Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Ricostruzione tridimensionale di siti architettonici

Ignazio Infantino

RT-ICAR-PA-03-06 **dicembre 2003**

Consiglio Nazionale delle Ricerche, Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR) –
Sede di Cosenza, Via P. Bucci 41C, 87036 Rende, Italy, URL: *www.icar.cnr.it*
– Sezione di Napoli, Via P. Castellino 111, 80131 Napoli, URL: *www.na.icar.cnr.it*
– Sezione di Palermo, Viale delle Scienze, 90128 Palermo, URL: *www.pa.icar.cnr.it*

Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Ricostruzione tridimensionale di siti architettonici

Ignazio Infantino[1]

Rapporto Tecnico N.6: **RT-ICAR-PA-03-06**

Data: **dicembre 2003**

---

[1] Istituto di Calcolo e Reti ad Alte Prestazioni, ICAR-CNR, Sezione di Palermo

Indice

Parte III
Implementazione in Matlab

3. Implementazione

Introduzione

Descrizione e finalità della ricerca

Nel corso dell'ultimo decennio la branca di ricerca relativa alla "computer vision" ha sviluppato metodologie robuste ed efficienti nel campo dell'estrazione dell'informazione tridimensionale a partire da una o più immagini di una scena reale. L'esperienza di ricerca ha avuto il fine di sviluppare applicazioni pratiche di ricostruzione tridimensionale automatica di siti architettonici e di oggetti tridimensionali in generale.
Partendo da una serie di immagini di una stessa scena reale, acquisite da telecamera o fotocamera non calibrate, si applicano algoritmi di autocalibrazione e ricerca dei punti all'infinito che caratterizzano le proiezioni prospettiche della scena. In maniera robusta, anche da poche immagini, è possibili risalire così ai parametri intrinseci di ciascuno dei dispositivi utilizzati per l'acquisizione ed avere delle prime stime delle loro posizioni relativamente alla scena ripresa. Utilizzando algoritmi di estrazione di punti di interesse sulle singole immagini (corner detector, line detector, etc.) , si cercano in maniera automatica corrispondenze tra le diverse immagini al fine di determinare e caratterizzare la geometria epipolare che lega. Si ottengono così delle prime stime delle matrici di proiezione di ogni vista consentendo, tramite la tecnica della triangolazione, di risalire all'informazione tridimensionale.
La robustezza del metodo è ottenuta applicando inoltre tecniche di ottimizzazione che in maniera iterativa permettono di minimizzare gli errori di ricostruzione e di trovare ulteriori corrispondenze tra le immagini sfruttando ad esempio la vincoli di parallelismo, perpendicolarità e planarità (vincolo omografico) tipici di scene architettoniche. Alle informazioni tridimensionali ottenute dalle immagini della scena si possono aggiungere ulteriori dettagli utilizzando vincoli ricavati da piante o introdotti dall'utente del software di ricostruzione.
Infine, estraendo la tessitura dalle immagini a disposizione e traducendo gli elementi geometrici ricostruiti in formato VRML, è possibile visualizzare ed esplorare la scena e gli oggetti ricostruiti tramite browser munito di apposito plug-in (facilmente reperibili ed installabili).

Risultati ottenuti

Fornire ricostruzioni tridimensionali di oggetti di interesse storico, di interi siti archeologici, di ambienti museali, etc, e rendere disponibili i risultati di tali elaborazioni anche attraverso il Web.

Parte I

Visione Artificiale

Geometria proiettiva.

1.1 Modello della telecamera.

Un'immagine rappresenta la mappatura tra punti tridimensionali e punti su un piano bidimensionale attraverso una proiezione centrale. Il dispositivo di acquisizione (telecamera, macchina fotografica o altro) può essere modellizzato in maniera semplice utilizzando la rappresentazione che prevede che tutti i raggi luminosi determinanti la formazione dei punti sull'immagine passino tutti per un unico punto per poi intersecare il piano dell'immagine (modello "pinhole"). Tale intersezione rappresenta per l'appunto la proiezione centrale del punto tridimensionale. Utilizzando le coordinate omogenee possiamo esprimere il processo proiettivo tramite la seguente relazione:

$$\lambda \hat{w} = K [R | t] \hat{X} \qquad (1.1.1)$$

Dove $\hat{w}$ è il vettore contenente le coordinate omogenee del pixel $[u\ v\ 1]^T$, $\hat{X}$ è il vettore 4x1 con le coordinate omogenee del punto tridimensionale $[X\ Y\ Z\ 1]^T$. R è la matrice 3x3 che individua l'orientazione della telecamera, mentre t il vettore 3x1 la sua traslazione. Infine λ è uno scalare e K la matrice 3x3 dei parametri intrinseci della telecamera (matrice di calibrazione) che ha la seguente forma.

$$K = \begin{bmatrix} \alpha_x & s & u_0 \\ 0 & \alpha_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \qquad (1.1.2)$$

Il modello pinhole assume che le coordinate dell'immagine siano coordinate euclidiane aventi scale uguali in entrambi le direzioni degli assi. Nel caso di una comune telecamera con sensori CCD, i fattori di scala (eventualmente differenti) sono introdotti attraverso i coefficienti m_x ed m_y, espressi in pixel per unità di distanza. Se f è la lunghezza focale in metri, allora $\alpha_x = f\ m_x$ ed $\alpha_y = f\ m_y$ rappresentano la lunghezza focale in pixel nelle direzioni x ed y. Il rapporto α_x / α_y è detto rapporto di scala (aspect ratio). Inoltre l'intersezione dell'asse ottico con il piano dell'immagine, punto principale, può trovarsi in un punto diverso dal centro geometrico

dell'immagine. Le coordinate del punto principale rispetto al centro dell'immagine sono u_0, v_0. Infine il parametro s introduce un'eventuale distorsione radiale (skew), anche se spesso si può considerare prossimo allo zero. Quindi i parametri da valutare per caratterizzare il processo di trasformazione del punto 3D a pixel sono: 4 per la telecamera, 3 per individuare l'orientazione, 3 per individuare la traslazione. In totale occorre stimare quindi 10 parametri.

E' possibile esprimere tutto il processo di trasformazione tramite un'unica matrice P 3x4, detta matrice di proiezione:

$$P = K [R | t] \tag{1.1.3}$$

$$\lambda \hat{w} = P \hat{X} \tag{1.1.4}$$

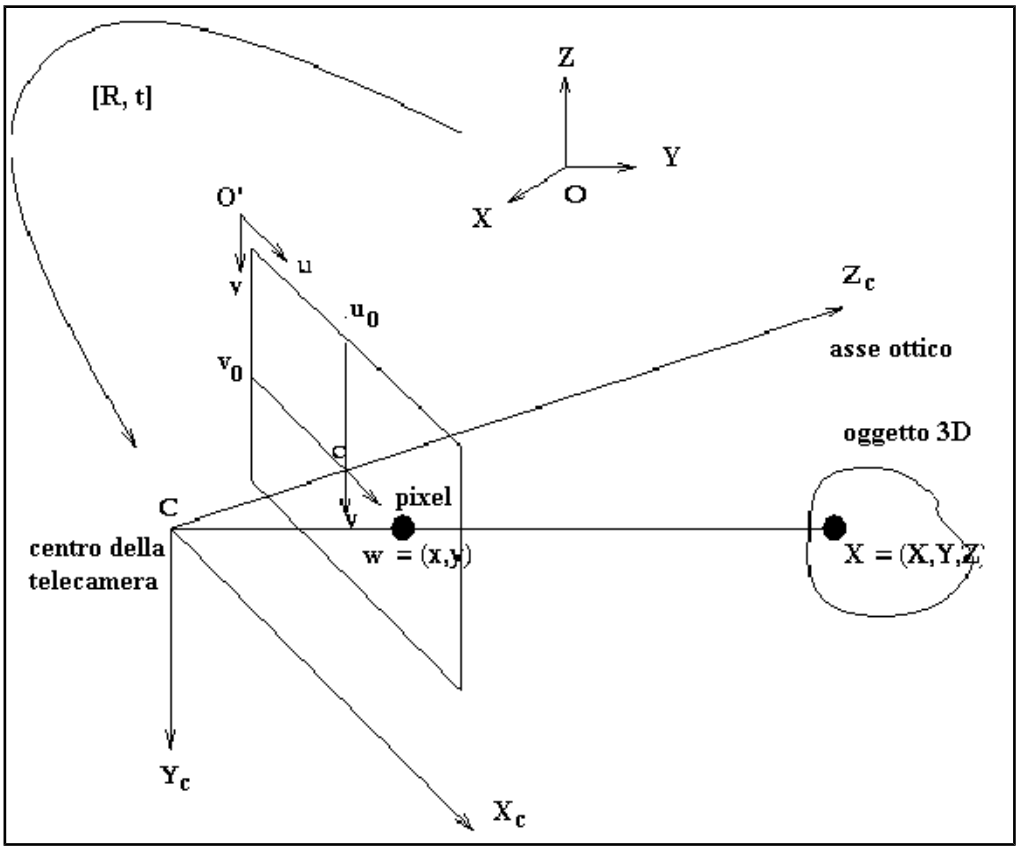


Figura 1.1 *Sistemi cartesiani di riferimento del modello pinhole della telecamera: O (X,Y,Z) è il sistema di riferimento dell'oggetto; C (Xc,Yc,Zc) è quello relativo alla telecamera; O' (u,v) è il sistema di riferimento bidimensionale relativo al piano dell'immagine.*

Utilizzando tale rappresentazione abbiamo il vantaggio di avere un'unica matrice ma con l'esigenza di dover stimare 11 parametri (considerando uno dei dodici elementi della matrice P pari ad 1). Per ricavare P, a meno

di un fattore di scala, occorre conoscere almeno 4 corrispondenze punto 3D – pixel, dato che si introducono 3 vincoli (equazioni) per ognuna di esse.

1.2 Autocalibrazione mediante i punti di fuga.

Una delle caratteristiche delle proiezioni prospettiche è che l'immagine di un entità che tende ad infinito acquista sul piano dell'immagine proprietà finite. Ad esempio rette parallele nello spazio euclidiano si intersecano in un punto all'infinito (punto di fuga), la cui proiezione però può giacere con coordinate finite sul piano dell'immagine. Il punto di fuga (vanishing point) e la sua proiezione dipende univocamente dalla direzione caratterizzante il fascio di rette. Analoghe considerazioni si possono fare per linee e piani all'infinito.

L'importanza dei vanishing points risiede nel fatto che conoscendone tre relativi a direzioni mutuamente ortogonali, si può risalire alla matrice dei parametri intrinseci della telecamera K e alla matrice di rotazione R. Inoltre è possibile ricavare il vettore di traslazione t a meno di un fattore di scala.

Consideriamo ad esempio le direzioni relative ai tre assi cartesiani del sistema di riferimento dell'oggetto tridimensionale. Le rette associate a tali direzioni si intersecano in tre punti all'infinito che indichiamo rispettivamente $\hat{I}_x = [1\ 0\ 0\ 0]^T$, $\hat{I}_y = [0\ 1\ 0\ 0]^T$, $\hat{I}_z = [0\ 0\ 1\ 0]^T$ (in coordinate omogenee un punto all'infinito ha come ultima componente 0). Le proiezioni sul piano dell'immagine saranno $\hat{w}_{Ix}=[u_{Ix}\ v_{Ix}\ 1]^T$, $\hat{w}_{Iy}=[u_{Iy}\ v_{Iy}\ 1]^T$, $\hat{w}_{Ix}=[u_{Iz}\ v_{Iz}\ 1]^T$ e derivano da:

$$\lambda_{Ix}\hat{w}_{Ix} = K\cdot\left[R \quad | \quad t\right]\cdot\begin{bmatrix}1\\0\\0\\0\end{bmatrix} = K\cdot r^1$$

$$\lambda_{Iy}\hat{w}_{Iy} = K\cdot\left[R \quad | \quad t\right]\cdot\begin{bmatrix}0\\1\\0\\0\end{bmatrix} = K\cdot r^2 \qquad (1.2.1)$$

$$\lambda_{Iz}\hat{w}_{Iz} = K\cdot\left[R \quad | \quad t\right]\cdot\begin{bmatrix}0\\0\\1\\0\end{bmatrix} = K\cdot r^3$$

Dove r^1, r^2 , r^3 sono i vettori colonna della matrice di rotazione R. Si indichi con U e Λ le seguenti matrici:

$$U = \begin{bmatrix} u_{vpx} & u_{vpy} & u_{vpz} \\ v_{vpx} & v_{vpy} & v_{vpz} \\ 1 & 1 & 1 \end{bmatrix} \quad (1.2.2)$$

$$\Lambda = \begin{bmatrix} \lambda_{vpx}^2 & 0 & 0 \\ 0 & \lambda_{vpy}^2 & 0 \\ 0 & 0 & \lambda_{vpz}^2 \end{bmatrix} \quad (1.2.3)$$

Si ottiene un sistema lineare di 6 equazioni dalla seguente relazione tra matrici:

$$U \Lambda U^T = K K^T \quad (1.2.4)$$

Se si conoscono quindi le coordinate dei tre vanishing point sul piano dell'immagine è possibile risalire ai parametri intrinseci della telecamera α ($= \alpha_x = \alpha_y$), u_0, v_0. Inoltre si ricava la matrice di rotazione R dalla seguente espressione:

$$R = K^{-1} U \Lambda \quad (1.2.5)$$

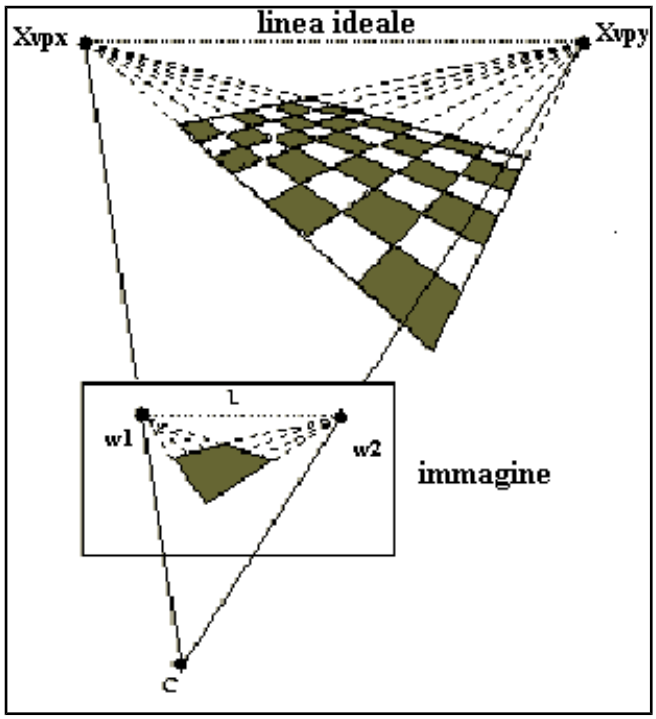


Figura 1.2 *Punti di fuga (vanishing points): insiemi di linee parallele nello spazio euclideo originano delle linee sul piano dell'immagine che si intersecano in un punto.*

Fissando un punto dell'immagine $\hat{w}_a$ come proiezione dell'origine del sistema di riferimento O(X,Y,Z), e conoscendo la corrispondenza di un ulteriore punto $\hat{X}_b$ è possibile ricavare il vettore di traslazione risolvendo il seguente sistema:

$$\lambda_a \hat{w}_a = K\, t$$
$$\lambda_b \hat{w}_b = K\, [R \mid t]\ \hat{X}_b \qquad (1.2.6)$$

Se non si conosce il secondo punto dalla prima parte del sistema precedente si può ricavare t a meno di un fattore di scala.

