

Això permet que les funcionalitats del nou entorn *off-line* es puguin executar amb la interfície de *Smart Flow* o amb la *Recording-GUI*. En tot cas el resultat serà el mateix i qualsevol canvi que es faci en un *client* es veurà reflectit en ambdues interfícies gràfiques.

A l'APÈNDIX A. Sistema de pre-compilació i generació de *Recording-Clients* es pot trobar codi d'exemple d'aquest procés de compilació que permet generar dos *clients* executables diferents d'un mateix fitxer de codi font.

4.5. Sincronisme

Un dels principals reptes en l'adquisició de dades de diferents sensors és el seu sincronisme. A nivell teòric dividim el sincronisme en dos tipus:

- *rate sync*, o sincronisme en freqüència. El ritme d'adquisició ha de ser igual per tots els sensors.
- *offset sync*, o sincronisme de fase. Tots els sensors han d'iniciar l'adquisició en el mateix instant.

A nivell pràctic, per garantir el sincronisme en una *smart-room* cal tenir en compte els següents punts::

- Captura d'imatges síncrona entre les diferents càmeres.
- Assignació síncrona de *timestamps*.
- Alineació de l'origen temporal de totes les càmeres.
- Control de possibles *dropped-frames*.

A la Figura 34 s'il·lustren aquests errors. No s'hi mostra específicament l'error produït per una mala assignació de *timestamps*.ja que, si el sistema assigna *timestamps* incorrectament es pot produir qualsevol tipus d'error de sincronisme.

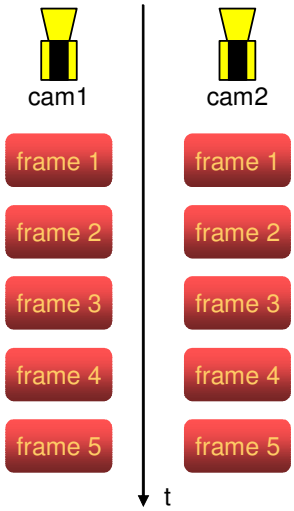


Figura 33. Esquema de gravació totalment sincronitzada.

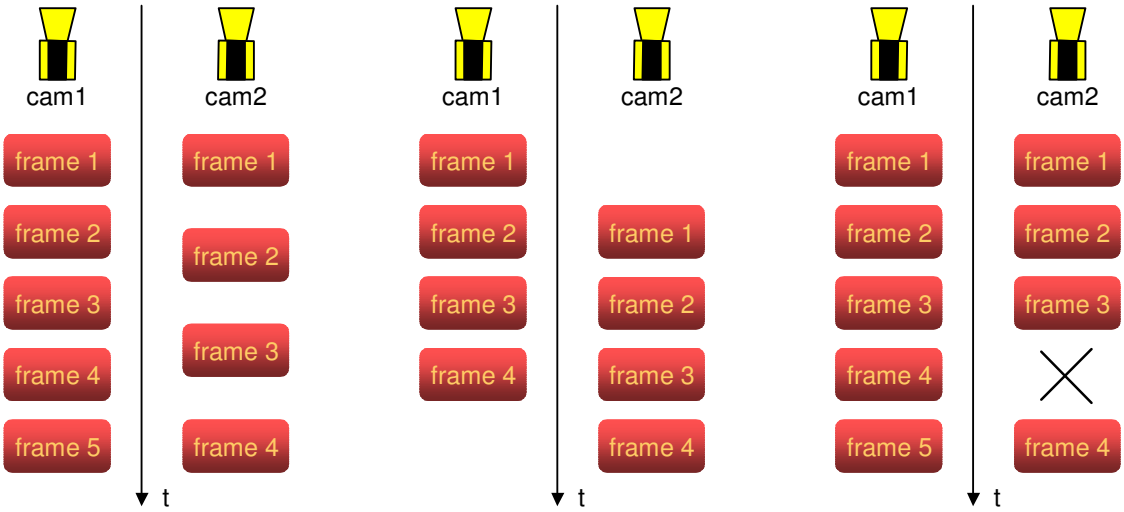


Figura 34. Esquemes dels possibles errors de sincronisme pràctics en una gravació multicàmera. Diferent freqüència de captura (esquerra), inici diferent de la captura (centre) i *dropped-frames* (dreta).

Com veurem en l'apartat 4.5.1, una de les petites però importants decisions pràctiques que s'han pres al llarg del projecte és la d'assignar el mateix número de *frame* a totes les imatges de les diferents càmeres que haurien de pertànyer al mateix instant de temps.

Amb aquesta premissa definim:

- Error de sincronisme (Δt): Qualsevol diferència temporal existent entre dues imatges de diferents càmeres amb el mateix número de *frame*.
- Error greu de sincronisme. Quan Δt és major o igual al 50% del temps entre *frames*[1] o quan imatges corresponents al mateix instant de temps tenen un número de *frame* diferent.

A continuació, a l'apartat 4.5.1, s'expliquen les diferents parts del sistema de sincronisme en la gravació. A l'apartat 4.5.2 s'explica el sistema de sincronisme en la lectura. El sistema de lectura ha de ser capaç de llegir una gravació de disc per generar els diferents *flows* de vídeo, un per cada càmera. Tot i que les dades a llegir

[1] S'utilitza com a temps entre *frames* al temps entre *frames* de la camera *master*. Per raons històriqued dins del projecte CHIL, s'escull la càmera 1 com a càmera *master*.

se suposen síncrones, com veurem en aquest apartat, la generació correcta de *flows* ha requerit un sistema de sincronisme especial sobre *Smart Flow*.

4.5.1. Sistema de sincronisme en una gravació

Separem el problema de sincronisme en l'adquisició de dades de vídeo en diferents parts:

- Captura síncrona de les càmeres.
- Assignació correcta de *timestamps*.
- Alineació de l'origen temporal.
- Control de *dropped-frames*.

4.5.1.1. Captura síncrona de les càmeres

Anomenarem Δt_c a l'error de sincronisme produït per una mala captura de les càmeres

S'utilitza el *MOTU Digital Timepiece*. Totes les càmeres disposem d'una entrada de sincronisme que està connectada al *MOTU*. Quan les càmeres reben el *trigger* provinent del *MOTU* fan la captura. A la Figura 35 es mostra aquest funcionament.

Cal fer notar que aquesta part no s'ha modificat respecte al sistema anterior desenvolupat per en Jordi Cenzano [11]. Com demostren els seus resultats d'aquest projecte i l'experiència pràctica, l'error de sincronisme en la captura és menyspreable ($\Delta t_c \approx 0$).

4.5.1.2. Assignació síncrona de timestamps

En aquest cas, sí s'ha canviat el sistema respecte al sistema precedent. Si anteriorment s'utilitzava el canal d'àudio del *MOTU* per enviar el mateix *timestamp* a totes capturadores, ara s'utilitza el rellotge intern de cada *PC* per assignar *timestamps*.

Això fa imprescindible que els rellotges interns de tots els *PCs* estiguin sincronitzats. Per assegurar aquest sincronisme s'utilitza el protocol *NTP* a la xarxa que formen els *PCs* d'adquisició. Aquest protocol monitoritza les diferències entre els diferents rellotges interns i els va corregint per evitar possibles derives[1].

Amb aquesta solució també s'aconsegueix evitar el problema del *offset* del senyal de *Linear Time Code* (LTC) del *MOTU*. En el sistema d'assignació de *timestamps* anterior capturava el senyal de LTC a través de la capturadora d'àudio i l'utilitzava com a marca temporal. La falta de control sobre l'*offset* temporal d'aquest senyal respecte la imatge capturada derivava en algunes ocasions en una mala sincronització.

El nou mecanisme funciona com es mostra en l'esquema de la Figura 35. Quan la capturadora de vídeo del *host* rep una nova imatge el *client* de captura que s'està executant li assigna el *system time* com a *timestamp*.

[1] Com es veu en l'apartat 3.2, el nou sistema *off-line* incorpora un mecanisme que comprova l'estat del sincronisme *NTP*. Per més informació sobre el protocol *NTP* consultar les referències [12] i [13].

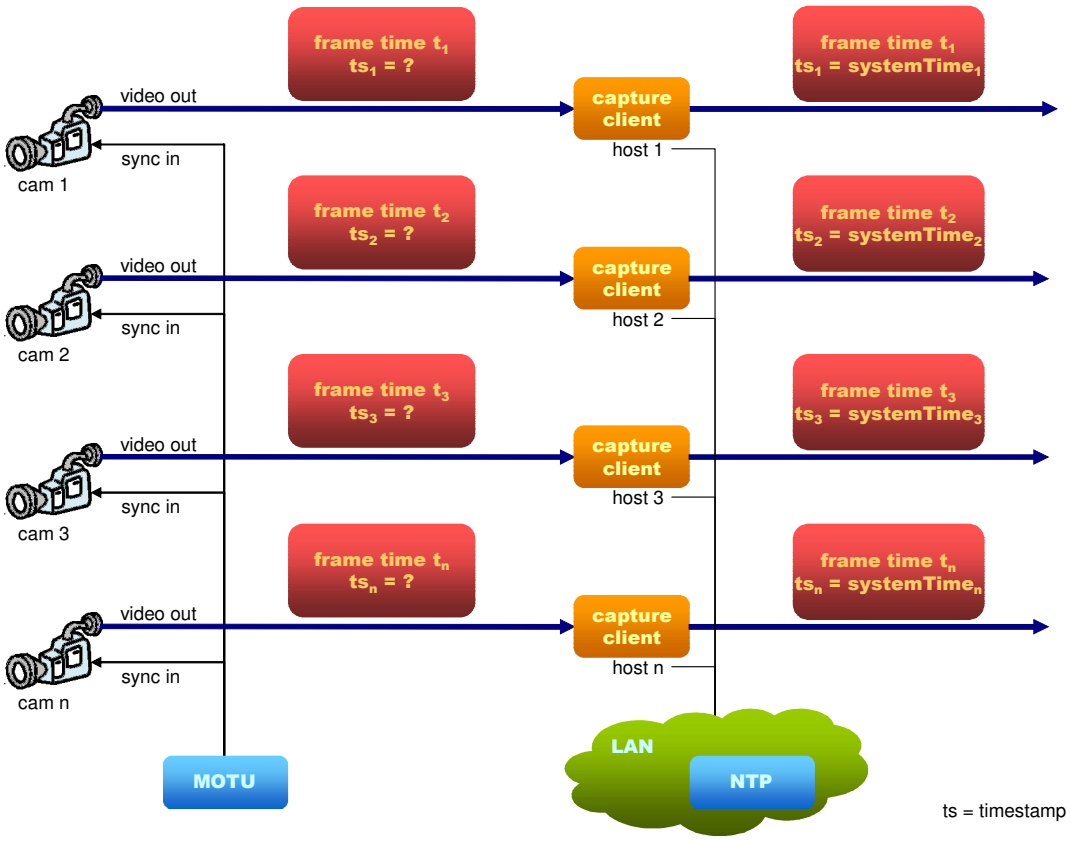


Figura 35. Esquema de sincronisme en la captura i l'assignació de timestamps.

Per tant, l'error de sincronisme introduït per un error en l'assignació de *timestamps* és deurà a la deriva entre els rellotges interns de cada host i l'anomenarem Δt_h.

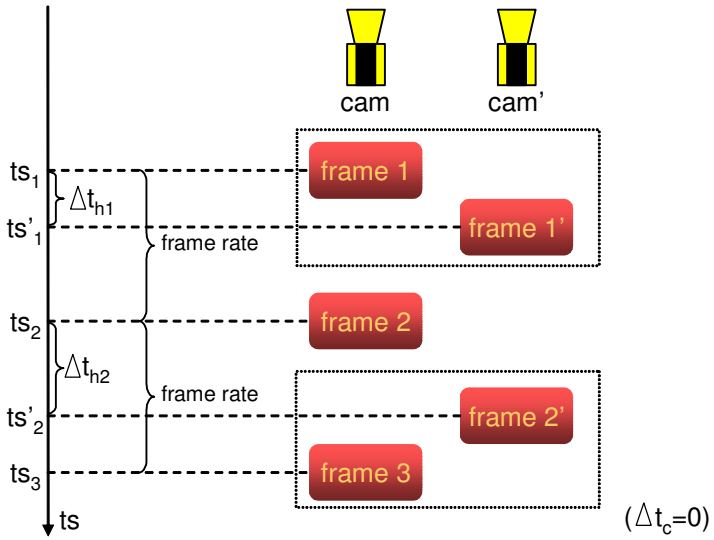


Figura 36. Esquema d'un error de sincronisme provocat per una mala assignació de timestamps.

A la Figura 36 s'ha suposat Δt_c=0, per tant els *frames* de les dues càmeres haurien d'estar totalment alineats. No obstant, si els hosts no estan sincronitzats es produirà una assignació de *timestamps* diferent. En la figura es mostren dos casos, un error normal, i un error greu.

Veiem que podem tenir un error de sincronisme greu si[1]:

$$\Delta t_h \geq \frac{\text{framerate}}{2}$$

En aquest cas el *frame* d'una càmera està més pròxim al següent de l'altre, tot i ser capturades en el mateix instant.

El *frame-rate* d'ús habitual a la *smart-room* de la UPC és de 25fps, i per tant el temps entre *frames* és de 40ms. Per tant, s'ha de complir que

$$\Delta t_h \leq 20\text{ms}$$

Les especificacions tècniques de *NTP* [12] garanteixen aquesta condició i a la pràctica s'ha comprovat que

$$\Delta t_h \leq 6\text{ms}$$

entre tots els *PCs* connectats a les càmeres.

Tot i que el marge d'error es considera suficient, cal comprovar l'estat del sincronisme abans de realitzar qualsevol gravació. Un dels inconvenients que té NTP és que per sincronitzar dos rellotges necessita aproximadament un temps equivalent al mateix error de sincronització.

4.5.1.3. Alineament de l'origen temporal de totes les càmeres

Degut a la que *Smart Flow* no garanteix que tots els *clients* de captura s'iniciïn al mateix instant de temps l'origen temporal de les imatges de cada càmera és diferent. Cada *client* de captura comença en un instant de temps diferent i el mateix passa amb els *clients* que escriuen les imatges a disc.

Hem comprovat que en un sistema de sincronisme en temps real com el que mostra la Figura 37 el nombre *dropped-frames* s'incrementa. Ni el transport de dades a la xarxa ni tampoc els *PCs* permeten realitzar les comprovacions computacionals necessàries per sincronitzar les càmeres i alhora mantenir constant el ritme necessari d'escriptura al disc.

És per això que s'ha considerat que la millor opció per solucionar aquest problema és realitzar un alineament temporal posterior com s'il·lustra a la Figura 38. Cada host inicia l'escriptura a disc en un instant diferent i posteriorment es porocessen totes les dades generades i s'alína l'origen temporal de totes les càmeres.

En l'apartat següent es descriu aquest post processat, doncs no només fa l'alineació de l'origen temporal..

[1] En general tindrem un error si $\Delta t_h + \Delta t_c \geq \frac{\text{framerate}}{2}$.

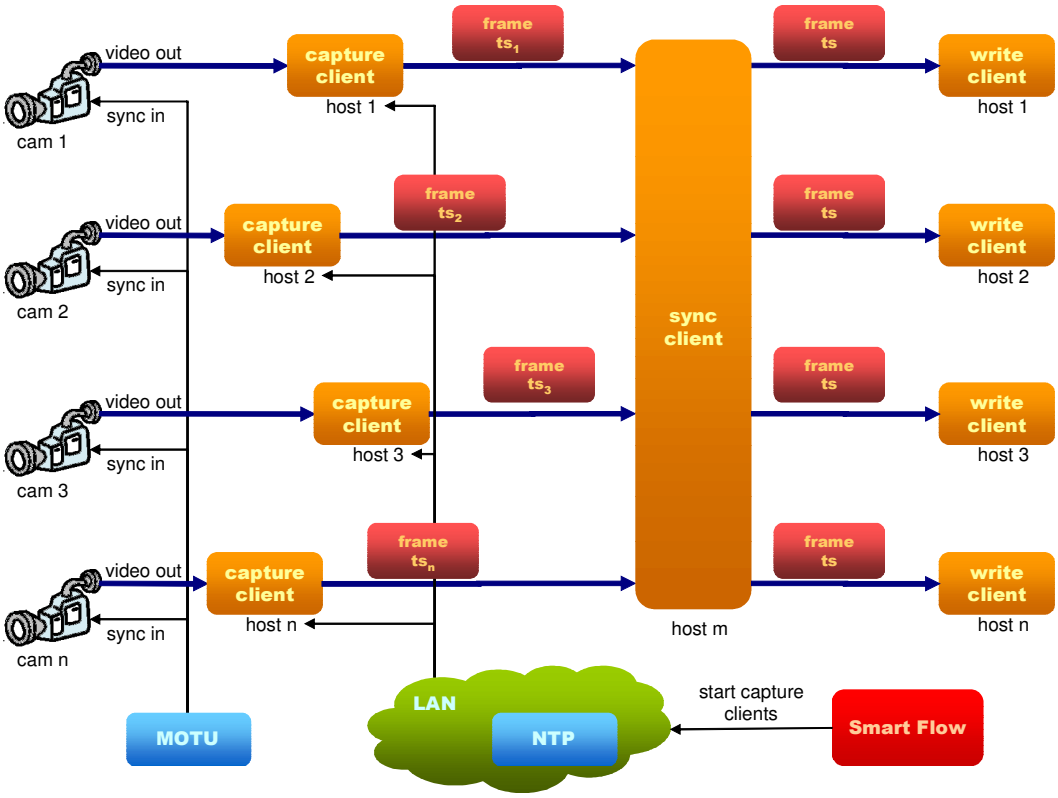


Figura 37. Esquema de sincronisme en temps real. Aquest sistema s'ha descartat.

