

Due o tre cose che so di Eveline

Ernesto Ferrari

ernesto@cinelinux.net

[Realizzato per l'esame del Corso di Progettazione dei contenuti per i nuovi media (prof. Pier Luigi Capucci), DAMS, Università di Bologna, A.A. 2002/2003]

Ho sempre voluto fare cinema.

Con *cinema* intendo quello che hanno fatto Dziga Vertov e Rossellini, Godard e Cassavetes, Bela Tarr e Kitano, registi che hanno capito che le immagini sono già tutte là fuori, che il cinema stesso c'è già: si tratta di guardarlo e montarlo per vedere se tra un cut e l'altro si può trovare il senso.

Poco dopo la metà degli anni Novanta sembrava essere diventato possibile gestire con il proprio Personal Computer almeno due delle tre fasi alle quali si può ridurre, in maniera sicuramente schematica ma efficace, l'elaborazione del prodotto *film*: la post produzione e la distribuzione. Per quanto riguardava la produzione il mercato prometteva molto, ma con un leggero ritardo: videocamere digitali con 500 discrete linee di risoluzione a prezzi che si abbassavano di anno in anno. Si trattava di aspettare l'anno giusto per il mio portafoglio.

Il vantaggio del digitale: io potevo girare in digitale con la mia videocamera Canon nuova di zecca, acquisire lo stream video direttamente dal mio PC tramite interfaccia IEEE1394, effettuare un montaggio *professionale* con la mia ricchissima (e costosissima) suite di editing non-lineare, post produrre aggiungendo titoli, effetti speciali, elaborare i singoli fotogrammi al pixel, gestire avanzati strumenti di *compositing* per magari far interagire scene "reali" con immagini di sintesi generate da un qualche software di animazione e modellazione 3D e infine distribuire il tutto via Internet, dove potenzialmente milioni di utenti avrebbero visionato il mio

capolavoro.

Dopotutto se Victor Navone, architetto trentacinquenne di Phoenix-Arizona che nel proprio tempo libero aveva fatto con Animation Master 8.0 (che usavo anche io!) il famigerato clip “Alien Song” (sì, quello con l’alieno che canta *I will survive*), lo aveva diffuso via Internet ed era stato chiamato a lavorare dalla Industrial Light and Magic di George Lucas al Ranch Skywalker – se tutto ciò stava accadendo davvero lì e in quel momento il futuro, compreso il mio, non poteva che tingersi della più rosea delle sfumature!

C’era un problema: tutta questa architettura poggiava su uno stuzzicadenti. Il sistema operativo del mio computer, Microsoft Windows 98 SE.

Tutti i miei meravigliosi programmi prima o poi mi lasciavano a piedi. Avevi affrontato con spirito di sacrificio e abnegazione la famigerata *learning curve* del software di animazione e modellazione 3D per poi non riuscire a lavorarci per più di 15 minuti senza che all’improvviso il tutto si piantasse lasciandoti lì con messaggi criptici ai quali potevi solo rispondere con “OK” o “Annulla”. Nessuna delle due scelte rappresentava una soluzione.

Lo stesso per i programmi di editing non-lineare e compositing.

Allora ci si rivolgeva tramite mailing list agli sviluppatori stessi del software o ad altri utenti che magari avevano risolto il problema. Gli sviluppatori dimostravano (o erano costretti a farlo) una notevole pazienza nel gestire le masse inferocite di utenti insoddisfatti. La risposta alle lamentele era spesso questa: “Ragazzi, non è colpa nostra, sviluppare un’applicazione su piattaforma Microsoft Windows è un delirio, facciamo il meglio che possiamo, ma vi giuriamo che non auguriamo a nessuno di dover programmare utilizzando le API di DirectX...”, eccetera.

Una volta un utente rispose qualcosa di illuminante: “Ma allora diteci chiaro e tondo che abbiamo comperato un software che non funziona, diteci chiaro e tondo che paghiamo per essere beta-tester dei vostri programmi!”

Verissimo: pagavamo per testare i programmi, cosa che avrebbe dovuto fare l’azienda produttrice dei medesimi prima di rilasciarli sul mercato. Intendiamoci, lo sapevo anche allora che il ciclo di vita di un programma è qualcosa che non si conclude mai, pena la morte del programma stesso, che un software è un continuo work in progress, ecc.; ma dei punti fermi, stabili, delle *major release* ci devono pur essere. E devono garantire le funzionalità che dichiarano di possedere.

Nel frattempo, per riuscire a produrre qualcosa, tra l’email di supporto e

ore passate a spulciare le mailing list, i forum degli utenti e le chiacchiere nei canali dedicati delle chat-line il mercoledì a notte fonda, stavo imparando a far funzionare il sistema operativo, per far girare i programmi.

Stavo diventando un sistemista, non un regista.

Poi un giorno, su un qualche canale irc, qualcuno dice che non ne può più, che ha provato a utilizzare un sistema operativo che si chiama Linux e che funziona, o meglio è stabile, fa quello che dice di fare, ti permette di lavorare. Ci sono applicazioni *multimediali* (all'epoca anche io utilizzavo il termine *multimediale* come prezzemolo linguistico in ambito informatico) per Linux? No, non ancora, ma ci sono tanti progetti in fase di sviluppo, sono Software Libero, ovvero il codice sorgente è rilasciato disponibile a chiunque a patto che a sua volta, se lo modifica, lo rilasci con la stessa licenza.

Ma io non so programmare, e non credo nemmeno di voler imparare. Però provo lo stesso, ci provo talmente tanto che per l'inizio del 2000 trovo pure un lavoro, grazie a GNU/Linux, e adesso faccio l'amministratore di reti di computer.

E il regista?

In questi anni è stato a guardare (un po' preoccupato) il sistemista, l'amministratore di reti e il programmatore che alla fine la programmazione l'ha studiata davvero. Incuriosito, prendendo nota di tutte quelle volte in cui percepiva che una determinata tecnologia avrebbe potuto aiutarlo a fare quello che lui intendeva per "cinema": guardare per cercare di capire, montare quello che si vede per scovare il senso che si trova tra un *cut* e quello successivo.

Curiosamente, più osservava più notava che non erano i programmi dedicati al cosiddetto *media-authoring* quelli che potevano tornargli utili, le grandi suite di montaggio video e post-produzione già utilizzate in passato e che ora stavano piano piano vedendo la luce anche sotto piattaforma GNU/Linux: per quanto preposte alla generazione dell'oggetto film non sembravano poter fare molto di più dei vecchi strumenti analogici per quanto concerneva la sua idea di cinema. Era invece sempre più evidente che programmi, a volte singole parti di programmi, pensati per fare tutt'altro che cinema potessero venire impiegati per contribuire allo sviluppo di un sistema coerente (il sistemista avrebbe usato il termine *framework*) che *facesse cinema*, e che lo facesse insieme all'intervento di altre persone (qui il termine è *interattività*).