1.3 Matrici di omografia.

Se consideriamo la proiezione dei punti che appartengono nello spazio tridimensionale ad un piano si possono ricavare delle forme semplificate delle equazioni che descrivono il processo proiettivo. Supponendo per semplicità che X sia un generico punto appartenente al piano Z=0 possiamo scrivere:

$$\lambda \hat{w} = K\, [\, R \mid t\,]\ \hat{X} = K\, [\, r^1\ \ r^2\ \ t\,]\ \hat{X}_{z=0} \qquad (1.3.1)$$

$$H = K\, [\, r^1\ \ r^2\ \ t\,] \qquad (1.3.2)$$

H rappresenta la matrice di omografia (o collineazione) che realizza la trasformazione di punti appartenenti a due piani (il piano dell'immagine ed il piano 3D). Rispetto al caso generale che coinvolge la matrice di proiezione P, adesso si devono stimare 8 parametri (trascurando il fattore di scala). Per fissare l'omografia occorrono 4 corrispondenze invece delle 6 richieste dalla determinazione di P.

Risulta utile considerare anche l'omografia relativa a due immagini di una stessa scena, considerando un insieme di punti appartenenti ad un piano nello spazio euclideo.

$$\hat{w}_{i2} \sim H_{21}\, \hat{w}_{i1} \qquad (1.3.3)$$

$$H_{21} = K_2 \cdot \begin{bmatrix} r_2^1 & r_2^2 & t_2 \end{bmatrix} \cdot \left(K_1 \cdot \begin{bmatrix} r_1^1 & r_1^2 & t_1 \end{bmatrix} \right)^{-1} \qquad (1.3.4)$$

L'omografia H_{21} è la trasformazione che ci porta dal piano dell'immagine 1 al piano dell'immagine 2. L'indice i è relativo all'i-esimo punto dell'immagine risultante dalla proiezione dell'i-esimo punto appartenente al piano Z=0.

1.4 Geometria epipolare.

La geometria epipolare è la geometria intrinseca proiettiva che permette l'analisi di due immagini acquisite da punti di vista differenti. Essa deriva dall'intersezione dei piani dell'immagine con il fascio di piani avente la linea di base (baseline) come asse. La linea di base è quella linea che unisce i centri ottici delle telecamere.

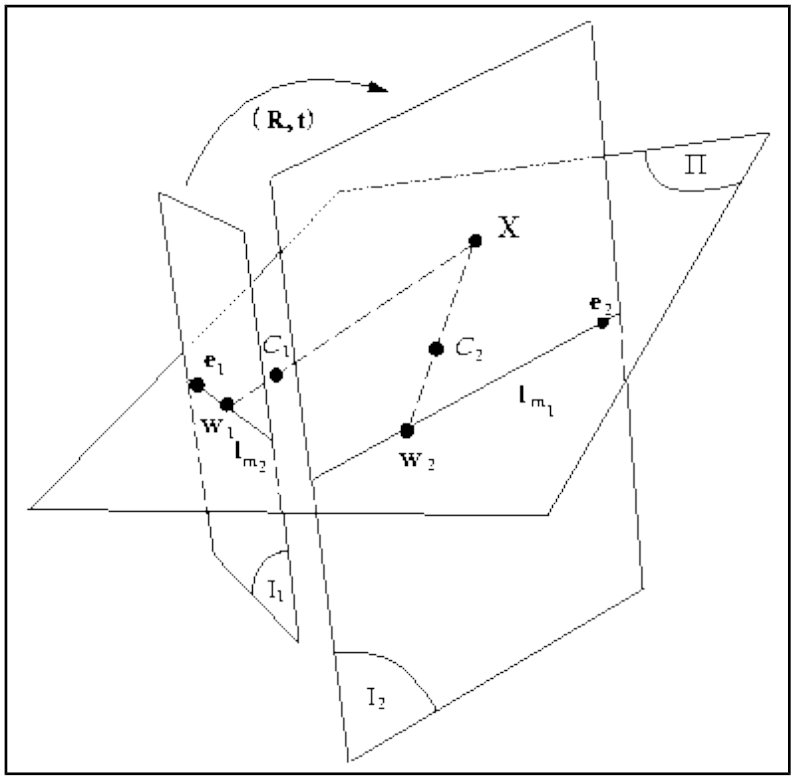


Figura 1.3 *Geometria epipolare: deriva dall'intersezione dei piani dell'immagine I_1 e I_2 con il fascio di piani avente la linea di base (baseline) come asse. La linea di base è quella linea che unisce i centri ottici C_1 e C_2 delle telecamere. Le proiezioni w_1 e w_2 del punto X giacciono rispettivamente sulle linee epipolare l_{m1} ed l_{m2}. Ogni fascio di linee epipolari di una delle immagini è caratterizzato dall'epipolo e_i. L'epipolo e_1 rappresenta anche la proiezione del punto C_2 sul piano dell'immagine I_1, così come e_2 è la proiezione di C_1 su I_2.*

Supponiamo che un punto X nello spazio 3D è visibile in due immagini differenti. Quale è la relazione che lega le corrispondenti proiezioni w_1 e w_2 sulle due immagini? Come è mostrato in figura 1.3. i punti w_1 e w_2 dell'immagine e X individuano un piano Π, sul quale giacciono anche i centri C_1 e C_2 delle telecamere. Le intersezione del piano Π con il piano dell'immagine individua I_1 la linea epipolare l_{w1}. Se si considerano le proiezioni di altri punti dello spazio 3D si avrà un fascio di rette

accomunato dal passaggio in un unico punto e_1 denominato epipolo. Analogamente è definito l'epipolo e_2 sulla seconda immagine.

La rappresentazione algebrica della linea epipolare è data dalla matrice fondamentale F. Supponiamo che l'origine del sistema di riferimento 3D sia coincidente con l'origine C_1 del sistema di riferimento della prima telecamera. Le matrici di proiezione relative alle due telecamere saranno rispettivamente $P_1 = K_1\ [\ I \mid 0\ ]$ e $P_2 = K_2\ [\ R \mid t\ ]$. Allora gli epipoli e_1 ed e_2 saranno dati dalle relazioni:

$$e_1 = K_1 R^T t$$
$$e_2 = K_2 t \qquad (1.4.1)$$

La matrice fondamentale F è allora espressa da:

$$F = [e_2]_x K_2 R (K_1)^{-1} = (K_2)^{-T} [t]_x R (K_1)^{-1}$$
$$= (K_2)^{-T} R [R^T t]_x (K_1)^{-1} = (K_2)^{-T} R (K_1)^T [e_1]_x \qquad (1.4.2)$$

L'espressione della matrice fondamentale può essere determinata in vari modi sia per via algebrica sia per via geometrica. La proprietà che rende così utile la matrice fondamentale è data da:

$$(w_2)^T F w_1 = 0 \qquad (1.4.3)$$

Tale relazione permette il calcolo della F a partire solo da corrispondenze trovate sulle due immagini (almeno 7). A seconda del moto che caratterizza lo spostamento da C_1 a C_2 (cioè R e t) si possono avere informazioni sulla struttura di F ed ottenere ulteriori vincoli per la sua determinazione. Conoscendo le matrici di proiezione delle due immagini è possibile ricercare lungo le linee epipolari corrispondenze che permettono di avere in seguito ricostruzioni più dense.

Se si suppone di lavorare con sistemi calibrati, ossia sono note la matrici K_1 e K_2 dei parametri intrinseci delle telecamere, possiamo utilizzare la cosiddetta matrice essenziale E derivandola direttamente da F:

$$E = (K_2)^T F K_1 = [t]_x R = R [R^T t]_x \qquad (1.4.4)$$

Il vincolo su punti corrispondenti sulle due immagini è allora esprimibile come:

$$(w_2)^T E w_1 = 0 \qquad (1.4.5)$$

1.5 Calcolo della matrice fondamentale e ricerca di corrispondenze.

Vari metodi sono stati proposti per il calcolo della matrice fondamentale e per determinare quindi la geometria epipolare relativa ad una coppia di immagini. Tra quelli che rivestono una notevole importanza e sono largamente usati è opportuno introdurre il metodo basato sul RANSAC (Random Sample Consensus). Tale algoritmo calcola in maniera automatica la matrice fondamentale, non utilizzando nessuna informazione a priori e restituisce inoltre un insieme di corrispondenze di punti d'interesse. Vediamo di esaminare i vari passi di tale metodologia:

i. Ricerca di punti d'interesse: tramite un algoritmo come quello di Harris si individuano punti d'interesse sulle due immagini. E' importante avere punti come angoli o intersezioni di contorni che sono meno sensibili alle variazioni d'intensità luminosa tra le due immagini e si possano localizzare anche con risoluzione sub-pixel.
ii. Corrispondenze putative: si determina un insieme di corrispondenze trai punti d'interesse basandosi su criteri di prossimità e similarità.
iii. Stima robusta tramite RANSAC: si ripete per N campioni, dove N è determinato adattativamente:
    a. Si seleziona un campione casuale di 7 corrispondenze e si calcola la matrice fondamentale F. Si avrà almeno una soluzione reale sulle tre possibili.
    b. Si calcola la distanza d per ogni corrispondenza putativa.
    c. Si calcola il numero di punti (inliers) che soddisfano entro una certa tolleranza la consistenza di F, cioè d < toll pixel.
    d. Se ci sono tre soluzioni reali di F occorre fare il calcolo del passo c. per ogni soluzione, e tenere solo quella che soddisfa più punti.

    Si sceglie la F con il più largo consenso, scegliendo a parità di inliers quella con la più bassa deviazione standard.
iv. Stima non – lineare: si stima nuovamente F utilizzando tutte le corrispondenze classificate come inliers minimizzando una data funzione di costo ed usando l'algoritmo di Levenberg - Marquardt.
v. Ricerca di corrispondenze guidata: ulteriori corrispondenze di punti di interesse sono determinate a partire dalla matrice F calcolata, definendo attorno le linee epipolari una zona di tolleranza.

Gli ultimi due passi devono essere iterati fino a quando si ottiene un numero di corrispondenze stabile.

1.6 Triangolazione.

Se si suppone che le matrici di calibrazione e la matrice fondamentale sono state stimate è possibile dalle corrispondenze di punti ricavare la posizione tridimensionale dei punti che hanno generato le proiezioni corrispondenti. Ricavando per ogni coppia di punti corrispondenti nelle due immagini i raggi di proiezione, si ricava la posizione nello spazio euclideo dalla loro intersezione.
Questo è in parole semplici il cosiddetto metodo di triangolizzazione. Naturalmente ci saranno degli errori dovuti al rumore sul calcolo delle posizioni sull'immagine dei punti w_{i1} e w_{i2}. Questo significa che le relazioni $w_{i1}=P_1X_i$ e $w_{i2}=P_2X_i$ non saranno perfettamente soddisfatte, così come il vincolo epipolare $(w_{i2})^T F w_{i1} = 0$. Occorre quindi dare robustezza al metodo di triangolizzazione cercando di limitare gli effetti di rumore e quindi l'errore di ricostruzione.
Una prima stima delle posizioni 3D dei punti può essere ricavata utilizzando una metodologia lineare e risolvendo appunto per ogni coppia di corrispondenze il sistema:

$$w_{i1}=P_1X_i$$

$$w_{i2}=P_2X_i \qquad (1.6.1)$$

Facendo una stima basata su ogni singolo punto le relazioni geometriche non saranno sicuramente soddisfate data la presenza dell'errore di misura e del modello semplificato del dispositivo di acquisizione. Per ciascuna coppia di immagini ad ogni corrispondenza sono associate le equazioni (1.6.1) che possono essere combinate in un sistema lineare AX=0. Il fattore di scala può essere eliminato operando un prodotto vettoriale ed ottenendo tre equazioni di cui due linearmente indipendenti. Per esempio, se si calcola w x (PX)=0, allora si ottengono:

$$u\,(p^{3T} X) - (p^{1T} X) = 0$$

$$w\,(p^{3T} X) - (p^{2T} X) = 0$$

$$u\,(p^{2T} X) - w\,(p^{1T} X) = 0 \qquad (1.6.2)$$

dove con p^{iT} sono indicati i vettori riga della matrice di proiezione P. Tali equazioni sono lineari nelle componenti del vettore X.

Quindi il sistema lineare AX=0 è rappresentato da:

$$A = \begin{bmatrix} u_1 p_1^{3T} - p_1^{2T} \\ w_1 p_1^{3T} - p_1^{2T} \\ u_2 p_2^{3T} - p_2^{2T} \\ w_2 p_2^{3T} - p_2^{2T} \end{bmatrix} \qquad (1.6.3)$$

dove sono state introdotte due equazioni per ogni immagine. Questo insieme di equazioni è ridondante dato che la soluzione è determinata a meno di un fattore di scala. Tale sistema può essere risolto in due modi:

1) Metodo omogeneo (DLT): si trova la soluzione come il vettore singolare unitario corrispondente al più piccolo valore singolare della matrice A. Per ottenere una migliore approssimazione è opportuno che si operi una normalizzazione preliminare.
2) Metodo non omogeneo: si ricerca soluzione con il metodo dei minimi quadrati servendosi direttamente delle equazioni non linearizzate (1.6.1). Possono sorgere dei problemi se X ha l'ultima coordinata prossima allo zero, quando invece per eliminare il fattore di scala si pone normalmente pari ad uno.