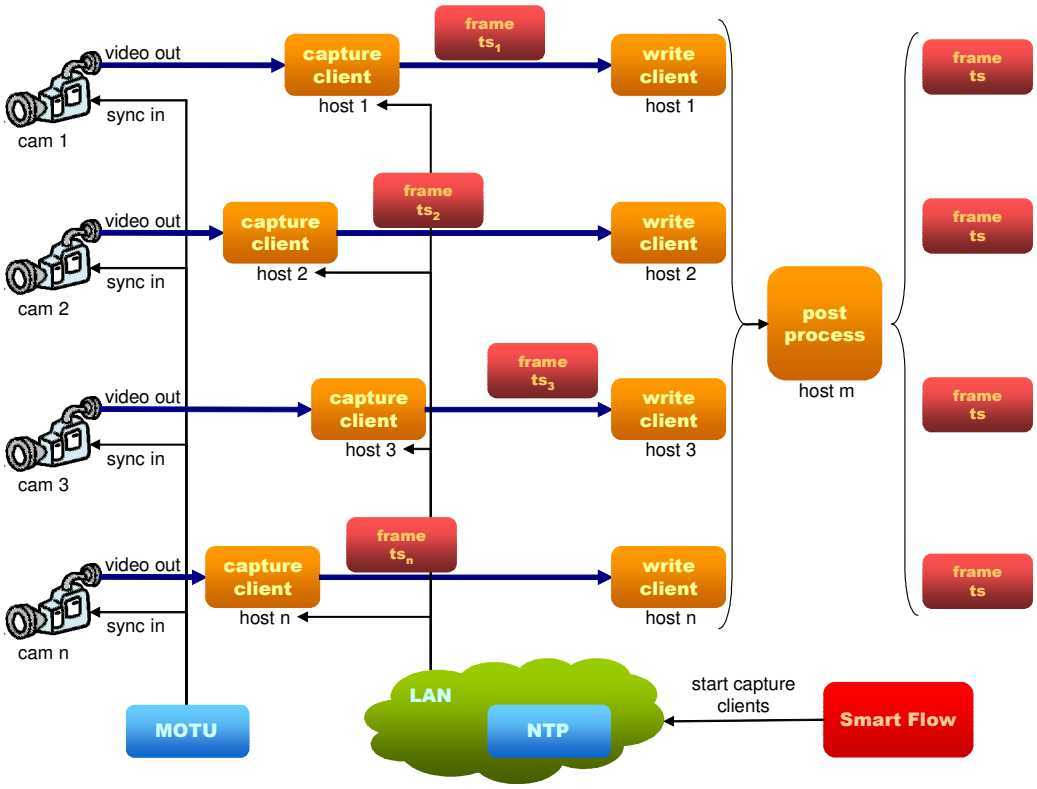


Figura 38. Esquema general del sistema de sincronisme utilitzat.

4.5.1.4. Post processat i update *d'una gravació.*

Com es detalla a l'APÈNDIX D. Estructura i format estàndard CHIL de gravacions, una gravació de vídeo consta principalment de:

- Les imatges (o *frames*) de cada càmera numerades seqüencialment.
- Un fitxer de *timestamps* per càmera on s'assigna un *timestamp* a cada imatge utilitzant el seu número de *frame*. Aquests fitxers s'anomenen `seq.index`.
- Altres fitxers de meta-dades.

Per tan, a cada *frame* li podem assignar les següents propietats:

- Càmera a què pertany.
- Numero dins la seqüència de vídeo.
- *Timestamp* associat.

Per comoditat farem servir la següent notació:

$ts_{i,n}$ = *timestamp* associat al frame *n* de la càmera *i*

L'objectiu de l'algoritme de post processat és aconseguir que la gravació resultant tingui les següents característiques:

- Totes les càmeres han de tenir el mateix nombre de *frames*.
- Totes les càmeres han de tenir el mateix origen temporal, i el mateix final.
- La numeració de *frames* ha de començar en zero, ser seqüencial i no tenir repeticions ni salts.
- Totes les càmeres han de seguir el *frame-rate* de la camera1, o càmera *master*[1].
- El *frame* n de cada càmera serà aquell que minimitzi l'error de sincronisme Δt_n on

$$\Delta t_n = |ts_{i,n} - ts_{master,n}|$$

A més l'algoritme ha de funcionar en el cas més general d'error de sincronisme, ha de considerar la possibilitat de:

- Errors de *rate-sync*.
- Errors de *offset-sync*.
- D*ropped-frames*.

A continuació s'il·lustra un cas típic d'error de sincronisme i els passos que segueix l'algoritme per processar una gravació[2]:

[1] S'escull la càmara 1 per raons històriques dins del projecte CHIL, però podria ser qualsevol. De fet existeix una versió del algoritme que escull la millor càmera com la *master*. Els parametres per escollir la millor càmera poden ser: el millor *frame-rate*, el *frame-rate* més estable, el mínim nombre de *dropped-frames*, etc...

[2] A l'APÈNDIX B. Post processat d'una gravació. Algoritme i resultats pràctics. es pot consultar el codi de l'algoritme de post processat.

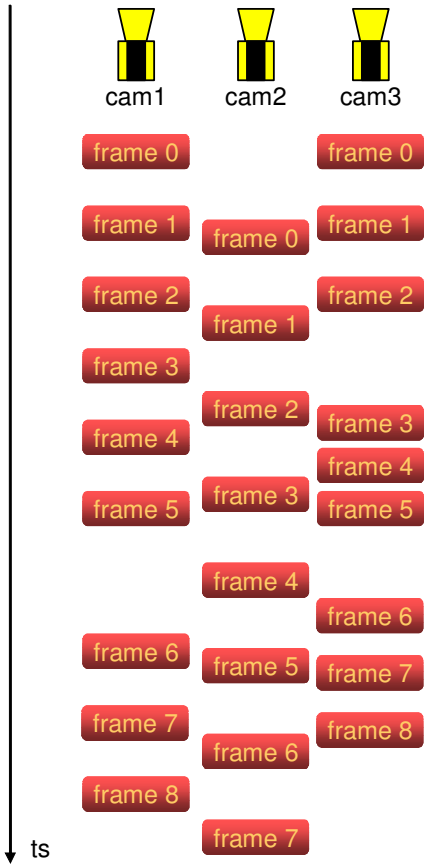


Figura 39. Gravació amb problemes de sincronisme que ha de ser processada.

1. Creació d'una càmera *master* seguint la càmera 1.

La **¡Error! No se encuentra el origen de la referencia.** ens mostra com es crea una càmera *master* artificial seguint el *frame-rate* de la càmera 1. Si es detecta un problema greu de sincronisme a la càmera 1, aquest es corregeix a la càmera *master*.

D'aquesta manera impedim que problemes ocorreguts a la captura de la càmera 1 afectin a tota la gravació.

2. Clipping de la càmera *master*.

La Figura 41 ens mostra com s'eliminen els *frames* de la càmera *master* de tal manera que totes les imatges comencin i acabin en el mateix instant. És a dir, la gravació començarà en el moment en que va començar l'última càmera i acabarà en el moment que es va aturar la primera càmera.

En el procés de *clipping* també és renumeren els *frames* de la càmera *master* de tal manera que comença sempre pel *frame* zero.

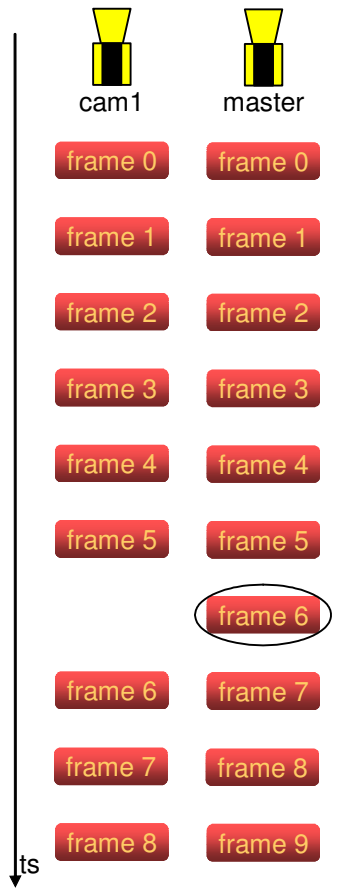


Figura 40. Creació de la càmera *master*.

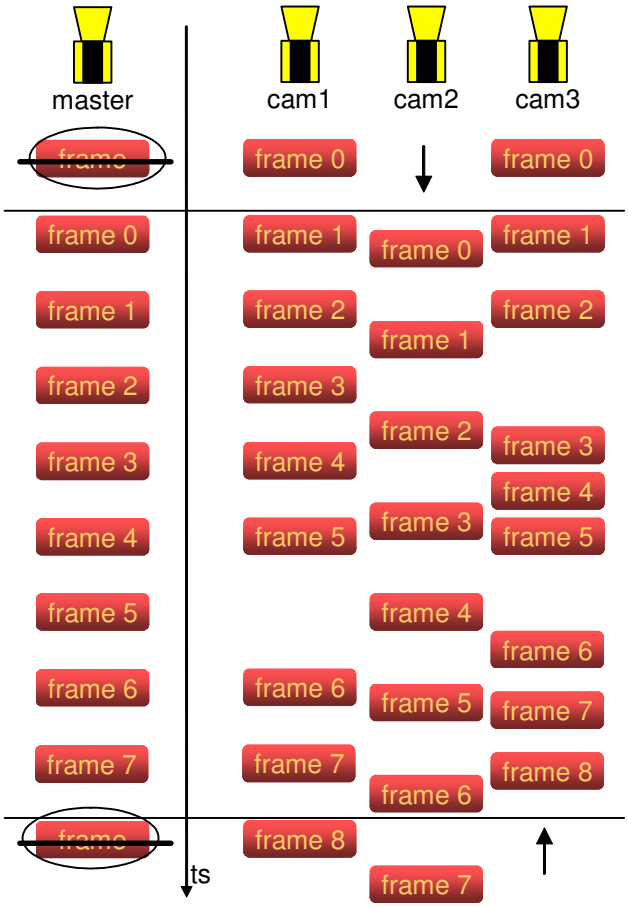


Figura 41. Clipping de la càmera *master*

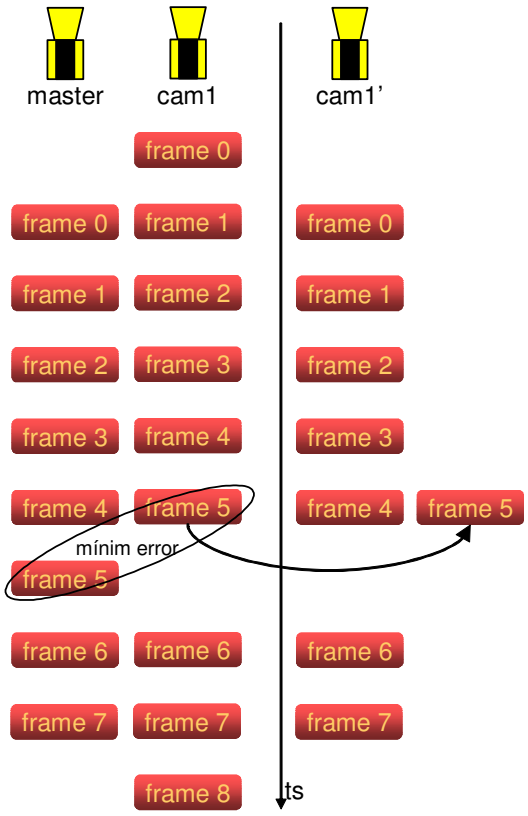


Figura 42. Sincronització de la càmera 1.

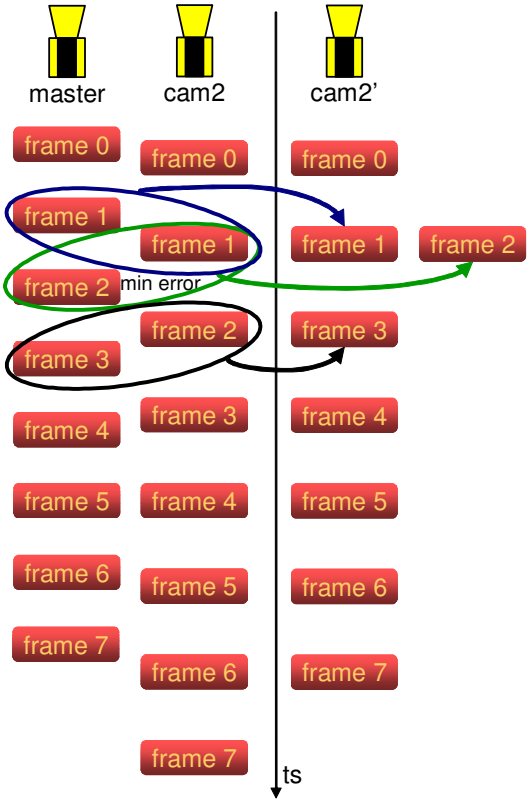


Figura 43. Sincronització de la càmera 2.

3. Sincronitzar cada càmera amb la *master* minimitzant l'error de sincronisme.

La Figura 42 ja ens mostra com en el procés de sincronització de la càmera 1 es renombren els seus *frames* seguint els de la càmera *master*. Com es pot observar això passarà amb totes les altres càmeres. D'aquesta manera, seguint totes la numeració de la *master* aconseguim també que totes tinguin el mateix número de *frames*.

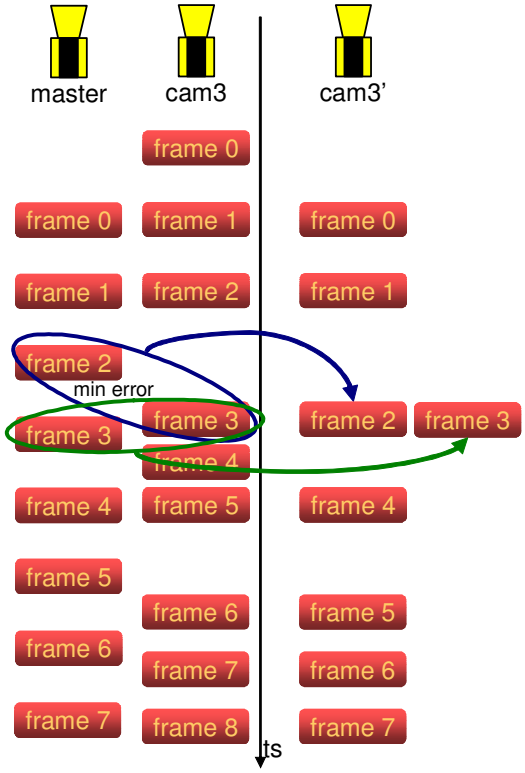


Figura 44. Sincronització de la càmera 3

La Figura 43 i la Figura 44 il·lustren com per cada *frame* de la càmera *master* l'algoritme busca el *frame* que minimitza l'error de sincronisme a la càmera a sincronitzar. En varies ocasions veu que l'algoritme ha de repetir un *frame*. En el cas de la càmera 3 també es veu el cas en que s'omet un *frame* de la càmera original, en aquest cas el *frame* 4 s'elimina de la gravació.

Quan l'algoritme detecta qualsevol d'aquests casos informa a l'usuari mitjançant un *log*.

