# UNIVERSITÀ DEGLI STUDI DI PADOVA

Facoltà di Ingegneria
Corso di Laurea in Ingegneria dell'Automazione L.M.

Progettazione di Sistemi di Controllo a.a. 2009/10

# Camera Selection

Tommaso Andreaus Matricola: 586325/IAM
email: *tom.andreaus@gmail.com*
Guido Cavraro Matricola: 603380/IAM
email: *guidocavraro@yahoo.it*
Andrea Vezzaro Matricola: 604255/IAM
email: *vezzright@gmail.com*


Padova, 15 Febbraio 2010

# Camera Selection

Tommaso Andreaus, Guido Cavraro, Andrea Vezzaro
Universitá degli studi di Padova
Dipartimento di Ingegneria dell'Informazione

15 Febbraio 2010

## Abstract

***Il progetto prevedeva la creazione di algoritmi per la* camera selection *al fine di velocizzare la ricostruzione della posizione nello spazio dei marker, nell'ambito del* motion capture*. Sono stati sviluppati due tipi di algoritmi. Il primo utilizza un approccio* dinamico*, cioè calcola dopo ogni nuova acquisizione* ex-novo *le associazioni fra videocamere. Il secondo sfrutta invece un approccio* statico*, cioè calcola a priori l'associazione, che non viene più modificata. Infine è stata testata l'efficienza di questi algoritmi, intesa come* rate *fra i marker associati ad ogni livello con quelli con quelli che rimanevano da associare dal livello successivo. Inoltre si è studiato come la suddetta efficienza venga modificata al variare del numero di marker o di telecamere presenti nella scena. In base a queste osservazioni, l'algoritmo migliore sembra essere quello di selezione statica, che dà prestazioni simili a quelli di selezione dinamica ma praticamente azzera i tempi di calcolo, essendo eseguito a priori dopo la calibrazione delle telecamere.***

## Introduzione

Col seguente lavoro ci si pone il problema dell'individuazione di algoritmi efficienti per l'associazione di videocamere in applicazioni di motion capture. Il sistema considerato consta di uno spazio tridimensionale nel quale si muove il soggetto di cui si vuole captare il movimento, ripreso da un elevato numero di videocamere. L'acquisizione delle informazioni sul moto avviene tramite il riconoscimento da parte delle videocamere di appositi markers, inserti di materiale riflettente di diverse dimensioni, posti sul soggetto considerato (nel caso, ad esempio, di human tracking su viso, arti e busto dell'attore ripreso). La stima delle posizioni dei vari markers nello spazio tridimensionale che si susseguono nella finestra temporale presa in considerazione fornisce la ricostruzione del movimento dell'intero soggetto.

Allo stato attuale, il sistema di *Vicon* acquisisce le immagini di tutte le videocamere utilizzate (con una frequenza di 120 Hz) e le elabora offline per ricostruire il movimento in 3D. Il problema principale di tale approccio consiste nell'elevato tempo di elaborazione delle immagini dovuto alla grande quantità di dati da processare, e al grande numero di confronti fra immagini, alla ricerca di raggi da *matchare*. L'elaborazione offline consiste infatti nel confrontare fra loro tutti gli $N$ frame acquisiti allo stesso istante da tutte le telecamere, per un totale di $N(N-1)/2$ operazioni.

Vista l'intenzione di Vicon di aumentare considerevolmente il numero di telecamere (fino a 200 secondo le nostre informazioni), si vede subito che c'è l'esigenza di trovare una nuova strategia per la ricostruzione della posizione dei marker, per evitare l'esplosione dei

tempi di calcolo (200! confronti per 120 acquisizioni al secondo). Di qui nasce l'idea sviluppata nel corso di *Progettazione di Sistemi di Controllo*, il cui scopo è ovviamente, creando un albero binario delle associazioni, quello di abbassare il numero di operazioni di confronto per frame da $\frac{N(N-1)}{2}$ a circa $N \sum_{i=1}^{log_2 N} \frac{1}{2}^i$ Ed è proprio la ricerca di algoritmi per la creazione di un albero binario ottimo lo scopo del nostro lavoro.

# 1 Space Partition

Sia l'algoritmo di associazione dinamica che quello di associazione statica lavorano su uno spazio discretizzato. La scena viene cioè divisa in molti *voxel*, ciascuno dei quali è etichettato con le coordinate del suo baricentro. Queste sono misurate rispetto al *world frame*, posto, per comodità, in maniera tale che non vi siano voxel con etichette a componenti negative.

Molto importante è la scelta della lunghezza $d$ dei lati dei voxel. Uno degli elementi di cui abbiamo tenuto conto in questa scelta è la dimensione dei marker. Sarebbe controproducente scegliere un valore $d$ inferiore alla dimensione massima (3-4 cm), o ad esso troppo vicino. La probabilità che un marker stia su più voxel contemporaneamente sarebbe troppo alta, così come la probabilità di avere un grandissimo numero di voxel vuoti. Ciò aumenterebbe a dismisura l'onerosità computazionale degli algoritmi di associazione. D'altro canto, anche un valore troppo grande presenta i suoi svantaggi. Soprattutto quando si deve discretizzare con accuratezza un volume, quale ad esempio il *field of view* (FOV) di una telecamera. Sarebbe infatti disdicevole approssimarlo con un volume troppo grande: si correrebbe il rischio di credere che siano visti alcuni marker che in realtà non lo sono, andando a compromettere l'efficienza degli algoritmi di associazione. Dunque con il decrescere della dimensione dei voxel aumenta l'accuratezza, ma si riduce anche la velocità del sistema. Perciò, come suggerito in [3], abbiamo scelto un valore $d = 20$ cm.

Grazie alla partizione si è in grado di stabilire per ciascuna telecamera quale porzione della scena è interna al suo FOV e, inversamente, da quale telecamera ciascun voxel è visto. L'algoritmo è implementato dalla funzione *space_partition*, e verrà di seguito illustrato.

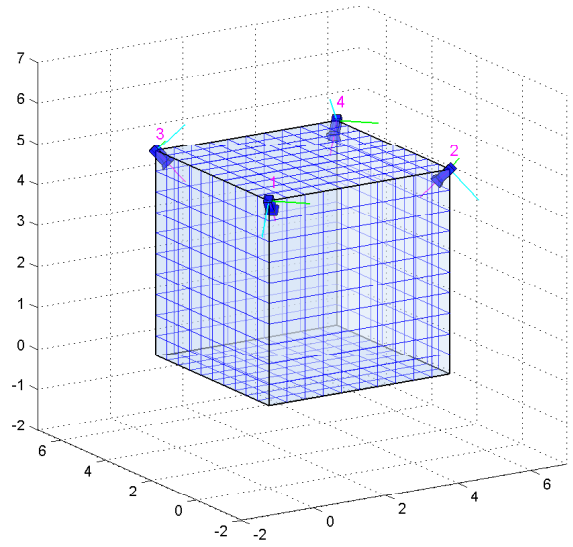


Figura 1: Partizione dello spazio in Voxels

Fissata una telecamera, se ne estraggono i parametri estrinseci $R_c^w$ e $T_c^w$, che rappresentano la sua orientazione e la sua posizione rispetto al *world frame*. Dopodichè comincia la conta dei voxel interni al suo campo di visibilità. A questo scopo, la posizione di ogni voxel viene espressa nel *camera frame*, il sistema di riferimento centrato sul centro ottico della telecamera e con l'asse $Z$ coincidente con il suo asse ottico, attraverso la

$$P^c = \begin{bmatrix} p_{xc} \\ p_{yc} \\ p_{zc} \end{bmatrix} = R_c^w \begin{bmatrix} p_{xw} \\ p_{yw} \\ p_{zw} \end{bmatrix} = (R_c^w)^T P^w \qquad (1)$$

dove $P^w$ è la posizione del baricentro del voxel (la sua etichetta) rispetto al *world frame*.

Il campo di visibilità di una telecamera è generalmente una piramide a base rettangolare (quadrata nel caso delle smart camera di Vicon). Il vertice di questa piramide è situato nel centro ottico, e il suo asse coincide con l'asse della telecamera (l'asse $z$ del *camera frame*). Sia $\theta^*$ l'angolo fra la base della piramide e le facce laterali. E' intuitivo vedere che il campo

di visibilità è costituito dall'intersezione di questi due insiemi:

$$I_1 = \left\{ \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} : [x_c\ z_c] \begin{bmatrix} 0 \\ 1 \end{bmatrix} = Kcos\theta, \theta < \theta^* \right\} \quad (2)$$

$$I_2 = \left\{ \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} : [y_c\ z_c] \begin{bmatrix} 0 \\ 1 \end{bmatrix} = Kcos\theta, \theta < \theta^* \right\} \quad (3)$$

che visualizzati nello spazio danno origine a due specie di prismi. Essi costituiscono il luogo dei punti del sottospazio $S = \{[x_c\ y_c\ z_c]^T : z_c \geq 0\} \subset R^3$ la cui proiezione, rispettivamente, sul piano $(x, z)$ o $(y, z)$ forma con il versore $u_z$ del *camera frame* un angolo inferiore a $\theta^*$.