All'inizio del 2003 ho cominciato a raccogliere i pezzi indicatimi dal regista negli ultimi tre anni e a cercare di metterli insieme per sviluppare

questa macchina interattiva per fare cinema.

Si chiama *Eveline*.

Di seguito vengono riportati e descritti gli elementi che la compongono, che cosa possono fare per Eveline e perché.

Content Based Image Retrieval Systems

Lo sviluppo del desktop publishing nell'ambito del personal computing, la progressiva convergenza verso la tecnologia digitale di numerosi dispositivi di produzione di immagini analogici di utilizzo oramai quotidiano (fotocamere, videocamere), la sempre maggiore accessibilità di strumenti di digitalizzazione di immagini analogiche un tempo riservati ai professionisti (scanner) e più in generale il progressivo aumento di potenza e capacità di storage dei moderni microcomputer nonché il loro sempre minore costo, hanno prodotto negli ultimi dieci anni un'enorme quantità di immagini digitali.

La tecnologia di Internet e lo spazio del World Wide Web ci permettono di distribuire e rendere reperibili queste immagini ad un numero sempre maggiore di persone, originando quelli che potremmo concepire come veri e propri database distribuiti di immagini sempre più estesi. Il testo è il tipo di oggetto mediale cui si riesce ad accedere nella maniera migliore. Fin dagli anni '60 sono state sviluppate diverse tecniche che aiutano nell'indicizzazione e recupero di testi elettronici, come inverted indexes, stop-word lists, clustering, relevance feedback e thesauri. Date queste premesse, il primo approccio all'indicizzazione di immagini fu quello di annotare le immagini stesse tramite parole-chiave e di utilizzare dunque gli strumenti già in possesso nell'ambito testuale per operare su queste annotazioni in fase di recupero. Fu però presto chiaro come l'utilizzo di query testuali per file di immagine fosse inadeguato, e di quanto si rendessero sempre più necessari sistemi di ricerca appropriati allo specifico dell'elemento da recuperare.

I *Content Based Image Retrieval System (CBIRS)* – sistemi di recupero immagini in base al contenuto – nascono per rispondere a questo tipo di esigenza e si basano su concetti presi in prestito da ambiti quali la computer vision, il pattern matching, la psicologia cognitiva e molti altri, al fine di estrapolare dalle immagini caratteristiche ad esse, e solo ad esse, proprie e intrinseche, funzionali all'indicizzazione e a processi di recupero automatizzati.

Un ambito di utilizzo tipico è quello pubblicitario, dove ci si trova spesso a dover cercare rapidamente tra migliaia un'immagine che corrisponda a

determinate caratteristiche, oppure in ambito giornalistico, dove ancora una volta si deve reperire in maniera veloce ed efficace un'immagine pregnante ad un determinato contesto.

Come funzionano più in dettaglio i CBIRS.

La maggior parte dei file di immagine sono mappature di entità appartenenti al mondo reale in una forma binaria. Questi file contengono due tipi base di informazioni: quelle che si riferiscono ad attributi e relazioni appartenenti ad entità del mondo reale (i contenuti) e quelle che sono specifiche alle rappresentazioni binarie di tali entità (la loro codifica). Qualunque sia la rappresentazione binaria (la codifica) che noi scegliamo, il contenuto di un'immagine dovrebbe rimanere lo stesso. Al tempo stesso il contenuto di un'immagine non dipende dalla sua risoluzione. Una bassa qualità o una bassa risoluzione potrebbero rendere più difficile estrapolare caratteristiche descrittive dell'immagine stessa: ciò che è importante sottolineare è che il formato dell'immagine e il tipo di compressione ad essa applicata, così come la risoluzione e i colori, sono elementi specifici alla codifica binaria dell'immagine stessa e non al suo contenuto. I CBIRS operano *a basso livello* sulla codifica delle immagini per consentire all'utente di recuperare in maniera veloce ed efficiente il loro contenuto. Date una serie di immagini queste vengono processate operando su di esse determinati algoritmi: questi algoritmi estrapolano dalle immagini determinate informazioni, definite *firma* (*signature*) dell'immagine.

In base all'algoritmo implementato queste informazioni possono dirci quali colori sono presenti in una data immagine, dove si trovano nell'immagine e in quale misura e percentuale, quali contorni presenta l'immagine, quali linee tendono ad essere presenti in maggiore percentuale all'interno dell'immagine, quale la loro direzione, etc.

Ogni *firma* di ogni immagine viene inserita in un database; quando un utente interroga il database presentando al sistema un'immagine di riferimento, questa stessa immagine subisce i processi precedentemente applicati a quelle già presenti nel database, ovverosia indicizzate: anche per essa viene dunque estrapolata una *firma* che verrà confrontata con quelle già presenti nel database, permettendo così all'utente di recuperare l'immagine che più assomigli a quella presentata.

Per il recupero di immagini in base al contenuto si identificano di norma le seguenti tipologie di richiesta (query):

- *Diretta*: l'utente sa esattamente che cosa sta cercando e conosce quali chiavi il sistema utilizza per identificare un particolare oggetto.
- *Per similitudine* (*Query-by-example*): l'utente seleziona uno o più documenti o parte di un documento che sa essere simili al tipo di

documento che sta cercando.

- *Per Prototipo (Query-by-sketch)*: questa tecnica è correlata con la precedente. Il prototipo in questione potrebbe essere un disegno improvvisato dall'utente al momento di effettuare la query o un oggetto interpretato dal sistema per produrre una particolare rappresentazione.

Per quanto concerne gli algoritmi con i quali vengono processate le immagini gli approcci sono essenzialmente due: da un lato l'utilizzo di funzionalità quali *color histogram* e caratteristiche relative alla forma come *circularity* e orientamento degli assi principali di regioni dell'immagine, così come combinazioni di queste tecniche, dall'altro i *wavelet transform*.

Color histogram

L'utilizzo di color histogram consiste nel sovrapporre all'immagine una griglia di celle, all'interno di ognuna delle quali vengono applicati algoritmi atti al rilevamento del tipo e della percentuale di colore ivi presente. Il limite principale dei color histogram consiste nel mancare di informazioni spaziali relative al colore: se l'immagine viene per esempio ruotata o traslata le informazioni ottenute non sono più valide. I color histogram restano comunque un valido strumento al servizio di un CBIRS, a patto che le immagini restino posizionate e orientate così com'esse si trovano al momento della loro indicizzazione.

Un esempio di CBIRS basato su *color histogram* è QBIC, sviluppato da IBM:

QBIC (1993 - <http://www.qbic.almaden.ibm.com>)

Il progetto di IBM Query By Image Content (QBIC) si basa su una combinazione di sistemi di indicizzazione automatici e semi-automatici. QBIC impiega funzionalità di color histogram, texture features, shape features e sketch features; le caratteristiche estrapolate dall'utilizzo di queste metodologie vengono salvate insieme alle immagini.