4.5.2. Sistema de sincronisme per la lectura

L'objectiu del sistema de lectura no és altre que el de llegir les imatges d'una gravació prèvia i els seus *timestamps* associats per tal de generar el mateix *flow* de dades que van generar els *clients* de captura en el moment de fer la gravació. Per tant, degut a què només es poden generar *flows* sobre *Smart Flow*, el sistema de lectura només té sentit sobre *Smart Flow*.

Tenint en compte és que el sistema de lectura ha de ser dual respecte al d'escriptura, és a dir que només llegiran gravacions que hagin estat post processades, podrem utilitzar el coneixement que tenim sobre com estan estructurades i sincronitzades les dades d'una gravació i podrem utilitzar determinades presumpcions que ens poden són de gran utilitat. En concret les més útils són:

- Mateix origen temporal per totes les càmeres.
- Mateix nombre d'imatges per totes les càmeres.
- El mateix número de *frame* a totes les càmeres correspon al mateix instant de temps, o al més pròxim.

Una gravació amb aquestes característiques facilita molt la feina del sincronisme en la reproducció. Si assumim que la gravació té aquestes característiques, el sistema de lectura només ha de llegir les imatges de cada càmera ordenadament.

La informació dels *timestamps* només serveix per saber si en la captura va haver-hi un error, però no cal processar-la per sincronitzar els *clients* entre si.

El problema el tenim en el fet que *Smart Flow* no garanteix que tots els *clients* de lectura comencin a llegir en el mateix instant de temps, ni que tots s'executin a la mateixa velocitat. Per tant els *flows* generats pel sistema de lectura no estaran sincronitzats.

La solució escollida ha estat la d'implementar els clients de lectura amb un *trigger* d'entrada i un nou *client* de sincronisme de lectura que genera *triggers*.

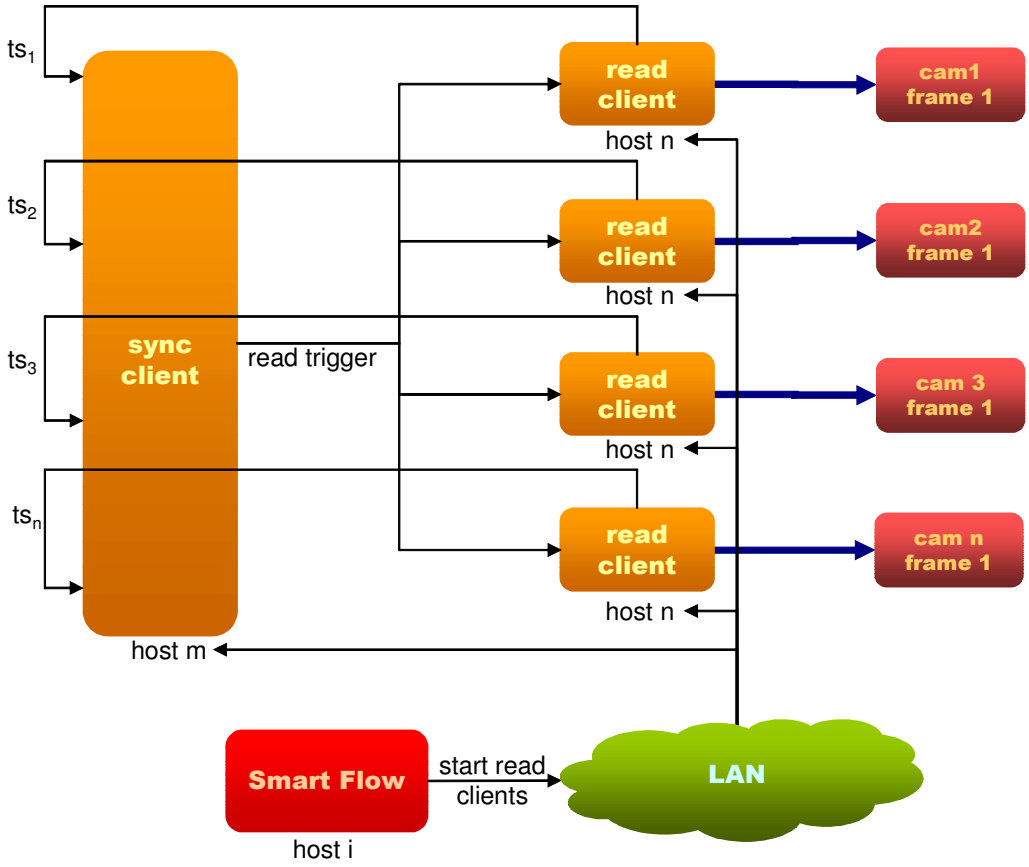


Figura 45. Esquema del sistema de sincronisme de lectura.

El mecanisme és el següent:

1. S'arranquen tots els *clients* de lectura des de Smart Flow.
2. Els *clients* de lectura esperen rebre un *trigger*.
3. S'arranca el *client* de sincronisme des de Smart Flow.
4. El *client* de sincronisme envia un *trigger*
5. El *client* de sincronisme espera rebre tots els *timestamps.*
6. Els *clients* de lectura reben el *trigger* i envien la imatge següent al *flow* de vídeo i el seu *timestamp* en un altre *flow* de dades.
7. Els clients de lectura esperen el temps que es correspon per poder seguir el ritme que marquen el *timestamps* de la gravació.
8. Els *clients* de lectura s'esperen rebre un nou *trigger*.

9. El *client* de sincronisme rep tots els *timestamps* i el mecanisme torna al punt 4 fins al final de la seqüència.

El *client* de sincronisme rep només el *timestamp* i no el *flow* de vídeo per evitar congestionar de la xarxa. Aquest mecanisme assegura que cap client s'avanci i el fet que el client de sincronisme rebi tots els *timestamps* li permet informar sobre problemes de sincronisme detectats a la gravació.

Gràcies a què aquest sistema s'ha estat utilitzant en el context del projecte *CHIL,* s'han rebut diferents comentaris i suggeriments que han donat com a resultat alguna millor interessant. Una d'elles és la implementació de l'opció d'utilitzar *trigger* global.

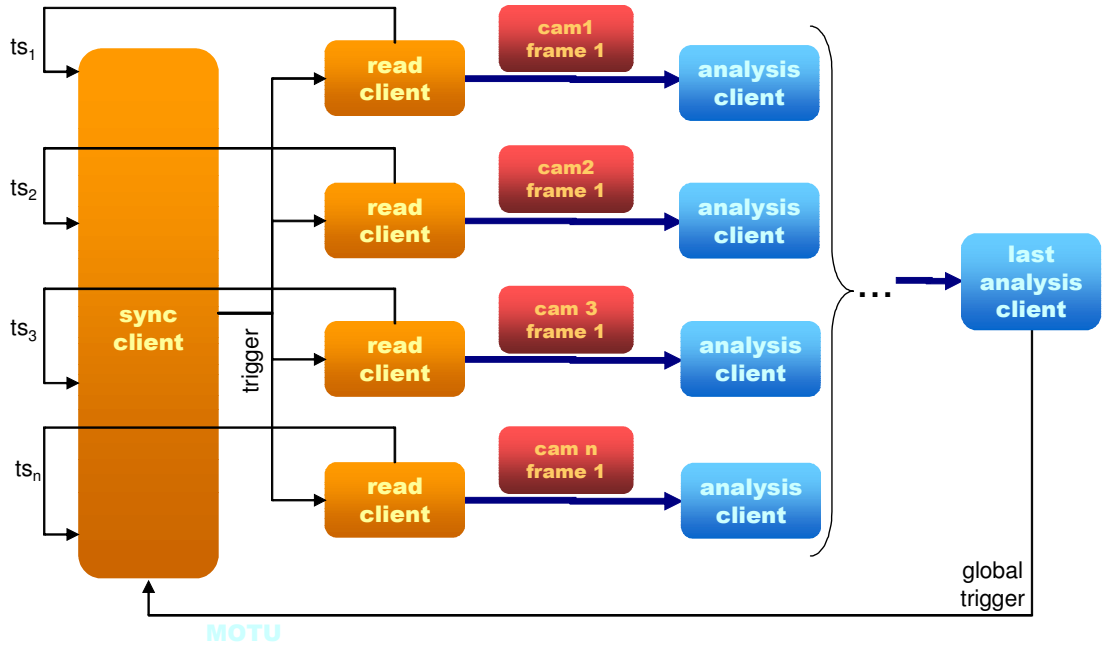


Figura 46. Esquema de lectura síncrona amb l'ús del *trigger* global.

La motivació és aprofitar que les imatges provenen de disc per llegir-les al ritme que es puguin processar, i no només al ritme que es van generar (temps real). Per això, com es mostra a la Figura 46 l'última versió del *client* de sincronisme (*triggerSync*) incorpora, no només un *trigger* de sortida que activa els *clients* de lectura, sinó que també incorpora un *trigger* d'entrada que està pensat per ser rebut de l'últim *client* de la cadena de processat.

La implementació d'aquest sistema de trigger global s'ha fet de tal manera que si es vol treballar al ritme normal, al que marquen els timestamps de les imatges, només cal realimentar el client de sincronisme com es mostra a la Figura 47.

En resum, gràcies al sistema de sincronisme de captura utilitzant el MOTU, al sincronisme de la xarxa que ens aporta *NTP*, a les correccions que realitza l'algoritme de post processat i, finalment, al client de *sincronisme* de lectura, el sistema *off-line* pot garantir la realització i reproducció de gravacions síncrones.

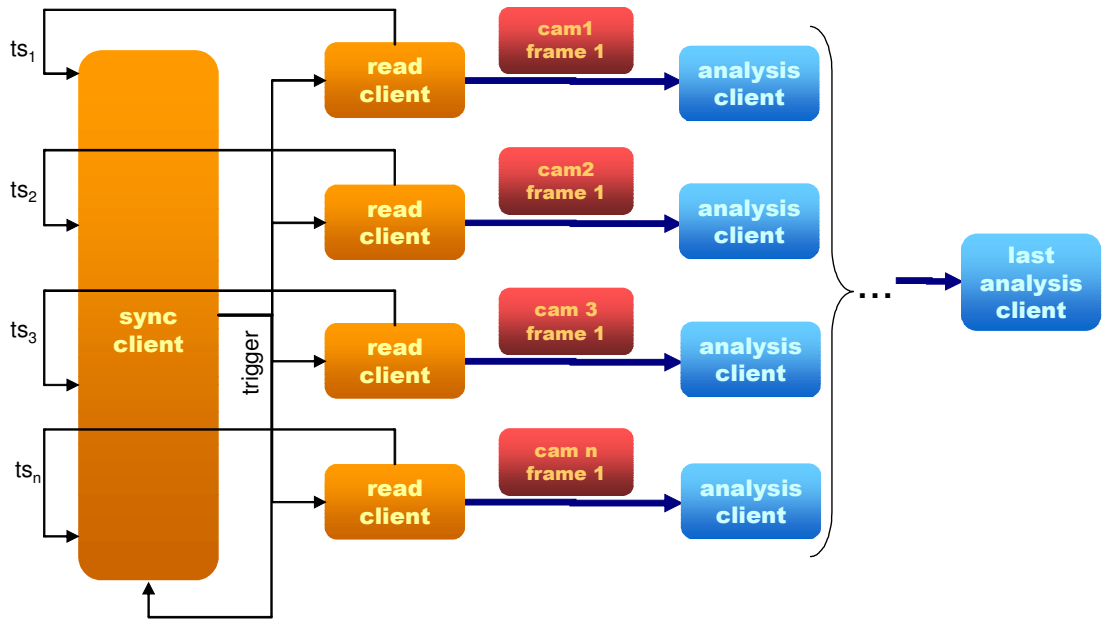


Figura 47. Esquema de lectura síncrona a temps real, és a dir, seguint els *clients* de lectura.

4.6. Altres

En aquest últim apartat es llisten en un format més breu d'altres aspectes de la implementació del nou entorn *off-line*. Aquests aspectes són de menor rellevància, però poden resultar d'utilitat al lector interessat.

4.6.1. Eines de desenvolupament de software

En el Grup d'Imatge i Vídeo de la UPC hi ha organitzat un entorn de desenvolupament de software basat en:

- Sistema multi-versió : *Subversion* [16]
- Sistema de *release* principal : *Scons* [15]
- Documentació : *Doxygen* [17] i *Wiki* [18]

Tot el software utilitzat per les aplicacions *on-line* està desenvolupat en aquest entorn. És per això que, tenint en compte que l'objectiu inicial d'aquest projecte és la fusió dels dos entorns de software existents en un de sol, s'ha considerat oportú desenvolupar el nou software utilitzant el mateix entorn que les aplicacions *on-line*, i no pas un de propi.

D'aquesta manera també es podrà aprofitar més fàcilment part del codi ja desenvolupat per l'entorn *on-line* i alhora el nou codi desenvolupat per treballar *off-line* podrà ser també aprofitat per altres aplicacions.

Com a contrapunt cal destacar que al integrar el nou software dins d'aquest entorn de desenvolupament podem crear moltes dependències amb unes llibreries que estan en continua evolució i en diferents nivells d'estabilitat. Això implica que, per tal que el sistema de *release* continuï funcionant correctament, caldrà fer un manteniment posterior un cop finalitzat el projecte.

Evidentment això implica un esforç encara més gran en la documentació. En aquest sentit, la funcionalitat està documentada el codi utilitzant *Doxygen* [17]. Això ens permet crear una documentació en format *html* de les diferents classes i mètodes implementats. A nivell d'algorísmica s'ha fet un esforç especialment gran per comentar tot el codi font, i finalment, a nivell d'usuari s'han generat diferents manuals i tutorials d'aquestes noves eines. Aquests manuals s'han integrat dins la pàgina wiki del grup d'imatge.

Aquí es llisten aquestes llibreries:

- XMLConfig
- xmlpp
- timestamp
- CIniFile
- CIndexFile
- Smart Flow
- UPC-SmartFlow (client.h)
- QT
- UPC::Count
- Pstream
- v4l
- ImageMagick++
- FastImage

4.6.2. Escalabilitat del sistema

La implementació de la *Recording-GUI* i la *Recording-Lib* no ha presentat cap dificultat per donar-li al sistema la possibilitat d'ampliar el nombre de càmeres. Però, el sistema de lectura basat en *Smart Flow* si té una dificultat important.

Com s'explica en l'apartat 4.5.2, el sistema de lectura utilitza un *client* de sincronització que té tants *flows* d'entrada com càmeres hi ha en el sistema. Per permetre que el sistema sigui escalable cal tenir una versió d'aquest *client* amb un nombre de *flows* configurable. Però com s'explica a l'apartat 4.2.2 el nombre de *flows* és fix per cada client. Smart Flow no permet definir el nombre *flows* en temps d'execució.

Per això s'ha definit en temps de compilació. Per disposar de la necessària escalabilitat, el *triggerSync* s'inclou en el procés de compilació varies vegades amb un paràmetre de pre-compilació que permet generar executables amb diferent numero de *flows*.

```
Import('env')
//...


# Client versions. Each copy will have diferent number of flows:
//...
my_env_x2 = env.Copy()
my_env_x3 = env.Copy()
//...

//...
my_env_x2.Append(
    CPPFLAGS =['-D_USE_DBG_MSG','`xml2-config --cflags`','-DINPUT_FLOWS=2'],
    CPPPATH=[flow_inc_path],
    LINKFLAGS ='`xml2-config --libs`',
    LIBS = ['recording', 'v4l', 'sflib', 'timestamp','iniFile', 'xmlpp'],
    LIBPATH = [flow_lib_path]
)
my_env_x3.Append(
    CPPFLAGS =['-D_USE_DBG_MSG','`xml2-config --cflags`','-DINPUT_FLOWS=3'],
    CPPPATH=[flow_inc_path],
    LINKFLAGS ='`xml2-config --libs`',
    LIBS = ['recording', 'v4l', 'sflib', 'timestamp','iniFile', 'xmlpp'],
    LIBPATH = [flow_lib_path]
)
//...

//...
my_env_x2.Object( 'triggerSync_x2', 'triggerSync.cpp' )
my_env_x3.Object( 'triggerSync_x3', 'triggerSync.cpp' )
//...

//..
my_env_x2.Program( '#bin/triggerSync_x2', 'triggerSync_x2.o' )
my_env_x3.Program( '#bin/triggerSync_x3', 'triggerSync_x3.o' )
//...
```

Figura 48. Fitxer de configuració de *SCons* que genera diferents executables d'un mateix fitxer font.