I due metodi sono abbastanza simili ma presentano comportamenti differenti rispetto al rumore. Il metodo non omogeneo suppone che il punto soluzione non sia un punto all'infinito. Ciò rappresenta una difficoltà quando si sta cercando di ottenere la ricostruzione proiettiva, dove i punti ricostruiti potrebbero appartenere al piano all'infinito. A partire da queste prime stime lineari è opportuno procedere con una stima non lineare cercando di minimizzare una funzione di costo data. In genere si utilizza come funzione d'errore quella dell'errore geometrico:

$$C(w_{i1}, w_{i2}) = d(w_{i1}, \hat{w}_{i1})^2 + d(w_{i2}, \hat{w}_{i2})^2$$

$$\text{soggetta al vincolo } (\hat{w}_{i2})^T F \hat{w}_{i1} = 0 \qquad (1.6.4)$$

dove d(*,*) è la distanza euclidiana tra il punto dell'immagine e la sua stima. Le stime dei punti sull'immagine sono relative all'assunzione di una distribuzione d'errore Gaussiana e allo stimatore di massima verosimiglianza. Questa funzione di costo è minimizzata utilizzando un metodo di minimizzazione numerica come ad esempio quello di Levenberg – Marquardt.
Una approssimazione vicina al minimo può essere determinata utilizzando un'approssimazione del primo ordine della funzione di costo geometrica, chiamata funzione d'errore di Sampson. La soluzione può essere ottenuta

anche non in maniera iterativa risolvendo un polinomio di sesto grado. L'approssimazione di Sampson è valida quando gli errori sulle misure sono piccoli rispetto le misure stesse. La correzione δ_w del punto $W=[u_1,w_1,u_2,w_2]$ è data da:

$$\delta_w = - J^T (JJ^T)^{-1} \varepsilon \qquad (1.6.5)$$

dove J è la matrice delle derivate parziali ed ε è il costo C(W) associato a W. Il punto corretto è allora:

$$\hat{W} = W + \delta_w = W - J^T (JJ^T)^{-1} \varepsilon \qquad (1.6.6)$$

Nel caso della varietà definita da $(w_2)^T F w_2 = 0$, l'errore è $\varepsilon = (w_2)^T F w_2$ e lo Jacobiano è dato da:

$$J = \partial\varepsilon/\partial w = [(F^T w_2)_1, (F^T w_2)_2, (F^T w_1)_1, (F^T w_1)_2] \qquad (1.6.7)$$

dove ad esempio $(F^T w_2)_1 = f_{11}u_1 + f_{21}w_1 + f_{31}$. Quindi l'approssimazione del primo ordine del punto corretto è la seguente:

$$\begin{bmatrix} \hat{u}_1 \\ \hat{w}_1 \\ \hat{u}_2 \\ \hat{w}_2 \end{bmatrix} = \begin{bmatrix} u_1 \\ w_1 \\ u_2 \\ w_2 \end{bmatrix} - \frac{w_2^T F w_1}{(Fw_1)_1^2 + (Fw_1)_2^2 + (F^T w_2)_1^2 + (F^T w_2)_2^2} \begin{bmatrix} (F^T w_2)_1 \\ (F^T w_2)_2 \\ (Fw_1)_1 \\ (Fw_1)_2 \end{bmatrix} \qquad (1.6.8)$$

L'approssimazione è accurata se la correzione in ogni immagine è piccola (meno di un pixel), ed ha il grande vantaggio di non essere onerosa dal punto di vista computazionale. Si noti che i punti corretti non soddisferanno il vincolo epipolare esattamente.

1.7 Ricostruzione ottimale.

In questa sezione sarà descritto un metodo di triangolazione che determina il minimo globale della funzione di costo (1.6.4) usando un algoritmo iterativo. Se il modello Gaussiano del rumore può essere ritenuto corretto, tale metodo è stato dimostrato ottimale [Hartley00].
Data una corrispondenza $w_1 \leftrightarrow w_2$ si ricerca una coppia di punti $\hat{w}_1$ e $\hat{w}_2$ che minimizza la somma dei quadrati delle distanze (1.6.4) soggetta al vincolo epipolare $(\hat{w}_2)^T F \hat{w}_1 = 0$.
Qualsiasi coppia di punti che soddisfa il vincolo epipolare deve giacere su una coppia di linee epipolare corrispondenti delle due immagini. Quindi il

punto ottimale $\hat{w}_1$ giace su una linea epipolare l_1 ed analogamente $\hat{w}_2$ su l_2. Sia $w_{1\#}$ il punto di l_1 più vicino a w_1 misurato, e $w_{2\#}$ il punto di l_2 più vicino a w_2. Si può allora dire che $d(w_1, \hat{w}_1) = d(w_1, l_1)$, dove $d(w_1, l_1)$ rappresenta la distanza perpendicolare del punto w_1 dalla retta l_1. Un'analoga espressione è valida per $d(w_2, l_2)$.
La minimizzazione allora si applica alla quantità:

$$d(w_1, l_1)^2 + d(w_2, l_2)^2 \qquad (1.7.1)$$

dove l_1 ed l_2 sono scelte tra tutte le coppie di linee epipolari corrispondenti. Il punto $\hat{w}_1$ è allora il punto più vicino della linea l_1 al punto w_1 , e similarmente è definito $\hat{w}_2$.

La strategia di minimizzazione della (1.7.1) è la seguente:

i. Parametrizzare il fascio di linee epipolari della prima immagine tramite il parametro t. Quindi indichiamo la generica linea epipolare della prima immagine $l_1(t)$.
ii. Utilizzando la matrice fondamentale si calcola la corrispondente linea epipolare generica $l_2(t)$ sulla seconda immagine.
iii. Si esprime la funzione di distanza $d(w_1, l_1(t))^2 + d(w_2, l_2(t))^2$ esplicitando la dipendenza dal parametro t.
iv. Si trova il valore di t che minimizza la funzione di costo.

In questo modo, il problema è stato ricondotto al trovare il minimo di una funzione di una sola variabile t:

$$\min_w \{C=d(w_1, \hat{w}_1)^2+d(w_2, \hat{w}_2)^2 \} = \min_t \{C=d(w_1,l_1(t))^2+d(w_2, l_2(t))^2\}$$

Facendo un'opportuna parametrizzazione del fascio di linee epipolari, la funzione di distanza è una funzione polinomiale razionale nella variabile t. Utilizzando semplici tecniche di calcolo, il problema della minimizzazione è ricondotto alla ricerca di radici reali di una polinomiale di sesto grado.

1.7.1 Dettagli della minimizzazione.

Se entrambi i punti delle immagini corrispondono con i rispettivi epipoli, allora il punto dello spazio giace sulla retta che passa per i centri delle telecamere. In tal caso è impossibile determinare la posizione del punto nello spazio. Se solamente uno dei punti coincide con un epipolo, allora si può affermare che l'altro coincide con il centro dell'altra telecamera. Di conseguenza si suppone che nessuno dei due punti delle immagini corrispondano agli epipoli.

Sotto queste ipotesi, possiamo semplificare l'analisi applicando una trasformazione rigida ad ciascuna immagine allo scopo di piazzare entrambi i punti w_1 e $\hat{w}_1$ all'origine, $[0\ 0\ 1]^T$ in coordinate omogenee.
Inoltre si possono piazzare gli epipoli rispettivamente nei punti $[1\ 0\ f_1]^T$ e $[1\ 0\ f_2]^T$. Un valore di f nullo significa che l'epipolo è all'infinito. L'applicazione di queste due trasformazioni rigide non influisce sulla funzione della somma dei quadrati delle distanze (1.7.1), e quindi lascia inalterato il problema della minimizzazione. La matrice fondamentale diviene allora:

$$F = \begin{bmatrix} f_1 f_2 d & -f_2 c & -f_2 d \\ -f_1 b & a & b \\ -f_1 d & c & d \end{bmatrix} \tag{1.7.2}$$

Si consideri una linea epipolare della prima immagine passante attraverso il punto $[0\ t\ 1]^T$ e l'epipolo $[1\ 0\ f_1]^T$. Indichiamo questa linea epipolare con $l_1(t)$. Il vettore che rappresenta questa linea è dato dal prodotto vettore $[0\ t\ 1]\ \times\ [1\ 0\ f_1] = [tf_1\ 1\ -t]$, ed la distanza al quadrato della linea dall'origine è

$$d(w_1, l_1(t))^2 = \frac{t^2}{1 + (tf_1)^2} \tag{1.7.3}$$

Usando la matrice fondamentale per trovare la corrispondente linea epipolare nella seconda immagine si ha

$$l_2(t) = F\,[0\ t\ 1]^T = [-f_1(ct+d),\ at+b,\ ct+d]^T \tag{1.7.4}$$

Questa è la rappresentazione della linea $l_2(t)$ come vettore omogeneo. La distanza al quadrato di questa linea dall'origine è uguale a

$$d(w_2, l_2(t))^2 = \frac{(ct+d)^2}{(at+b)^2 + f_2^2 (ct+d)^2} \tag{1.7.5}$$

La distanza al quadrato totale è data da :

$$s(t) = \frac{t^2}{1 + (tf_1)^2} + \frac{(ct+d)^2}{(at+b)^2 + f_2^2 (ct+d)^2} \tag{1.7.6}$$

Questa è la funzione della quale ricerchiamo il minimo. Determiniamo la derivata di tale funzione per poi uguagliarla a zero:

$$s'(t) = \frac{2t}{1+(tf_1)^2} + \frac{2(ad - bc)(at+b)(ct+d)}{(at+b)^2 + f_2^2(ct+d)^2} \qquad (1.7.7)$$

Massimi e minimi corrispondono all'annullamento della derivata prima. Raggruppando i due termini della derivata sotto un comune denominatore ed eguagliando il numeratore a zero si ottiene la seguente condizione:

$$g(t) = t((at+b)2 + (f_1)2\,(ct+d)2)2-(ad-bc)\,(1+ (f_1)2t2)2(at+b)(ct+d) = 0 \qquad (1.7.8)$$

I minimi o massimi della s(t) saranno anche le radici del polinomio g(t). Dato che tale polinomio ha grado sei , al più si possono avere sei radici reali corrispondenti a tre minimi e tre massimi della funzione s(t).
Il minimo assoluto di s(t) può essere trovato trovando le radici di g(t) e valutando la funzione s(t) per ognuna delle radici reali. Più semplicemente, si può controllare solo il valore di s(t) della parte reale di ogni radice, eliminando il problema di controllare se la radice è reale o complessa. Si può anche controllare il valore asintotico si s(t) al tendere di t ad infinito per vedere se la distanza minima corrisponde a $t \rightarrow \infty$, corrispondente ad una linea epipolare $f_1 w = 1$ nella prima immagine.

1.7.2 Ricostruzione di linee e calcolo dei punti di fuga.

Si supponga che una retta nello spazio 3D sia proiettati in due immagini nelle due rette l_1 ed l_2. La linea nello spazio tridimensionale può essere ricostruita dalla proiezione inversa di ognuna delle linee ed intersecando i due piani così determinati.
Siano $\pi_1 = P_1^T l_1$ e $\pi_2 = P_2^T l_2$ i due piani in questione. Spesso è conveniente parametrizzare la linea dello spazio tridimensionale utilizzando i due piani definiti dalle linee delle immagini, e si rappresenta matrice la seguente matrice 2x4:

$$L = \begin{bmatrix} l_1^T P_1 \\ l_2^T P_2 \end{bmatrix} \qquad (1.7.9)$$

Nel caso del sistema originato da una corrispondenza di punti in due immagini, si avevano quattro equazioni per determinare tre incognite (la posizione del punto nello spazio). Di converso, nel caso di corrispondenze di linee, si hanno quattro equazioni e quattro incognite e quindi la soluzione è ancora determinabile. Per quanto riguarda le condizioni che provocano degenerazioni al problema, queste sono più onerose rispetto quelle relative alle corrispondenze di punti. Le rette che giacciono sul piano epipolare non possono essere determinate a partire dalle loro immagini in due differenti viste. Nel caso dei punti si aveva una famiglia di punti determinati al

variare di un solo parametro. Nel caso delle linee invece la famiglia delle degenerazioni dipende addirittura da tre parametri.
Un'applicazione della determinazione di linee nello spazio tridimensionale è data dalla ricerca dei punti di fuga. Si è visto nelle sezioni precedenti che rette parallele nello spazio tridimensionale danno luogo a linee sulle immagini che possono intersecarsi.
Se consideriamo un fascio di rette sull'immagine che corrisponde ad una particolare direzione nello spazio, origineranno un'intersezione corrispondente alla proiezione del punto di fuga.
Data la presenza di rumore delle misure, generalmente l'intersezione di tali linee non sarà unica. Spesso il vanishing point è fissato allora nel punto più vicino a tali rette. Tale soluzione non è tuttavia ottimale.
Occorre quindi ricorrrere allo stimatore di massima verosimiglianza (MLE) sia per il vanishing point che le linee che lo generano. Per fare ciò si minimizza la somma dei quadrati delle distanze ortogonali di due punti stimati dalla linea misurata. Anche in questo caso è opportuno ricorrere al metodo di minimizzazione di Levemberg-Marquardt.

1.8 Rettificazione.

Un'utile trasformazione proiettiva è quella della rettificazione. Il problema consiste, nel caso più generale, nel determinare nuove immagini a partire da quelle date. Si parla di rettificazione ad esempio nella tecnica del mosaicing che consiste nel creare un'immagine panoramica risultante dall'unione di più immagini. Naturalmente la fusione deve considerare la differenza dei punti di vista e creare quindi la nuova immagine in maniera consistente. Nel caso di due immagini, per esempio, si può pensare di ricondursi a due nuovi punti di vista che portino le due immagini ad essere accostate (o sovrapposte) sullo stesso piano (vedi figura 1.4).
Si tratta quindi in genere nel determinare omografie che possano trasferire i punti di un'immagine in un altro dato piano rappresentante la nuova immagine rettificata. Supponiamo che P sia la matrice di proiezione relativa ad una delle due immagini. Possiamo estrarre le lunghezze focali a ed a sfruttando le seguenti relazioni:

$$\alpha_x = \| p^1 \circ p^3 \|$$

$$\alpha_y = \| p^2 \circ p^3 \| \qquad (1.8.1)$$

Il centro ottico è dato da :

$$C = - ([p^1 \; p^2 \; p^3] \; p^4)^{-1} \qquad (1.8.2)$$

Se P_1 e P_2 sono le matrici di proiezioni delle due immagini , la normale al piano retinale è data da:

$$n = (p_{3\_P1})^T \circ (p_{3\_P2})^T \qquad (1.8.3)$$

Dalle precedenti relazioni è immediato ricavare le omografie H_{1_rect} ed H_{2_rect} che realizzano la rettificazione voluta.