Caratteristica di QBIC è quella di consentire all'utente la composizione di una richiesta in base a diversi attributi visivi; per esempio l'utente può specificare una particolare composizione cromatica (x% del colore 1, y% del colore 2, etc.), una particolare texture, alcune caratteristiche relative alla forma e uno schizzo dei contorni dominanti nell'immagine target, insieme ai pesi relativi da dare ad ognuno di questi attributi.

QBIC viene utilizzato all'interno del sito web del museo dell'Hermitage per consentire agli utenti di reperire opere simili a quelle selezionate o per trovare opere che presentino colori e forme disegnate dall'utente stesso su

una tavolozza. Questo è un buon esempio delle funzionalità dei color histogram: i dipinti saranno sempre presentati e usufruiti dall'utente in una determinata posizione, le informazioni spaziali possono quindi venire ignorate.

E' possibile vedere QBIC all'opera sul sito web del museo dell'Hermitage, presso

<http://www.hermitagemuseum.org/cgi-bin/db2www/qbicSearch.mac/qbic?selLang=English>

Wavelet transform

Il wavelet transform è uno strumento emerso dalla comunità matematica nel corso dell'ultima decade per analizzare funzioni a diversi livelli di dettaglio. Le wavelet sono funzioni matematiche che dividono i dati ad esse sottoposti in diverse componenti di frequenza e successivamente studiano ogni componente con una risoluzione proporzionata alla sua scala. Nell'ambito delle immagini il wavelet transform è stato impiegato soprattutto per quanto concerne la compressione. Salvando i coefficienti più ampi di un'immagine sottoposta ad un algoritmo di wavelet transform (e buttando via tutti i coefficienti più piccoli) è possibile ottenere una rappresentazione dell'immagine in gran parte ancora accurata/corretta. Si può pensare ad un CBIRS che applichi ad ogni immagine un wavelet transform, si limiti a salvare un numero ridotto di coefficienti per ogni canale di colore (abbiamo visto come la "descrizione" dell'immagine sarà comunque ancora corretta) e distilli una piccola "firma" per ognuna delle immagini. Grazie alle dimensioni così ridotte di tale firma, la sua ricerca in un database potrà avvenire in tempi molto brevi. Rispetto ai tradizionali metodi di tipo Fourier (come i color histogram) le wavelet presentano numerosi vantaggi nell'ambito dell'analisi, primo fra tutti quello di poter formulare query ad ogni risoluzione, eventualmente diversa da quella dell'immagine target, nonché il fatto di preservare le informazioni spaziali relative all'immagine. Inoltre il tempo richiesto dalla creazione del database di immagini è indipendente dalla risoluzione delle immagini stesse. Tramite l'utilizzo di wavelet le informazioni di firma possono venire estrapolate da versioni *wavelet-compressed* dell'immagine stessa consentendo al database di firma di venir creato in maniera veloce e conveniente direttamente a partire da un set di immagini compresse.

Un esempio di CBIRS basato su *wavelet transform* è ImgSeek, sviluppato da Ricardo Niederberger Cabral e rilasciato sotto licenza GNU GPL:

ImgSeek (2002 – <http://imgseek.sourceforge.net>)

Per utilizzare ImgSeek l'utente formula una richiesta ad un database di immagini utilizzando l'interfaccia utente per "dipingere" uno schizzo di ciò che vuole trovare o fornendo al sistema un'immagine-esempio. La ricerca dell'immagine maggiormente simile può venire ostacolata da diversi fattori. L'immagine-esempio è di norma diversa dall'immagine-target, quindi il metodo di recupero deve consentire alcune distorsioni. Se l'immagine proviene da uno scanner potrebbe presentare artefatti quali micro spostamenti di colore, scarsa risoluzione, effetti di dithering, etc. Se invece l'immagine fosse dipinta dall'utente, presenterebbe tutti gli eventuali limiti artistici dell'utente stesso. Ciò nonostante, l'obiettivo è quello di recuperare le immagini in maniera veloce e interattiva anche con database contenente migliaia di immagini.

La scelta degli sviluppatori di ImgSeek è caduta sul wavelet transform.

Durante la fase di preprocessing viene effettuato un wavelet transform su ogni immagine presente nel database. Considerando solo i coefficienti più alti restituiti dalla decomposizione si genera una piccola "firma" per ogni immagine. Le firme vengono salvate e organizzate in maniera tale da rendere facile e veloce la comparazione di ognuna di esse con una nuova firma. Quando un utente sottopone una richiesta viene effettuata una wavelet transform che genera una firma per l'immagine sottoposta. Questa nuova firma viene confrontata con quelle relative alle immagini del database e quella che corrisponde meglio viene recuperata e consegnata all'utente.

Le firme generate da ogni wavelet transform sono molto piccole, la qual cosa rende ImgSeek veramente molto veloce nell'indicizzazione e nella ricerca e decisamente efficace nel trovare il cosiddetto "best match".

Dai Content Based Image Retrieval Systems ai Content Based Video Retrieval Systems

Il passo da sistemi di indicizzazione che operano su immagini a sistemi che operano su segmenti video è breve: i secondi applicano ai primi la dimensione temporale. Se nel caso delle immagini applico algoritmi su due dimensioni spaziali il video introduce il vettore temporale, ovverosia dovrò operare su n fotogrammi di x per y dimensioni.

I *Content Based Video Retrieval Systems* (CBVRS) permettono di trovare sequenze video in un database. Operano di norma dividendo il video in segmenti (*video segmentation*), indicizzando questi segmenti e fornendo all'utente funzionalità di interrogazione del database generato (*video querying*).

La segmentazione di uno stream video, ovverosia rilevare i cambiamenti bruschi tra gruppi vicini di fotogrammi, è molto simile alla ricerca di immagini così come viene intesa da un CBIRS: si tratta di confrontare immagini. La ricerca di una sequenza in un database di video è analoga alla ricerca di un'immagine in un database di immagini.

Anche nell'ambito dei CBVRS l'utilizzo dei color histogram è stato fino a poco tempo fa l'approccio maggiormente seguito. Operando nel dominio dei pixel, cercando di rilevare i tagli (*cut*) presenti in uno stream video in base alla similitudine nella distribuzione dei colori nei fotogrammi, si sono raggiunti buoni risultati, specie abbinando le caratteristiche rilevate sotto questo profilo con quelle relative al movimento, alla forma e al contorno. Il problema principale è sempre stato quello relativo alle notevoli risorse computazionali necessarie a eseguire tali operazioni per ogni fotogramma disponibile, il che vuol dire per esempio nel caso di uno stream video proveniente da una videocamera PAL lavorare su 25 immagini al secondo alla risoluzione di 720x576 pixel per 24 bit di profondità di colore. La maggior parte delle applicazioni di CBVRS basate su color histogram girano su workstation Silicon Graphics ad alte prestazioni.