En la Figura 48 es veu com la variable *INPUT_FLOWS* canvia en cada compilació del mateix *client [15]* generant executables amb diferent nom. El codi font del client utilitza aquesta variable per generar i controlar un nombre indeterminat de *flows*.

4.6.3. Drivers per Linux

Un dels aspectes crítics que s'han de tenir en compte alhora de migrar de Windows a Linux és la disponibilitat o no dels *drivers* necessaris. En el cas que ens ocupa, el dispositiu hardware que utilitzem és la capturadora de vídeo *Osprey 210* de *ViewCast Corporation* [24].

Si bé en aquest cas el fabricant no proporciona cap *driver* específic per Linux, sí que ens diu que Linux té suport natiu del seu producte [23]. De fet, en l'entorn *on-line* ja s'estaven provant aquestes targetes amb els *drivers* natius de *Video For Linux*. i funcionaven correctament.

No obstant això, a la UPC s'utilitza un *wrapper* anomenat *v4l* que encapsula aquest *drivers* i en simplifica l'ús. Durant el desenvolupament de la *Recording-Lib* s'ha detectat que *v4l* mostra un error quan s'atura la captura.

Quan s'aturen els *clients* de captura des de *Smart Flow*, reporten un "*assertion failed*" i s'avorta l'execució del *client*. Aquest error no afecta al sistema *on-line* perquè quan s'atura la captura es vol aturar el processat, mentre que el sistema *off-line* vol seguir treballant.

Per tal de poder controlar aquest error s'ha modificat el codi font del *wrapper* canviant la instrucció `assert()` per una excepció.

```
// File: v4l.cpp

...

// This method has been modified by Albert Gil to avoid an uncontroled error
// It must be fixed.
const unsigned char* Video4Linux::buffer()
{
  assert( _state!=stopped );

  _vmap.frame = _bufferIndex;

  // Removed to avoid smartflow clients final abort.
  //assert( ioctl( _fd , VIDIOCSYNC , &_vmap.frame ) != -1 );

  //Changed to exception/error control.
  if ( ioctl( _fd , VIDIOCSYNC , &_vmap.frame ) == -1 )
    throw std::logic_error("v4l: ioctl( _fd, VIDIOCSYNC, &_vmap.frame ) == -1");

  return _frameBuf + _vm.offsets[_vmap.frame];
}

...
```

Figura 49. Modificació realitzada al *wrapper v4l* de *Video For Linux*.

Tot i aquesta modificació, l'error necessita d'un estudi específic per localitzar-ne la causa exacta.

4.6.4. Sistema de fitxers

En l'actualitat existeixen molt sistemes de fitxers diferents. Degut a la velocitat d'accés necessària en el procés de captura, l'elecció d'un sistema o altre té una influència decisiva.

En concret, el sistema que ha donat millors resultats és el ext2. Per això tots els PCs amb capturadora disposen també d'un disc amb aquest format.

Per tan, per tal que la gravació sigui correcte cal utilitzar aquest disc. En concret totes els PCs tenen muntat aquest disc a `/CHILrecording`.

4.6.5. Orientació a objectes

Durant el desenvolupament del software s'ha topat amb la necessitat de processar diferents estructures de dades i fitxers. En tots els casos s'ha intentat sempre generar una petita classe o llibreria per treballar-hi. En alguns casos, si la llibreria ja existia, s'ha ampliat o modificat per suportar les noves funcionalitats desitjades.

Els exemples més destacats són:

- La llibreria *timestamp*, que ha incorporat i modificat varis mètodes relacionats amb el seu format d'entrada i sortida estàndard, així com d'altres mètodes relacionats amb operacions aritmètiques.
- La creació des de zero de la llibreria CIndexFile que gestiona totes les operacions relacionades amb els fitxers de meta-dades seq.index[1].
- Les llibreries XMLConfig i xmlpp, que permeten treballar amb fitxers XML mitjançant objectes C++.

4.6.6. Nou format estàndard per gravacions

La nova estructura de gravació estàndard és basa en la definida per l'última especificació de la normativa del projecte CHIL, *CHIL Data Collection - Requirement for Interactive Seminars*, amb data de Juny del 2006.

Els canvis o adaptacions d'aquesta normativa més destacats són els següents:

- Els fitxers de calibració no tenen format .ini, sinó .txt i segueixen l'estructura pròpia de la UPC[2], TCameraParameters.
- Totes les càmeres han de tenir el mateix nombre d'imatges i el mateix origen temporal.
- Els fitxers seq.ini han de complir el format *.ini* estàndard, han d'estar dividits en seccions i tenir-hi les corresponents entrades.
- Tots els directoris s'escriuen en minúscules.
- Totes les gravacions han d'incloure els fitxers de configuració de *Smart Flow* per ser llegides.

4.6.7. QT com a llibreria gràfica

Per la implementació de la Recording-*GUI* s'ha escollit la llibreria QT de Trolltech principalment per dos motius:

- Paral·lelament a aquest projecte s'han desenvolupat d'altes aplicacions gràfiques en l'entorn de la *smart-room* de la UPC i totes elles han utilitzat QT[3]. Per facilitar el posterior manteniment de l'aplicació s'ha utilitzat una tecnologia coneguda pel grup.
- El fet que QT sigui una llibreria per C++ facilita molt la comunicació amb la *Recording-Lib*.

---

[1] Per més informació sobre aquests fitxers consultar a l'APÈNDIX D. Estructura i format estàndard CHIL de gravacions. A l'APÈNDIX B. Post processat d'una gravació. Algoritme i resultats pràctics. hi podem trobar la documentació de la classe CIndexFile en format Doxygen.

[2] Les llibreries i utilitats relacionades amb els paràmetres de calibració de càmeres i desenvolupades a la UPC treballen sobre uns fitxers amb un format concret. Els fitxers descriptors utilitzats en el projecte CHIL tenen un format diferent. Per tal de evitar la modificació d'aquestes llibreries i utilitats es transformen els fitxers de calibració del projecte CHIL al estàndard intern de la UPC.

[3] L'autor vol destacar la informació extreta del codi de l'aplicació *DisplayAcousticEvents* de Carlos Segura. Ha servit d'ajuda per al desenvolupar el mòdul *checkCalibration* de la *Recording-GUI*.

No obstant això, degut a l'especial forma de compilació i *link* de QT, la *Recording-GUI* no s'ha pogut integrar dins del sistema de compilació global gestionat per *scons*.

4.6.8. XML com a llenguatge de configuració

Una de les decisions que s'han pres al llarg del projecte ha estat l'ús sempre que sigui possible del format descriptor XML. Per fer més útil aquest format, també s'ha participat en el desenvolupament de diferents llibreries que gestionen aquests fitxers d'una manera específica.

A l'APÈNDIX E. Fitxers de configuració podem trobar diferents exemples d'aquests fitxers. En concret destaca el fitxer de configuració del *showRoom*.

Capítol 5. Resultats

Si a l'inici de l'informe ens plantejàvem els diferents objectius a complir, en aquest apartat els revisem i n'exposem els resultats obtinguts. També es presenten els resultats d'alguns dels tests realitzats al sistema i es mostra quin nivell d'estabilitat del sistema s'ha assolit.

5.1. Objectius aconseguits i limitacions del sistema

- Sistema *off-line* sobre *Smart Flow*

S'ha provat que *Smart Flow* garanteix un nivell de *dropped-frames* suficientment baix com per generar gravacions de qualitat en múltiples càmeres. No obstant, per assegurar aquest nivell de qualitat cal que no s'executin processos amb una càrrega computacional o d'accés a disc alts als *PCs* on es capturen i guarden les imatges

Per altra banda s'ha comprovat que amb *Smart Flow* no es pot garantir l'alineament temporal d'una gravació multicàmera i fa necessari un post processat. El problema és que no es pot controlar amb l'exactitud desitjada l'instant de temps en el qual els diferents *clients* comencen la gravació.

Un dels punts més negatius de l'ús de *Smart Flow* és la impossibilitat de controlar-ne els *daemons* des d'una altra aplicació que no sigui la pròpia *GUI* de *Smart Flow*. Això fa que externament no es puguin utilitzar *flows*, dificultant així la compatibilitat entre els sistemes *on-line* i *off-line*.

- Estandardització de les dades generades per l'entorn *off-line*

Al llarg del projecte s'ha arribat a un consens al grup d'imatge que ha permès generar un nou format estàndard per les futures gravacions, grans o petites, realitzades en l'entorn de la *smart-room* de la UPC.

S'han creat les eines necessàries per generar dades en aquest format, i per poder llegir-les posteriorment. També s'han generat les eines per poder actualitzar gravacions més antigues, o per corregir gravacions que no compleixin les especificacions de sincronisme definides en el nou format.

Gràcies a aquestes eines s'han actualitzat totes les gravacions generades des de l'any 2005 en el context del projecte CHIL.

- Sistema dual i compatible

El sistema desenvolupat és totalment dual: tota gravació generada amb el sistema d'escriptura pot ser reproduïda amb el sistema de lectura.

També s'ha aconseguit la compatibilitat entre el sistema de lectura i el de captura, són totalment intercanviables.

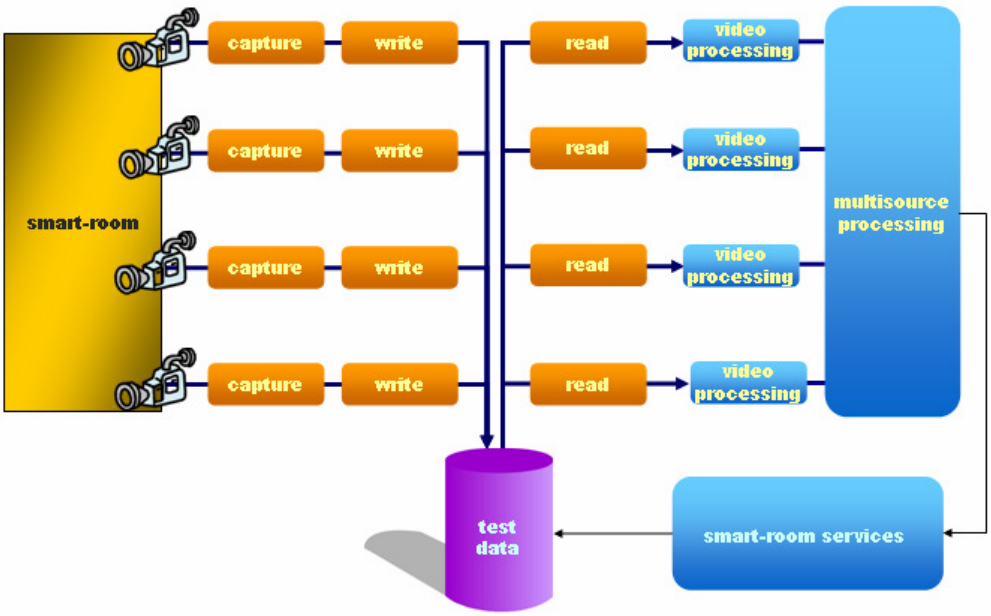


Figura 50. Esquema global de dualitat (escriptura/lectura) i compatibilitat (captura/lectura).

Actualment és possible repetir tota mena d'experiments a la *smart-room* de la UPC de manera *off-line*. Només cal generar gravacions amb el nou sistema i llegir-les després amb les noves eines desenvolupades.

- Base de dades de meta-dades

Si bé existeix un *link* entre la *Recording-GUI* i la base de dades que permet afegir una gravació nova de manera força automàtica, no s'ha aconseguit integrar totalment la base de dades en el sistema de gestió *off-line*.

- Interfície gràfica

La *Recording-GUI* és una interfície que permet controlar tot el sistema *off-line* de manera còmoda i segura.

Pel que fa a la compatibilitat amb la interfície de *Smart Flow*, els resultats no són exactament els desitjats. Si bé els *clients* que s'executen amb la *Recording-GUI* provenen del mateix fitxer de codi font que els clients de *Smart Flow*, els executables són diferents.

No obstant, la *Recording-GUI* permet automatitzar la configuració de Smart Flow per treballar amb el sistema *off-line* reduint així el cost d'utilitzar la seva interfície.

5.2. Tests

A la taula següent hi veiem els resultats de diferents gravacions realitzades i cronometrades.

Gravació	Número càmeres	Mida de la gravació	*Dropped frames*	Post processat	*Temps total aproximat (gravació , transferència i processat)*
3' 22''	5	5GB	0 de 5083	7' 42'''	20min
11' 56''	5	13GB	15 de 17923	41' 26'''	1h 20min
24' 42''	5	24,9GB	0 de 37050	3h 14' 28'''	4h 30min
5' 58''	3	2,7GB	0 de 8983	8' 51'''	15min
8' 53''	3	4,7GB	0 de 13338	17' 20'''	30min
26' 55''	3	9,2GB	12 de 40393	1h 26' 6'''	2h

Taula 4. Tests

Primer de tot que els temps de post processat son molt elevats. Si els observem en detall ens adonem que linealment amb el número de càmeres, i de manera exponencial amb el temps de gravació.

Si bé l'increment lineal amb el numero de càmeres sembla lògic i comprensible, l'increment exponencial en funció del temps de gravació no. L'algorísmica del processat hauria de ser lo suficientment eficient com per evitar aquest efecte. Com veurem a l'últim capítol aquest és un dels punts a millorar del sistema.

Per altra banda, si ens fixem en el nombre de *dropped-frames* veiem que és nul en la majoria de gravacions, però elevat en algunes. També observem que no té una relació directa ni amb el nombre de càmeres ni amb el temps de gravació.

Gràcies a què les gravacions de cada càmera és realitza totalment de manera local al *PCs* corresponent, sembla lògic que el nombre de càmeres no afecti al número de *dropped-frames*, cada una treballa independentment. Però sí que sembla lògic que mantenir un nivell alt d'escriptura a disc és més complicat com més temps es vol mantenir. No obstant, els tests demostren que els *PCs* poden mantenir un ritme d'escriptura alt durant 25 minuts sense masses dificultats. En concret, els *dropped-frames* detectats es concentren en només una de les màquines i són deguts a què d'altres usuaris han realitzat operacions que l'han sobrecarregat.

5.3. Estat final del sistema

A la següent taula es llisten els diferents sistemes i *clients* i es detalla el seu grau d'estabilitat (*alfa*, *beta*, *gamma*, *gold*) [28] així com la seva utilització actual.

Sistema / software	Estabilitat	Tests i utilització
Sistema de captura	Gamma /Gold	Utilitzat per tots els desenvolupadors. S'ha utilitzat en dues presentacions públiques de la *smart-room*. No presenta cap problema.
Sistema d'escriptura	Alpha	L'han utilitzat uns pocs usuaris i amb supervisió de l'autor. Actualment s'estan afegint nous formats i l'ampliació de funcionalitat porta inestabilitat.
Sistema de lectura	Gamma	Ha estat utilitzat per diferents desenvolupadors durant les Avaluacions 2007 del projecte CHIL. No ha donat cap problema.
Sistema de sincronisme	Gamma	La captura síncrona utilitzant el *MOTU* i el sincronisme de xarxa basat en *NTP* han estat àmpliament utilitzats pel grup CHIL-UPC.
Recording-GUI	Alpha	Només l'ha utilitzat l'autor. Requereix una millora de l'aparença.
Post processat	Gamma	Tot i que només l'ha utilitzat l'autor el sistema ha servit ja per actualitzar moltes gravacions d'anys anteriors del projecte CHIL.
Automatització de *Smart Flow*	Beta	Només l'ha utilitzat l'autor. Pot necessitar una actualització al canviar la versió de *Smart Flow*.
Sistema de calibració	Aplha	Necessita un increment de les funcionalitats i una integració amb els *scripts* de *Matlab*.

Taula 5. Estabilitat i estat actual del nou sistema off-line

Veiem que el sistema té actualment un nivell d'estabilitat elevat i que algunes de les seves parts estan en ple rendiment. Per altra banda, algunes, com la interfície gràfica, encara no s'han utilitzat.

D'altres, com el sistema d'escriptura, tot i existir-ne una versió estable actualment estan ampliant-se i per tant les últimes versions no han estat encara validades.

Capítol 6. Valoracions i millores

El principal objectiu d'aquest projecte era canviar el sistema de gravació i calibració existent en la *smart-room* de la UPC, el denominat sistema *off-line*, per un de nou compatible amb el sistema *on-line* basat en l'arquitectura de *Smart Flow*.

En aquest sentit podem dir que el projecte ha estat un èxit, doncs actualment es poden realitzar totes les funcionalitats del sistema *off-line* utilitzant la interfície de *Smart Flow*.