Per quello che interessa per il prosieguo si è interessati ad una rettificazione più semplificata: data un immagine visualizzante una proiezione di una struttura planare tridimensionale, si vuole determinare l'omografia che porti ad un nuovo piano dell'immagine parallelo al piano considerato. Si avrà così una nuova immagine con la struttura planare non deformata dalla prospettiva, e caratterizzata da una tessitura simile (a meno di un fattore di scala) a quella del piano originale.

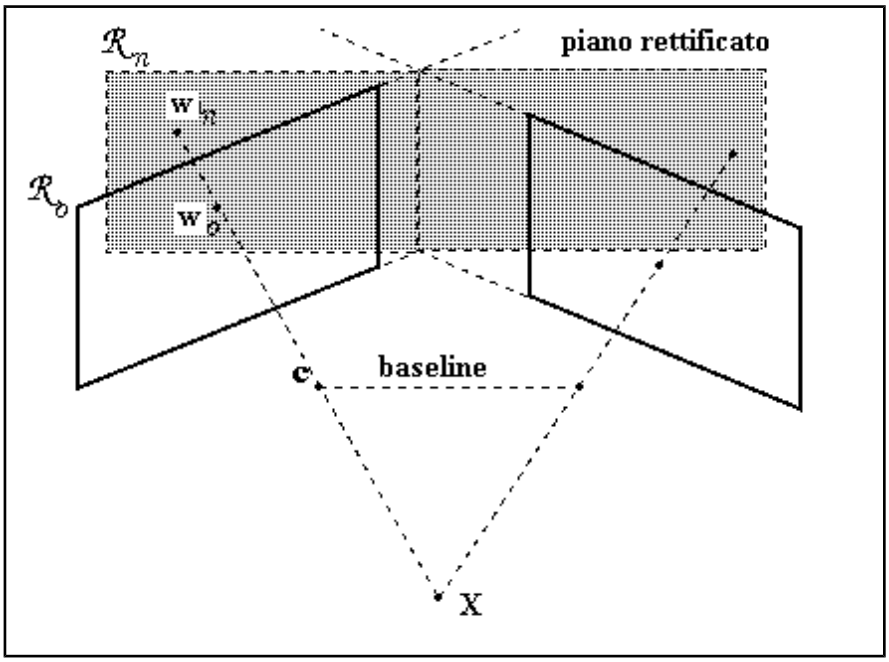


Figura 1.4 *Rettificazione per il mosaicing: date due immagini I_1 e I_2 si vogliono determinare nuove immagini che siano parallele alla vaseline e che siano contigue o sovrapposte se hanno porzioni d'immagini che si riferiscono agli stessi punti 3D.*

Supponiamo che w sia la proiezione del punto X appartenente al piano Z=0, allora si avrà l'omografia $H = [r^1\ r^2\ t]$ tale che:

$$\lambda\ \hat{w} = H\ X^{\wedge}_{z=0} \qquad (1.8.4)$$

Se vogliamo posizionare il nuovo piano dell'immagine ad una distanza d_{ca} pari alla distanza del centro ottico nell'imagine di partenza dalla struttura planare, allora l'omografia che realizza il parallelismo H_{par} è data da:

$$H_{par} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -d_{ca} \end{bmatrix} \qquad (1.8.5)$$

Combinando H ed H otteniamo l'omografia di rettificazione H:

$$H_{rect} = H_{par}\, H^{-1} \qquad (1.8.6)$$

La trasformazione rettificante riduce anche il problema della ricerca di corrispondenze in una coppia d'immagini, semplificando la determinazione della geometria epipolare. Avendo le due immagini su uno stesso piano, è possibile trovare le corrispondenze esaminando righe di pixel delle due immagini, senza quindi doverle ricercare su rette non orizzontali. La semplificazione si ripercuote nel fatto di poter implementare algoritmi più veloci ed efficienti dal punto di vista computazionale.
L'approccio classico nel fare questo tipo di trasformazione è quello di ricondurre l'epipolo di ogni immagine al punto all'infinito. In questo modo si hanno appunto linee epipolari parallele all'asse x dell'immagine. Nella determinazione della matrice di trasformazione H_{rect} si hanno quattro gradi di libertà.
Una condizione che conduce a buoni risultati è quella di imporre che la trasformazione H_{rect} sia quanto più possibile una trasformazione rigida nell vicinanze di un dato punto w_0 dell'immagine. Ciò significa che l'approssimazione del primo ordine della trasformazione in vicinanza del punto w_0 sia risultante solo da una rotazione ed una traslazione, e quindi mantenendo intatta l'apparenza originale dell'area attorno ad esso. Un'appropriata scelta di w_0 può essere rappresentata dal centro dell'immagine.
Supponiamo di fissare l'origine del sistema su w_0 e che le coordinate dell'epipolo siano e = [f 0 1], cioè giaccia sull'asse x. Si consideri la seguente trasformazione:

$$G = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1/f & 0 & 1 \end{bmatrix} \qquad (1.8.7)$$

Questa trasformazione porta l'epipolo $[f,0,1]^T$ al punto all'ininito $[f,0,0]^T$ come richiesto. Un generico punto $[u,v,1]^T$ viene trasferito da G nel punto $[u^*,v^*,1]^T = [u,v,1-u/f]^T$. Se $|u / f| < 1$ allora si può scrivere:

$$[u^*,v^*,1]^T = [u,v,1-u/f]^T = [u(1+u/f+...),v(1+x/f+...),1]^T \qquad (1.8.8)$$

e lo Jacobiano è:

$$\frac{\partial(u^*, v^*)}{\partial(u, v)} = \begin{bmatrix} 1 + 2u/f & 0 \\ v/f & 1 + u/f \end{bmatrix} \tag{1.8.9}$$

più gli altri termini di ordine superiore in u e v. Se u = v = 0 allora lo Jacobiano diventa la matrice d'identità. In altre parole, G è approssimata (al primo ordine) nell'origine dalla matrice d'identità.
Per un'arbitrario punto d'interesse w_0 ed epipolo e, la trasformazione richiesta è H_{rect}=GRT, dove R e T sono la rotazione e la traslazione per portare w_0 nell'origine e l'epipolo nel punto [f,0,1].

Parte II

Algoritmi per la ricostruzione tridimensionale

2.1 Introduzione.

La ricostruzione di scene architettoniche è uno dei temi di ricerca trattato in maniera ampia negli ultimi anni. Oltre ad essere un valido terreno di sperimentazione delle tecniche sviluppate dalla visione artificiale, presenta numerosi risvolti di natura applicativo – commerciale. Si vuole ottenere in maniera automatica la ricostruzione a partire da una serie di immagini non calibrate. In questo capitolo è presentata una metodologia che fa uso delle informazioni geometriche ottenibili sia dalle immagini, sia dalla conoscenza della pianta della scena in esame (vedi figura 7.1). Si suppone che siano acquisite coppie di immagini relative ad elementi architettonici con la presenza predominante di strutture planari come facciate. Sotto questa ipotesi, si fa largo impiego delle stime delle omografie estendendole quindi alla ricostruzione vera e propria.

Gli approcci utilizzati in questo contesto di ricerca sono stati numerosi e differenti fra loro. Alcuni tentano la ricostruzione tridimensionale basandosi su coppie di immagini stereo calibrate [Narayanan98]. Altri applicano la triangolarizzazione ed i vincoli ricavati dalla geometria epipolare su sequenze estese di immagini della scena [Beardsley96, Pollefeyes98, Tomasi90], ovvero utilizzando vincoli di trilinearità [Hartley96, Luong96]. Altri approcci consistono nella visualizzazione basata sull'immagine, senza ricavare un modello esplicito tridimensionale [Faugeras88, Gortler96, Seitz96, Szeliski97, Szeliski98]. Si possono ricavare modelli tridimensionali anche da mosaico panoramico di immagini e vincoli geometrici [Kang97, Shum98]. L'approccio scelto è invece legato all'utilizzo esteso di vincoli ricavati dalla stessa scena, come parallelismo, ortogonalità e corrispondenza di elementi geometrici. Tale approccio è stato utilizzato con fruttuosità da molti ricercatori ed è ampiamente trattato in letteratura [Cipolla99, Debevec96, Liebowitz98]. Il contributo originale della presente trattazione consiste nell'utilizzo dei vincoli ricavabili dalla pianta e che permettono da poche viste della scena di ottenere ricostruzioni suggestive e precise della scena.

2.2 Schema del sistema di ricostruzione proposto.

Il sistema di ricostruzione utilizza una o più coppie di immagini di facciate delle strutture architettoniche da ricostruire. Per ogni coppia

vengono introdotte corrispondenze di elementi geometri (punti, rette, etc.) tra immagine e pianta. Il sistema considera tutti i possibili vincoli ricavati da tali corrispondenze e elabora un modello consistente e coerente ad essi. L'elaborazione è suddivisa in parecchi passi lineari che garantiscono semplicità e efficienza computazionale.

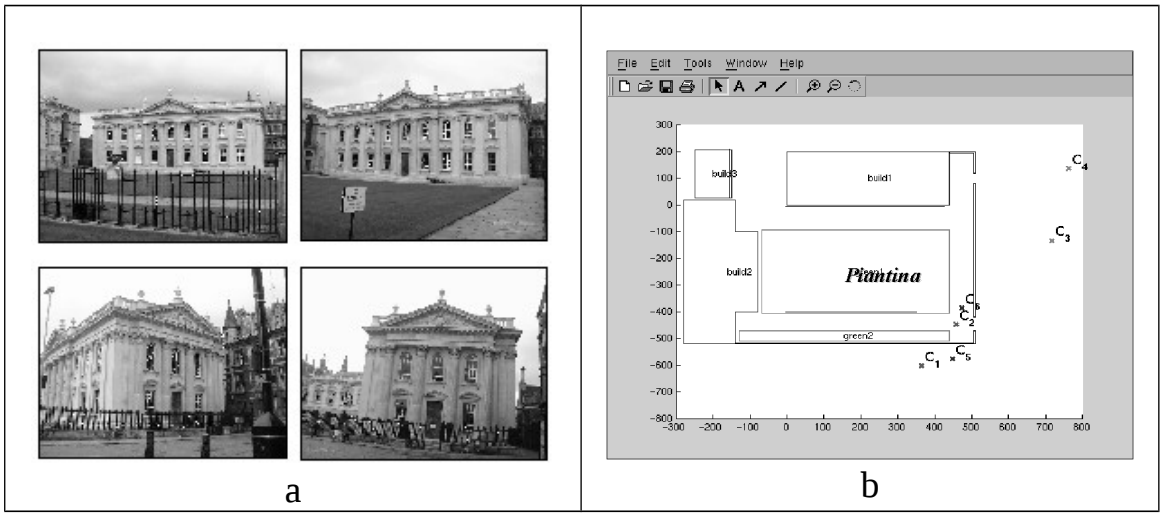


Figura 2.1 Ricostruzione di una scena architettonica: (a) n-immagini della scena con la presenza predominante di strutture piane (facciate); (b) pianta della scena da ricostruire.

Sulla singola immagine si realizzano le seguenti fasi:

- Stima dei parametri intrinseci della telecamera e della sua orientazione tramite la localizzazione dei punti all'infinito dell'immagine prospettica.

- Stima della traslazione della telecamera fissando due punti noti sulla pianta.

- Rettificazione dell'immagine basata sull'omografia rispetto ad una struttura planare di riferimento.

In questo modo, per ogni singolo punto di vista, si hanno una prima stima della matrice di proiezione e la tessitura degli elementi planari. Allo scopo di avere stime migliori delle matrici di proiezione si utilizza l'omografia tra due immagini che inquadrano la stessa superficie planare e si ricavano in maniera automatica corrispondenze per la successiva ricostruzione. Il modello finale è ricavato dopo aver eseguito una ottimizzazione globale, sfruttando tutte le immagini, la pianta, i vincoli introdotti e le

corrispondenze ricavate automaticamente. Le fasi finali dell'elaborazione sono dunque le seguenti:

- Ottimizzazione globale utilizzando tutti i vincoli della pianta.
- Triangolazione e ricostruzione grezza del modello.
- Ricostruzione della tessitura e posizionamento sul modello.
- Esportazione del modello nello standard VRML.

2.3 Calibrazione e stima della matrice di proiezione.

Introducendo su ogni singola immagine una serie di linee corrispondenti alle direzioni orizzontale e verticale è possibile ricavare due punti all'infinito della proiezione prospettica (vedi immagine 2.2). Per una completa calibrazione occorrerebbero tre di tali punti corrispondenti a direzioni mutuamente ortogonali, ma anche con solamente due di essi è possibile avere valide informazioni.