Ecco quindi che il vettore trovato in precedenza $P^c = [p_{xc}\ p_{yc}\ p_{zc}]^T$, viene proiettato sui piani $(x, z)$ e $(y, z)$ e le proiezioni normalizzate

$$P^c_{xz} = \frac{1}{||[x_c\ z_c]||} \begin{bmatrix} x_c \\ z_c \end{bmatrix}\ P^c_{yz} = \frac{1}{||[y_c\ z_c]||} \begin{bmatrix} y_c \\ z_c \end{bmatrix} \quad (4)$$

Gli angoli fra questi vettori e il versore $u_z$ semplicemente si trovano con le

$$\theta_{xz} = \arccos(P^c_{xz} \cdot u_z) \qquad \theta_{yz} = \arccos(P^c_{yz} \cdot u_z) \quad (5)$$

Se $\theta_{xz} \leq \theta$ e $\theta_{yz} \leq \theta$, allora il voxel è interno al campo di visibilità della telecamera.

Spesso si devono porre delle limitazioni alla profondità del campo visivo delle telecamere, per modellare le imprecisioni e le difficoltà nella vista di marker troppo vicini o troppo lontani. Nella fattispecie, è stata introdotta una distanza minima, 0.5 m, e una massima, 5 m, dalla telecamera entro le quali devono giacere i voxel per poter dire di essere visti. Questa limitazione si ottiene semplicemente verificando che

$$||P^c|| < 0.5 \qquad ||P^c|| > 5 \quad (6)$$

# 2 Associazione Dinamica

Ora verranno illustrati gli algoritmi da noi implementati per ottenere l'associazione dinamica delle telecamere. Il loro svolgimento può essere suddiviso in due parti:

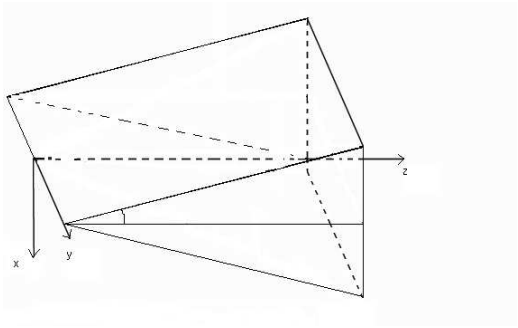


Figura 2: rappresentazione grafica dell'insieme $I_1$

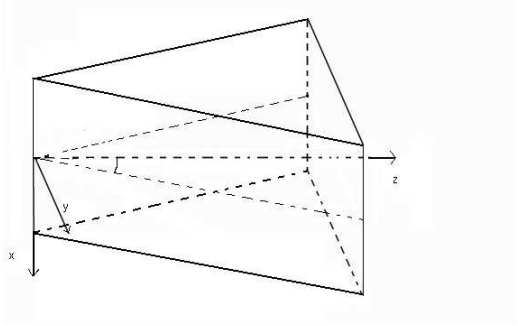


Figura 3: rappresentazione grafica dell'insieme $I_2$

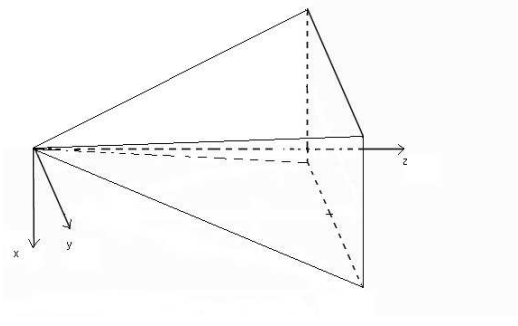


Figura 4: rappresentazione grafica del campo di visibilità di una telecamera

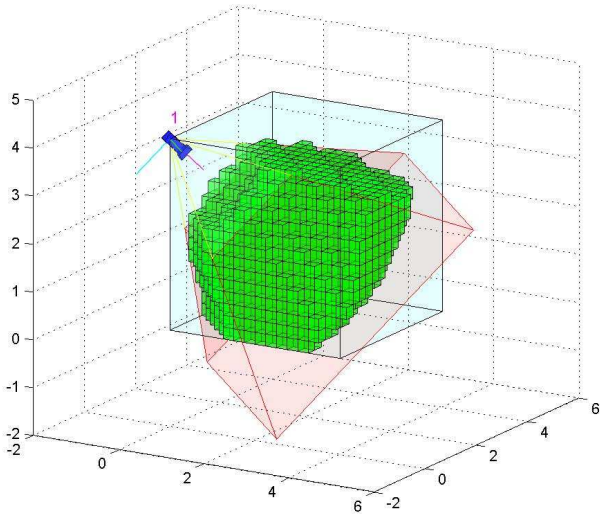


Figura 5: Partizione del campo di visibilità di una telecamera

1. una decentralizzata, in cui ogni smart camera può elaborare indipendentemente dalle altre i dati da essa acquisiti

2. una centralizzata, in cui il sistema centrale utilizza i risultati delle elaborazioni precedenti per ottenere l'associazione vera e propria

## 2.1 Sovrapposizione di immagini

L'elaborazione decentralizzata consiste nel confrontare i blob estratti dai frame correnti con le predizioni delle posizioni dei marker. Se $I_N(t)$ è l'immagine al tempo $t$ registrata dall'$N$-esima telecamera, essa viene elaborata per verificare la presenza in essa dei marker la cui posizione predetta ($\hat{x}(t|t-1)$) risulta essere all'interno del cono di visibilità della telecamera. Questa è la differenza fondamentale rispetto all'approccio statico, in cui non si verifica se i marker che, secondo le predizioni, dovrebbero essere visibili lo sono veramente. Tali elaborazioni vogliono associare, in maniera probabilistica, a ciascun blob rilevato nel frame, un marker. Ciò viene eseguito dalla funzione *im_overlap*.

Concettualmente, essa compie una sovrapposizione fra l'immagine reale $I_N(t)$ e un'*immagine fittizia* $\hat{I}_N(t)$, nella quale sono riportate le aree in cui, in base alle predizioni, i marker dovrebbero giacere. Spieghiamoci meglio: il modello di stato più utilizzato nelle applicazioni di motion capture per rappresentare l'evoluzione temporale della traiettoria di un marker è la passeggiata aleatoria:

$$\begin{array}{rcccl} x(t+1) & = & Ax(t) & + & w(t) \\ y(t) & = & Cx(t) & + & v(t) \end{array} \tag{7}$$

con $x(t) = \begin{bmatrix} p_x(t) & p_y(t) & p_z(t) & \end{bmatrix}'$,

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

con $w(t)$ gaussiano a media nulla e varianza $Q$ e $v(t)$ gaussiano a media nulla e varianza $R$. Dalla teoria della stima segue che $\hat{y}(t|t-1)$ è un vettore aleatorio gaussiano di media

$$\begin{bmatrix} \hat{p}_x(t|t-1) \\ \hat{p}_y(t|t-1) \\ \hat{p}_z(t|t-1) \end{bmatrix} = \begin{bmatrix} p_x(t-1) \\ p_y(t-1) \\ p_z(t-1) \end{bmatrix}$$

e varianza

$$\sigma^2 \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Perciò ogni marker all'istante $t+1$ si troverà all'interno di una sfera di centro $[p_x(t)\ p_y(t)\ p_z(t)]^T$ e raggio $3\sigma$ (in realtà si troverà all'interno della sfera con una probabilità circa del 99,7%). Queste sfere proiettate sui piani immagine delle telecamere danno origine ad aree circolari di raggio proporzionale a $3\sigma$ (tanto più grande quanto più la posizione predetta del marker è vicina alla telecamera) e centrate in $P[p_x(t)\ p_y(t)\ p_z(t)]^T$, dove $P$ è la matrice di proiezione prospettica della particolare telecamera. Possiamo immaginare che questi cerchi compongano l'immagine fittizia. La funzione *im_overlap* ad essa sovrappone quella reale, cioè va a verificare se nell'area interna alle circonferenze si trova il centroide di qualche blob. Da questa verifica nasce una tabella, nella quale si registra per ogni blob il cerchio o i cerchi su cui giace e la sua distanza normalizzata dai loro centri. Se ad

esempio il centroide dell'$i$-esimo blob giace all'interno del $j$-esimo cerchio, viene registrato nella tabella il valore

$$w_{i,j} = \frac{d_{i,j}}{r_j} \tag{8}$$

Questa misura è proporzionale alla probabilità che l'$i$-esimo blob sia la proiezione sul piano immagine del $j$-esimo marker. Se invece il centroide dell'$i$-esimo blob non giace all'interno del $j$-esimo cerchio viene registrato il valore arbitrario $w_{i,j} = \infty$.

Successivamente, si normalizzano gli elementi di ogni riga, cioè

$$w_{i,j} \leftarrow \frac{w_{i,j}}{\sum_j w_{i,j}} \tag{9}$$

dato che

$$\sum_j P[\text{blob i-esimo} = \text{proiezione marker j}] = 1 \tag{10}$$

Leggendo opportunamente questa tabella si ottiene l'associazione blob-marker ricercata. La lettura viene compiuta per righe.