No obstant, en aquest apartat es pretén valorar la feina feta tenint en compte els objectius inicials, les dificultats i facilitats trobades, i els resultats finals. Tenint en compte aquestes valoracions també llistarem algunes de les possibles millores implementables en un futur.

6.1. Valoració d'objectius

Tal i com dèiem, l'objectiu principal del projecte s'ha assolit plenament. Actualment es poden realitzar el mateix tipus de gravacions i calibracions que anteriorment, però utilitzant *Smart Flow*.

S'han assolit també d'altres objectius importants. Un dels més ben valorats dins del grup d'investigadors que treballen a la *smart-room* és el de la creació eines duals. Aquestes eines duals els permeten reproduir experiments còmodament sense usuaris presents a la *smart-room*, i els dóna una eina universal per llegir qualsevol gravació generada en el context del projecte CHIL. En aquest sentit la valoració de la feina feta és força positiva tenint en compte que la dificultat elevada del disseny i la implementació dels diferents sistemes de sincronisme sobre *Smart Flow*..

També destaca l'eina que post processa gravacions actuals i, alhora, també permet actualitzar gravacions antigues. De fet, aquest sistema s'ha utilitzat, no només per actualitzar-ne d'antigues, sinó també per adaptar algunes de les últimes gravacions generades pel projecte CHIL i incloses dins del procés d'Avaluacions de Tecnologies 2007. Aquestes gravacions s'han actualitzat poc temps abans de la presentació d'aquest projecte, al Març del 2007. Concretament alguns d'aquests seminaris no complien amb el sincronisme esperat i s'han hagut d'actualitzar utilitzant aquest nou sistema. Un cop actualitzats s'hi ha pogut treballar utilitzant el nou sistema de lectura. En aquest sentit, els comentaris rebuts dels investigadors implicats en aquest procés d'avaluacions han estat positius.

A nivell d'enginyeria de software, es vol destacar especialment la compatibilitat aconseguida entre la *GUI* de *Smart Flow*, i la *Recording-GUI*. Aquest punt ha estat especialment important davant la impossibilitat de controlar els *daemons* de *Smart Flow.* En aquest moment semblava que la conclusió final del projecte seria que, si es volia treballar amb *clients* de *Smart Flow*, s'havia d'utilitzar la seva interfície i cap altra. La recerca d'una solució pel control remot d'aplicacions mantenint la compatibilitat amb *Smart Flow* ha estat llarga, però finalment, la solució basada en la llibreria *pstreams* que s'exposa a l'apartat 4.3 representa l'èxit que permet presentar uns resultats positius pel que fa a la interfície gràfica que integra tot l'entorn *off-line* i manté un nivell elevat de compatibilitat amb *Smart Flow*.

En aquest mateix sentit, una de les conclusions del projecte és que, tot i que *Smart Flow* ens permet realitzar totes les funcionalitats desitjades, no ens permet fer-ho tal i com voldríem:

- Degut a la falta de sincronisme al iniciar els *clients* de *Smart Flow*, ens trobem amb la necessitat de post processar les gravacions. Això fa que el temps necessari per realitzar-ne sigui superior al desitjat.
- La complexitat en el control dels *daemons* limita les possibilitats futures de la *Recording-GUI* ja que no es poden controlar totalment *clients* amb *flows* sense la *GUI* de *Smart Flow*.

En aquest sentit existeix un projecte anomenat *CHIL-Flow* que justament intenta prestar totes les capacitats de *Smart Flow* per línia de comandes. Això hauria d'eliminar aquesta limitació, i hauria de ser possible controlar *clients* amb *flows* des de qualsevol altra aplicació.

Una de les limitacions del sistema és el *link* amb la base de dades. L'objectiu era utilitzar aquesta base de dades per gestionar les gravacions i poder treballar amb elles. En el moment de presentar el projecte només existeix un *link* que facilita la introducció de noves gravacions a la base de dades.

Degut a la velocitat amb la que s'ha intentat desenvolupar aquest projecte no s'han desenvolupat tots els tests unitaris necessaris per garantir-ne l'estabilitat. Tot i això, part del software ha estat ja utilitzat àmpliament per usuaris i per tan se'n pot assumir l'estabilitat. A l'apartat 5.2 es detalla el nivell d'estabilitat de les diferents parts del nou sistema *off-line*.

Com conclusió final, i tenint en compte l'ús que ja s'ha fet del sistema, l'autor no té dubte que la introducció de les eines desenvolupades en aquest projecte seran de gran utilitat per tal de continuar millorant les tecnologies de reconeixement que es desenvolupen a la UPC. En aquest sentit la valoració és positiva, doncs el treball realitzat té ja una utilitat intensa. I just aquest era el seu objectiu últim.

6.2. Possibles millores i ampliacions del sistema

Gràcies a què les conclusions del projecte han estat positives, la UPC vol donar-ne continuïtat. Les idees sorgides al llarg del projecte han estat moltes. Moltes d'elles s'han implementat, d'altres s'han descartat, i moltes s'han aplaçat.

L'objectiu és que aquest projecte continuï amb la implementació d'algunes de les següents millores.

1. <u>Sincronització entre vídeo i àudio</u>

Tot el projecte està pensat per gestionar només dades de vídeo. Una de les extensions immediates d'aquest projecte és la sincronització amb el sistema de gravació d'àudio. Seria de gran utilitat desenvolupar un sistema de gravació i reproducció comú per ambdós tipus de dades.

2. <u>Actualització de la versió de *Smart Flow*</u>

La *smart-room* de la UPC ha d'estudiar la possibilitat d'actualitzar la versió de Smart Flow. Aquest canvi pot implicar moltes incompatibilitats amb els diferents sistemes implementats. Un d'ells el nostre, el sistema de gestió de vídeo *off-line*. Cal estudiar-ne l'impacte i assegurar-ne la compatibilitat.

Una altra possibilitat és estudiar també la possibilitat d'utilitzar *CHIL-Flow*. Una interfície que ha de permetre les mateixes funcionalitats que Smart Flow però controlable per línia de comandes. Aquesta opció ajudaria molt, tant a l'actual sistema off-line com a qualsevol sistema extern que vulgui automatitzar la configuració i controlar *clients* amb *flows*.

3. <u>Millores en l'algoritme de post processat</u>

Els resultats presentats a la Taula 4 motiven de sobre un esforç per reduir els temps de post processat de gravacions. Sembla clar que s'ha de millorar la gestió dels fitxers de *timestamps* (p. ex. els algoritmes de recerca en vectors) i optimitzar la transferència o còpia de fitxers.

En aquest sentit, una de les propostes que pot reduir aquest temps és la de modificar el sistema per tal que l'algoritme s'executi de manera distribuïda, utilitzant un *server* diferent per sincronitzar cada una de les càmeres amb la *master*.

Per altra banda, per tal que el post processat de gravacions sigui més útil, cal donar-li més flexibilitat fent-lo més parametritzable. Algunes de les opcions o paràmetres que s'han pensat són:

- Opció de fer només un *check* del sincronisme, sense modificar res.
- Opció de fer només un *clipping* de totes les càmeres.
- Primer i últim *frames* seleccionables per *timestamp*.
- Càmera *master* seleccionable.
- Sincronització amb dades de gravacions d'àudio.

4. Nous formats pel sistema dual de gravació / reproducció

L'èxit del sistema dual escriptura / lectura ha fet que es pensi en la creació de sistemes iguals per l'escriptura i lectura no només imatges en format *jpg* o *bmp*, sinó en estructures de dades utilitzades en processat d'imatges com ara màscares, vòxels, volums, etiquetes...

5. Millores a la *Recording-GUI*

La *Recording-GUI* és una interfície molt senzilla. És molt convenient una millora de les seves funcionalitats com a interfície gràfica:

- Millorar la interfície de l'assistent per gravacions.
- Fitxers de configuració que es guardin i es carreguin automàticament.
- Separar els *logs* dels diferents *clients*.
- Afegir controls sobre l'estat de les connexions de la xarxa.
- Afegir controls sobre l'espai disponible a tots els discos.
- Afegir un mòdul de gestió en SQL que permeti treballar amb totes les possibilitats de la base de dades.

6. Control de càmeres actives

A l'entorn de la *smart-room* de la UPC té diferents projectes engegats. Un d'ells és el d'afegir noves càmeres actives. El sistema *off-line* implementat permet afegir qualsevol número de càmeres. Però seria una bona opció que també permetés controlar també la posició *pan-tilt-zoom* d'aquest tipus de càmeres.

APÈNDIX A. Sistema de pre-compilació i generació de *Recording-Clients*

En aquest annex s'il·lustra amb un exemple com es generen dos *clients* amb la mateixa funcionalitat, a partir del mateix fitxer de codi font, però un compatible amb *Smart Flow* i l'altre controlable per la *Recording-Lib*.

Primer es mostra el fitxer SConscript que és utilitzat per Scons en el procés de compilació. Cal observar com un dels clients inclou el *path* i la llibreria de *Smart Flow* (`sflib`) i l'altre compila amb el *flag* `_RECORDING`.

```
Import('env')
Import('flow_inc_path')
Import('flow_lib_path')


#
# Smartflow clients building
#

sf_env = env.Copy()

sf_env.Append(
    CPPFLAGS =['-D_USE_DBG_MSG', '`xml2-config --cflags`' ],
    CPPPATH=[flow_inc_path],
    LIBS = ['sflib','recording', 'iniFile','indexFile','timestamp','xmlpp'],
    LIBPATH = [flow_lib_path]
)

sf_env.Object( 'lib/processRecording',  'processRecording.cpp' )
sf_env.Object( 'lib/cleanRecording',    'cleanRecording.cpp ' )
sf_env.Object( 'lib/transferRecording', 'transferRecording.cpp' )

sf_env.Program( 'bin/processRecording',  'lib/processRecording.o' )
sf_env.Program( 'bin/cleanRecording',    'lib/cleanRecording.o' )
sf_env.Program( 'bin/transferRecording', 'lib/transferRecording.o' )


#
# Recording clients building
#

rec_env = env.Copy()

rec_env.Append(
    CPPFLAGS =['-D_USE_DBG_MSG', '-D_RECORDING', '`xml2-config --cflags`' ],
    LINKFLAGS ='`xml2-config --libs`',
    LIBS = ['recording', 'iniFile', 'indexFile', 'timestamp', 'xmlpp'],
)

rec_env.Object( 'lib/rec_cleanRecording'     , 'cleanRecording.cpp' )
rec_env.Object( 'lib/rec_transferRecording'  , 'transferRecording.cpp' )
rec_env.Object( 'lib/rec_processRecording'   , 'processRecording.cpp' )

rec_env.Program( 'bin/rec_cleanRecording'     , 'lib/rec_cleanRecording.o' )
rec_env.Program( 'bin/rec_transferRecording'  , 'lib/rec_transferRecording.o' )
rec_env.Program( 'bin/rec_processRecording'   , 'lib/rec_processRecording.o' )
```

Ara veiem com s'utilitza el *flag* al codi del *client*:

```
/*!
 * This client cleans a recording directory.
 * It looks for recordings in a directory and removes them.
```

```
 *
 * \note This client uses basic smartflow library, not the cpp wrapper class.
 *
 * \author Albert Gil
 */

#include <string>
#include <iomanip>
#include <iostream>

#include <XMLConfig.hpp>
#include <recording.hpp>

#ifndef _RECORDING
#include <sflib.h>
#endif

using namespace std;

int main ( int argc, char** argv ) {

#ifndef _RECORDING
    // Smartflow varables
    sf_init_param iparam;
    sf_error err;
    sf_handle* sfHandle;
#endif

    // Config variables
    string configFileName;
    string outputLocalPath;

    try {

#ifndef _RECORDING
        // Initialize client parameters
        sf_init_param_init( &iparam );
        sf_init_param_set_name( &iparam, "cleaner" );

        // Create no flows
        sf_init_param_set_flow_count( &iparam, 0, 0 );

        // Initialize client
        sf_init_param_args( &iparam, &argc, argv );
        sfHandle = sf_init( &iparam, &err );
        if(err.error_code != SFE_OK) {
            throw string("Couldn't start system");
        }
#endif

        Recording::initClient();

        // Check configuration parameters
        // ...

        // Read config file
        // ...


        // Look for recordings
        // ...

        // Remove recordings
        // ...

        // End client
        exit(0);
    }
    catch( const std::exception& e ) {
      cout << e.what() << std::endl;
    }
}
```

APÈNDIX B. Post processat d'una gravació. Algoritme i resultats pràctics.

En aquest annex s'exposa el codi i la documentació del *client* de post processat. Hi trobem el codi font del *client* i la documentació de la classe que gestiona els fitxers seq.index, la classe CIndexFile.