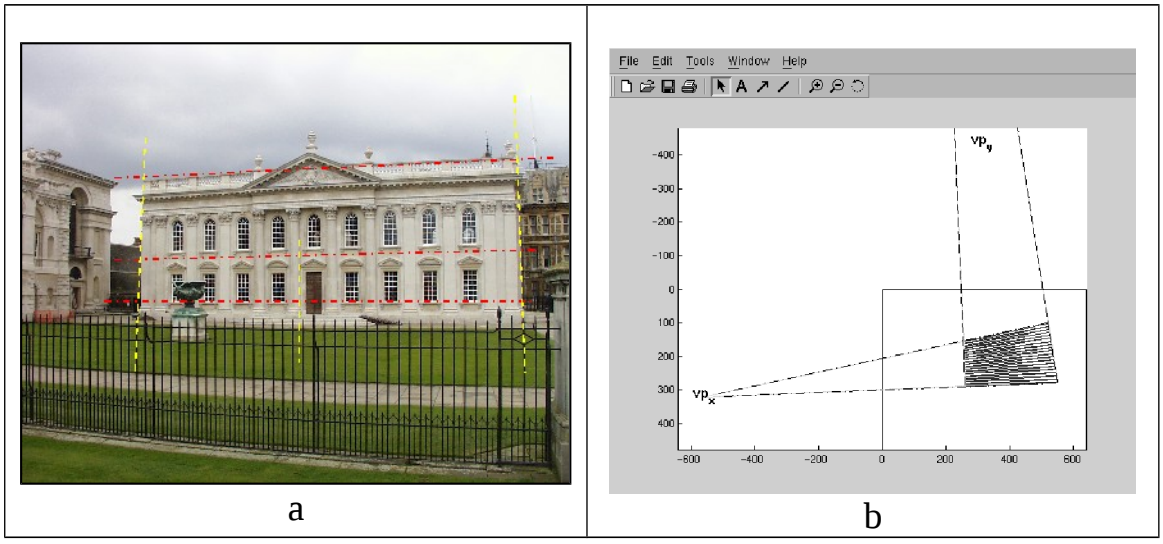


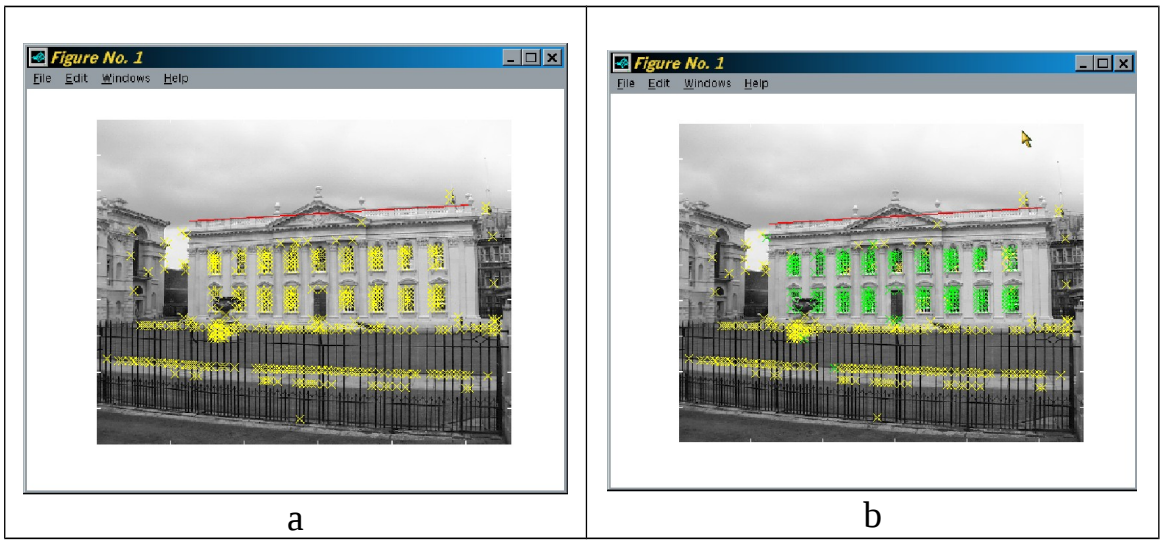
Figura 2.2. Calibrazione tramite punti all'infinito: (a) insiemi di linee evidenziate dall'utente per individuare due direzioni mutuamente perpendicolari; (b) calcolo delle intersezioni dei due fasci di retta e determinazione dei punti all'infinito.

Supponendo che il punto principale coincida con il centro geometrico dell'immagine e il parametro di distorsione radiale sia trascurabile, si ottiene una stima significativa della lunghezza focale e quindi della matrice di calibrazione K. Inoltre abbiamo una stima della matrice di rotazione R della telecamera e, fissando due punti noti sulla pianta, anche il vettore di

traslazione t. Tutte queste componenti ci danno una stima approssimativa della matrice di proiezione legata alla singola vista.

2.4 Ricerca di nuove corrispondenze.

Per migliorare le stime delle matrici di proiezione e ricavare corrispondenze tra punti di immagini differenti si impiega una tecnica basata su omografia. Considerando due immagini di una stessa struttura planare, si ricavano su entrambe insiemi di punti caratteristici.



a b

Figura 2.3. *Ricerca di nuove corrispondenze: (a) insieme di punti ricavato dall'applicazione del rilevatore di angoli di Harris; (b) selezione degli angoli (in verde) per la nuova stima dell'omografia.*

Applicando ad esempio l'algoritmo di Harris, si rilevano un certo numero di angoli presenti su ogni immagine. Per stabilire la corrispondenza di tali angoli nelle due diverse immagini si impiega l'omografia ricavata dalle precedenti stime delle matrici di proiezione relative ai due punti di vista. Considerando anche una certa tolleranza si ottengono le corrispondenze volute appartenenti alla struttura planare presa in considerazione. Tali corrispondenze sono inoltre sfruttate per una nuova stima dell'omografia in esame e, tramite decomposizione, della matrice di proiezione che lega le due immagini.

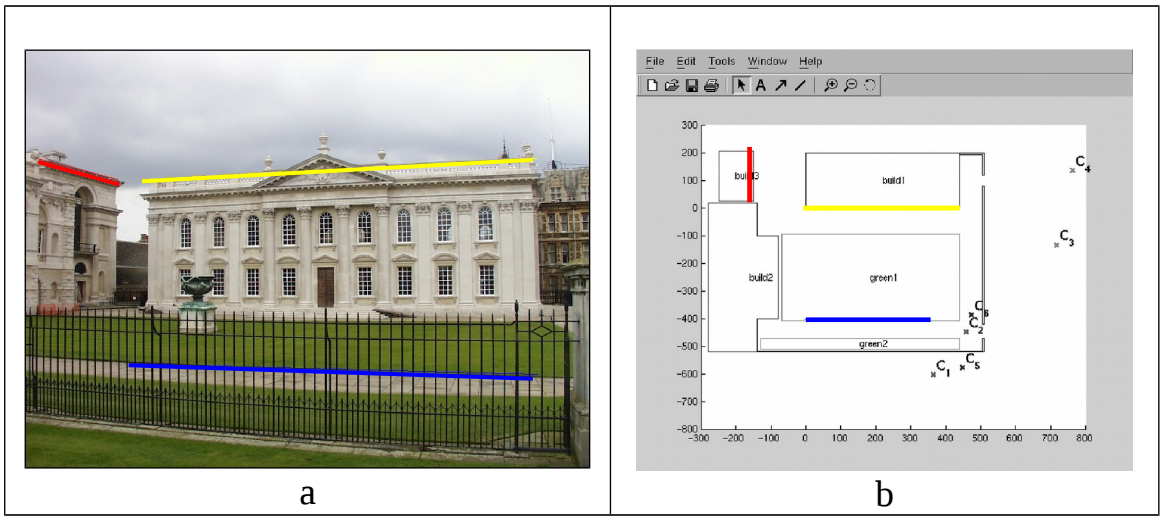
2.5 Vincoli ricavati dalla pianta.

La pianta costituisce una sorgente ricca d'informazioni geometriche utili alla caratterizzazione del modello tridimensionale da ricavare. I vincoli utilizzabili in maniera semplice scaturiscono dalle seguenti corrispondenze:

1. punto – punto
2. punto –linea
3. linea –linea

Per ogni immagine si introducono le possibili corrispondenze, considerando che anche i punti all'infinito ricavati in precedenza costituiscono dei vincoli punto – punto.
La corrispondenza punto – punto introduce tre vincoli sulla matrice di proiezione relativa alla vista, come facilmente si può evincere dalla relazione:

$$\lambda w = P X \qquad (2.5.1)$$



a b

Figura 2.4 *Vincoli ricavati dalla pianta: (a) (b) esempio di corrispondenze linea-linea individuate tra un'immagine e la pianta.*

La corrispondenza punto linea è invece esprimibile tramite la seguente relazione:

$$L \cdot P_{4x4}^{-1} \cdot \begin{bmatrix} \lambda w \\ 1 \end{bmatrix} = 0_{3x1} \qquad (2.5.2)$$

dove L è una matrice 3x4 contenente i coefficienti dell'equazione della retta 3D, e P_{4x4} è la matrice P con l'aggiunta finale della riga [0 0 0 1].
Infine, la corrispondenza linea – linea è esprimibile tramite la relazione:

$w_1 \wedge w_2 = (P X_3) \wedge (P X_4)$ (2.5.3)

dove w_1 e w_2 sono due generici punti della retta dell'immagine, mentre X_3 e X_4 sono due generici punti della retta sulla pianta.

2.6 Rettificazione.

L'omografia ricavata dalle strutture planari di riferimento può essere utilizzata per la rettificazione delle immagini. La trattazione analitica è stata già trattata nella parte teorica alla quale si rimanda.

Figura 2.5 *Rettificazione basata su un piano di riferimento: (a) la facciata presente nell'immagine costituisce la struttura planare di riferimento adoperata per la rettificazione; (b) risultato dell'operazione di rettificazione, con il piano dell'immagine parallelo alla struttura planare di riferimento.*

Nella figura 2.5 è mostrato il risultato dell'operazione di rettificazione su una delle immagini della sequenza presa in esame. Avendo nella nuova immagine la struttura planare parallela al piano dell'immagine è possibile estrarre la tessitura della facciata per collocarla direttamente nel modello tridimensionale della ricostruzione.
Le altre strutture presenti nell'immagine e che tramite i punti ricostruiti completeranno il modello, saranno in maniera analoga rettificati per estrarne la tessitura.

2.7 Ricostruzione 3D.

Tutti i vincoli ricavati dalla pianta, le nuove corrispondenze trovate tramite l'utilizzo delle omografie confluiscono nel processo finale di ottimizzazione della ricostruzione desiderata. Tale processo tenta di minimizzare in maniera iterativa l'errore di riproiezione, ossia la differenza tra le immagini della sequenza e quelle ottenute dal modello considerando i parametri proiettivi stimati. Anche utilizzando solo le tre coppie di immagini mostrate nell'esempio proposto in questa trattazione, si ottengono risultati di ricostruzione soddisfacente. L'errore medio di riproiezione rilevato per questa ricostruzione e per le altre sperimentate è stato all'incirca di 1,4 pixel. Tale errore è da considerarsi buono, anche perché non incide in maniera significativa sulla precisione del modello tridimensionale della scena ottenuto. A partire dai punti tridimensionali stimati tramite triangolazione dalle corrispondenze ottenute in maniera automatica, è possibile raffinare il modello in parti ritenute interessanti (particolari architettonici di rilievo). A tal fine si potrebbero utilizzare tecniche di "space carving" , o di ricostruzione a partire da insiemi densi di punti.
Un'ulteriore estensione sarebbe di affrontare l'estrazione dei parametri di illuminazione della scena per ottenere un modello ancora più realistico e visibile sotto condizioni di illuminazioni differenti da quelle originali.

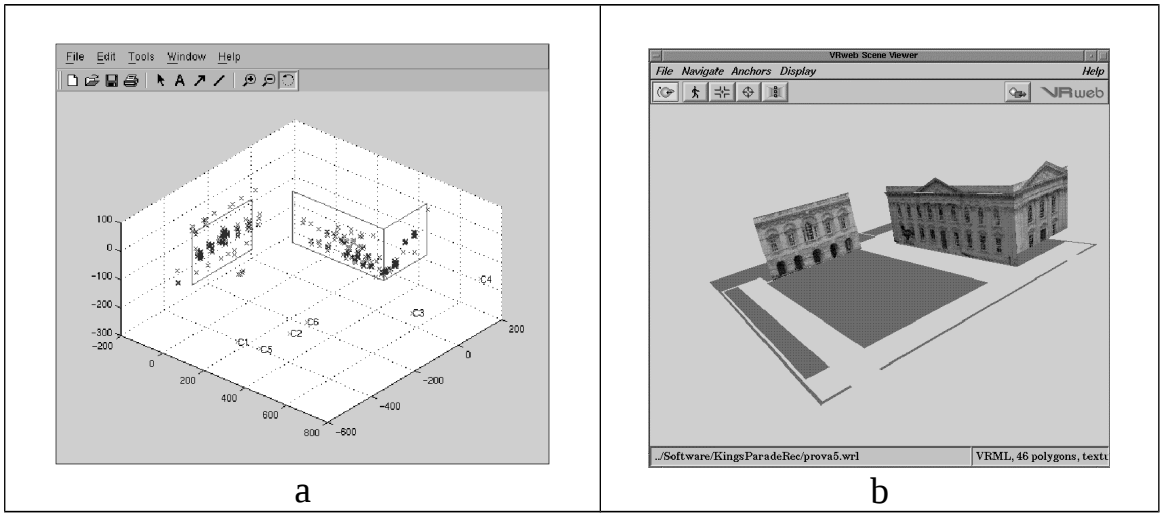


Figura 2.6 *Ricostruzione dei piani di riferimento: (a) punti e piani di riferimento ricostruiti; (b) ricostruzione con la tessitura e posizionamento sulla pianta.*