Applicate alla segmentazione e indicizzazione di stream video le wavelet si sono rivelate essere ancora una volta uno strumento estremamente efficace. Abbiamo già visto come consentano un'approssimazione molto buona al contenuto dell'immagine anche con solo pochi coefficienti: questa proprietà è di fondamentale importanza quando si ha a che fare con compressione di tipo lossy, il che è quanto accade nella maggioranza dei casi nell'ambito della compressione video. I coefficienti risultano di una decomposizione tramite wavelet forniscono informazioni indipendenti dalla risoluzione originale dell'immagine, permettendo così confronti tra immagini, quindi anche video, di diversa risoluzione.

L'autore di ImgSeek, Ricardo Niederberger Cabral, aveva cominciato a sviluppare un CBVRS basato su wavelet prima di decidere di dedicarsi esclusivamente a un CBIRS, con l'intenzione di aggiungere funzionalità di recupero video in futuro. La decisione da parte di Cabral di rilasciare il codice già scritto su SourceForge mi ha permesso di procedere più velocemente nello sviluppo di Eveline, senza dover reimplementare da zero determinate funzionalità già sufficientemente operative.

Il CBVRS di Cabral, VideoQuery (<http://videoquery.sourceforge.net>), utilizza i wavelet transform nel seguente modo:

- in una prima fase i video vengono segmentati, ovverosia divisi in scene, in base ai cambiamenti bruschi rilevati tra fotogrammi vicini
- le scene vengono indicizzate in un database; per ogni scena vengono salvate informazioni relative al file di appartenenza, il fotogramma chiave

che identifica questa scena, la *wavelet signature* relativa al fotogramma stesso con i relativi coefficienti e una miniatura dell'immagine

- il motore di interrogazione del database prevede che l'utente fornisca un'immagine al sistema, che questa subisca i medesimi processi di wavelet transform applicati in precedenza ai video indicizzati ed infine che i coefficienti propri della sua wavelet signature vengano confrontati con quelli delle altre nel database.

- il fotogramma la cui signature risulta essere più simile a quella della query viene restituito all'utente, insieme a informazioni relative al file cui appartiene, il numero di fotogramma all'interno del video e il coefficiente di similitudine in base alla metrica del sistema.

Il paradigma utilizzato per la ricerca in VideoQuery è quello del *query-by-example*, nel caso specifico limitato alle sole immagini statiche come esempio da fornire al sistema, ma è già pronto il codice per permettere all'utente di fornire come query al sistema uno stream MPEG2. Il sistema restituisce all'utente un'immagine, o meglio una sua anteprima, ma anche in questo caso ho modificato il codice affinché effettuasse il play del file video cui appartiene la signature che corrisponde maggiormente, e lo effettuasse a partire dal fotogramma relativo alla signature stessa.

Vedremo più avanti qual è la parte che svolge questo Content Based Video Retrieval System in Eveline: prima parliamo di webcam e script Perl.

Webcam, feh-cam e webspiders

Canon WebView Livescope (<http://www.canon.com/www/livescope/index.htm>) è un software che permette di controllare in remoto le webcam Canon motorizzate. Ognuna di esse può ruotare il proprio campo visivo di 360 gradi sull'asse orizzontale e 45 gradi sull'asse verticale. Al proprio interno include un'interfaccia di rete Ethernet e un web server. Ci si può connettere alla videocamera e controllarla tramite un apposito client, Canon WebView Livescope, oppure tramite un browser Web utilizzando un applet java. Una volta che si è avuto accesso alla webcam, se ne può prendere possesso per un determinato intervallo di tempo e in quell'intervallo di tempo la si può ruotare sui due assi e operare sullo zoom; si può anche sentire l'audio proveniente dal microfono integrato e salvare un'immagine che ci piace particolarmente sul nostro hard-disk. La velocità di aggiornamento delle immagini ricevute è di norma piuttosto bassa: nulla impedisce a chi installa e configura il server interno alla videocamera di impostare l'aggiornamento su 25 o 30 immagini al secondo, ma questa impostazione richiederebbe un consumo di banda che

difficilmente chi impiega questi sistemi si può permettere. La videocamera viene così impostata su 12-15 fotogrammi al secondo e il client si occuperà di riceverli ad una velocità relativa alla banda disponibile all'host sul quale si trova in esecuzione.

Con Canon WebView Livescope, così come con altri sistemi di questo tipo prodotti da altre aziende, si può dunque controllare temporaneamente una videocamera che si trova dall'altro capo del pianeta rispetto al proprio computer e alla propria postazione di lavoro, ma l'esperienza in sé non risulta essere particolarmente esaltante. Ciò è dovuto soprattutto a limiti tecnici: l'impossibilità già accennata di usufruire di un'immagine in movimento fluida cui si accompagna l'ascolto di un audio a dir poco incomprensibile. Le immagini in sé però non sono male, grazie anche ad alcune caratteristiche tecniche di recente introduzione quali i CCD a scansione progressiva e l'aumento del numero di pixel sui CCD stessi.

Feh (<http://www.linuxbrit.co.uk/feh/>) è un visualizzatore di immagini molto semplice e spartano, ma anche molto veloce. Feh-cam è uno script in Perl che si occupa di invocare feh e di dargli come argomento un URL che verosimilmente punti ad una webcam, o meglio all'immagine che dalla webcam viene salvata sull'host remoto cui è di norma interfacciata. Feh-cam può prendere come argomento delle keywords che siano associate a un URL, per esempio "jenni" può puntare a <http://www.jennicam.org/webcam/cam.jpg>, e queste keywords possono venire salvate su un file.

I webspiders, o webcrawlers, sono programmi che si occupano di percorrere il Web per recuperare determinate informazioni. Chi li impiega intende di norma processare il maggior numero di documenti presenti sul Web al fine di estrapolare da essi determinate parole chiave o espressioni regolari che possono avere per lui una particolare valenza.

Si può dire che queste tre tecnologie operano sui medesimi oggetti, cioè immagini codificate in formato binario; le webcam unite a un server web si occupano di riprodurle, codificarle e distribuirle, feh-cam di recuperarle e presentarle all'utente, un webcrawler di indicizzare il Web al fine di aggiornare la lista di URL relativi a webcam che feh-cam utilizza, nonché di verificare periodicamente l'effettiva consistenza dei valori riportati, ovverosia se il server di una webcam è ancora operativo, se ha cambiato indirizzo IP, ecc. E' facile integrarle, o meglio farle dialogare assieme. Quando l'ho fatto e indirizzato feh-cam ad un URL relativo ad un dispositivo WebView, con sufficiente banda a mia disposizione da arrivare a poter visualizzare almeno 5 immagini al secondo, è successa una cosa che non avevo previsto: l'inquadratura cambiava, o meglio si spostava. Effettuava panoramiche, zoomate, si soffermava sul dettaglio di un mazzo di fiori rossi tenuti in mano da un tizio giapponese con camicia bianca a

maniche corte e pantaloni neri fermo immobile al centro di un marciapiede affollato di passanti lungo una strada centrale di Tokio. Mi ci è voluto un po' per capire cosa stesse accadendo: un utente su un computer da qualche parte del mondo aveva deciso di dare un'occhiata a quello che stava accadendo in quel momento nel quartiere Shibuya di Tokio e per farlo aveva attivato una sessione di controllo della webcam posizionata in quel particolare luogo, eseguendo il client WebView sul suo PC o lanciando l'applet Java su una pagina Web: io mi ero collocato tra lui e la webcam, tra il suo sguardo e le strade di Tokio.