Se nella riga $i$ esiste un solo valore minore di infinito (cioè un $w_{i,j} = 1$), allora sicuramente l'$i$-esimo blob è la proiezione sul piano immagine del $j$-esimo marker e ad esso viene associato con probabilità 1 (ovvero con probabilità d'errore 0).

Se invece vi sono diversi valori $w_{i,j} < \infty$ (e ovviamente $\sum_j w_{i,j} = 1$), si ricerca fra essi quello minore $w_{i,j*}$. Si associa cioè l'$i$-esimo blob con il $j*$-esimo marker, cioè quello da cui dista meno, e la probabilità di aver preso una decisione corretta è $1 - w_{i,j*}$ (cioè con probabilità d'errore proprio $w_{i,j*}$).

Dopo ogni associazione, per evitare che uno stesso marker venga associato a più blob, l'intera colonna relativa al marker sceto viene rimossa dalla tabella. Può succedere che, dopo aver esaurito i blob, rimangano dei marker non associati. Fra questi, quelli nel cui cerchio d'incertezza non è stato rilevato alcun centroide di blob, con certezza sono soggetti a occlusioni da parte di oggetti della scena o da parte dell'attore stesso. Gli altri lo sono con alta probabilità.

Il risultato finale è quindi, per ciascuna telecamera, una tabella che dice quali marker sono visti e con che

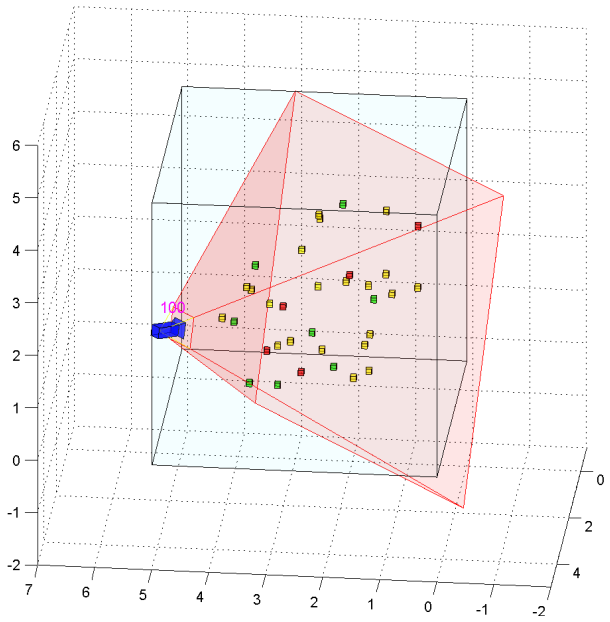


Figura 6: Insieme dei marker visti dalla telecamera: 1)in verde i marker visti con p=1 2)in giallo con p¡1 e 3)in rosso quelli occlusi

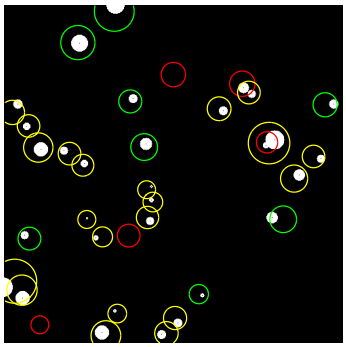

Figura 7: Rappresentazione nel piano immagine dei punti visti dalla telecamera presentata nella precedente figura

probabilità.
E' fondamentale per la risoluzione del problema poter disporre di una adeguata *funzione costo*, che esprima la bontà dell'associazione delle telecamere.

## 2.2 Funzione Costo

Il risultato dell'elaborazione precedente abbiamo visto essere, per ogni telecamera, una tabella che rappresenta la stima dei marker visti con la loro probabilità. Cioè, se $i$ è un marker giudicato visto dalla telecamera $j$, nella tabella è registrata la probabilità di vedere $i$ fissata la telecamera $j$ ($p(i,j)$). Per poter fare l'associazione delle telecamere dobbiamo confrontare fra loro tutte queste tabelle associando ad ogni confronto un punteggio, creando un ulteriore tabella. Quest'ultima avrà dimension $N$x$N$, con $N$ numero delle telecamere, e sarà simmetrica. Infatti

$$C(i,j) = C(j,i) = \text{score associazione telecamere i-j}$$

Questo è il compito della *funzione costo*. Sfruttando l'inteligenza delle smart-camera, anche il calcolo della matrice dei costi può essere fatto in maniera distribuita. Abbiamo elaborato vari tipi di *funzione costo*, che causano ovviamente diverse associazioni, accumunate ovviamente dal fatto di volere in ingresso due tabelle di due telecamere distinte.

La prima, confronta le tabelle e discrimina i marker visti da entrambe le telecamere da quelli non comuni. Fissate due telecamere $j$ e $k$, per ogni marker comune $i$, calcola il prodotto

$$p(i,j,k) = p(i,j)p(i,k) \tag{11}$$

$p(i,j)$ e $p(i,k)$ che dà la probabilità che il marker sia visto da entrambe le telecamere (abbiamo assunto che $p(i,j)$ e $p(i,k)$ siano indipendenti). Successivamente, somma tutti questi valori

$$w(j,k) = \sum_i p(i,j,k) \tag{12}$$

Ricordiamo che la ricostruzione 3D di un punto dello spazio a partire dalle sue proiezioni su due piani immagine distinti (triangolazione) è tanto più precisa quanto più i piani immagine sono ortogonali. La funzione ricava l'angolo fra le telecamere

$$\theta_{jk} = \arccos(u_{zj} \cdot u_{zk}) \tag{13}$$

con $u_{zj}$ ($u_{zk}$) versore dell'asse della telecamera $j$ ($k$). Per premiare l'ortogonalità, assegnamo

$$c(j,k) = |\sin(\theta_{jk})| w(j,k) = |\sin(\theta_{jk})| \sum_i p(i,j,k) \tag{14}$$

che, a parità di $w(j,k)$, è tanto maggiore quanto più le telecamere sono ortogonali (infatti $|\sin(\theta)| = 1$ se $\theta = \pi/2 + k\pi$, mentre $|\sin(\theta)| = 0$ se $\theta = k\pi$).

Questa funzione premia la quantità dei marker comuni: più alto è questo numero e più alta è la probabilita che $c(j,k)$ sia grande.

La seconda *funzione costo* agisce in maniera leggermente diversa. Essa infatti, memore del numero totale (sia esso N) di marker comuni, registra il valore normalizzato

$$c(j,k) = \frac{|\sin(\theta_{jk})| w(j,k)}{N} = \frac{|\sin(\theta_{jk})| \sum_i p(i,j,k)}{N} \tag{15}$$

Questa funzione premia la qualità dei marker comuni. Infatti due telecamere$j$ e $k$ con pochi marker comuni ma con $p(i,j,k)$ mediamente alta, ottengono uno score maggiore rispetto ad un altra coppia ($l$ e $m$) con molti marker in comune ma con $p(i,l,m)$ mediamente bassa.

Il secondo tipo di *funzione costo* garantisce quindi un rapporto *punti matchati* su raggi maggiore rispetto al primo, ma non necessariamente un numero effettivamente superiore di match. Sta all'utente scegliere il tipo di approccio a lui più conveniente.

# 3 Creazione dell'Albero delle Associazioni

Una volta ottenuta la matrice dei costi di associazione tramite l'utilizzo della funzione costo, si procede alla creazione dell'albero delle associazioni vero e proprio. Il criterio di base è quello di creare coppie di videocamere o di gruppi di videocamere

basandosi sul punteggio relativo assegnato dalla matrice costo ad ogni possibile associazione. La creazione dell'albero delle associazioni è suddivisa in 3 fasi fondamentali:

- Una prima funzione crea le associazioni di primo livello ovvero la selezione di coppie di videocamere
- Una seconda funzione associa i gruppi di videocamere per i livelli successivi al primo
- Infine tramite l'uso combinato delle prime due funzioni si crea l'albero completo.

## 3.1 Associazioni di primo livello

Al primo livello vengono create coppie di videocamere. La lettura di una riga della matrice dei costi fornisce la valutazione della potenziale bontà dell'associazione di una videocamera con ognuna delle altre

$$\mathcal{C} = \begin{pmatrix} 0 & c_{12} & \cdots & c_{1N} \\ c_{12} & 0 & \cdots & c_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ c_{1N} & c_{2N} & \cdots & 0 \end{pmatrix}$$

$c_{ij} \rightarrow$ *punteggio associazione tra le videocamere i,j*

L'algoritmo di associazione che si è sviluppato è dunque basato interamente sulla matrice dei costi ed è chiaro che esso cercherà di selezionare i punteggi più elevati compatibilmente con le scelte già effettuate. Sono stati sviluppati due approcci al problema dell'associazione ed a ciascuno di essi corrisponde una specifica funzione.

***association*** Il primo è di tipo *greedy*: la matrice costo viene letta per righe a partire dalla prima e ad ogni riga viene assegnata la colonna che garantisce il punteggio relativo più alto compatibilmente con le scelte già fatte alle righe precedenti. In pseudocodice:

scegli $c_{ij} \Leftrightarrow c_{ij} \geq c_{ik}$ ; $\forall k = (i+1)..., N$ ; $i, j \notin I$

dove $I$ è l'insieme degli indici già associati tra di loro.