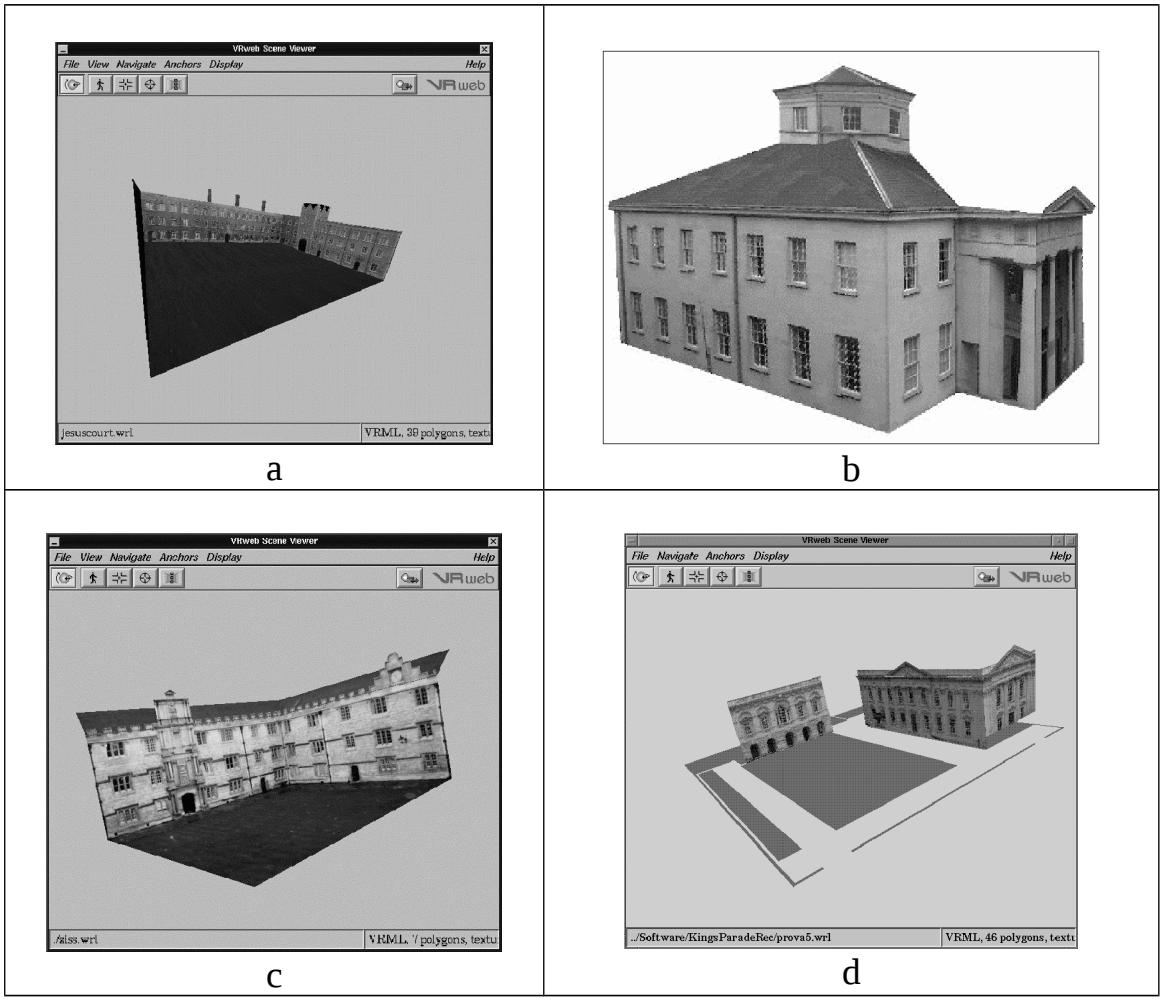


Figura 2.7 *Esempi di ricostruzioni in formato VRML*

Figura 2.8 *Particolare del modello VRML del Senate House, Cambridge (UK).*

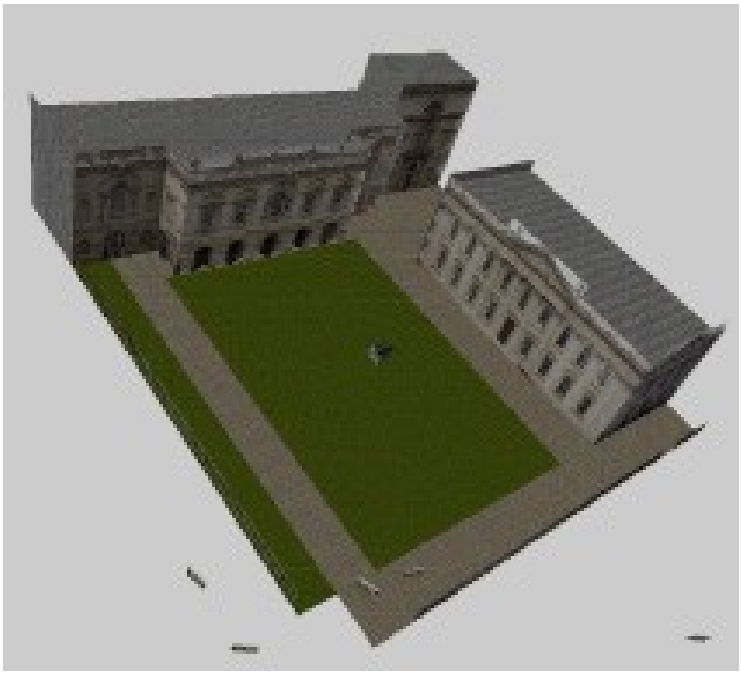


Figura 2.9 *Particolare del modello VRML del Senate House, Cambridge (UK).*

Figura 2.10 *Particolare del modello VRML del Senate House, Cambridge (UK).*

Parte III

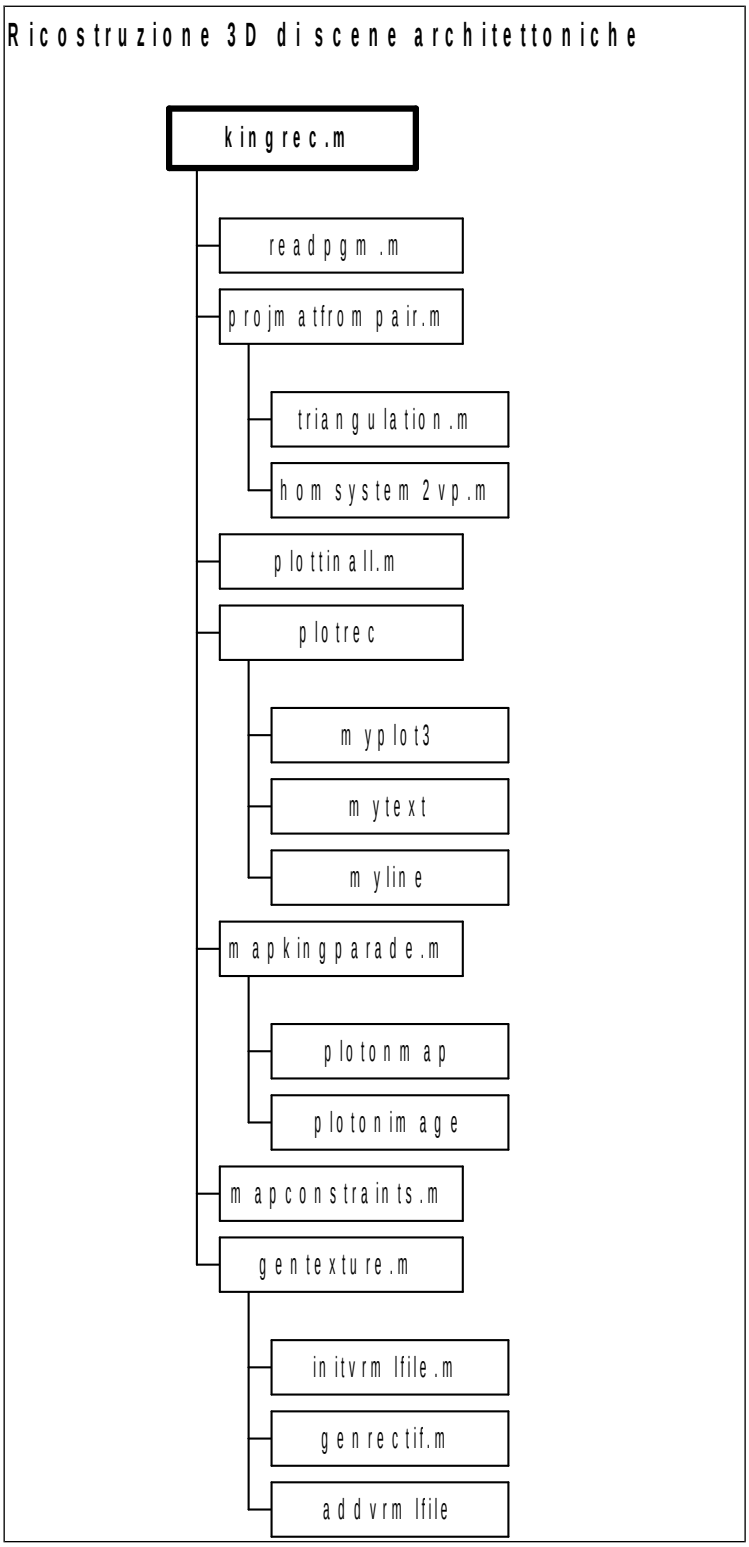
Implementazione in Matlab

3.1 Introduzione.

Nelle pagine seguenti sono riportate le funzioni matlab implementate per le sperimentazioni sulla ricostruzione delle scene architettoniche. Tali sorgenti sono stati anche convertiti in un listato in linguaggio C sfruttando gli appositi strumenti forniti dal Matlab. Dai sorgenti C si è quindi ottenuto del codice eseguibile indipendente dall'interprete Matlab e con prestazioni computazionali notevolmente incrementate.

3.2 Albero delle procedure.

E’ riportarto lo schema delle chiamate delle varie funzioni implementate.



File kingrec.m

```
%King's Parade reconstruction
%image sixe 562 x 450

w0=[281;225]; u0=w0(1); v0=w0(2);

%-------------------------------------------------------------------------
-
% Pair img1-img2
%-------------------------------------------------------------------------
-

img1=readPGM('facadeA1.pgm'); figimg1=1;
vp1_1=[-1.7328e+03;259.1731;1]; vp2_1=[292.7971;-7.8315e+03;1];
wa1=[126;142;1]; wb1=[504;119;1]; d1=440;
%!tjdat2txt facadeA1_200.tjdat > corFacA1_200.dat
filecorn1='corFacA1_200.dat';

img2=readPGM('facadeA3.pgm'); figimg2=2;
vp1_2 = [-991.0428;252.7856;1]; vp2_2 = [421.9412;-4.5335e+03;1];
% terrace
wa2=[66;136;1]; wb2=[500;89;1]; d2=440;
%!tjdat2txt facadeA3_200.tjdat > corFacA3_200.dat
filecorn2='corFacA3_200.dat';

%-----------------------------------------------------------

[K1,R1,t1,H1,Xca1,cp1, K2,R2,t2,H2,Xca2,cp2, ....
          H21,corr,ncorr,Xrec,lsplane,H21new,R21new,t21new,E12] ....
= projmatfrompair(w0,vp1_1,vp2_1,img1,wa1,wb1,d1,filecorn1, ....
                     vp1_2,vp2_2,img2,wa2,wb2,d2,filecorn2);

%-----------------------------------------------------------

plottingall(img1,figimg1,wa1,wb1,cp1,corr,ncorr);
plottingall(img2,figimg2,wa2,wb2,cp2,corr(:,3:4),ncorr);

%-----------------------------------------------------------
figrec=3;
plane=[0 0 0; 440 0 0; 440 180 0; 0 180 0;0 0 0];
Ra1_a1=[1 0 0; 0 1 0; 0 0 1];
ta1_a1=[0;0;0];
plotrec(figrec,Xrec,Ra1_a1,ta1_a1,Xca1,'C1',Xca2,'C2',plane,lsplane,1)

%-----------------------------------------------------------


%-------------------------------------------------------------------------
-
% Pair img3-img4
%-------------------------------------------------------------------------
-

img3=readPGM('facadeB1.pgm'); figimg3=4;
vp1_3=[1.0724e+03;365.7302;1]; vp2_3=[351.7184; -2.5200e+03;1];
wa3=[180;67;1];wb3=[415;144;1]; d3=200;
%!tjdat2txt facadeB1_200.tjdat > corFacB1_200.dat
filecorn3='corFacB1_200.dat';
```

```
img4=readPGM('facadeB2.pgm'); figimg4=5;
vp1_4=[-2.9504e+03;66.7313;1]; vp2_4=[507.4475;-2.3771e+03;1];
wa4=[167;114;1]; wb4=[493;119;1]; d4=200;
%!tjdat2txt facadeB2_200.tjdat > corFacB2_200.dat
filecorn4='corFacB2_200.dat';

%-------------------------------------------------------------

[K3,R3,t3,H3,Xca3,cp3, K4,R4,t4,H4,Xca4,cp4, ....

H43,corr43,ncorr43,Xrec43,lsplane43,H43new,R43new,t43new,E43] ....
= projmatfrompair(w0,vp1_3,vp2_3,img3,wa3,wb3,d3,filecorn3, ....
                     vp1_4,vp2_4,img4,wa4,wb4,d4,filecorn4);

%-------------------------------------------------------------

plottingall(img3,figimg3,wa3,wb3,cp3,corr43,ncorr43);
plottingall(img4,figimg4,wa4,wb4,cp4,corr43(:,3:4),ncorr43);


%-------------------------------------------------------------

plane43=[440 0 0; 440 0 200; 440 180 200; 440 180 0;440 0 0];
Ra1_a3=[0 0 -1;0 1 0;1 0 0];
ta1_a3=[440;0;0];
plotrec(figrec,Xrec43,Ra1_a3,ta1_a3,Xca3,'C3',Xca4,'C4',plane43,lsplan
e43,0);

%-------------------------------------------------------------

%-------------------------------------------------------------------
-
% Pair img5-img6
%-------------------------------------------------------------------
-

img5=readPGM('facadeC1.pgm'); figimg5=6;
vp1_5=[1.1720e+03;286.7446;1]; vp2_5=[ 227.9593;-5.8039e+03;1];
wa5=[94;107;1]; wb5=[355;150;1]; d5=300;

%!tjdat2txt facadeC1_300.tjdat > corFacC1_300.dat
filecorn5='corFacC1_300.dat';

img6=readPGM('facadeC3.pgm'); figimg6=7;
vp1_6=[2.1153e+03;312.7557;1]; vp2_6=[363.6234;-5.1812e+03;1];
wa6=[109;130;1]; wb6=[392;155;1]; d6=300;
%!tjdat2txt facadeC3_300.tjdat > corFacC3_200.dat
filecorn6='corFacC3_300.dat';

%-------------------------------------------------------------

[K5,R5,t5,H5,Xca5,cp5, K6,R6,t6,H6,Xca6,cp6, ....

H65,corr65,ncorr65,Xrec65,lsplane65,H65new,R65new,t65new,E65] ....
= projmatfrompair(w0,vp1_5,vp2_5,img5,wa5,wb5,d5,filecorn5, ....
                     vp1_6,vp2_6,img6,wa6,wb6,d6,filecorn6);

%-------------------------------------------------------------

plottingall(img5,figimg5,wa5,wb5,cp5,corr65,ncorr65);
plottingall(img6,figimg6,wa6,wb6,cp6,corr65(:,3:4),ncorr65);
```

```
%-------------------------------------------------------------

plane65=[-80 0 -385; -80 0 -110;-80 180 -110;-80 180 -385;-80 0 -385];
Ra1_a5=[0 0 -1;0 1 0;1 0 0];
ta1_a5=[-80;0;-400];
plotrec(figrec,Xrec65,Ra1_a5,ta1_a5,Xca5,'C5',Xca6,'C6',plane65,lsplan
e65,0);

%-------------------------------------------------------------

%gentexture;
%rectovrml;
figmap=8;
mapKingParade;
%recimprov;
mapconstraints;

figure(1); colormap(gray(255)); image(img1); hold on; zoom on;
figure(2); colormap(gray(255)); image(img2); hold on; zoom on;
figure(4); colormap(gray(255)); image(img3); hold on; zoom on;
figure(5); colormap(gray(255)); image(img4); hold on; zoom on;
figure(6); colormap(gray(255)); image(img5); hold on; zoom on;
figure(7); colormap(gray(255)); image(img6); hold on; zoom on;
```