Stavo guardando lo sguardo di un'altra persona.

Il cinema ha più volte cercato di arrivare a questo operando fino all'estremo sulla propria dimensione temporale, ma si è inevitabilmente ritrovato a dover fare i conti con immagini inevitabilmente *già viste* al momento della loro fruizione.

La diretta è stata fino ad oggi patrimonio esclusivo della tecnologia video, in particolare della televisione. I costi necessari per gestire l'infrastruttura necessaria ad inviare immagini in movimento da una parte all'altra del pianeta sono talmente alti da avere mortificato quello che più volte è stato indicato come "specifico televisivo". Anche per la televisione si può parlare, nonostante le potenzialità della diretta, di immagini *già viste*, cioè di immagini già passate e ripassate al vaglio di codici culturali anestetizzanti, pietrificate di fronte a qualsiasi ipotesi di destabilizzazione di anche solo uno dei diversi soggetti che popolano il sistema-televisione e per questo costrette a una coazione a ripetere infinita.

Con un semplice script in Perl, una webcam controllabile in remoto e la suite di protocolli TCP/IP posso liberare la diretta dalla mortificazione televisiva e restituire al cinema uno dei suoi obiettivi più ambiziosi, la condivisione dello sguardo. Ma per farlo devo prima uscire dal Web.

Fuori dal Web...

Io sono d'accordo con Hitchcock (e con Rohmer e Chabrol): il cinema deve essere "larger than life" [1]. Aggiungo anche "larger than the Web". Per tutta la seconda metà degli anni '90 sembrava che portare qualsiasi costrutto della mente umana sul Web fosse un obbligo. Ciò valeva specialmente per i prodotti medialti dell'uomo, la musica, le arti visive, il cinema e il teatro. Tutte queste discipline, e i loro prodotti, sembravano possedere addirittura una naturale tendenza a trovare nel Web un proprio spazio per evolversi e prosperare sotto nuove e rinnovate forme. Il teatro ha resistito meglio di tutti, anzi è riuscito a invertire il processo, ha costretto

il Web ad andare a teatro. Questo perchè quando il teatro c'è, in quel punto e in quel momento accade qualcosa di talmente *larger than life* da rifuggire da qualsiasi tentativo di riduzione e imbrigliamento.

Il cinema, che sembra aver perso da tempo le forze necessarie per farsi *larger than life*, si è lasciato subito sedurre dal Web, e con lui tutti coloro i quali in assoluta buona fede si sono convinti che il Web potesse aiutarli a fare cinema: nel giro di pochi anni tutto quello che il cinema aveva ottenuto dal Web era grande quanto il francobollo animato che vedete apparire al centro del vostro browser quando cliccate sull'immagine d'anteprima di quel cortometraggio del filmmaker americano indipendente di cui tutti vi hanno parlato tanto bene. Spesso il (breve) francobollo animato viene interrotto da messaggi relativi alla rete congestionata e al buffering in corso. Ma dietro al francobollo c'è un lavoro enorme. C'è qualcuno che ha perso il sonno per acquisire materiale audio/video con un computer, lo ha montato con un'applicazione che spesso e volentieri è andata in crash senza poi permettergli di ripristinare quanto elaborato fino a quel momento, ha aspettato tempi biblici per effettuare il rendering finale e la conversione in un formato adatto allo streaming su Internet, salvo scoprire che non può permettersi le cifre richieste da un Content Delivery Network per garantire una risoluzione di 320x240 pixel a 15 fotogrammi al secondo con audio campionato a 22Khz. Quindi noi vediamo il francobollo. E lo vediamo -soli- davanti al nostro monitor.

Alla fine del XIX secolo i cittadini americani conoscevano modalità di fruizione delle immagini in movimento molto simili. Il kinetoscope Edison permetteva a una sola persona di vedere un breve filmato grande proprio quanto un francobollo, o poco più. Qualcuno ha notato come questa modalità di visione si presentasse di nuovo poco dopo la diffusione della tecnologia video in ambito domestico, attraverso le produzioni pornografiche girate a bassa risoluzione (spesso in VHS) e visionate da pochi di fronte ad un televisore [2].

Siamo costretti, per dirla con Renato Barilli, alla “ripetizione differente” [3] oppure grazie a Internet e alle tecnologie digitali il cinema può tornare ad essere *larger than life*?

Finchè il cinema pretenderà di trovare un proprio spazio all'interno del Web quello che vedremo sarà un “nickelodeon” che stupisce e meraviglia sempre di meno e mortifica sempre di più.

Se invece il cinema decide di uscire dal Web rifiutando di essere uno dei tanti documenti *pubblicati* in esso e ribalta le regole del gioco servendosi del Web come di un enorme magazzino di tracce e segni da montare per produrre senso e tornare a raccontarci, allora sarà nuovamente qualcosa di

più della vita. Sarà di nuovo cinema.

...per servirsi del Web

Eveline è stata pensata per fare cinema e quindi se ne sta fuori dal Web. Eveline crede ancora che per condividere lo sguardo ci si debba incontrare, come si fa adesso quando si va a vedere un film. Una persona da sola seduta su una sedia scomoda davanti al monitor di un computer in attesa che compaia il francobollo animato è un'immagine patetica e straziante.

Così come ci si sposta per andare al cinema, e sappiamo che questa pratica implica qualcosa di molto simile all'andare al tempio per riunirsi e partecipare a un rito collettivo, allo stesso modo ci si dovrà spostare per andare da Eveline, perchè si starà andando a vedere cinema. E a farlo con gli altri.

Quella che segue è una descrizione del modo un cui ho pensato che Eveline debba funzionare.

Ricordate il Content Based Video Retrieval System? Può venire considerato come il cuore di Eveline.

Si occupa di accettare stream video, dividerli in scene e indicizzarli. Può accettare stream video live provenienti da una videocamera collegata al sistema così come file video disponibili su CD-ROM o DVD-ROM. Può anche ricevere in upload file video provenienti da un host remoto: questa operazione, che in base alle dimensioni del file potrebbe richiedere molto tempo e consumare molta banda verrà effettuata durante le ore di inattività del sistema.

Man mano che vengono indicizzati nel database i file video sono pronti per essere montati. Cosa determina il montaggio, come vengono decisi i cut?

Il sistema di script che gestiscono feh-cam si occupa di recuperare le immagini provenienti da una sessione di utilizzo di Canon WebView (o di un altro sistema di gestione di webcam motorizzata con server web integrato) e le fornisce al motore di query del CBVRS. A quel punto la query restituisce il fotogramma che più rassomiglia all'immagine ricevuta, e un player comincia a riprodurre il filmato cui appartiene il fotogramma esattamente a partire dal fotogramma stesso.