La struttura del codice prevede due cicli innestati l'uno dentro l'altro, il più esterno deputato alla scansione delle righe e il più interno all'analisi delle colonne, di modo che avvienga una lettura per righe e che ad ogni riga venga assegnata, alla fine dell'iterazione, la colonna con punteggio più favorevole, compatibilmente con le associazioni già eseguite. Per tenere conto degli indici già assegnati (e che quindi non possono più essere utilizzati per un'associazione), l'algoritmo gestisce un *array* denominato *checked* nel quale vengono memorizzati i suddetti indici. Il contenuto del vettore verrà poi confrontato con gli indici presi in esame ad ogni iterazione dei cicli (operazione realizzata dalla funzione *confronto*) evitando l'accoppiamento di videocamere non più disponibili. Tale approccio conferisce una buona velocità di esecuzione (se la matrice ha dimensione $n \times n$, allora $\lfloor n/2 \rfloor$ righe vengono confrontate con al più con $n - i$ colonne se la riga in questione è la $i$–esima) ma non garantisce l'ottimalità delle associazioni, intesa come la scelta dei punteggi più alti possibile compatibilmente con le scelte già attuate dall'algoritmo. Questo perchè di fatto dopo le prime iterazioni alcuni indici sono stati esclusi per la possibili associazioni successive ed essi potrebbero portare ad accoppiamenti migliori di quanto poi in realtà l'algoritmo non sia costretto a generare.

***association1*** Un secondo approccio che tiene conto di questa considerazione è quello realizzato dalla funzione *association1* che contrariamente a quanto realizzato da *association* non crea l'accoppiamento ad ogni lettura di riga, ma esegue un una prima lettura completa di ogni riga della matrice costo individuando l'associazione con il punteggio più elevato per ciascuna di esse. Successivamente l'algoritmo individua il massimo punteggio assoluto, che appartiene naturalmente al gruppo dei punteggi massimi sopracitato, creando in questo modo l'associazione. L'algoritmo viene poi richiamato iterativamente per l'individuazione degli accoppiamenti successivi.

scegli $c_{ij} \Leftrightarrow c_{ij} \geq a_{hk}$ ; $\forall h, k = 1..., N$ ; $i, j \notin I$

L'ottimalità di questo procedimento secondo i criteri discussi in precedenza comporta, per contro, un costo computazionale più elevato (tutte le righe della matrice costo vengono scansionate interamente).

In *Matlab* l'algoritmo è stato implementato tramite l'utilizzo delle funzioni *highest*, che esamina la matrice costo per righe e individua l'accoppiamento di indici cui corrisponde il punteggio più elevato, e *association1*, che richiama ciclicamente *highest* escludendo ad ogni iterazione gli indici già accoppiati.

In entrambi i casi, approccio *greedy* e massimizzazione del costo, le funzioni producono in uscita una tabella, di tipo *array* di celle che memorizza le coppie di indici e i punteggi tramite i quali gli indici sono stati associati.

## 3.2 Associazione ai livelli successivi

Ai livelli successivi le coppie di telecamere formatesi dopo la prima associazione devono essere riaccoppiate per proseguire nella ricerca dei *markers* ancora *unmatched*. In tale contesto si pone un problema fondamentale: per le associazioni al primo livello il concetto intuitivo di buona associazione si traduceva in un punteggio numerico calcolabile a partire dai parametri fisici delle videocamere in questione. Ciò era reso possibile dal fatto che le entità da mettere in relazione erano oggetti fisici, la cui descrizione poteva essere ricondotta a parametri riscontrabili nel mondo reale. Nelle associazioni ai livelli successivi gli oggetti da mettere in relazione sono costituiti invece da coppie, e più in generale gruppi, di videocamere, entità totalmente astratte dellle quali risulta difficile, se non addirittura insensato, fornire una descrizione basata su di un parametro con una qualche attinenza col mondo reale. In particolare non risultava affatto chiaro come e se ricalcolare una qualche matrice di costo per associazioni di gruppi di videocamere. La strategia adottata per superare l'ostacolo è stata allora quella di continuare a fare riferimento alle singole videocamere, considerando stavolta la loro appartenenza ad un gruppo specifico.

***levels_association*** Come già anticipato, il criterio utilizzato per costruire la associazioni di livello successivo si rifà alle valutazioni sugli accoppiamenti tra singole telecamere, in particolare verrà riutilizzata la matrice dei costi di associazione, esattamente la stessa su cui si basavano le procedure di associazione al primo livello.

L'algoritmo procede come segue: in ingresso riceve, oltre alla già nominata matrice dei costi, una tabella in cui sono memorizzati gli indici delle associzaioni effettuate al livello precedente. Ciascun gruppo di videocamere viene considerato singolarmente, uno dopo l'altro, a partire dal primo (per questo motivo è possibile affermare che la soluzione adottata è in qualche modo riconducibile ad una approccio di tipo *greedy*)ed in particolare ogni elemento appartenente al gruppo in questione viene confrontato con tutti gli elementi dei gruppi successivi(ad esclusione di quelli appartenenti a gruppi già associati), e messo in relazione ad un secondo elemento, appartenente ad un altro gruppo, l'associazione col quale viene valutata con il punteggio migliore (per il singolo elemento del gruppo in esame) dalla matrice dei costi di associazione. In un secondo momento si effettua un confronto fra tutte le potenziali associazioni evidenziate al passo precedente, scegliendo quella cui corrisponde il punteggio di associazione più elevato. Se il costo prescelto è identificato all'interno della matrice di associazione dall'indice $a_{kh}$, allora il gruppo sotto esame (di cui fa parte l'indice $k$) viene associato al gruppo cui fa capo la telecamera indicizzata con $h$.

| **Gruppi** | **1** | $\cdots$ | **i** | $\cdots$ | **j** | $\cdots$ | **n** |
|---|---|---|---|---|---|---|---|
| ***Labels* di videocamere** | $\cdots$ | $\cdots$ | *k,y,z,..* | $\cdots$ | *f,g,h,..* | $\cdots$ | $\cdots$ |

$c_{kh} > c_{rs}$ ; $\forall r \in i, \forall s \notin i$ ; $j \notin C \Rightarrow$ associo $i$ e $j$

dove ancora si è indicato con $I$ l'insieme degli indici già associati.
Questo modo di operare è giustificato dal seguente ragionamento: la nozione di buona associazione, come in parte già messo in evidenza precedentemente,

viene espressa matmaticamente da un valore numerico calcolato a partire da delle caratteristiche, l'angolo tra due telecamere e il numero di *markers* comuni, in qualche modo riconducibili alla collocazione nello spazio delle videocamere stesse (se due videocamere inquadrano gli stessi *markers* allora inquadrano, nel medesimo istante, uno stessa porzione di spazio e ciò è dovuto in ultima analisi alla loro locazione spaziale ed alla loro orientazione). Pertanto è possibile sostenere una sorta di proprietà transitiva della buona associazione: in pratica se una videocamera ha un punteggio elevato rispetto all'associazione con una seconda videocamera, e quest'ultima ottiene un punteggio elevato rispetto all'associazione con una terza, allora con buona probabilità anche il costo relativo tra la prima e la terza sarà elevato.

$$c_{ij} \text{ alto } \wedge c_{jk} \text{ alto } \Rightarrow c_{ik} \text{ alto}$$

La funzione *levels_association* implementa il suddetto algoritmo richiamando a sua volta la funzione *element_scan*, la quale si occupa, una volta passatole come parametro d'ingresso il gruppo da esaminare (varaibile denominata *group*), di effettuare nel modo sopraindicato la singola associazione. *levels_association*, richiama iterativamente *element_scan*, organizzandone le uscite in una tabella con struttura *array* di celle, e gestendo inoltre il vettore degli indici (questa volta non più relativi al gruppo di appartenenza e non più corrispondenti a *labels* di telecamere) già associati che viene utilizzato ancora una volta dalla funzione *confronto* per evitare che nell'esecuzione di *element_scan* vengano create associazioni con gruppi già accoppiati in precedenza. L'uscita finale, come già preannunciato, è una tabella strutturata come*array* di celle, che tiene memoria non solo degli indici delle telecamere costituenti il nuovo gruppo, ma anche degli indici relativi ai gruppi che, dal livello precedente, sono stati associati per formare il nuovo gruppo in questione.
La tabella avrà cioè la seguente struttura:

| | **group 1** | ... | **group n** |
|---|---|---|---|
| **Indici relativi al livello precedente** | *i,j* | $\cdots$ | *h,k* |
| **Punteggio di associazione** | $\cdots$ | $\cdots$ | $\cdots$ |
| ***Labels* delle videocamere appartenenti al gruppo** | *r,s,* $\cdots$ *, u* | $\cdots$ | $\cdots$ |

## 3.3 Creazione dell'albero

L'algoritmo conclude creando l'albero binario completo delle associazioni. Per riuscire nell'intento esso ricorre all'uso combinato delle funzioni di associazione di primo livello (una delle due sviluppate) e di *levels_association*.