El codi font de post processat és el següent

```
/*!
 * \brief This client reads a recorded session to synchronize it.
 *
 * Based on seq.index files from different cameras, it removes out of range images
 * and renames the others to minimize the error with a master cam.
 *
 * \note This client uses basic smartflow library, not the cpp wrapper class.
 *
 * \ingroup VideoSmartFlowClient
 * \author Albert Gil
 */


// System libraries
#include <string>
#include <iomanip>
#include <iostream>
#include <fstream>

// Smartflow library
#ifndef _RECORDING
#include <sflib.h>
#endif

// Utilities
#include <XMLConfig.hpp>
#include <timestamp.hpp>
#include <indexFile.hpp>
#include <iniFile.hpp>

// Recording variables
#include <recording.hpp>


#define MASTER_CAM 0

using namespace std;

int main ( int argc, char** argv ) {

#ifndef _RECORDING
    // Smartflow varables
    sf_init_param iparam;
    sf_error err;
    sf_handle* sfHandle;
#endif

    // Global variables
    vector<string> cameraName;
    vector<string> cameraName_tmp;
    xmlpp::XMLStringVector configCameraNames;

    // Config variables
    string basicPath;
    string imageExtension;
    double fpsExpected;
```

```
    string configFileName;

    // Seq files
    vector<string> indexFileName;
    vector<string> iniFileName;
    vector<CIniFile> iniFile;
    vector<CIndexFile> newFile;
    vector<CIndexFile> oldFile;

    try {
#ifndef _RECORDING
        // Initialize Smart Flow client parameters
        // ...
#endif

        Recording::initClient();

        // Check parameters
        // ...

        configFileName = argv[1];

        // Read configFile
        // ...


        size_t numberOfCameras = configCameraNames.size();
        string sessionPath     = basicPath /*+ "/" + sessionName*/;

        // Wtite log header to stdout
        // ...

        // Create index files
        newFile.resize( numberOfCameras, CIndexFile() );
        oldFile.resize( numberOfCameras, CIndexFile() );
        iniFile.resize( numberOfCameras, CIniFile()   );

        // Get camera names and read seq.index files
        for ( size_t i = 0; i < numberOfCameras; i++ ) {
            string name     = sessionPath + "/video/" + configCameraNames(i);
            string name_tmp = sessionPath + "/video/" + configCameraNames(i) + "_tmp";
            cameraName.push_back(name);
            cameraName_tmp.push_back(name_tmp);

            string indexName = name + "/seq.index";
            string iniName   = name + "/seq.ini";

            indexFileName.push_back ( indexName );
            iniFileName.push_back   ( iniName   );

            cout << "Reading files from " << configCameraNames(i) << "...   ";
            oldFile.at(i).read( indexFileName.at(i) );
            cout << oldFile.at(i).size() << " frames." << endl;
            if ( !iniFile.at(i).OpenIniFile( iniFileName.at(i).c_str() ) )
                throw string( "Error opening seq.ini file: " +  iniFileName.at(i) );

            // To make post processing an idempotent client,
            // index files shouldn't have repeated frames.

            oldFile.at(i).removeRepeatedFrames();
        }

        // Select master camera
        cout << endl << "Creating master camera at " << fpsExpected << "fps from "
             << configCameraNames(MASTER_CAM) << "..." << endl;

        CIndexFile master = oldFile.at(MASTER_CAM).getMaster(fpsExpected);

        // Create the output seq.index file from every camera seq.index
        cout << "Clipping master with all cameras...   ";
        for ( size_t i = 0; i < numberOfCameras; i++ ) {
            master.clip( oldFile.at(i) );
        }

        // Rename frames to start at frame zero
        master.renameBlocks();
        cout << "Master camera has " << master.size() << " frames." << endl;
```

```
        // Create new files
        for ( size_t i = 0; i < numberOfCameras; i++ ) {
            cout << "Synchronizing " << configCameraNames(i) << " index file with master..." << endl;
            newFile.at(i) = oldFile.at(i).getSync( master );
            assert ( newFile.at(i).size() == master.size() );
        }

        // Rename old image files and remove out of range images
        for ( size_t j = 0; j < numberOfCameras; j++) {

            // Get the vector of old images from old seq.index file from each camera.
            vector <IndexFileEntry> oldEntries = oldFile.at(j).getEntriesVector();
            vector <IndexFileEntry> newEntries = newFile.at(j).getEntriesVector();
            IndexFileEntry newEntry;
            IndexFileEntry prvEntry;
            IndexFileEntry oldEntry;

            // Check if every timestamp/frame is still present in final seq.index file
            cout "Removing clipped images files from " << configCameraNames(j) << "... " << endl;
            size_t oldSize = oldEntries.size();
            for ( size_t i = 0; i < oldSize; i++) {

                // Get old entry
                oldEntry = oldEntries.at(i);

                stringstream oldImageFileName;
                oldImageFileName << cameraName.at(j) << "/seq_" << setfill('0') << setw(6)
                                 << oldEntry.block << imageExtension;

                // If timestamp/frame isn't in final seq.index file, remove image file.
                if ( !newFile.at(j).hasTimestamp( oldEntry.ts ) ) {
                    call_system( "rm -f " + oldImageFileName.str());
                }
            }

            // Move camera images to temporal directory and mantain old directory
            cout << "Saving temporal files from " << configCameraNames(j) << "... " << endl;
            try {
                call_system ( "mv "    + cameraName.at(j) + " " + cameraName_tmp.at(j) );
            }
            catch( const exception& e ){
                cerr << "Error creating temporal files." << endl;
                throw logic_error(e.what());
            }
            try {
                call_system ( "mkdir " + cameraName.at(j) );
            }
            catch( const exception& e ){
                cerr << "Error creating camera directory." << endl;
                throw logic_error(e.what());
            }

            // Rename image files
            cout << "Creating new image files from " << configCameraNames(j) << "... " << endl;
            size_t newSize = newEntries.size();
            for ( size_t i = 0; i < newSize; i++) {

                // Get new and old entries from new and old index files
                // NOTE: "getEntry" returns first entry found with timestamp passed as parameter
                newEntry = newEntries.at(i);
                prvEntry = newFile.at(j).getEntry( newEntry.ts );
                oldEntry = oldFile.at(j).getEntry( newEntry.ts );

                // For debug
                assert( prvEntry.ts == newEntry.ts);
                assert( oldEntry.ts == newEntry.ts);

                // If block has changed, rename file
                string command;

                // Set files names
                stringstream newImageFileName;
                stringstream prvImageFileName;
                stringstream tmpImageFileName;

                newImageFileName << cameraName.at(j)     << "/seq_" << setfill('0') << setw(6)
                                 << newEntry.block << imageExtension;
                prvImageFileName << cameraName.at(j)     << "/seq_" << setfill('0') << setw(6)
```

```
                                << prvEntry.block << imageExtension;
                tmpImageFileName << cameraName_tmp.at(j) << "/seq_" << setfill('0') << setw(6)
                                << oldEntry.block << imageExtension;

                // If old file is still in temporal directory move it
                // Else look for file in new directory using newFile to get previous ts entry
                if (fileExists(tmpImageFileName.str()))
                    command = "mv " + tmpImageFileName.str() + " " + newImageFileName.str();
                else
                    command = "cp " + prvImageFileName.str() + " " + newImageFileName.str();

                try { call_system( command ); }
                catch( const exception& e ){
                    cout << "Error creating image file." << endl;
                    throw logic_error(e.what());
                }
            }

            // Remove temporal directory
            cout << "Removing temporal files from " << configCameraNames(j) << "... " << endl;
            try { call_system ( "rm -rf " + cameraName_tmp.at(j) ); }
            catch( const exception& e ){
                cout << "Error removing temporal files." << endl;
                throw logic_error(e.what());
            }
        }

        // Overwrite old seq.index files
        cout << endl << "Writing new index and ini files..." << endl;;
        for ( size_t i = 0; i < numberOfCameras; i++ ) {
            newFile.at(i).write( indexFileName.at(i) );

            iniFile.at(i).WriteInt( SEQ_INI_SECTION.c_str(), "frames", newFile.at(i).size() );
            if ( ! iniFile.at(i).WriteIniFile( iniFileName.at(i).c_str() ) ) {
                cout << "Error creating seq.ini file: " << iniFileName.at(i) << endl;
            }
        }

        // Create a main info file
        call_system("cp " + iniFileName.at(0) + " " + sessionPath + "/video/seqInfo.ini");

        cout << endl;
        cout << "=== RECORDING PROCESS DONE ===" << endl;

        // End client
        exit(0);
    }
    catch( const std::exception& e ) {
      std::cout << "Exception error: " << e.what() << std::endl;
    }
}
```

A continuació tenim el resultat de l'execució d'aquest client en dos gravacions diferents.

La primera gravació no té cap error de sincronisme:

```
=== RECORDING PROCESS LOG ===

Session location  : /mnt/nas/data/CHIL_EVALUATION_DATA_2006/IBM_ALBERT/IBM_20050819/Segment1
Number of cameras : 5 ( cam1 cam2 cam3 cam4 cam5 )
Frame rate expeted: 15fps

Reading files from cam1...   4501 frames.
Reading files from cam2...   4501 frames.
Reading files from cam3...   4501 frames.
Reading files from cam4...   4501 frames.
Reading files from cam5...   4501 frames.

Creating master camera at 15fps from cam1...
Clipping master with all cameras...   Master camera has 4501 frames.

Synchronizing cam1 index file with master...
Synchronizing cam2 index file with master...
```

```
Synchronizing cam3 index file with master...
Synchronizing cam4 index file with master...
Synchronizing cam5 index file with master...

Removing clipped images files from cam1...
Saving temporal files from cam1...
Creating new image files from cam1...
Removing temporal files from cam1...

Removing clipped images files from cam2...
Saving temporal files from cam2...
Creating new image files from cam2...
Removing temporal files from cam2...

Removing clipped images files from cam3...
Saving temporal files from cam3...
Creating new image files from cam3...
Removing temporal files from cam3...

Removing clipped images files from cam4...
Saving temporal files from cam4...
Creating new image files from cam4...
Removing temporal files from cam4...

Removing clipped images files from cam5...
Saving temporal files from cam5...
Creating new image files from cam5...
Removing temporal files from cam5...

Writing new index and ini files...

=== RECORDING PROCESS DONE ===
```

Aquesta gravació té diferents errors de sincronisme detectats:

```
=== RECORDING PROCESS LOG ===

Session location  : /mnt/nas/data/CHIL_EVALUATION_DATA_2006/ITC_ALBERT/ITC_20050607/Segment1
Number of cameras : 4 ( cam1 cam2 cam3 cam4 )
Frame rate expeted: 15fps

Reading files from cam1...   4487 frames.
Reading files from cam2...   4490 frames.
Reading files from cam3...   4488 frames.
Reading files from cam4...   4481 frames.

Creating master camera at 15fps from cam1...
Drop frame detected at master camera. Added teorical ts: 1118139153.230666
Drop frame detected at master camera. Added teorical ts: 1118139299.023666
Drop frame detected at master camera. Added teorical ts: 1118139304.776666
Drop frame detected at master camera. Added teorical ts: 1118139322.066666
Clipping master with all cameras...   Master camera has 4490 frames.

Synchronizing cam1 index file with master...
Drop frame detected. Added ts: 1118139153.164000 as frame 308
Drop frame detected. Added ts: 1118139298.957000 as frame 2490
Drop frame detected. Added ts: 1118139304.710000 as frame 2576
Drop frame detected. Added ts: 1118139322.000000 as frame 2835
Synchronizing cam2 index file with master...
Drop frame detected. Added ts: 1118139159.091000 as frame 395
Drop frame detected. Added ts: 1118139170.730000 as frame 570
Drop frame detected. Added ts: 1118139279.973000 as frame 2204
Drop frame detected. Added ts: 1118139299.169000 as frame 2491
Drop frame detected. Added ts: 1118139301.781000 as frame 2532
Drop frame detected. Added ts: 1118139301.781000 as frame 2533
Drop frame detected. Added ts: 1118139302.071000 as frame 2534
Drop frame detected. Added ts: 1118139302.071000 as frame 2536
Drop frame detected. Added ts: 1118139322.237000 as frame 2836
Drop frame detected. Added ts: 1118139339.713000 as frame 3097
Drop frame detected. Added ts: 1118139369.078000 as frame 3537
Synchronizing cam3 index file with master...
Drop frame detected. Added ts: 1118139145.786000 as frame 196
Drop frame detected. Added ts: 1118139157.686000 as frame 374
Drop frame detected. Added ts: 1118139164.620000 as frame 478
```

```
Drop frame detected. Added ts: 1118139169.557000 as frame 551
Drop frame detected. Added ts: 1118139187.507000 as frame 821
Drop frame detected. Added ts: 1118139187.707000 as frame 822
Drop frame detected. Added ts: 1118139205.417000 as frame 1089
Drop frame detected. Added ts: 1118139224.117000 as frame 1369
Drop frame detected. Added ts: 1118139229.125000 as frame 1444
Drop frame detected. Added ts: 1118139241.036000 as frame 1621
Drop frame detected. Added ts: 1118139252.761000 as frame 1798
Drop frame detected. Added ts: 1118139258.705000 as frame 1887
Drop frame detected. Added ts: 1118139259.649000 as frame 1901
Drop frame detected. Added ts: 1118139259.817000 as frame 1902
Drop frame detected. Added ts: 1118139264.658000 as frame 1976
Drop frame detected. Added ts: 1118139270.594000 as frame 2065
Drop frame detected. Added ts: 1118139294.245000 as frame 2419
Drop frame detected. Added ts: 1118139295.256000 as frame 2433
Drop frame detected. Added ts: 1118139300.985000 as frame 2520
Drop frame detected. Added ts: 1118139301.866000 as frame 2532
Drop frame detected. Added ts: 1118139301.866000 as frame 2533
Drop frame detected. Added ts: 1118139302.207000 as frame 2538
Drop frame detected. Added ts: 1118139307.215000 as frame 2612
Drop frame detected. Added ts: 1118139312.070000 as frame 2686
Drop frame detected. Added ts: 1118139313.009000 as frame 2700
Drop frame detected. Added ts: 1118139318.108000 as frame 2775
Drop frame detected. Added ts: 1118139324.050000 as frame 2863
Drop frame detected. Added ts: 1118139342.334000 as frame 3138
Drop frame detected. Added ts: 1118139342.421000 as frame 3140
Drop frame detected. Added ts: 1118139342.616000 as frame 3141
Drop frame detected. Added ts: 1118139347.737000 as frame 3219
Drop frame detected. Added ts: 1118139353.771000 as frame 3308
Drop frame detected. Added ts: 1118139360.640000 as frame 3411
Drop frame detected. Added ts: 1118139365.660000 as frame 3486
Drop frame detected. Added ts: 1118139371.597000 as frame 3575
Drop frame detected. Added ts: 1118139372.533000 as frame 3589
Drop frame detected. Added ts: 1118139383.480000 as frame 3754
Drop frame detected. Added ts: 1118139389.707000 as frame 3846
Drop frame detected. Added ts: 1118139395.425000 as frame 3933
Drop frame detected. Added ts: 1118139396.577000 as frame 3949
Drop frame detected. Added ts: 1118139401.581000 as frame 4024
Drop frame detected. Added ts: 1118139407.454000 as frame 4113
Drop frame detected. Added ts: 1118139419.336000 as frame 4291
Drop frame detected. Added ts: 1118139425.451000 as frame 4381
Drop frame detected. Added ts: 1118139431.425000 as frame 4472
Synchronizing cam4 index file with master...
Drop frame detected. Added ts: 1118139181.676000 as frame 732
Drop frame detected. Added ts: 1118139187.644000 as frame 823
Drop frame detected. Added ts: 1118139193.418000 as frame 910
Drop frame detected. Added ts: 1118139193.707000 as frame 911
Drop frame detected. Added ts: 1118139193.707000 as frame 912
Drop frame detected. Added ts: 1118139259.841000 as frame 1902
Drop frame detected. Added ts: 1118139302.272000 as frame 2539
Drop frame detected. Added ts: 1118139302.413000 as frame 2540
Drop frame detected. Added ts: 1118139342.359000 as frame 3139
Drop frame detected. Added ts: 1118139347.776000 as frame 3220
Drop frame detected. Added ts: 1118139347.947000 as frame 3222
Drop frame detected. Added ts: 1118139383.441000 as frame 3754
Drop frame detected. Added ts: 1118139383.663000 as frame 3755
Drop frame detected. Added ts: 1118139429.188000 as frame 4437
Drop frame detected. Added ts: 1118139429.251000 as frame 4439

Removing clipped images files from cam1...
Saving temporal files from cam1...
Creating new image files from cam1...
Removing temporal files from cam1...

Removing clipped images files from cam2...
Saving temporal files from cam2...
Creating new image files from cam2...
Removing temporal files from cam2...

Removing clipped images files from cam3...
Saving temporal files from cam3...
Creating new image files from cam3...
Removing temporal files from cam3...

Removing clipped images files from cam4...
Saving temporal files from cam4...
Creating new image files from cam4...
Removing temporal files from cam4...
```

```
Writing new index and ini files...

=== RECORDING PROCESS DONE ===
```

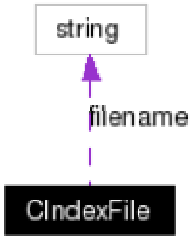
Per més detall, aquí es mostra la documentació generada per doxygen de la classe CIndexFile.

CIndexFile Class Reference

To manage timestamps files of CHIL recordings.

```
#include <indexFile.hpp>
```

Collaboration diagram for CIndexFile:



Public Member Functions

- void read (const std::string fileName, double fps=25)
- void write (const std::string fileName)
- void **setFrameRate** (double fps)
- bool **hasSimilarTimestamp** (const Timestamp &ts, double epsilon) const
- bool **hasTimestamp** (const Timestamp &ts) const
- void **addEntry** (const Timestamp &ts, unsigned int block, int offset)
- void **addEntry** (const IndexFileEntry &newEntry)
- void **clear** (void)
- void removeRepeatedFrames (void)
- void clip (const CIndexFile &master)
- CIndexFile getSync (const CIndexFile &master) const
- void renameBlocks (void)
- IndexFileEntry **getEntry** (const Timestamp &ts) const
- int **size** (void) const
- CIndexFile getMaster (const double tsDistance) const
- IndexFileEntry **getNearestEntry** (const Timestamp &ts) const
- void **print** (void) const
- std::vector< IndexFileEntry > **getEntriesVector** (void) const

Private Attributes

- std::string **filename**
- double **tsDistance**

- std::vector< IndexFileEntry > **entries**

Detailed Description

To manage timestamps files of CHIL recordings.

Timestamps files are used in CHIL recordings to associate a timestamp to each frame. These files are usually placed at recording/video/camN/seq.index. This class reads nad writes this files and allows any kind of editing operation.

Member Function Documentation

void CIndexFile::clip (const CIndexFile & master)

This method clips the object with master. Entries with minor timestamp than the first first timestamp of master are removed. And also the ones that have a greater timestamp than the maximum of master are removed.

CIndexFile CIndexFile::getMaster (const double tsDistance) const

This method creates a new object fixing sync errors of current one. It is considered an error if temporal distance beetwen two consecutive frames are bigger than tsDistance parameter.

CIndexFile CIndexFile::getSync (const CIndexFile & master) const

This method creates a new object with same number of frames of master. Each frame of new object have the timestamp of the current object nearest to master timestamp.

void CIndexFile::read (const std::string fileName, double fps = 25)

This methods reads the content of the file.

Parameters:

fileName The file to read from.
fps The frame rate expecetd of fileName file.