File readpgm.m

```
function picture = readPGM(filename);

% picture = readPGM(filename)
%
% This function reads a .pgm file and returns a matrix with the
% values of the gray levels.
%
% filename - name of the .pgm file;
% picture - matrix of gray levels.

fid = fopen(filename);
PGM_type = fscanf(fid, '%c', 2);
if (PGM_type ~= 'P5')
      error('*** error in PGM file! ***');
else
      dummy = fscanf(fid, '%c', 1);
      comment = fscanf(fid, '%c', 1);
end
if (comment == '#')
      while (double(comment) ~= 10)
            comment = fscanf(fid, '%c', 1);
      end
      comment = fscanf(fid, '%c', 1);
end


max_x = 0;
aux = double(comment);
while (aux ~= 32)
      max_x = 10*max_x + str2num(comment);
      comment = fscanf(fid, '%c', 1);
      aux = double(comment);
end
max_y = fscanf(fid, '%d', 1);
```

```
max_color = fscanf(fid, '%d');

picture = reshape(fread(fid), max_x, max_y)';
fclose(fid);

return
```

File projmatfrompair.m

```
function [K1,R1,t1,H1,Xca1,cp1, K2,R2,t2,H2,Xca2,cp2, ....
          H21,corr,ncorr,Xrec,lsplane,H21new,R21_new,t21_new,E12] ....
= projmatfrompair (w0,vp1_1,vp2_1,img1,wa1,wb1,d1,filecorn1, ....
                       vp1_2,vp2_2,img2,wa2,wb2,d2,filecorn2);

u0=w0(1); v0=w0(2);

[K1,R1,t1,f1,vp3_1]=homsystem2vp(u0,v0,vp1_1(1),vp1_1(2), ....
              vp2_1(1),vp2_1(2), wa1,wb1,d1);

Xca1=-inv(R1)*t1;
H1=K1*[R1(1:3,1) R1(1:3,2) t1];

disp('calibration 1st image ... done');

disp('loading corners 1st image');
%-------------------------------------------------
% loading corners img1

fid=fopen(filecorn1,'r');
 cp1temp=fscanf(fid,'%f');
fclose(fid);

nc1=max(size(cp1temp))/3;
cp1=reshape(cp1temp,3,nc1)';

%-----------------------------------------------------------


disp('calibration 2st image ... done');

[K2,R2,t2,f2,vp3_2]=homsystem2vp(u0,v0,vp1_2(1),vp1_2(2), ....
              vp2_2(1),vp2_2(2), wa2,wb2,d2);

Xca2=-inv(R2)*t2;
H2=K2*[R2(1:3,1) R2(1:3,2) t2];

%-------------------------------------------------------------------
% loading corners img2

disp('loading corners 1st image');

fid=fopen(filecorn2,'r');
 cp2temp=fscanf(fid,'%f');
fclose(fid);

nc2=max(size(cp2temp))/3;
cp2=reshape(cp2temp,3,nc2)';


%-----------------------------------------------------------
```

```
% Finding correspondences using homography of the two images


disp('finding correspondences by homography');

H21=H2*inv(H1);

dmax=(2*u0)^2+(2*v0)^2;

for i=1:nc1,
 cp1(i,3)=0;
 cp1(i,4)=dmax;
 w1=[cp1(i,1); cp1(i,2);1];
 w2=[(H21(1,1:3)*w1)/(H21(3,1:3)*w1);(H21(2,1:3)*w1)/
(H21(3,1:3)*w1);1];
 for j=1:nc2,
  w2temp=[cp2(j,1); cp2(j,2);1];
  dtemp=(w2temp-w2)'*(w2temp-w2);
  if dtemp<cp1(i,4), cp1(i,4)=dtemp; cp1(i,3)=j; end;
 end;
end;

k=0; corr=0;
for i=1:nc1,
 if  cp1(i,4)<16,
   k=k+1;
   corr(k,1:4)=[cp1(i,1),cp1(i,2),cp2(cp1(i,3),1),cp2(cp1(i,3),2)];
 end;
end;
ncorr=k;

%-------------------------------------------------------------
%triangulation

Xrec=0;
Xrec=triangulation(K1,R1,t1,K2,R2,t2,corr(:,1:2),corr(:,3:4));

%-------------------------------------------------------------
%finding least-square solution plane

[Up Dp Vp]=svd([Xrec,ones(ncorr,1)]);
a=Vp(1,3); b=Vp(2,3); c=Vp(3,3); d=Vp(4,3);

if c<0, a=-a; b=-b; c=-c; d=-d; end;

lsplane=[a;b;c;d];

%-------------------------------------------------------------
% H21 improvement
% inv(K2)*H21*K1


disp('homography matrix new estimation');

H21sys=zeros(3*ncorr,9+ncorr);

for i=1:ncorr,
 j=(i-1)*3+1;
 w1=[corr(i,1:2) ,1]';
 w2=[corr(i,3:4) ,1]';
 H21sys(j:j+2,1:9)=[  w1'  0 0 0  0 0 0; ....
                   0 0 0    w1'  0 0 0; ....
```

```
                    0 0 0  0 0 0    w1'];
 H21sys(j:j+2,9+i)=-w2;
end;

[Uh Dh Vh]=svd(H21sys);
x=Vh(1:9,9+ncorr);
H21new=reshape(x',3,3)'; %proportional of H21

%-------------------------------------------------------------
% (d*R+t*n') scomposition   % t proporzionale
% t21=t21_npo*no'*inv(K1)*w1/d


disp('homography scomposition');


[Uho Dho Vho]=svd(inv(K2)*H21*K1);
s1o=Dho(1,1); s2o=Dho(2,2); s3o=Dho(3,3);

if ((s1o~=s2o) & (s2o~=s3o)),
 x1o=sqrt((s1o^2-s2o^2)/(s1o^2-s3o^2)); % x2o=0;
 x3o=sqrt((s2o^2-s3o^2)/(s1o^2-s3o^2));
end;

t21_pp=Uho*(s1o-s3o)*[x1o;0;-x3o];
t21_pn=Uho*(s1o-s3o)*[x1o;0;x3o];
t21_np=Uho*(s1o-s3o)*[-x1o;0;-x3o];
t21_nn=Uho*(s1o-s3o)*[-x1o;0;+x3o];

t21=t2-R2*inv(R1)*t1;
t21norm=t21./t21(1);
t21_pp_norm=t21_pp./t21_pp(3);
t21_pn_norm=t21_pn./t21_pn(3);
t21_np_norm=t21_np./t21_np(3);
t21_nn_norm=t21_nn./t21_nn(3);

d_pp=(t21norm-t21_pp_norm)'*(t21norm-t21_pp_norm);
d_pn=(t21norm-t21_pn_norm)'*(t21norm-t21_pn_norm);
d_np=(t21norm-t21_np_norm)'*(t21norm-t21_np_norm);
d_nn=(t21norm-t21_nn_norm)'*(t21norm-t21_nn_norm);
[mind i]=min([d_pp d_pn d_np d_nn]);

if i==1, s_x1o=1; s_x3o=1; t21_new=t21_pp_norm;
 elseif i==2, s_x1o=1; s_x3o=-1; t21_new=t21_pn_norm;
 elseif i==3,  s_x1o=-1; s_x3o=1; t21_new=t21_np_norm;
 else s_x1o=-1; s_x3o=-1; t21_new=t21_nn_norm;
end;

sin_po=(s_x1o*s_x3o)*sqrt((s1o^2-s2o^2)*(s2o^2-s3o^2))/
((s1o+s3o)*s2o);
cos_po=(s2o^2+s3o*s1o)/((s1o+s3o)*s2o);

Rpo=[cos_po 0 -sin_po; 0 1 0;sin_po 0 cos_po];
so=det(Uho)*det(Vho');
R21_new=so*Uho*Rpo*Vho';
% no=Vho*[s_x1o*x1o;0;s_x3o*x3o];


%-------------------------------------------------------------
% Essential matrix and reconstruction
```

```
disp('essential matrix estimation');

nxm=size(img1);

R12=R21_new; t12=t21_new;;

T12x=[   0    -t12(3)  t12(2); ....
       t12(3)    0    -t12(1); ....
      -t12(2)  t12(1)    0  ];
E12=T12x*R12;
```

File homsystem2vp.m

```
function
[K_est,R_est,t_est,f_est,vp3]=homsystem2vp(u0,v0,u1,v1,u2,v2,wa,wb,d);

f_est= sqrt((u0-u1)*(u2-u0)+(v0-v1)*(v2-v0));
K_est=[f_est 0 u0;0 f_est v0; 0  0 1];

u10=u1-u0;u20=u2-u0;
v10=v1-v0;v20=v2-v0;

den=(v20*u10)-(v10*u20);

u3=(v10*(v1*v20+u1*u20)-v20*(v2*v10+u2*u10))/(-den);
v3=(u10*(u1*u20+v1*v20)-u20*(u2*u10+v2*v10))/den;

vp3=[u3;v3];

sysA=[ u1  u2  u3; ....
       v1  v2  v3; ....
        1   1   1];

b=[u0;v0;1];

[U D V]=svd(sysA);

D1=0*D';
for i=1:3 if D(i,i)~=0 D1(i,i)=1/D(i,i); end; end;
x=V*D1*U'*b;

l1=sqrt(x(1));
l2=sqrt(x(2));
l3=sqrt(x(3));

if u1<u0, l1=-l1; end;
if v2<v0, l2=-l2; end;

R_est=inv(K_est)*[l1*u1 l2*u2 l3*u3; l1*v1 l2*v2 l3*v3; l1 l2 l3];

if wb(1)<wa(1), d=-d; end;
Xb_a=[d;0;0];

%----------------
%NOTE: t_est=t+Xa

KRX=K_est*R_est*Xb_a;
if wa(1)~=wb(1),
 la=(wb(1)*KRX(3)-KRX(1))/(wa(1)-wb(1));
else
 la=(wb(2)*KRX(3)-KRX(2))/(wa(2)-wb(2));
```

```
end;
t_est=la*inv(K_est)*wa;
```

File triangulation.m

```
function [Xrec]= triangulation(K1,R1,t1,K2,R2,t2,W1,W2);

Wsize=size(W1);
npoints=Wsize(1);

bsys=[-K1*t1;-K2*t2];

for i=1:npoints,
 Asys=[K1*R1 -[W1(i,1:2),1]' zeros(3,1); ....
       K2*R2 zeros(3,1)  -[W2(i,1:2),1]'];
 [U D V]=svd(Asys,0);
 D1=0*D';
 for j=1:5 if D(j,j)~=0, D1(j,j)=1/D(j,j); end; end;
 x=V*D1*U'*bsys;
 Xrec(i,1:3)=[x(1),x(2),x(3)];
end;


% finding exact d and true Xrecs

%A1=zeros(3*npoints,3*npoints);
%A2=A1;
%for i=1:npoints,
% ii=1+npoints*(i-1);
% A1(ii:ii+2,ii:ii+2)=R1;
% A2(ii:ii+2,ii:ii+2)=R2;
% Aa1(ii:ii+2,1)=-t1;
% Aa2(ii:ii+2,1)=-t2;
% bsys(ii:ii+2,1)=R1*Xrec(i,1:3)';
% bsys(ii+3*npoints:ii+3*npoints+2,1)=R1*Xrec(i,1:3)';
%end;
%
%Asys=[A1 Aa1; A2 Aa2];
%
%[U D V]=svd(Asys);
%cond(Asys)
%[U D V]=svd(Asys,0);
%D1=0*D';
%for j=1:3*npoints if D(j,j)~=0, D1(j,j)=1/D(j,j); end; end;
%x=V*D1*U'*bsys;
%
%for i=1:npoints
% ii=1+3*(npoints-1);
% Xrec(i,1:3)=x(ii:ii+2,1)';
%end;
```

File plottingall.m

```
function plottingall(img1,fig1,wa1,wb1,cp1,corr,ncorr)
```

```
figure(fig1); colormap(gray(255));
image(img1); zoom on; hold on;


W1=[wa1(1:2,1)';wb1(1:2,1)'];
%hl=line(W1(:,1),W1(:,2));
%set(hl,'Color','Red');

%plot(cp1(:,1), cp1(:,2), 'xw');
%plot(cp1(:,1)+1, cp1(:,2)+1, 'xw');

plot(corr(:,1), corr(:,2), 'xg');
%plot(corr(:,1)+1, corr(:,2)+1, 'xk');

for i=1:ncorr,
 ht=text(corr(i,1)+1,corr(i,2)+1,sprintf('%d',i));
 set(ht,'Color','Red');
end;
```

File plotrec.m

```
function
plotrec(fig3,Xrec,Ra1_a3,ta1_a3,Xca1,lab1,Xca2,lab2,plane,lsplane,flag
)

ncorrtemp=size(Xrec);
ncorr=ncorrtemp(1);

for i=1:ncorr,
 Xrecnew(i,1:3)=(Ra1_a3*Xrec(i,1:3)'+ta1_a3)';
end;

figure(fig3);

if flag>0, hold on; rotate3d; grid on; end;

myplot3(Xrecnew(:,1),Xrecnew(:,2),Xrecnew(:,3),'xb');

Xca1new=(Ra1_a3*Xca1+ta1_a3);
Xca2new=(Ra1_a3*Xca2+ta1_a3);

myplot3(Xca1new(1),Xca1new(2),Xca1new(3),'xr');
mytext(Xca1new(1)+1,Xca1new(2)+1,Xca1new(3)+1,lab1)
myplot3(Xca2new(1),Xca2new(2),Xca2new(3),'xg');
mytext(Xca2new(1)+1,Xca2new(2)+1,Xca2new(3)+1,lab2);


hl=myline(plane(:,1), plane(:,2), plane(:,3));
set(hl,'color','Red');

a=lsplane(1); b=lsplane(2); c=lsplane(3); d=lsplane(4);

z1=-(a*plane(1,1)+b*plane(1,2)+d)/c;
z2=-(a*plane(2,1)+b*plane(2,2)+d)/c;
z3=-(a*plane(3,1)+b*plane(3,2)+d)/c;
z4=-(a*plane(4,1)+b*plane(4,2)+d)/c;
```