***association_tree*** La funzione *association_tree* assurge al ruolo di *main function* dell'intero algoritmo, dal momento che, ricevendo in ingresso la sola matrice dei costi di associazione, essa costruisce l'albero delle associazioni completo.
Innanzitutto la *routine* esegue, una sola volta, una funzione di associazione di primo livello (*association* o *association1*) producendo una prima tabella di accoppiamenti. Successivamente tale tabella viene utilizzata come ingresso per la funzione *levels_association* che da quel momento in poi verà richiamata in maniera ricorsiva, andando a creare in uscita delle tabelle di associazione con la struttura già descritta nella sezione precedente (ogni tabella prodotta costituisce un livello dell'albero).
Il risultato finale sarà un oggetto, strutturato di nuovo come *array di celle*, estremamente flessibile in quanto facilmente esplorabile per livelli e per singoli elementi (gruppi) di livello.
Tale oggetto potrà essere passato come parametro di ingresso ad un qualche algoritmo che lo utilizzi per ottimizzare il processo di elaborazione delle immagini provenienti dal set di telecamere al fine di ricostruire le posizioni dei *markers* per eseguirne, ad esempio, il *tracking*.

# 4 Associazione Statica

L'associazione statica differisce sostanzialmente da quella dinamica. Infatti non dipende dall'evoluzione della scena (e quindi non è tempo-variante), ma dipende solamente dalla particolare disposizione dell'impianto di acquisizione. Ovviamente, rispetto a quella dinamica, l'associazione statica comporta il vantaggio di azzerare i tempi di calcolo, potendo essere prodotta subito dopo la calibrazione delle telecamere e la partizione dello spazio, prima che gli attori entrino in gioco.

## 4.1 Funzione Costo e Associazione

Fondamentale, anche in questo caso, è disporre di una adeguata funzione costo. Essa è concettualmente simile al primo tipo di funzione costo dinamica. La differenza sta nel fatto che anzichè sommare le probabilità si sommano i voxel che i due FOV delle telecamere hanno in comune. Ovviamente anche in questo caso si tiene conto della mutua posizione fra le telecamere. Similmente a quanto fatto con la funzione costo dinamica, se $j$ e $k$ sono le due telecamere di cui si vuole quantificare la bontà di una loro associazione, l'algoritmo restituisce il valore

$$c(j,k) = |\sin(\theta_{jk})|N \tag{16}$$

con, al solito

$$\theta_{jk} = \arccos(u_{zj} \cdot u_{zk}) \tag{17}$$

dove $u_{zj}$ ($u_{zk}$) è il versore dell'asse della telecamera $j$ ($k$), e $N$ è il numero di voxel comuni. La mole dei dati da elaborare è molto superiore a quella dell'algoritmo per l'associazione dinamica (migliaia di voxel VS centinaia di marker), per questo l'algoritmo è decisamente più lento. Ma essendo l'elaborazione a priori, sottolineiamo ancora come questo fatto sia irrilevante.

La formazione dell'albero delle chiamate viene svolta dalle stesse funzioni che lo creano per l'associazione dinamica.

# 5 Apparato di Simulazione

Oltre alle funzioni che realizzano gli algoritmi di associazione, abbiamo dovuto costruire tutto l'apparato di simulazione. Ciò ha richiesto molto tempo, inevitabilmente sottraendone allo sviluppo degli algoritmi di associazione, in particolare all'esplorazione dei risultati che si avrebbero variandone i moltissimi parametri.

Probabilmente i risultati ottenuti sono fortemente condizionati dal fatto abbiamo lavorato su dati completamente sintetici. Il problema è che non sappiamo se questi dati sintetici sono o meno buone approssimazioni della realtà, non avendo nè avuto l'opportunità di vedere il campo nè un numero sufficiente di informazioni. Dunque

- abbiamo creato una funzione che genera casualmente le predizioni sulla posizione nello spazio dei marker. Queste rappresentano un ingresso (l'altro è l'immagine reale) per i nostri algoritmi di associazione. Successivamente la funzione crea dalle predizioni la posizione reale dei marker.
- abbiamo sfruttato, modificandola in parte per venire incontro alle nostre esigenze, la funzione *camera-simulation* del gruppo 2 (che si occupa del tracking), che dalle posizione nello spazio dei marker restituisce le loro coordinate sui piani immagine.
- abbiamo sfruttato, modificandolo in parte per venire incontro alle nostre esigenze, il simulatore di immagini del gruppo 1. Questo richiede in ingresso la posizione nel piano immagine dei marker e i parametri della telecamera restituendo un immagine in cui sono simulate anche le occlusioni.
- abbiamo creato una funzione che simula il match fra raggi di due piani immagine. Questa, date due telecamere associate, considera ricostruiti i loro marker comuni. Inoltre restituisce i *rate* raggi ricostruiti su raggi totali per ogni livello dell'albero.
- abbiamo creato una funzione che posiziona le telecamere nelle scena.

- abbiamo infine creato un'interfaccia grafica per facilitare all'utente le simulazioni. In questa interfaccia è possibile variare qualsiasi tipo di parametro: dal numero di telecamere a quello dei marker, dalla dimensione della scena al tipo di funzione costo da utilizzare. Si può anche vedere la scena o parte di essa, selezionando ad esempio le telecamere da visualizzare. E' uno strumento potente e estremamente flessibile, adatto a fare qualsiasi tipo di simulazione.

Le simulazioni sono state eseguite con i seguenti parametri:

- il numero di telecamere è variato da 20 a 100
- abbiamo imposto che il FOV delle telecamere formasse un angolo di $\pi/3$ rad e abbiamo posto una limitazione inferiore, 1 m, e una superiore, 5 m, alla sua profondità
- la scena è un parallelepipedo di 5x5x5 m
- le telecamere sono state poste sia sul soffitto che sui muri perimetrali della scena, automaticamente, dalla nostra funzione.
- abbiamo imposto che la deviazione standard della posizione dei marker reali rispetto alle sua posizione fosse $0.05I_2$. Non disponendo di dati reali è un valore che molto probabilmente non è veritiero.

# 6 Commento dei Risultati Sperimentali e Conclusioni

Per valutare la bontà del lavoro svolto sono state sviluppate delle simulazioni. Innanzitutto vanno messi ben in evidenza i criteri su cui si basa la valutazione. Il tempo di esecuzione dell'algoritmo non rappresenta un parametro di interesse; infatti è numericamente poco significativo, dato che nelle applicazioni reali i calcoli vengono eseguiti in parallelo, con elaborazione multiprocessore, e quindi i dati che si fornirebbero in questa sede, ottenuti su *personal computer*, avrebbero scarso valore. Gran parte dell'algoritmo sviluppato poi, costruisce dati che nel suo uso integrato, in una applicazione di *motion capture*, verrebbero normalmente forniti come ingresso o comunque elaborati in una diversa regione di competenza del sistema (ad esempio la generazione dell'immagine reale da confrontare con quella fittizia) e la cui produzione non dovrebbe pertanto essere considerata nel tempo di esecuzione.

Ciò che realmente conta nel valutare le prestazioni dell'algoritmo è quanto bene esso realizzi l'associazione delle videocamere, ovvero quanto il procedere nell'elaborazione delle immagini delle varie telecamere secondo l'ordine indicato dalla gerarchia di accoppiamento rappresentata dall'albero di associazione porti a completare la ricostruzione della scena in pochi passaggi, oppure ad un alto numero di raggi *matchati* per livello. Per dare una misura dell'efficienza dell'algoritmo ci si è basati su di un parametro costruito come rapporto tra il numero di *markers* ricostruiti ad un dato livello e il numero di *markers* ancora *unmatched* al livello precedente. In pratica si valuta quale apporto abbiano fornito le associazioni ad un dato livello nel completare la ricostruzione del *set* di *markers*. Non potendosi basare su prove reali, si è scelto di considerare ricostruito un *marker* in due situazioni diverse: quando esso è inquadrato da due telecamere distinte, e, più realisticamente, quando è visto da almeno 3 videocamere.

Analizzando l'andamento del *Rate* nelle simulazioni riportate si nota innanzitutto che le associazioni che si basano sulla lettura *greedy* della matrice dei costi portano a risultati dello stesso ordine di grandezza, e spesso migliori, rispetto alle altre, e sono perciò preferibili. Inoltre, facendo riferimento alla seconda assunzione sulla ricostruzione (ovvero quella per cui c'è *matching* di un *marker* quando esso è visto da tre videocamere), come in tutti i casi (approccio *greedy* e non, associazione statica e dinamica) si nota che le percentuali di ricostruzione al secondo livello siano molto elevate (ovviamente al primo livello non può esserci alcun match). Con questo tipo di nozione di *matching*, gli algoritmi risultano in grado di coprire l'operazione di ricostruzone per gran parte già al primo livello, fase in cui il raffronto delle immagini risulta poco oneroso, dato che avviene al massimo tra due videocamere alla volta.