Note:

The fps parameter will be used only if you want to create a master from this file.

void CIndexFile::removeRepeatedFrames (void)

This method cleans all repeated frames of the object. Repeated frames are added in a sync process.

void CIndexFile::renameBlocks (void)

This method check if the object begins with frame 0. If not, all frames are renamed (renumbered) to start with 0.

void CIndexFile::write (const std::string fileName)

This methods writes the object content to the file

APÈNDIX C. Documentació de la *Recording-Lib*

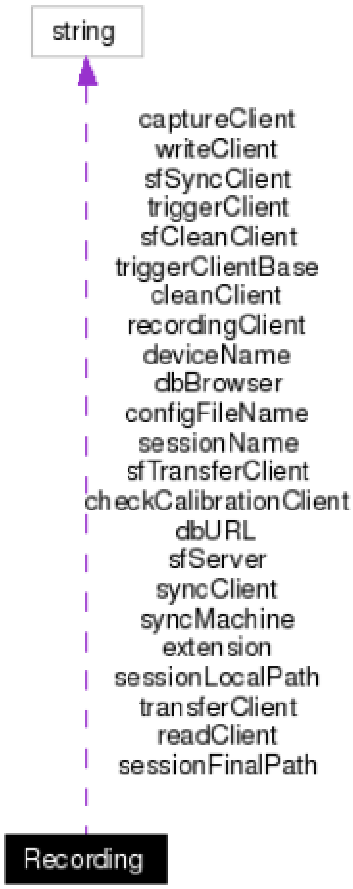
La següent és la documentació de la *Recording-Lib* generada automàticament per *Doxygen*.

Recording Class Reference

Middleware class to be used by any GUI to manage Recording-Clients.

```
#include <recording.hpp>
```

Collaboration diagram for Recording:



Public Types

1. enum Mode { Clean, Capture, Write, Read, Record, Transfer, Sync, CheckCalibration, Ntp, DataBase }

Public Member Functions

2. Recording ()

3. Recording (const std::string fileName)
4. unsigned int getWidth ()
5. unsigned int getHeight ()
6. unsigned int getFrameRateExpected ()
7. std::string captureInit (std::string camName)
8. void capture (unsigned char *saveBuffer, Timestamp &ts)
9. bool getCalibrationMode ()
10. void createFiles ()
11. void start (Mode mode=Record)
12. void sfStart (Mode mode)
13. bool getLog (unsigned int clientNumber, std::string &log)
14. unsigned int getNumberOfLogs ()
15. void stop ()
16. std::string **getFinalPath** ()
17. std::string **getCalibrationFile** (unsigned int camNumber)
18. std::string **getCameraName** (unsigned int camNumber)
19. std::string **getCalibrationImageName** (unsigned int camNumber)
20. void **setLocalPath** (std::string path)
21. void **setFinalPath** (std::string path)
22. void **setSessionName** (std::string name)
23. void **setExtension** (std::string ex)
24. void **setCameraNames** (std::vector< std::string > camNames)
25. void **setCalibrationFiles** (std::vector< std::string > calFiles)
26. void **setComputerNames** (std::vector< std::string > computerNames)
27. void **setCalibrationMode** (bool enabled)
28. void **setWriteBackground** (bool enabled)
29. void **setBackgroundImages** (unsigned int images)
30. void **setCalibrationImages** (unsigned int images)
31. void **setCleanClient** (std::string client)
32. void **setRecordingClient** (std::string client)
33. void **setTransferClient** (std::string client)
34. void **setSyncClient** (std::string client)
35. void **setReadClient** (std::string client)
36. void **setTriggerClient** (std::string client)
37. void **setSmartflowServer** (std::string server)
38. void **setSyncComputer** (std::string computer)
39. void **setDeviceName** (std::string device)
40. void **setChannel** (int _channel)
41. void **setBuffer** (int _buffer)
42. void **setWidth** (unsigned int _width)
43. void **setHeight** (unsigned int _height)
44. void **setUseEveryNFrame** (unsigned int _useEveryNFrame)
45. void **setFrameRateExpected** (unsigned int _frameRateExpected)

Static Public Member Functions

46. static void initClient ()

Private Member Functions

47. pid_t **receivePid** (redi::ipstream &log)
48. void **xmlAddSFFlowsInfo** (xmlpp::XMLElement &client, size_t cameraNumber, const std::string &readPrefix="", bool triggerClient=false)
49. xmlpp::XMLElement **xmlGetConfigServers** (std::string serverPath)

Private Attributes

50. std::vector< redi::ipstream * > **logs**
51. std::vector< pid_t > **pids**
52. std::string **configFileName**
53. std::string **sessionLocalPath**
54. std::string **sessionFinalPath**
55. std::string **sessionName**

56. std::string **extension**
57. unsigned int **backgroundImages**
58. unsigned int **calibrationImages**
59. bool **writeBackground**
60. bool **readBackground**
61. bool **calibrationMode**
62. xmlpp::XMLStringVector **cameraName**
63. xmlpp::XMLStringVector **tsName**
64. xmlpp::XMLStringVector **machineName**
65. xmlpp::XMLStringVector **calibrationFile**
66. std::string **syncMachine**
67. std::string **writeClient**
68. std::string **captureClient**
69. std::string **syncClient**
70. std::string **cleanClient**
71. std::string **readClient**
72. std::string **transferClient**
73. std::string **recordingClient**
74. std::string **checkCalibrationClient**
75. std::string **triggerClient**
76. std::string **triggerClientBase**
77. std::string **sfServer**
78. std::string **sfSyncClient**
79. std::string **sfCleanClient**
80. std::string **sfTransferClient**
81. std::string **dbURL**
82. std::string **dbBrowser**
83. VideoIO::Video4Linux * **v4l**
84. std::string **deviceName**
85. int **channel**
86. int **bufferSize**
87. int **useEveryNFrame**
88. unsigned int **width**
89. unsigned int **height**
90. unsigned int **frameRateExpected**

Detailed Description

Middleware class to be used by any GUI to manage Recording-Clients.

This class can be used for Recording-Clients, and for any GUI to control them. It's a middleware layer beetwen recording clients and a user application.

Member Enumeration Documentation

enum Recording::Mode

Enum type to define the operation mode.

Enumerator:

Clean To clean an existing recording.

Capture To start sf capture clients

Write To start sf write clients

Read To start sf read clients

Record To start a new recording.

Transfer To transfer a recording from local place to the final place.

Sync To post process the recording placed in final place.

CheckCalibration To check if calibration is correct.

Ntp To check ntp status.

DataBase To start data base

Constructor & Destructor Documentation

Recording::Recording ()

Default constructor

Recording::Recording (const std::string fileName*)*

Constructor wich reads configuration from the fileName

Member Function Documentation

*void Recording::capture (unsigned char ** saveBuffer*, Timestamp &* ts*)*

This method launch a new capture.

Parameters:

saveBuffer The buffer pointer where captured frame will be saved.
ts The timestamp obtained from the capture

std::string Recording::captureInit (std::string camName*)*

This method initialize the capture funcionality and returns the calibration file.

Parameters:

camName The name of tha camera that is initialized

void Recording::createFiles ()

This method creates all config files used by Smart Flow. All this files will be placed at final storage place.

bool Recording::getCalibrationMode () `[inline]`

This method returns the frame rate parameter

unsigned int Recording::getFrameRateExpected () `[inline]`

This method returns the frame rate parameter

unsigned int Recording::getHeight () `[inline]`

This method returns the height parameter

bool Recording::getLog (unsigned int clientNumber*, std::string &* log*)*

This method returns the last log captured by a client.

Parameters:

clientNumber Selects which log will be returned. It must be less than the total number of clients running.
log The log is returned in this parameter.

Returns:

Returns true if client is still running, return false if client is over.

unsigned int Recording::getNumberOfLogs ()

This method returns the the number of logs (or clients) running.

unsigned int Recording::getWidth () `[inline]`

This method returns the width parameter

static void Recording::initClient () `[static]`

This method should be used by a Recording-Client. It sends its process information. This information will be recieved by the start method that will launch the client.

void Recording::sfStart (Mode mode*)*

This method starts the corresponding SmartFlow-Clients.

Clients lod is controled by Smart Flow. Smart Flow servers must be running before calling this method.

Parameters:

mode Selects which Recording-Clients will be launched. creates all config files used by Smart Flow.

void Recording::start (Mode mode = `Record`*)*

This method starts the corresponding Recording-Clients.

This method also capture all clients log in order to be readed with getLog method.

Parameters:

mode Selects which Recording-Clients will be launched. creates all config files used by Smart Flow.

void Recording::stop ()

This method stops all Recording-Clients running.

APÈNDIX D. Estructura i format estàndard CHIL de gravacions

En aquest annex es mostra l'estructura bàsica de les gravacions realitzades en el projecte CHIL [6].

1. Estructura de directoris:

- audio
  - hammerfall
  - markIII_1
  - markIII_2

- video
  - cam1
    - seq_000000.jpg
    - seq_000001.jpg
    - ...
    - seq_XXXXXX.jpg
    - seq.index
    - seq.ini
  - cam2
  - cam3
  - ....
- info
  - calibration
    - cam1.ini
    - cam1
      - cal_000000.bmp
      - ...
      - cal_000010.bmp
    - cam2.ini
    - cam2
      - cal_000000.bmp
      - ...
      - cal_000009.bmp
    - ...
  - background
    - cam1
      - seq_000000.jpg
      - ...
      - seq_000099.jpg
    - cam2
      - seq_000000.jpg
      - ...
      - seq_000099.jpg
    -
  - slides

2. Fitxers de metadades:

- Fitxers seq.ini:

; Total number of frames
frames=39303
; Image width
width=640
; Image height
height=480
; Image depth
depth=3
; Frames per second (approximately)
fps=25
; Compression: 0=none, 1=JPG, 2=MPEG2
compression=1
; Number of images concatenated to a single file
blocksize=1
; Description of image format
description=RGB

- Fitxers seq.index:

; time stamp	block	offset
1150389928.757	0	0
1150389928.823	1	0
1150389928.890	2	0
1150389928.956	3	0
1150389929.023	4	0
1150389929.089	5	0
1150389929.155	6	0
1150389929.222	7	0
1150389929.288	8	0
1150389929.355	9	0

La primera columna és el *timestamp* de la imatge amb una precisió de 1ms.
La segona és el número de bloc (o número de fitxer).
La tercera és l'offset de la imatge en el bloc.

APÈNDIX E. Fitxers de configuració

Smart Flow utilitza dos fitxers de configuració principals. Un d'ells defineix les propietats de tots els *clients* i *servers* que s'utilitzaran, i l'altre defineix com es configuren i connecten. Tot i que els fitxers tenen l'extensió .rc, realment són fitxers descriptors *XML*.

A mode d'exemple, aquí es mostren dos fitxers generats pel nou sistema *off-line*. Tot i les dimensions dels fitxers, l'exemple s'ha generat per tal de treballar amb només dues cameres.

1. Fitxer de configuració (llista de servers i clients):

```
<!--File: example_sf_conf.rc -->

<?xml version="1.0" encoding="UTF-8"?>
<config>
  <servers>
    <server>
      <name>v1</name>
      <host>v1</host>
      <path>/mnt/smartflow/bin/sfd</path>
      <desc>v1</desc>
      <active>true</active>
    </server>
    <server>
      <name>v2</name>
      <host>v2</host>
      <path>/mnt/smartflow/bin/sfd</path>
      <desc>v2</desc>
      <active>true</active>
    </server>
    <server>
      <name>v7</name>
      <host>v7</host>
      <path>/mnt/smartflow/bin/sfd</path>
      <desc>v7</desc>
      <active>true</active>
    </server>
  </servers>
  <clients>
    <client>
      <name>cleaner</name>
      <group>default</group>
      <path>/mnt/smartflow/clients/user/albert/clients/cleanRecording</path>
    </client>
    <client>
      <name>transfer</name>
      <group>default</group>
      <path>/mnt/smartflow/clients/user/albert/clients/transferRecording</path>
    </client>
    <client>
      <name>capture</name>
      <flows>
        <output>
          <video width="720" height="480">
            <name>video</name>
            <group>default</group>
          </video>
          <data size="4" name="UPC::Count">
            <name>ts</name>
            <group>default</group>
          </data>
        </output>
      </flows>
      <group>default</group>
      <path>/mnt/smartflow/clients/user/albert/clients/capture</path>
```

```
    </client>
    <client>
      <name>writeVideo</name>
      <flows>
        <input>
          <video width="720" height="480">
            <name>to_write</name>
            <group>default</group>
          </video>
        </input>
      </flows>
      <group>default</group>
      <path>/mnt/smartflow/clients/user/albert/clients/writeVideo</path>
    </client>
    <client>
      <name>processRecording</name>
      <group>default</group>
      <path>/mnt/smartflow/clients/user/albert/clients/processRecording</path>
    </client>
    <client>
      <name>readVideo</name>
      <flows>
        <input>
          <data size="4" name="UPC::Trigger">
            <name>trigger</name>
            <group>default</group>
          </data>
        </input>
        <output>
          <video width="720" height="480">
            <name>video</name>
            <group>default</group>
          </video>
          <data size="4" name="UPC::Count">
            <name>ts</name>
            <group>default</group>
          </data>
        </output>
      </flows>
      <group>default</group>
      <path>/mnt/smartflow/clients/user/albert/clients/readVideo</path>
    </client>
    <client>
      <name>triggerSync_x2</name>
      <flows>
        <input>
          <data size="4" name="UPC::Count">
            <name>ts_2</name>
          </data>
          <data size="4" name="UPC::Count">
            <name>ts_2</name>
          </data>
          <data size="4" name="UPC::Trigger">
            <name>trigger</name>
          </data>
        </input>
        <output>
          <data size="4" name="UPC::Trigger">
            <name>triggerSync</name>
          </data>
        </output>
      </flows>
      <group>default</group>
      <path>/mnt/smartflow/clients/user/albert/clients/triggerSync_x2</path>
    </client>
  </clients>
</config>
```

2. Fitxer de mapa (configuració de clients i flows):

```
<!--File: example_sf_map.rc -->

<?xml version="1.0" encoding="UTF-8"?>
<application>
  <clients>
```

```
    <client>
      <name>cleaner</name>
      <param>/mnt/smartflow/clients/user/albert/config/pfc_example.xml</param>
      <server>v1</server>
      <pos x="10" y="20"/>
      <disable>true</disable>
    </client>
    <client>
      <name>capture</name>
      <flows>
        <output>
          <video width="720" height="480">
            <name>cam1</name>
            <group>default</group>
          </video>
          <data size="4" name="UPC::Count">
            <name>ts1</name>
            <group>default</group>
          </data>
        </output>
      </flows>
      <param>/mnt/smartflow/clients/user/albert/config/pfc_example.xml
/mnt/config/camera/v1/TCameraParameters_cam1.txt</param>
      <server>v1</server>
      <pos x="100" y="20"/>
      <disable>true</disable>
    </client>
    <client>
      <name>writeVideo</name>
      <flows>
        <input>
          <video width="720" height="480">
            <name>cam1</name>
            <group>default</group>
          </video>
        </input>
      </flows>
      <param>/mnt/smartflow/clients/user/albert/config/pfc_example.xml cam1</param>
      <server>v1</server>
      <pos x="200" y="10"/>
      <disable>true</disable>
    </client>
    <client>
      <name>transfer</name>
      <param>/mnt/smartflow/clients/user/albert/config/pfc_example.xml</param>
      <server>v1</server>
      <pos x="300" y="20"/>
      <disable>true</disable>
    </client>
    <client>
      <name>readVideo</name>
      <flows>
        <input>
          <data size="4" name="UPC::Trigger">
            <name>triggerSync</name>
            <group>default</group>
          </data>
        </input>
        <output>
          <video width="720" height="480">
            <name>rcam1</name>
            <group>default</group>
          </video>
          <data size="4" name="UPC::Count">
            <name>rts1</name>
            <group>default</group>
          </data>
        </output>
      </flows>
      <param>/mnt/smartflow/clients/user/albert/config/pfc_example.xml cam1</param>
      <server>v7</server>
      <pos x="600" y="10"/>
      <disable>true</disable>
    </client>
    <client>
      <name>cleaner</name>
      <param>/mnt/smartflow/clients/user/albert/config/pfc_example.xml</param>
      <server>v2</server>
```

```
      <pos x="10" y="100"/>
      <disable>true</disable>
    </client>
    <client>
      <name>capture</name>
      <flows>
        <output>
          <video width="720" height="480">
            <name>cam2</name>
            <group>default</group>
          </video>
          <data size="4" name="UPC::Count">
            <name>ts2</name>
            <group>default</group>
          </data>
        </output>
      </flows>
      <param>/mnt/smartflow/clients/user/albert/config/pfc_example.xml
/mnt/config/camera/v2/TCameraParameters_cam2.txt</param>
      <server>v2</server>
      <pos x="100" y="100"/>
      <disable>true</disable>
    </client>
    <client>
      <name>writeVideo</name>
      <flows>
        <input>
          <video width="720" height="480">
            <name>cam2</name>
            <group>default</group>
          </video>
        </input>
      </flows>
      <param>/mnt/smartflow/clients/user/albert/config/pfc_example.xml cam2</param>
      <server>v2</server>
      <pos x="200" y="90"/>
      <disable>true</disable>
    </client>
    <client>
      <name>transfer</name>
      <param>/mnt/smartflow/clients/user/albert/config/pfc_example.xml</param>
      <server>v2</server>
      <pos x="300" y="100"/>
      <disable>true</disable>
    </client>
    <client>
      <name>readVideo</name>
      <flows>
        <input>
          <data size="4" name="UPC::Trigger">
            <name>triggerSync</name>
            <group>default</group>
          </data>
        </input>
        <output>
          <video width="720" height="480">
            <name>rcam2</name>
            <group>default</group>
          </video>
          <data size="4" name="UPC::Count">
            <name>rts2</name>
            <group>default</group>
          </data>
        </output>
      </flows>
      <param>/mnt/smartflow/clients/user/albert/config/pfc_example.xml cam2</param>
      <server>v7</server>
      <pos x="600" y="90"/>
      <disable>true</disable>
    </client>
    <client>
      <name>processRecording</name>
      <param>/mnt/smartflow/clients/user/albert/config/pfc_example.xml</param>
      <server>v7</server>
      <pos x="400" y="90"/>
      <disable>true</disable>
    </client>
    <client>
```

```
      <name>triggerSync_x2</name>
      <flows>
        <input>
          <data size="4" name="UPC::Count">
            <name>rts1</name>
          </data>
          <data size="4" name="UPC::Count">
            <name>rts2</name>
          </data>
          <data size="4" name="UPC::Trigger">
            <name>triggerSync</name>
          </data>
        </input>
        <output>
          <data size="4" name="UPC::Trigger">
            <name>triggerSync</name>
          </data>
        </output>
      </flows>
      <param>/mnt/smartflow/clients/user/albert/config/pfc_example.xml</param>
      <server>v7</server>
      <pos x="600" y="190"/>
      <disable>true</disable>
    </client>
  </clients>
</application>
```