Un'immagine che qualcuno sta guardando determina un taglio di montaggio.

Alle tre del pomeriggio di un Mercoledì di Gennaio avvio Eveline, che diligente comincia a effettuare il play di uno dei video presenti nel suo

database. Qualche minuto dopo in una strada del quartiere Shibuya di Tokio si ferma una macchina nera dalla quale scende un tizio con camicia bianca a maniche corte, pantaloni neri e un mazzo di fiori rossi in mano. Il tizio resta fermo immobile tra la macchina e il marciapiede, affollato di gente che cammina in fretta. Non attraversa il marciapiede, se ne resta lì con i fiori rossi in mano. Qualcuno da qualche parte del pianeta lo guarda da una pagina Web, probabilmente lo colpisce la sua immobilità, si chiede, come del resto me che sto assistendo alla scena in una finestra sul mio desktop, perché il tizio non attraversi il marciapiede. Chi controlla la webcam decide di effettuare uno zoom, per vedere meglio. L'inquadratura si compone ora di un primo piano delle spalle del giapponese, i suoi capelli neri e i fiori rossi, *questi ultimi al centro dell'inquadratura*. Eveline decide di utilizzare questa immagine per interrogare il proprio database ed effettuare un cut sul video che stava renderizzando. L'ho impostata per operare in modalità automatica, sceglie lei quando tagliare.

cut.

Piano Americano su un bancone di legno, sul quale sono appoggiati vasi di fiori, *la maggior parte rossi, alcuni viola*. Alle spalle del bancone un'ampia vetrina, oltre la quale si vede gente che passa. Siamo all'interno del negozio di un fioraio, a San Francisco, nel 1940.

Uno dei video che avevo indicizzato è un filmato amatoriale in 8mm girato a San Francisco nel 1940, l'ho scaricato da www.open-video.org.

Lo sguardo di qualcuno determina lo svolgersi dello sguardo di qualcun altro.

Immagino Eveline come un interminabile film montato da immagini che si riconoscono, si ritrovano e si rincorrono l'un l'altra.

I video che vengono forniti a Eveline e man mano indicizzati, chiamiamoli contributi, dovrebbero essere stati girati direttamente dall'utente che li fornisce, con una videocamera digitale. Questo per evitare di fornire al sistema materiale ancora una volta *già visto*, già pieno di significato attribuitogli da altri contesti culturali. I contributi dovrebbero cioè rivelarsi come neutri a chi li guarda, per permettere a Eveline e a chi interagisce con essa di investirli di significato tramite il montaggio e l'utilizzo dell'interfaccia utente. Un ottimo esempio di contributi che ho immaginato di utilizzare per Eveline sono quelli prodotti dal progetto "Autoritratto Italiano", esperienza in corso di svolgimento nell'ambito del laboratorio Ipotesi Cinema (www.ipotesicinema.it) presso la Cineteca del Comune di Bologna. E' stato chiesto ai partecipanti di utilizzare le proprie videocamere digitali per tornare a rivolgere il proprio sguardo sull'Italia nella quale vivono, dato che il cinema italiano istituzionale non sembra

essere più in grado di farlo. Attualmente il progetto consiste in centinaia di cassette MiniDV: per ognuna di esse l'autore ha segnalato a chi si occuperà di acquisirle su PC i valori del timecode per gli intervalli, le sequenze, che ritiene significative. Queste sequenze sono per ogni autore il proprio sguardo migliore sulle cose che li circondano. Mi piacerebbe che Eveline operasse su questi sguardi.

Immagino dunque Eveline come un ambiente dove si va per portare i propri sguardi e metterli in circolo insieme a quelli degli altri, dove si può guardare il discorso sviluppato dal sistema in base al montaggio descritto sopra.

Allora può solo guardare chi si reca al “tempio”? L'unica interazione con il sistema concessa all'utente è collegarvi la propria videocamera o inserire un DVD masterizzato e fare l'upload dei propri contributi?

L'utente può in qualsiasi momento richiedere una sessione di lavoro su Eveline, interrompere il montaggio automatico e operare su di esso tramite un'interfaccia utente.

Quali sono le operazioni che si possono compiere su Eveline?

L'utente può innanzitutto selezionare su quale webcam Eveline debba *sintonizzarsi* e può decidere se Eveline debba procedere a un montaggio automatico o semiautomatico. In modalità automatica Eveline utilizza le immagini provenienti dalla località scelta dall'utente e a intervalli casuali sceglie un'immagine come sorgente per la query. In modalità semiautomatica è l'utente stesso a selezionare quale immagine verrà sottoposta al motore di interrogazione del database. In quest'ultimo caso è anche possibile selezionare un'immagine ma senza presentarla subito al sistema di query: si può operare su di essa, per effettuare una selezione per esempio, se vogliamo fornire al sistema solo una parte che ci interessa particolarmente.

E' anche possibile fare in modo che sia il video del quale si sta effettuando il play a gestire il montaggio. Nel momento in cui l'utente seleziona il quadro del player il fotogramma renderizzato in quel momento viene scelto per interrogare il database. In questo modo sono i video a montarsi tra di loro, il loro fotogrammi a cercarsi l'un l'altro.

Interagire per narrare

Fin qui abbiamo dunque un sistema che ci permette di montare contributi video in base a immagini provenienti da webcam controllate in remoto o dai contributi stessi. Si potrebbe facilmente estendere il sistema

per ricevere come input immagini disegnate direttamente da noi. L'utilizzo di wavelet transform rende Eveline molto veloce nel processare e indicizzare le immagini, così come nell'interrogare il database e restituire i risultati. Grazie a questa velocità il risultato è quello di un montaggio in tempo reale, decido l'immagine e subito ottengo un taglio.

E' possibile dedurre una qualche storia dai tagli effettuati, dal montaggio che Eveline opera da sola o con l'interazione dell'utente? Affinché ci sia narrazione ci deve essere una storia: è possibile raccontare una storia a partire da singoli contributi eterogenei, montati secondo gli algoritmi che guidano Eveline?

I contributi girati da me, i video prodotti dai partecipanti al laboratorio "Postazione della Memoria" in Ipotesi Cinema, i filmati amatoriali girati in 8mm dalle famiglie americane negli anni '40 e '50 che si trovano su www.open-video.org corrispondono a quelle che la narratologia definisce come "descrizioni" [4]. E la narratologia ci dice che solo con le descrizioni non c'è la storia, ci vogliono anche le azioni per portare avanti il discorso. Il montaggio effettuato da Eveline procede per ellissi, salta da una descrizione all'altra. Non che questo sia un limite dal punto di vista filmico, anzi: spesso Eveline mi spiazzava con certe illuminazioni che mi ricordano tagli di montaggio tipici di Takeshi Kitano, regista che utilizza l'ellissi come figura fondamentale per svolgere il proprio discorso. Si potrebbe addirittura aggiungere che già l'ellissi in sé implica un forte grado di interattività: richiede all'utente lo sforzo mentale di riempire il vuoto creato tra un cut e l'altro, esige un'operazione di completamento spesso molto elaborata.