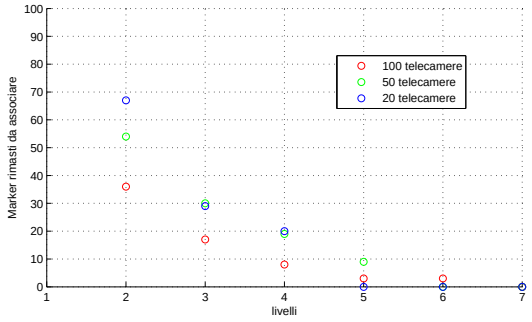


Figura 8: Andamento dei markers rimasti da associare dopo ogni livello tenendo fisso il numero di marker nella scena (qui 300). Minore è il numero di telecamere, minore è il numero dei livelli dell'albero di associazione

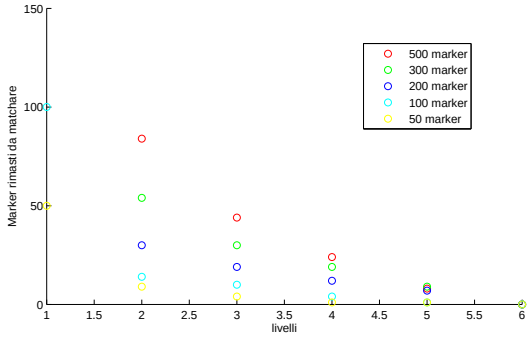


Figura 9: Andamento dei markers rimasti da associare dopo ogni livello tenendo fisso il numero di telecamere nella scena (qui 50)

L'algoritmo che in genere dà risultati migliori è quello di associazione dinamica con funzione di costo che premia la quantità dei marker comuni, anche se la sua efficienza non è molto maggiore rispetto a quella degli altri due (quello statico e quello dinamico con funzione di costo che premia la qualità dei marker comuni). In virtù di questa osservazione, si può affermare che l'algoritmo migliore risulta essere quello di associazione statica, dato che fornisce prestazioni simili a quelli di associazione dinamica ma praticamente azzera i tempi di calcolo.

Si può notare poi come in alcuni casi, nei livelli superiori al primo, il *rate* possa essere nullo. Ciò avviene nel caso, tutt'altro che desiderabile, in cui l'associazione di due gruppi di videocamere non porta contributo alla ricostruzione della scena. Un simile risultato può apparire quantomeno insolito, dal momento che l'avvenuta associazione è diretta conseguenza dell'esistenza di un punteggio relativo tra i due gruppi (ricavato dalla matrice dei costi) quntomeno positivo, e che quindi un certo numero di videocamere appartenenti a due gruppi distinti inquadrano dei *markers* comuni. In realtà questo tipo di esiti possono essere spiegati facendo riferimento ancora una volta alla nozione di buona associazione. In effetti se il punteggio relativo di associazione elevato è conseguenza del fatto che le telecamere in questione inquadrano una porzione comune di spazio visivo, è altamente probabile che dopo alcuni livelli, le associazioni abbiano portato al formarsi di gruppi di videocamere quasi disgiunti, volendo con questo termine indicare gruppi la cui associazione non porti ulteriore informazione al fine della ricostruzione. Ciò, nella pratica, si verifica quando due gruppi vengono associati sulla base di un punteggio relativo tra due videocamere calcolato con riferimento ad un gruppo di *markers* il cui *matching* in realtà è già stato interamente realizzato ad un livello precedente, rendendo di fatto nullo il contributo dell'associazione alla ricostruzione della scena.

Infine qualche considerazione sull'andamento dei *match* in rapporto al numero di marker o di telecamere nella scena. I grafici sono stati ricavati dalle simulazioni con associazione greedy e usando la funzione costo che premia la quantità dei marker comuni (cioè quella che dà migliori risultati). In primo luogo, è stato tenuto fisso il numero dei markers e variato il numero di telecamere nella scena. Si nota ovviamente che maggiore è il numero di telecamere, maggiore è il numero di match che avviene in ogni livello (e quindi minore è il numero dei marker la cui posizione non è stata ricostruita). Conseguentemente, minore è il numero di telecamere e maggiore sarà il peso computazione negli ultimi livelli.

In secondo luogo, è stato tenuto fisso il numero di telecamere e variato il numero di marker nella scena. Si nota che il rapporto fra i marker rimasti in due

livelli contigui (percentuale di diminuzione dei marker rimasti da associare dopo ogni livello) è circa costante per ogni numero di marker nella scena.

# 7 Sviluppi Futuri

Ci siamo resi conto, con lo svolgimento del progetto, che l'algoritmo di associazione dinamica può risultare pesante, e, vista l'elevata frequenza di acquisizione delle immagini (tipicamente 120 HZ in Vicon), il suo tempo di svolgimento può superare senza dubbio il tempo *intra-frame*. D'altronde si nota immediatamente che in questo intervallo temporale la posizione dei marker non può variare di molto, dato che sono attaccati ad un operatore umano. Per questo, un'attento studio della dinamica degli attori nella scena può portare allo sviluppo di un algoritmo di associazione ibrido. Questo non calcola ex-novo dopo ogni acquisizione l'associazione (120 volte al secondo), ma solamente dopo un $\Delta t$ appropriato, legato alla velocità di variabilità dei marker nella scena. L'algoritmo mantiene cioè la stessa associazione dinamica per un intervallo della durata $\Delta t$.

Particolare attenzione potrebbe essere posta inoltre su di un raffinamento nel calcolo della matrice dei costi e sulle tecniche di associazione: nello specifico un possibile sviluppo futuro potrebbe constare nel portare memoria dei *matching* già effettuati al momento delle associazioni successive, risolvendo in tal modo il problema esposto nel caso di *rate* nulli.

# Appendice

# A Formazione delle Immagine

## A.1 Immagini Digitali

Un sistema di acquisizione di immagini digitali consiste di tre componenti fondamentali: una telecamera, un frame grabber ed un calcolatore host. La telecamera è composta dall'ottica, che approssimiamo con una lente sottile, e dal sensore, una matrice di CCD (Charged Copuled Device) che costituisce il piano immagine (un CCD misura circa 1x1 cm e contiene circa $10^6$ elementi). Possiamo vedere quest'ultima come una matrice $n$x$m$ di celle rettangolari fotosensibili, ciascuna dei quali converte l'intensità della radiazione luminosa incidente in un potenziale elettrico. L'uscita della telecamera a CCD è un segnale elettrico analogico, ottenuto leggendo il potenziale degli elementi della matrice CCD per righe. Il segnale video (analogico) viene letto dal frame grabber (la scheda di acquisizione) che lo digitalizza, convertendolo in una matrice $N$x$M$ di valori interi memorizzati in un'opportuna area di memoria (frame buffer). Gli elementi della matrice prendono il nome di pixel (picture element). La dimensione $n$x$m$ della matrice CCD non è necessariamente la stessa della immagine $N$x$M$ (matrice dei pixel); per questo motivo la posizione di un punto del piano immagine è diversa se misurata in elementi CCD piuttosto che in pixel. Si indica con I(u; v) il valore dell'immagine (luminosita) nel pixel individuato dalla riga v e colonna u (sistema di coordinate u; v con l'origine nell'angolo in alto a sinistra). Il calcolatore host acquisisce l'immagine trasferendola dal frame buffer alla memoria centrale. La frequenza temporale che si raggiunge normalmente è attorno ai 25 Hz, ovvero un frame ogni 40 ms. Dunque una telecamera trasferisce circa 12 Mb/s.

## A.2 Pinhole Model

Un apparato di acquisizione di immagini funziona raccogliendo la luce riflessa dagli oggetti della scena e creando una immagine bidimensionale. Il modello geometrico più semplice della formazione dell'immagine è la pinhole camera. Sia $P = [X, Y, Z]$ un punto

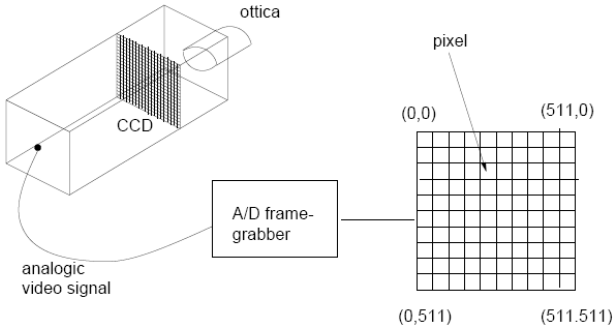


Figura 10: sistema di acquisizione di immagini digital

della scena espresso in coordinate relative al sistema di riferimento centrato sul centro ottico della telecamera e con l'asse $Z$ coincidente con il suo asse ottico (camera frame). Allora è immediato verificare che, per la similarità dei tringoli, le coordinate di $P$ sul piano immagine (o piano retina) sono date dalle

$$x = -\frac{fX}{Z},\ y = -\frac{fY}{Z} \tag{18}$$

dove $f$ è la lunghezza focale (focal length). SI nota immediatamente che l'immagine risulta essere rovesciata. Per eliminare questo inconveniente solitamente si pone il piano immagine davanti al centro ottico, sempre ad una distanza da esso di $f$. In tal caso, le coordinate di $P$ sul piano immagine diventano

$$x = \frac{fX}{Z},\ y = \frac{fY}{Z} \tag{19}$$

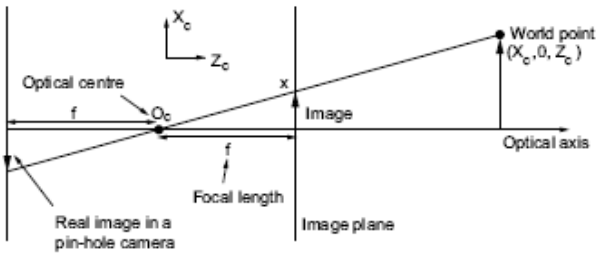


Figura 11: rappresentazione schematica di una pinhole camera

Introducendo le coordinate omogenee, esse diventano

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{20}$$

In realtà, i punti della scena sono rappresentati con coordinate relative al sistema di riferimento proprio della scena (world frame), e non al camera frame. Ma ad esso si possono ricondurre molto semplicemente attraverso la