I aquí es mostra el fitxer de configuració del *client showRoom*:

```
<?xml version="1.0"?>
<!--
    X, Y and Z refer to the coordinate system of Oscar:
    origin in corner floor/window/Philippe
    X towards door to TV lab
    Y towards entrance door
    Z towards Allah
    The unit distances are floats in centimeters
-->

<roomSetup>
    <configuration>
        <roomSize unit="mm">
            <x>3980</x>
            <y>5200</y>
            <z>3500</z>
        </roomSize>
        <talkingHead description= "talking head" inRoom = "1">
            <position unit="mm">
                <x>3000</x>
                <y>5200</y>
                <z>1700</z>
            </position>
            <straightAhead>
                <pan>180</pan>
                <tilt>0</tilt>
            </straightAhead>
        </talkingHead>
        <hotSpot description= "center of the room" inRoom = "1">
            <x>200</x>
            <y>5000</y>
        </hotSpot>
        <table description= "central table" inRoom = "1">
            <centre unit="mm">
                <x>1990</x>
                <y>2600</y>
                <z>600</z>
            </centre>
            <rectangular unit="mm">
                <x>1500</x>
                <y>1800</y>
            </rectangular>
        </table>
        <table description= "coffee table" inRoom = "1">
            <centre unit="mm">
```

```
                <x>310</x>
                <y>310</y>
                <z>600</z>
            </centre>
            <rectangular unit="mm">
                <x>600</x>
                <y>600</y>
            </rectangular>
        </table>
        <door description= "entrance" inRoom = "1">
            <doorRotationAxis unit="mm">
                <x>10</x>
                <y>5200</y>
            </doorRotationAxis>
            <doorClosedAt unit="mm">
                <x>1100</x>
                <y>5200</y>
            </doorClosedAt>
            <angle alwaysOpen="0" > 90 </angle>
            <height unit="mm">1800</height>
        </door>
        <whiteBoard description = "large whiteboard" inRoom = "1">
            <position1 unit="mm">
                <x>4980</x>
                <y>1000</y>
                <z>1000</z>
            </position1>
            <position2 unit="mm">
                <x>4980</x>
                <y>2000</y>
                <z>1000</z>
            </position2>
            <lowerEdge unit="mm">50</lowerEdge>
            <upperEdge unit="mm">170</upperEdge>
        </whiteBoard>
        <window description = "window to street" inRoom = "1">
            <position1 unit="mm">
                <x>2220</x>
                <y>2800</y>
                <z>1000</z>
            </position1>
            <position2 unit="mm">
                <x>2600</x>
                <y>2800</y>
                <z>1000</z>
            </position2>
            <lowerEdge unit="mm">50</lowerEdge>
            <upperEdge unit="mm">170</upperEdge>
        </window>
       <camera number = "1" description = "camera1" inRoom = "1">
               <intrinsics>
                       <imageResolution>
                               <x>768</x>
                               <y>576</y>
                       </imageResolution>
                       <focalLenght unit="1/mm">
                               <x>15.1482</x>
                               <y>19.9002</y>
                       </focalLenght>
                       <principalPoint>
                               <x>376.393</x>
                               <y>309.476</y>
                       </principalPoint>
                       <distortion>
                               <k1>-0.339839</k1>
                               <k2>0.138678</k2>
                               <k3>0</k3>
                               <k4>0.000763519</k4>
                               <k5>0.00016387</k5>
                       </distortion>
               </intrinsics>
               <extrinsics>
                       <translation unit="mm">
                               <x>44.5034</x>
                               <y>2030.19</y>
                               <z>1159.11</z>
                       </translation>
                       <rotation unit="mm">
```

```
                        <R11>0.683143</R11>
                        <R12>-0.729868</R12>
                        <R13>0.0246839</R13>
                        <R21>-0.394876</R21>
                        <R22>-0.397607</R22>
                        <R23>-0.82824</R23>
                        <R31>0.61432</R31>
                        <R32>0.556059</R32>
                        <R33>-0.55983</R33>
                </rotation>
        </extrinsics>
        <realWorldCoordinates unit="cm">
                <X>5.92057</X>
                <Y>19.5166</Y>
                <Z>232.929</Z>
        </realWorldCoordinates>
</camera>
<camera number = "2" description = "camera2" inRoom = "1">
        <intrinsics>
                <imageResolution>
                        <x>768</x>
                        <y>576</y>
                </imageResolution>
                <focalLenght unit="1/mm">
                        <x>14.963</x>
                        <y>19.6359</y>
                </focalLenght>
                <principalPoint>
                        <x>400.415</x>
                        <y>300.362</y>
                </principalPoint>
                <distortion>
                        <k1>-0.329981</k1>
                        <k2>0.135106</k2>
                        <k3>0</k3>
                        <k4>-0.00112156</k4>
                        <k5>0.000526491</k5>
                </distortion>
        </intrinsics>
        <extrinsics>
                <translation unit="mm">
                        <x>-3039.02</x>
                        <y>640.816</y>
                        <z>3163.68</z>
                </translation>
                <rotation unit="mm">
                        <R11>0.74793</R11>
                        <R12>0.661732</R12>
                        <R13>0.0520783</R13>
                        <R21>0.371468</R21>
                        <R22>-0.35225</R22>
                        <R23>-0.85903</R23>
                        <R31>-0.550102</R31>
                        <R32>0.661839</R32>
                        <R33>-0.50927</R33>
                </rotation>
        </extrinsics>
        <realWorldCoordinates unit="cm">
                <X>377.528</X>
                <Y>14.2899</Y>
                <Z>231.991</Z>
        </realWorldCoordinates>
</camera>
<camera number = "3" description = "camera3" inRoom = "1">
        <intrinsics>
                <imageResolution>
                        <x>768</x>
                        <y>576</y>
                </imageResolution>
                <focalLenght unit="1/mm">
                        <x>14.9758</x>
                        <y>19.6728</y>
                </focalLenght>
                <principalPoint>
                        <x>360.659</x>
                        <y>284.019</y>
                </principalPoint>
                <distortion>
```

```
                        <k1>-0.330189</k1>
                        <k2>0.131664</k2>
                        <k3>0</k3>
                        <k4>0.00100907</k4>
                        <k5>0.000416527</k5>
                </distortion>
        </intrinsics>
        <extrinsics>
                <translation unit="mm">
                        <x>-687.804</x>
                        <y>-1524.18</y>
                        <z>6499.65</z>
                </translation>
                <rotation unit="mm">
                        <R11>-0.721851</R11>
                        <R12>0.691851</R12>
                        <R13>-0.0165039</R13>
                        <R21>0.392454</R21>
                        <R22>0.389596</R22>
                        <R23>-0.833183</R23>
                        <R31>-0.570009</R31>
                        <R32>-0.607911</R32>
                        <R33>-0.552751</R33>
                </rotation>
        </extrinsics>
        <realWorldCoordinates unit="cm">
                <X>380.654</X>
                <Y>502.088</Y>
                <Z>231.141</Z>
        </realWorldCoordinates>
</camera>
<camera number = "4" description = "camera4" inRoom = "1">
        <intrinsics>
                <imageResolution>
                        <x>768</x>
                        <y>576</y>
                </imageResolution>
                <focalLenght unit="1/mm">
                        <x>14.9928</x>
                        <y>19.6559</y>
                </focalLenght>
                <principalPoint>
                        <x>391.117</x>
                        <y>287.478</y>
                </principalPoint>
                <distortion>
                        <k1>-0.328041</k1>
                        <k2>0.120835</k2>
                        <k3>0</k3>
                        <k4>0.000848644</k4>
                        <k5>3.53488e-05</k5>
                </distortion>
        </intrinsics>
        <extrinsics>
                <translation unit="mm">
                        <x>3477.96</x>
                        <y>232.353</y>
                        <z>4353.31</z>
                </translation>
                <rotation unit="mm">
                        <R11>-0.725383</R11>
                        <R12>-0.686346</R12>
                        <R13>0.0524326</R13>
                        <R21>-0.394737</R21>
                        <R22>0.352365</R22>
                        <R23>-0.848541</R23>
                        <R31>0.563917</R31>
                        <R32>-0.636214</R32>
                        <R33>-0.526525</R33>
                </rotation>
        </extrinsics>
        <realWorldCoordinates unit="cm">
                <X>15.9662</X>
                <Y>507.484</Y>
                <Z>230.693</Z>
        </realWorldCoordinates>
</camera>
<camera number = "5" description = "camera5" inRoom = "1">
```

```
                <intrinsics>
                        <imageResolution>
                                <x>768</x>
                                <y>576</y>
                        </imageResolution>
                        <focalLenght unit="1/mm">
                                <x>8.06848</x>
                                <y>10.5809</y>
                        </focalLenght>
                        <principalPoint>
                                <x>369.198</x>
                                <y>284.786</y>
                        </principalPoint>
                        <distortion>
                                <k1>-0.218084</k1>
                                <k2>0.0564824</k2>
                                <k3>0</k3>
                                <k4>-0.000464405</k4>
                                <k5>-0.00159565</k5>
                        </distortion>
                </intrinsics>
                <extrinsics>
                        <translation unit="mm">
                                <x>-2419.06</x>
                                <y>-1946.49</y>
                                <z>3469.55</z>
                        </translation>
                        <rotation unit="mm">
                                <R11>0.0123443</R11>
                                <R12>0.997146</R12>
                                <R13>-0.074488</R13>
                                <R21>0.999915</R21>
                                <R22>-0.0120056</R22>
                                <R23>0.00499317</R23>
                                <R31>0.00408464</R31>
                                <R32>-0.0745433</R32>
                                <R33>-0.997209</R33>
                        </rotation>
                </extrinsics>
                <realWorldCoordinates unit="cm">
                        <X>196.201</X>
                        <Y>264.742</Y>
                        <Z>328.94</Z>
                </realWorldCoordinates>
        </camera>
</configuration>
</roomSetup>
```

Referències

[1] Mitchell M., Oldham J., Samuel A (CodeSourcery LLC). *Advanced Linux Programming.* 1ª ed. California: New Riders Publishing, 2001. ISBN 0-7357-1043-0. Disponible a <http://www.advancedlinuxprogramming.com/> [consulta: març- 2007]

[2] Nate Kohl. *C/C++ Reference.* [el línia] [actualitzat: març-2007] Disponible a: <http://www.cppreference.com/> [consulta: de setembre-2006 a febrer-2007]

[3] NIST. *Smart space laboratory*. [en línia], [actualitzat: febrer, 2007], Disponible a: <http://www.nist.gov/smartspace/nsfs.html> [consulta: de setembre a desembre del 2007]

[4] Martin, C., Galibert, O., (i altres). *NIST Smart Flow System. User's Guide.* Versió 3.7, Gaithersburg: National Institute of Standards and Technology, 2000. Disponible a: <http://www.nist.gov/smartspace/toolChest/nsfs/dataFlow/userG/NSFS_userGuide.pdf>

[5] Martin, C., Galibert, O., (i altres). *NIST Smart Flow System. Programmer's Guide.* Versió 3.7, Gaithersburg: National Institute of Standards and Technology, 2000. Disponible a: <http://www.nist.gov/smartspace/toolChest/nsfs/dataFlow/progG/NSFS_programmerGuide.pdf>

[6] Cédrick Rochet. *CHIL Data Collection. Requirement for Interactive Seminars*. Document intern del projecte CHIL.

[7] Josep Ramon Casas, Rainer Stiefelhagen. *Multi-camera/multi-microphone system design for continuous room monitoring.* Document intern del projecte CHIL, 2004.

[8] James Crowely, Michael Okon (et al). *Cooperative Information System Software Desig.* Document intern del projecte CHIL, 2004.

[9] *CHIL Home Page*. [en línia] Disponible a: <http://chil.server.de/>

[10] Vinoski, S., Henning, M. *Advanced CORBA programming with C++*. Addison-Wesley, cop. 1999. ISBN 0-201-37927-9.

[11] Jordi Cenzano Ferret. *Adquisició de senyal de vídeo multicàmera per a una smartroom*. [projecte final de carrera] UPC, 2004.

[12] *NTP: The Network Time Protocol*. [en línia] [actualitzat: octubre-2006] Disponible a: <http://www.ntp.org/> [consulta: octubre-2006]

[13] Peter Harrison. *The NTP server.* A: Peter Harrison. *Linux Home Networking.* [en línia] [actualitzat: Març, 2007] Disponible a: <http://www.linuxhomenetworking.com/wiki/index.php/Quick_HOWTO_:_Ch24_:_The_NTP_Server > A: <http://www.linuxhomenetworking.com/>

[14] *Open SSH.* [en línia] [actualitzat: febrer-2006] Disponible a: <http://www.openssh.com/>

[15] Knight, S. (SCons Foundation). *Scons*. [en línia] Disponible a: <http://www.scons.org/>

[16] Collins-Sussman, B. *Version control with Subversion*. O'Reilly Media [en línia] Disponible a: <http://subversion.tigris.org/> i <http://svnbook.red-bean.com/>

[17] Dimitri van Heesch, *Doxygen*. [en línia] [actualitzat març-2007] Disponible a: <http://www.stack.nl/~dimitri/doxygen/>

[18] *Wikipedia, The free Encyclopedia: Wiki article.* [en línia] Disponible a: <http://en.wikipedia.org/wiki/Wiki>

[19] Jonathan Wakely, *PStreams.* [en línia] [actualitzat: juny-2005] Disponible a: <http://pstreams.sourceforge.net/>

[20] *NoMachine Home Page*. [en línia] Disponible a: <http://www.nomachine.com/>

[21] Jasmin Blanchette, Mark Summerfield - C++ GUI Programming with Qt 4 - Prentice Hall

[22] W3C. *Extensible Markup Language (XML).* [en línia] Disponible a: <http://www.w3.org/XML/ >

[23] *Television with Linux: Linux TV*. [en línia] Disponible a: <http://www.linuxtv.org/>

[24] *ViewCast Home Page: analogue capture card Osprey 210.* [en línia] Disponible a: <http://www.viewcast.com> A: <http://www.viewcast.com/products/osprey/osprey210.html>

[25] Zerowait Corp. *NAS-SAN.com Working together for the future of storage.* [en línia] Disponible a: <http://www.nas-san.com/>

[26] Josep Ramon Casas. *UPC smart room demo for 3GSM*. 2007 [document electrònic].

[27] Josep Ramon Casas, Dušan Macho. *UPC smart room. Sensors setup*. 2004 [document electrònic]

[28] Benjamin, Hill. *Free Software Project Management*. [en línia]. Disponible a: <http://www.tldp.org/HOWTO/Software-Proj-Mgmt-HOWTO/>

[29] *Kubuntu Home Page*. [en línia] Disponible a: <http://www.kubuntu.org/>