Ma la storia ancora non c'è. Tra un cut e l'altro ci si può a volte trovare il senso (e non è mica poco!), ma la storia lì non la si trova.

Se Eveline in sé non è in grado di raccontare una storia con i propri contributi bisogna che lo lasci fare a qualcun altro. Magari all'utente, che sarebbe ora cominciasse a raccontare i propri desideri, la propria percezione del mondo, invece di continuare a subire quella di qualche presunto "autore" in un rapporto di continua passività rispetto alle informazioni ricevute. Del resto Godard, che autore lo è veramente, non ha forse detto che l'unico modo di fare veramente critica cinematografica sarebbe quello di rimontare un film che non ci è piaciuto e mostrarlo nuovamente al regista?

Per fare in modo che l'utente racconti una storia Eveline gli permette di lasciare delle *tracce* del proprio passaggio, tracce che arricchiscono semanticamente i contributi video e permettono di stabilire relazioni tra gli stessi.

Se l'utente sceglie tra le tante possibili immagini una in particolare è perchè probabilmente l'ha colpito in un qualche modo. Che cosa c'è in quell'immagine, cosa gli ha ricordato, che cosa gli ha raccontato, quali associazioni ha innescato nella sua mente? Non sarebbe meraviglioso se l'utente potesse raccontare agli altri questa sua storia, lasciare sull'immagine delle note, dei riferimenti ad altri media object, se potesse mostrarci la sua mente? I grandi registi non fanno altro che mostrarci quello che accade all'interno della loro mente: ciò è ad esempio evidentissimo in ogni film di Kubrick, così come nei momenti migliori del cinema di Scorsese.

Se l'utente potesse lasciare delle tracce di se stesso, delle annotazioni testuali, sonore e visive, pre-meditare che accadano determinati eventi quando gli altri utenti si ritrovano a interagire con le sue tracce, allora probabilmente ci sarebbe narrazione, staremmo manipolando segni che ci raccontano la storia di qualcuno e nel manipolarli potremmo modificare il racconto.

C'è una tecnologia che permette questo, che permette in sostanza di creare collegamenti ipertestuali non più solo tra pagine HTML ma anche tra singoli oggetti, siano essi testo – audio – video, che popolano il Web?

Esistono diversi strumenti che cercano di mettere in grado l'utente di ottenere questa ricchezza semantica, il più promettente dei quali è SMIL, il Synchronized Multimedia Integration Language (<http://www.w3c.org/AudioVideo/>), standard consigliato dal World Wide Web Consortium (W3C) [5].

SMIL è un linguaggio di markup derivato da XML. Consente di generare presentazioni multimediali interattive in maniera dinamica. E' un formato aperto, basato su testo e media-agnostico. Il linguaggio non conosce direttamente i singoli oggetti presenti in un proprio documento, la responsabilità di effettuarne il rendering e le modalità di resa sono proprie dell'implementazione del player di file SMIL. Un file .smil si occupa di dichiarare quali oggetti invocare in una presentazione, dove si trovano e come si debbano comportare, relativamente alle dimensioni dello spazio e del tempo e in base all'interazione offerta all'utente. Questi oggetti possono essere immagini, stream audio e video, semplice testo, pagine HTML, altri file smil, ecc. Possono trovarsi sul computer locale o su un host remoto, per SMIL è indifferente, si limita a referenziarli. Con SMIL si può impostare un singolo oggetto, per esempio un file video, affinché si comporti come un link a un altro video qualora lo si selezioni. Si può decidere che una determinata area della superficie del video in un determinato momento attivi il play di un file audio su protocollo rtsp (Real Time Streaming Protocol) quando il puntatore del mouse ci passa sopra.

Un file SMIL è un file di testo, e come tale modificabile e aggiornabile al volo: non si trova mai in una forma finale, come altri formati di presentazione che in quanto file binari conoscono già a priori tutte le interazioni possibili, tutti i percorsi e le scelte consentite all'utente. Tramite SMIL l'utente può non solo operare sulle scelte predefinite dal sistema, ma può operare sul file per impostare le proprie scelte, i propri percorsi, lasciare le proprie tracce interattive.

Si pensi a un sistema che generi dinamicamente codice SMIL in base all'interazione dell'utente e che dia questo codice in pasto a un player per effettuarne il parsing in tempo reale. La *timeline* della presentazione potrebbe venire decisa al momento della presentazione stessa.

Un'interfaccia utente per Eveline basata su SMIL potrebbe per esempio permettere all'utente di associare del testo all'immagine selezionata dalla webcam, o semplicemente a parti di essa, oppure associare una traccia audio a un determinato intervallo temporale del video cui appartiene un fotogramma scelto durante il play. Quando qualche altro utente opererà su Eveline e si vedrà restituire dal CBVRS il file video *annotato* dall'utente precedente, riceverà insieme ad esso anche tutte le tracce lasciate dal passaggio di quella persona. Queste tracce saranno una frase sovrainpressa sul video, una traccia audio, un link a qualche documento presente pubblicato sul Web.

Le potenzialità offerte da SMIL per un cinema interattivo sono enormi, e sono ancora in buona parte da investigare.

Quello che mi sembra renda SMIL particolarmente adatto ad essere impiegato come motore narrativo per questa nuova mutazione che il cinema dovrà subire se vuole tornare ad essere *larger than life* è il suo essere intrinsecamente omologo all'idea fondamentale di tutto il cinema migliore che ha nutrito i nostri occhi: le immagini sono già tutte là fuori, il cinema c'è già. SMIL è un linguaggio cosiddetto *dichiarativo*, non si occupa di generare contenuti, non codifica formati, per SMIL i media object sui quali operare sono già dati a priori, sono già tutti là, nel Web.

Se l'interazione con Eveline consiste essenzialmente nell'effettuare scelte, c'è poi tanta differenza tra questo modo di vedere/fare cinema e la modalità classica di visione di un film proiettato in pellicola? Se in quest'ultimo caso mi ritrovo a seguire i percorsi narrativi decisi dal regista, con Eveline non mi ritrovo forse a seguire i percorsi narrativi sempre comunque decisi da qualcun altro? Le scelte che posso operare non sono forse tutte già prestabilite, se non altro dallo sviluppatore del programma quando ha scritto il codice e deciso quali messaggi si possono inviare a Eveline e come Eveline debba comportarsi quando li riceve?