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} & R_A^C & & T_A^C \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \\ 1 \end{bmatrix} \tag{21}$$

Di solito, le coordinate sul piano immagine sono adimensionali (misurate in pixel), e si ottengono attraverso la

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}, \tag{22}$$

$$S_x = \frac{1}{dim_x\ del\ pixel}\ , S_y = \frac{1}{dim_y\ del\ pixel} \tag{23}$$

$S_x$ e $S_y$ sono uguali nel caso di pixel quadrati. Convenzionalmente, le coordinate sul piano immagine sono espresse in un sistema di riferimento, con coordinate $(i, j)$ e con l'origine posta sull'angolo in alto a sinistra dell'immagine, diverso dal precedente, con coordinate $(u, v)$. I due frame sono però legati da una rototraslazione

$$\begin{bmatrix} i \\ j \\ 1 \end{bmatrix} = \begin{bmatrix} R_z(\theta) & & O_u \\ & & O_v \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{24}$$

Mettendo insieme le equazioni precedenti si ottiene finalmente la

$$\lambda \begin{bmatrix} i \\ j \\ 1 \end{bmatrix} = \begin{bmatrix} R_z(\theta) & & O_u \\ & & O_v \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} * \tag{25}$$

$$* \begin{bmatrix} & R_A^C & & T_A^C \\ & & & \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \\ 1 \end{bmatrix}$$

Se, come accade di solito, $(i, j)$ e $(u, v)$ sono orientati nello stesso modo,essa diviene

$$\lambda \begin{bmatrix} i \\ j \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_u & 0 & O_u \\ 0 & \alpha_v & O_v \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R_A^C & & T_A^C \\ & & \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \\ 1 \end{bmatrix} =$$

$$K \; [R_A^C \; T_A^C] \; \begin{bmatrix} P_0 \\ 1 \end{bmatrix} \tag{26}$$

dove $K$ è detta matrice dei parametri intrinseci, mentre $[R_A^C \; T_A^C]$ matrice dei parametri estrinseci della telecamera.

# B Risultati Sperimentali

Riportiamo qui tutti i risultati sperimentali ottenuti. Nelle seguenti tabelle ogni colonna riporta gli *association rate* (punti matchati su raggi in ingresso) per una particolare simulazione. Le simulazioni sono state eseguite variando il numero di telecamere, il tipo di algoritmo di associazione (greedy o non greedy, statico), le funzioni costo (quella statica e le due dinamiche), e giudicando che un marker fosse ricostruito solo dopo l'intersezione di due (2r nelle didascalie) o tre raggi (3r nelle didascalie).

| | costo1 e ass. din. greedy | costo2 e ass. din. greedy | ass. stat. greedy | costo1 e ass. din. nn greedy | costo2 e ass. din. nn greedy | ass. stat. nn greedy |
|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0.8180 | 0.8900 | 0.8800 | 0.7840 | 0.8520 | 0.8520 |
| 3 | 0.5934 | 0.5273 | 0.6167 | 0.6759 | 0.5541 | 0.7162 |
| 4 | 0.5946 | 0.6154 | 0.4348 | 0.5143 | 0.4545 | 0.2857 |
| 5 | 0.5333 | 0.7000 | 0.5833 | 0.4118 | 0.5000 | 0.4667 |
| 6 | 0.4286 | 0.6667 | 0.6000 | 0.2000 | 0.1111 | 0.1250 |

Figura 1: Simulazione con 500 marker, 100 telecamere, 3r

| | costo1 e ass. din. greedy | costo2 e ass. din. greedy | ass. stat. greedy | costo1 e ass. din. nn greedy | costo2 e ass. din. nn greedy | ass. stat. nn greedy |
|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0.7525 | 0.8249 | 0.8209 | 0.7500 | 0.8000 | 0.8260 |
| 3 | 0.5203 | 0.5632 | 0.4944 | 0.6320 | 0.4400 | 0.5747 |
| 4 | 0.4407 | 0.5143 | 0.5111 | 0.5122 | 0.5098 | 0.3784 |
| 5 | 0.4848 | 0.2353 | 0.3182 | 0.5000 | 0.4583 | 0.4091 |

Figura 2: Simulazione con 500 marker, 50 telecamere, 3r

| | costo1 e ass. din. greedy | costo2 e ass. din. greedy | ass. stat. greedy | costo1 e ass. din. nn greedy | costo2 e ass. din. nn greedy | ass. stat. nn greedy |
|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0.6436 | 0.7536 | 0.6843 | 0.6337 | 0.7032 | 0.7137 |
| 3 | 0.5886 | 0.5124 | 0.6194 | 0.4598 | 0.4468 | 0.5074 |
| 4 | 0.5455 | 0.4746 | 0.3898 | 0.3511 | 0.3733 | 0.3333 |

Figura 3: Simulazione con 500 marker, 20 telecamere, 3r

| | costo1 e ass. din. greedy | costo2 e ass. din. greedy | ass. stat. greedy | costo1 e ass. din. nn greedy | costo2 e ass. din. nn greedy | ass. stat. nn greedy |
|---|---|---|---|---|---|---|
| 1 | 0.8600 | 0.9140 | 0.9120 | 0.8460 | 0.9120 | 0.8740 |
| 2 | 0.6857 | 0.6977 | 0.7045 | 0.5195 | 0.4545 | 0.5079 |
| 3 | 0.1364 | 0.1538 | 0.0769 | 0.4865 | 0.2083 | 0.4194 |
| 4 | 0.2632 | 0.5455 | 0.0833 | 0.2105 | 0.1053 | 0.3333 |
| 5 | 0.3571 | 0.4000 | 0.4000 | 0.2000 | 0.5294 | 0 |
| 6 | 0.3333 | 0.3333 | 0.3333 | 0.4167 | 0.1250 | 0.5000 |

Figura 4: Simulazione con 500 marker, 100 telecamere, 2r

| | costo1 e ass. din. greedy | costo2 e ass. din. greedy | ass. stat. greedy | costo1 e ass. din. nn greedy | costo2 e ass. din. nn greedy | ass. stat. nn greedy |
|---|---|---|---|---|---|---|
| 1 | 0.7807 | 0.8833 | 0.8913 | 0.8080 | 0.8620 | 0.8500 |
| 2 | 0.6514 | 0.4138 | 0.3704 | 0.5521 | 0.4493 | 0.6667 |
| 3 | 0.1316 | 0.3529 | 0.2353 | 0.4419 | 0.1842 | 0.2000 |
| 4 | 0.4242 | 0.0714 | 0.1154 | 0.3889 | 0.2400 | 0 |
| 5 | 0.3158 | 0.9231 | 0.3333 | 0.2727 | 0.2222 | 0.6316 |

Figura 5: Simulazione con 500 marker, 50 telecamere, 2r

| | costo1 e ass. din. greedy | costo2 e ass. din. greedy | ass. stat. greedy | costo1 e ass. din. nn greedy | costo2 e ass. din. nn greedy | ass. stat. nn greedy |
|---|---|---|---|---|---|---|
| 1 | 0.6986 | 0.8432 | 0.7454 | 0.6779 | 0.7789 | 0.8084 |
| 2 | 0.6486 | 0.3636 | 0.6480 | 0.5752 | 0.5143 | 0.3626 |
| 3 | 0.3654 | 0.4082 | 0.3409 | 0.3385 | 0.0588 | 0.2241 |
| 4 | 0.4643 | 0.3103 | 0.3103 | 0.1395 | 0.3111 | 0.1136 |

Figura 6: Simulazione con 500 marker, 20 telecamere, 2r

| | costo1 e ass. din. greedy | costo2 e ass. din. greedy | ass. stat. greedy | costo1 e ass. din. nn greedy | costo2 e ass. din. nn greedy | ass. stat. nn greedy |
|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0.8350 | 0.8653 | 0.8721 | 0.8384 | 0.8653 | 0.8687 |
| 3 | 0.4898 | 0.5250 | 0.3947 | 0.5417 | 0.4500 | 0.5128 |
| 4 | 0.3200 | 0.4211 | 0.3913 | 0.3182 | 0.4545 | 0.5789 |
| 5 | 0.4118 | 0.3636 | 0.1818 | 0.2667 | 0.4167 | 0.5000 |
| 6 | 0.7000 | 0.2857 | 0.6250 | 0.3636 | 0.5714 | 0.5000 |

Figura 7: Simulazione con 300 marker, 100 telecamere, 3r

| | costo1 e ass. din. greedy | costo2 e ass. din. greedy | ass. stat. greedy | costo1 e ass. din. nn greedy | costo2 e ass. din. nn greedy | ass. stat. nn greedy |
|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0.7216 | 0.8488 | 0.8351 | 0.7973 | 0.8557 | 0.8419 |
| 3 | 0.5926 | 0.5000 | 0.3958 | 0.3729 | 0.5238 | 0.4130 |
| 4 | 0.3333 | 0.4091 | 0.4483 | 0.5143 | 0.4000 | 0.2692 |
| 5 | 0.5000 | 0.4615 | 0.6250 | 0.4118 | 0.2500 | 0.4211 |

Figura 8: Simulazione con 300 marker, 50 telecamere, 3r

| | costo1 e ass. din. greedy | costo2 e ass. din. greedy | ass. stat. greedy | costo1 e ass. din. nn greedy | costo2 e ass. din. nn greedy | ass. stat. nn greedy |
|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0.5830 | 0.7173 | 0.6926 | 0.6725 | 0.7500 | 0.7641 |
| 3 | 0.3814 | 0.4250 | 0.3563 | 0.4946 | 0.5352 | 0.4328 |
| 4 | 0.4722 | 0.2727 | 0.3750 | 0.4412 | 0.2121 | 0.2400 |

Figura 9: Simulazione con 300 marker, 20 telecamere, 3r

| | costo1 e ass. din. greedy | costo2 e ass. din. greedy | ass. stat. greedy | costo1 e ass. din. nn greedy | costo2 e ass. din. nn greedy | ass. stat. nn greedy |
|---|---|---|---|---|---|---|
| 1 | 0.8721 | 0.8889 | 0.8956 | 0.8721 | 0.9057 | 0.8956 |
| 2 | 0.3158 | 0.3636 | 0.2903 | 0.4474 | 0.4286 | 0.3226 |
| 3 | 0.3077 | 0.4286 | 0.3182 | 0.2857 | 0.0625 | 0.2381 |
| 4 | 0.2222 | 0.4167 | 0.2667 | 0.3333 | 0.4000 | 0.3750 |
| 5 | 0.5000 | 0 | 0.4286 | 0.1000 | 0.3333 | 0.3000 |
| 6 | 0.2857 | 0.1429 | 0.6667 | 0.3333 | 0 | 0.1429 |

Figura 10: Simulazione con 300 marker, 100 telecamere, 2r

| | costo1 e ass. din. greedy | costo2 e ass. din. greedy | ass. stat. greedy | costo1 e ass. din. nn greedy | costo2 e ass. din. nn greedy | ass. stat. nn greedy |
|---|---|---|---|---|---|---|
| 1 | 0.8076 | 0.8763 | 0.8694 | 0.8660 | 0.9038 | 0.8866 |
| 2 | 0.3393 | 0.3056 | 0.2895 | 0.4615 | 0.2857 | 0.3333 |
| 3 | 0.2703 | 0.4000 | 0.1481 | 0.2857 | 0.4000 | 0.1818 |
| 4 | 0.5556 | 0.1333 | 0.4348 | 0.1538 | 0 | 0.2353 |
| 5 | 0.2500 | 0.2308 | 0.5385 | 0.3636 | 0.3333 | 0.1538 |

Figura 11: Simulazione con 300 marker, 50 telecamere, 2r

| | costo1 e ass. din. greedy | costo2 e ass. din. greedy | ass. stat. greedy | costo1 e ass. din. nn greedy | costo2 e ass. din. nn greedy | ass. stat. nn greedy |
|---|---|---|---|---|---|---|
| 1 | 0.7244 | 0.7739 | 0.8057 | 0.8134 | 0.8415 | 0.8063 |
| 2 | 0.4615 | 0.3750 | 0.2000 | 0.2264 | 0.2222 | 0.3455 |
| 3 | 0.3095 | 0.1250 | 0.2045 | 0.1951 | 0.2286 | 0.1667 |
| 4 | 0.1786 | 0.3750 | 0.2857 | 0.2857 | 0.2593 | 0.3333 |

Figura 12: Simulazione con 300 marker, 20 telecamere, 2r

| | costo1 e ass. din. greedy | costo2 e ass. din. greedy | ass. stat. greedy | costo1 e ass. din. nn greedy | costo2 e ass. din. nn greedy | ass. stat. nn greedy |
|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0.7900 | 0.8900 | 0.8750 | 0.8300 | 0.8900 | 0.8350 |
| 3 | 0.6190 | 0.5455 | 0.4800 | 0.6471 | 0.5000 | 0.6061 |
| 4 | 0.3750 | 0.6000 | 0.5385 | 0.5833 | 0.7273 | 0.3846 |
| 5 | 0.6000 | 0.5000 | 0.3333 | 0.6000 | 0.3333 | 0.3750 |
| 6 | 0.7500 | 0.5000 | 0.7500 | 1 | 0.5000 | 0.6000 |

Figura 13: Simulazione con 200 marker, 100 telecamere, 3r

| | costo1 e ass. din. greedy | costo2 e ass. din. greedy | ass. stat. greedy | costo1 e ass. din. nn greedy | costo2 e ass. din. nn greedy | ass. stat. nn greedy |
|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0.7938 | 0.8093 | 0.8402 | 0.7665 | 0.8376 | 0.7766 |
| 3 | 0.5500 | 0.4324 | 0.4194 | 0.4889 | 0.3750 | 0.5000 |
| 4 | 0.5556 | 0.3500 | 0.3529 | 0.4783 | 0.4500 | 0.4545 |
| 5 | 0.5000 | 0.6154 | 0.7273 | 0.4167 | 0.4545 | 0.4167 |

Figura 14: Simulazione con 200 marker, 50 telecamere, 3r

| | costo1 e ass. din. greedy | costo2 e ass. din. greedy | ass. stat. greedy | costo1 e ass. din. nn greedy | costo2 e ass. din. nn greedy | ass. stat. nn greedy |
|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0.7128 | 0.7181 | 0.6968 | 0.7128 | 0.7333 | 0.7436 |
| 3 | 0.4630 | 0.4340 | 0.4912 | 0.4464 | 0.3542 | 0.3800 |
| 4 | 0.3448 | 0.5172 | 0.4828 | 0.4667 | NaN | 0.4667 |

Figura 15: Simulazione con 200 marker, 20 telecamere, 3r

| | costo1 e ass. din. greedy | costo2 e ass. din. greedy | ass. stat. greedy | costo1 e ass. din. nn greedy | costo2 e ass. din. nn greedy | ass. stat. nn greedy |
|---|---|---|---|---|---|---|
| 1 | 0.8650 | 0.9100 | 0.9100 | 0.8450 | 0.9000 | 0.8850 |
| 2 | 0.3704 | 0.3889 | 0.3333 | 0.5484 | 0.5500 | 0.4348 |
| 3 | 0.3529 | 0.4545 | 0.2500 | 0.3571 | 0.2222 | 0.5385 |
| 4 | 0.3636 | 0.6667 | 0.4444 | 0.2222 | 0.5714 | 0.1667 |
| 5 | 0.5714 | 0 | 0.4000 | 0.1429 | 0.3333 | 0.2000 |
| 6 | 0 | 0.5000 | 0.3333 | 0.5000 | 0.5000 | 0.5000 |

Figura 16: Simulazione con 200 marker, 100 telecamere, 2r

| | costo1 e ass. din. greedy | costo2 e ass. din. greedy | ass. stat. greedy | costo1 e ass. din. nn greedy | costo2 e ass. din. nn greedy | ass. stat. nn greedy |
|---|---|---|---|---|---|---|
| 1 | 0.8144 | 0.8454 | 0.8763 | 0.7614 | 0.8579 | 0.8274 |
| 2 | 0.5278 | 0.4000 | 0.3333 | 0.4468 | 0.3571 | 0.5294 |
| 3 | 0.1765 | 0.0556 | 0.2500 | 0.2400 | 0.2222 | 0.1250 |
| 4 | 0.2143 | 0.2500 | 0.1000 | 0.1579 | 0.1429 | 0.2857 |
| 5 | 0.6364 | 0.3333 | 0.4444 | 0.5000 | 0.7273 | 0.3000 |

Figura 17: Simulazione con 200 marker, 50 telecamere, 2r

| | costo1 e ass. din. greedy | costo2 e ass. din. greedy | ass. stat. greedy | costo1 e ass. din. nn greedy | costo2 e ass. din. nn greedy | ass. stat. nn greedy |
|---|---|---|---|---|---|---|
| 1 | 0.7074 | 0.7447 | 0.7660 | 0.6769 | 0.8513 | 0.8051 |
| 2 | 0.5818 | 0.2708 | 0.3182 | 0.6667 | 0.2759 | 0.3947 |
| 3 | 0.1739 | 0.6286 | 0.3667 | 0.1905 | 0.1579 | 0.3043 |
| 4 | 0.1579 | 0.1667 | 0.5263 | 0.1875 | NaN | 0.2667 |

Figura 18: Simulazione con 200 marker, 20 telecamere, 2r

## Riferimenti bibliografici


[1] Fusiello A., 2006, *Visione Computazionale: appunti delle lezioni.*

[2] Trucco E., Verri A., 1998, *Introductory Techniques for 3-D Computer Vision*, Prentice-Hall.

[3] V Rama Aravind , 2004, *VICON Volume Visualization*

[4] url:http://www.c3d.org/