Fin tanto che l'attuale tecnologia ci permette di operare su unità minime (i bit) che conoscono solo due possibili stati (1/0, vero/falso, ecc.) la programmazione che ne scaturisce non può che essere condizionale. I limiti del mondo possibile generato dal software sono dunque tutti quelli previsti dal suo sviluppatore. Quello che posso fare è aprire la porta al mondo reale, analogico, e fare entrare nel sistema un po' di entropia. Le webcam sulle quali si sintonizza Eveline possono ruotare sul proprio asse orizzontale di 360 gradi, su quello verticale di 45. All'interno di questo spazio euclideo si trovano tutte le immagini che qualcuno può vedere in diretta e fornire a Eveline per effettuare un cut. E' uno spazio limitato, ma potremmo immaginare un futuro molto vicino nel quale videocamere dotate di interfaccia Wi-Fi e sufficientemente miniaturizzate da poter essere indossate con facilità inviano stream a 25 fotogrammi al secondo su un'infrastruttura di rete cittadina che rende disponibile per il trasporto dati una velocità nell'ordine dei gigabit per secondo. A quel punto Eveline sarebbe in grado di ampliare i propri limiti fin dove arrivano gli sguardi di un intero quartiere e successivamente di intere città.

Il cinema allora non sarebbe un paese in più sulla carta geografica del mondo, come ce l'ha consegnato il furore di *père* Godard: saremmo in presenza di una vera e propria *mappatura* [6] del cinema sul mondo, lo spazio rappresentato dalla carta geografica coinciderebbe con lo spazio dello sguardo.

Note

- 1) Antoine de Baecque, *Assalto al cinema*, Milano, Il Saggiatore, 1993
- 2) Antonio Costa, *Saper vedere il cinema*, Milano, Bompiani, 1994
- 3) Renato Barilli, *Scienza della cultura e fenomenologia degli stili*, Bologna, Il Mulino, 1991
- 4) Angelo Marchese, *L'officina del racconto*, Milano, Mondadori, 1990
- 5) Per i motivi che mi hanno spinto a scegliere questa particolare tecnologia e a ritenerla la più idonea ad affrontare in maniera corretta concetti quali multimedialità e interattività sul Web si veda <http://eveline.cinelinux.net/testi/smil.html>
- 6) E' interessante notare come il concetto di mappatura informi di sé l'intero ambito informatico: si "mappano" gli indirizzi IP privati in indirizzi IP pubblici, si "mappano" le entità nella programmazione orientata agli oggetti al fine di astrarre sempre di più il linguaggio dalla macchina, si "mappano" le directory di un filesystem su disco, ecc...

Bibliografia

CBIRS e CBVRS:

“Fast Multiresolution Image Querying”, C. E. Jacobs, A. Finkelstein, D. H. Salesin, Department of Computer Science and Engineering - University of Washington, Seattle, USA

“Content-Based Retrieval from Digital Video”, V. Roth, Fraunhofer Gesellschaft Institut fuer Graphische - Darmstadt, Germany, 1999

“Distributed Image Indexing and Retrieval with Mobile Agents”, V. Roth, Fraunhofer Gesellschaft Institut fuer Graphische - Darmstadt, Germany, 2002

“A Robust CBIR Approach Using Local Color Histograms”, S. Wang, Department of Computing Science - University of Alberta Edmonton, Alberta, Canada, 13 October 2001

“Content-based query of image databases, inspiration from text retrieval: inverted files, frequency-based weights and relevance feedback”, D. McG. Squire, W. Muller, H. Muller, J. Raki, Computer Vision Group Computing Science Center - University of Geneva, Geneva, Switzerland, 1 December 1998

“MRML: A Communication Protocol for Content-Based Image Retrieval”, W. Muller, H. Muller, S. Marchand-Maillet, T. Pun, Computer Vision Group - University of Geneva, Geneva, Switzerland, 14 February 2000

“Content-based Video Retrieval: An overview”, S. Marchand-Maillet, Viper Team – CUI Université de Genève , Geneva, Switzerland, 2000

“A Society of Models for Video and Image Libraries”, Rosalind W. Picard, Media Lab, Cambridge M.A. 23 April 1996

“Wavelet-Based Video Indexing and Querying for a Smart VCR”, Xiaodoung Wen, D. Huffmire, Helen H. Hu, A. Finkelstein, Princeton University 2002

“An Introduction to Wavelets”, Amara Graps, Institute of Electrical and Electronics Engineers 1995

SMIL – Synchronized Multimedia Integration Language:

Bulterman, Dick, e Rutledge Lloyd SMIL 2.0 – *Interactive Multimedia for*

Web and mobile devices (sarà pubblicato nel 2004 per Springer – gli autori mi hanno dato accesso al manoscritto)

Bulterman, Dick, *SMIL 2.0 – Part1 Overview, Concepts, and Structure*. Siemens Corporate Research. <http://www.computer.org/multimedia/mu2001/pdf/u4082.pdf>

Bulterman, Dick, *Ambulant Phase1: A Multi-Profile SMIL Player for Mobile and Desktop Systems*.

Arciniegas, Fabio, *A Realist's SMIL Manifesto*. XML.com. <http://www.xml.com/pub/a/2002/05/29/smil.html>

Arciniegas, Fabio, *A Realist's SMIL Manifesto, Part II*. XML.com. <http://www.xml.com/pub/a/2002/07/17/smil.html>

Yoshimura, T., Yonemoto, Y., Ohya, T., Etoh, M. e Wee, S., *Mobile Streaming Media CDN Enabled by Dynamic SMIL*. NTT DoCoMo, Multimedia Laboratories. <http://www2002.org/CDROM/refereed/515/>

Kumar, Sujai e Miller, Kevin, *Let SMIL be your umbrella: Computerized tools for automating presentation and analysis of digital video in behavioral research*. University of Illinois at Urbana-Champaign. http://www.psych.uiuc.edu/~kmiller/smil/smil_umbrella.htm

Narrazione e interazione nei nuovi media:

Lev Manovich, *The Language of New Media*, Cambridge (Mass.), MIT Press, 2001

Michael Dertouzos, *La rivoluzione incompiuta*, Milano, Apogeo, 2002

Angelo Marchese, *L'officina del racconto*, Milano, Mondadori, 1990

“Hyperdoc: An Adaptive Narrative System for Dynamic Multimedia Presentations”, in David E. Millard, Christopher Bailey, Timothy Brody, David Dupplaw, Wendy Hall, Steve Harris, Kevin R. Page, Guillermo Power, Mark J. Weal, *Intelligence, Agents, Multimedia*, Dept. of Electronics and Computer Science, University of Southampton, U. K. - 2002

“Dynamic Generation of Intelligent Multimedia Presentations through Semantic Inferencing” - Suzanne Little, Joost Geurts, Jane Hunter - ITEE Dept. University of Queensland, CWI – Amsterdam, DSTC, Brisbane – 2002

“Finding the Story – Broader Applicability of Semantics and Discourse for Hypermedia Generation” - Lloyd Rutledge, Matin Albernik, Rogier

Brussee, Stanislav Pokraev, William van Dienen e Mettina Veenstra –
CWI e Telematica Instituut – 2002

“Hybrid Narrative and Categorical Strategies for Interactive and
Dynamic Video Presentation Generation” - Craig A. Lindley, Frank Nack –
CSIRO e CWI – 2003