File myplot3.m

```
function myplot3(X,Y,Z,string);

plot3(X,Z,-Y,string);
```

File mytext.m

```
function mytext(X,Y,Z,string);

text(X,Z,-Y,string);
```

File myline.m

```
function [hl]=myline(X,Y,Z);

hl=line(X,Z,-Y);
```

File mapkingparade.m

```
figmap=8;

R_I=[1 0 0; 0 1 0; 0 0 1]; t_I=[0;0;0];

building1=[0 0 0; 440 0 0; 440 0 210; 0 0 210; 0 0 0];
R_build1_O=[1 0 0;0 1 0;0 0 1]; t_build1_O=[0;0;0];
bu1_0=plotonmap(figmap,building1,R_build1_O,t_build1_O,0,0,'Blue','bui
ld1');

building2=[0 0 0; 300 0 0; 300 0 60; 420 0 60; 420 0 200;
           -120 0 200; -120 0 60; 0 0 60; 0 0 0];
R_build2_O=[0 0 -1;0 1 0; 1 0 0]; t_build2_O=[-80;0;-400];
bu2_0=plotonmap(figmap,building2,R_build2_O,t_build2_O,1,0,'Blue','bui
ld2');

building3=[0 0 0; 180 0 0; 180 0 100; 0 0 100; 0 0 0];
R_build3_O=[0 0 -1;0 1 0; 1 0 0]; t_build3_O=[-180;0;25];
bu3_0=plotonmap(figmap,building3,R_build3_O,t_build3_O,1,0,'Blue','bui
ld3');

green1=[0 0 0;510 0 0;510 0 340;0 0 340;0 0 0];
R_green1_O=[1 0 0;0 1 0; 0 0 1]; t_green1_O=[-70;0;-415];
gr1_0=plotonmap(figmap,green1,R_green1_O,t_green1_O,1,0,'Green','green
1');

green2=[0 0 0;570 0 0;570 0 40;0 0 40;0 0 0];
R_green2_O=[1 0 0;0 1 0; 0 0 1]; t_green2_O=[-130;0;-510];
gr2_0=plotonmap(figmap,green2,R_green2_O,t_green2_O,1,0,'Green','green
2');

gates1=[0 0 0;650 0 0;650  0 50; 645 0 50; 645 0 5;0 0 5;0 0 0];
R_gates1_O=[1 0 0;0 1 0; 0 0 1]; t_gates1_O=[-140;0;-520];
ga1_0=plotonmap(figmap,gates1,R_gates1_O,t_gates1_O,1,0,'Black','');

gates2=[0 0 0;5 0 0;5 0 500;0 0 500;0 0 0];
R_gates2_O=[1 0 0;0 1 0; 0 0 1]; t_gates2_O=[505;0;-420];
ga2_0=plotonmap(figmap,gates2,R_gates2_O,t_gates2_O,1,0,'Black','');

gates3=[0 0 0;5 0 0;5  0 90;-65 0 90; -65 0 85;0 0 85;0 0 0];
R_gates3_O=[1 0 0;0 1 0; 0 0 1]; t_gates3_O=[505;0;120];
```

```
ga3_0=plotonmap(figmap,gates3,R_gates3_O,t_gates3_O,1,0,'Black','');

plot3(Xca1(1),Xca1(3),0,'bx'); text(Xca1(1)+10,Xca1(3)+10,0,'C1');
plot3(Xca2(1),Xca2(3),0,'bx'); text(Xca2(1)+10,Xca2(3)+10,0,'C2');
Xca3_a1=Ra1_a3*Xca3+ta1_a3;plot3(Xca3_a1(1),Xca3_a1(3),0,'rx');
text(Xca3_a1(1)+10,Xca3_a1(3)+10,0,'C3');
Xca4_a1=Ra1_a3*Xca4+ta1_a3;plot3(Xca4_a1(1),Xca4_a1(3),0,'rx');
text(Xca4_a1(1)+10,Xca4_a1(3)+10,0,'C4');
Xca5_a1=Ra1_a5*Xca5+ta1_a5;plot3(Xca5_a1(1),Xca5_a1(3),0,'gx');
text(Xca5_a1(1)+10,Xca5_a1(3)+10,0,'C5');
Xca6_a1=Ra1_a5*Xca6+ta1_a5;plot3(Xca6_a1(1),Xca6_a1(3),0,'gx');
text(Xca6_a1(1)+10,Xca6_a1(3)+10,0,'C6');

axis([-300 800 -800 300 0 200]);

%reprojection map building on images using recovered h.

h_build1=180;
plotonimage(figimg1,K1,R1,t1,Xca1,Ra1_a1,ta1_a1,bu1_0,h_build1,'Blue')
;
plotonimage(figimg2,K2,R2,t2,Xca2,Ra1_a1,ta1_a1,bu1_0,h_build1,'Blue')
;

plotonimage(figimg3,K3,R3,t3,Xca3,Ra1_a3,ta1_a3,bu1_0,h_build1,'Blue')
;
plotonimage(figimg4,K4,R4,t4,Xca4,Ra1_a3,ta1_a3,bu1_0,h_build1,'Blue')
;

plotonimage(figimg5,K5,R5,t5,Xca5,Ra1_a5,ta1_a5,bu1_0,h_build1,'Blue')
;

h_build2=180;
plotonimage(figimg1,K1,R1,t1,Xca1,Ra1_a1,ta1_a1,bu2_0,h_build2,'Red');

plotonimage(figimg4,K4,R4,t4,Xca4,Ra1_a3,ta1_a3,bu2_0,h_build2,'Red');

plotonimage(figimg5,K5,R5,t5,Xca5,Ra1_a5,ta1_a5,bu2_0,h_build2,'Red');
plotonimage(figimg6,K6,R6,t6,Xca6,Ra1_a5,ta1_a5,bu2_0,h_build2,'Red');

%t_build3_O=[-180;-50;30];
h_build3=230;
%h_build3=180;
bu3h_0=bu3_0;
bu3h_0(:,2)=bu3_0(:,2)-(h_build3-h_build1);
plotonimage(figimg1,K1,R1,t1,Xca1,Ra1_a1,ta1_a1,bu3h_0,h_build3,'Yello
w');
plotonimage(figimg2,K2,R2,t2,Xca2,Ra1_a1,ta1_a1,bu3h_0,h_build3,'Yello
w');

plotonimage(figimg3,K3,R3,t3,Xca3,Ra1_a3,ta1_a3,bu3h_0,h_build3,'Yello
w');
plotonimage(figimg4,K4,R4,t4,Xca4,Ra1_a3,ta1_a3,bu3h_0,h_build3,'Yello
w');

plotonimage(figimg5,K5,R5,t5,Xca5,Ra1_a5,ta1_a5,bu3h_0,h_build3,'Yello
w');
plotonimage(figimg6,K6,R6,t6,Xca6,Ra1_a5,ta1_a5,bu3h_0,h_build3,'Yello
w');


h_green1=0;
gr1h_0=gr1_0;
```

```
gr1h_0(:,2)=gr1_0(:,2)-(h_green1-h_build1);
%R_green1_O=[1 0 0;0 1 0; 0 0 1]; t_green1_O=[-20;0;-420];

plotonimage(figimg1,K1,R1,t1,Xca1,Ra1_a1,ta1_a1,gr1h_0,h_green1,'Green
');
plotonimage(figimg2,K2,R2,t2,Xca2,Ra1_a1,ta1_a1,gr1h_0,h_green1,'Green
');

plotonimage(figimg3,K3,R3,t3,Xca3,Ra1_a3,ta1_a3,gr1h_0,h_green1,'Green
');
plotonimage(figimg4,K4,R4,t4,Xca4,Ra1_a3,ta1_a3,gr1h_0,h_green1,'Green
');

plotonimage(figimg5,K5,R5,t5,Xca5,Ra1_a5,ta1_a5,gr1h_0,h_green1,'Green
');
plotonimage(figimg6,K6,R6,t6,Xca6,Ra1_a5,ta1_a5,gr1h_0,h_green1,'Green
');

h_green2=0;
gr2h_0=gr2_0;
gr2h_0(:,2)=gr2_0(:,2)-(h_green2-h_build1);

plotonimage(figimg1,K1,R1,t1,Xca1,Ra1_a1,ta1_a1,gr2h_0,h_green2,'Green
');
plotonimage(figimg5,K5,R5,t5,Xca5,Ra1_a5,ta1_a5,gr2h_0,h_green2,'Green
');
%plotonimage(figimg6,K6,R6,t6,Xca6,Ra1_a5,ta1_a5,gr2h_0,h_green2,'Gree
n');


h_gates1=25;
ga1h_0=ga1_0;
ga1h_0(:,2)=ga1_0(:,2)-(h_gates1-h_build1);
plotonimage(figimg1,K1,R1,t1,Xca1,Ra1_a1,ta1_a1,ga1h_0,h_gates1,'White
');
ga1h2_0=[ga1h_0(1,:);ga1h_0(2,:);[510 150
-515];ga1h_0(6,:);ga1h_0(1,:)];
plotonimage(figimg5,K5,R5,t5,Xca5,Ra1_a5,ta1_a5,ga1h2_0,h_gates1,'Whit
e');

h_gates2=25;
ga2h_0=ga2_0;
ga2h_0(:,2)=ga2_0(:,2)-(h_gates2-h_build1);
plotonimage(figimg1,K1,R1,t1,Xca1,Ra1_a1,ta1_a1,ga2h_0,h_gates2,'White
');
plotonimage(figimg4,K4,R4,t4,Xca4,Ra1_a3,ta1_a3,ga2h_0,h_gates2,'White
');
ga2h2_0=ga2h_0;ga2h2_0(1,3)=-220;ga2h2_0(2,3)=-220;ga2h2_0(2,5)=-220;
plotonimage(figimg3,K3,R3,t3,Xca3,Ra1_a3,ta1_a3,ga2h2_0,h_gates2,'Whit
e');

h_gates3=25;
ga3h_0=ga3_0;
ga3h_0(:,2)=ga3_0(:,2)-(h_gates3-h_build1);
plotonimage(figimg1,K1,R1,t1,Xca1,Ra1_a1,ta1_a1,ga3h_0,h_gates3,'White
');
plotonimage(figimg2,K2,R2,t2,Xca2,Ra1_a1,ta1_a1,ga3h_0,h_gates3,'White
');
plotonimage(figimg3,K3,R3,t3,Xca3,Ra1_a3,ta1_a3,ga3h_0,h_gates3,'White
');
plotonimage(figimg4,K4,R4,t4,Xca4,Ra1_a3,ta1_a3,ga3h_0,h_gates3,'White
');
```

File plotonmap.m

```
function [object2] =
plotonmap(fig,object,R,t,flagfig,flagloop,linecol,label);


if flagfig==0, figure(fig); hold on; rotate3d; end;

nxm=size(object);

npoints=nxm(1)-1;


for i=1:npoints+1,
 object2(i,1:3)=(R*object(i,1:3)'+t)';
end;


if flagloop==0,
 hl=myline(object2(:,1),object2(:,2),object2(:,3));
 set(hl,'Color',linecol);
 x_text=sum(object2(1:npoints,1))/npoints;
 y_text=sum(object2(1:npoints,2))/npoints;
 z_text=sum(object2(1:npoints,3))/npoints;
 mytext(x_text,y_text,z_text,label);
else
 for i=2:npoints,
  hl=myline(object2(i-1:i,1),object2(i-1:i,2),object2(i-1:i,3));
  set(hl,'Color',linecol);
 end;
 mytext(object2(1,1)+2,object2(1,2)+2,object2(1,3),label);
end;
```

File plotonimage.m

```
function plotonimage(figimg,K1,R1,t1,Xca1,Ra1_a1,ta1_a1, ....
                     building1,h_build1,line_color);


figure(figimg);

nxm=size(building1);
npoints=nxm(1)-1;

for i=1:npoints+1,

building1_h(i,1:3)=[building1(i,1),building1(i,2)+h_build1,building1(i
,3)];

 building1_0(i,1:3)=(inv(Ra1_a1)*(building1(i,1:3)'-ta1_a1))';
 building1_h(i,1:3)=(inv(Ra1_a1)*(building1_h(i,1:3)'-ta1_a1))';

 w_0(i,1:3)=(K1*(R1*building1_0(i,1:3)'+t1))';
 w_0(i,1:3)=w_0(i,1:3)./w_0(i,3);
 w_h(i,1:3)=(K1*(R1*building1_h(i,1:3)'+t1))';
 w_h(i,1:3)=w_h(i,1:3)./w_h(i,3);
end;

hl=line(w_0(:,1),w_0(:,2));
```

```
 set(hl,'Color',line_color);

hl=line(w_h(:,1),w_h(:,2));
 set(hl,'Color',line_color);

for i=1:npoints,
 hl=line([w_0(i,1) , w_h(i,1) ],[w_0(i,2) , w_h(i,2)]);
 set(hl,'Color',line_color);
end;
```

File mapconstraints.m

```
P1=K1*([R1*Ra1_a1, t1-R1*Ra1_a1*ta1_a1]);
P2=K2*([R2*inv(Ra1_a1), t2-R2*inv(Ra1_a1)*ta1_a1]);
P3=K3*([R3*inv(Ra1_a3), t3-R3*inv(Ra1_a3)*ta1_a3]);
P4=K4*([R4*inv(Ra1_a3), t4-R4*inv(Ra1_a3)*ta1_a3]);
P5=K5*([R5*inv(Ra1_a5), t5-R5*inv(Ra1_a5)*ta1_a5]);
P6=K6*([R6*inv(Ra1_a5), t6-R6*inv(Ra1_a5)*ta1_a5]);


%-------------------------------------
% Img1 project matrix re-estimation

%point to point correspondences image-map

%wpp1=[; ; 1];  Xpp1=[; 0; 0;1];
%wpp2=[; ; 1];  Xpp2=[; 0; 0;1];
%wpp3=[; ; 1];  Xpp3=[0;0; 0;1];
wpp1=vp1_1;  Xpp1=[1; 0; 0;0];
wpp2=vp2_1;  Xpp2=[0; 1; 0;0];

Wpp=[wpp1';wpp2'];
Xpp=[Xpp1';Xpp2'];

%Wpp=[];Xpp=[];
Xp0=[P1(1,:)'; P1(2,:)';  P1(3,:)'];

%line to line correspondences image-map
% L-> aX+bZ+c=0;
%      Y=h;
wll1=[19 136 52 146]; Xll1=[1 0 140 0]; % X+140=0
wll2=[43 125 97 143]; Xll2=[1 0 180 -50];
wll3=[16 331 546 354]; Xll3=[0 1 415 180];
wll4=[14 347 537 375]; Xll4=[0 1 470 180];

Wll=[wll1;wll2;wll3;wll4];
Xll=[Xll1;Xll2;Xll3;Xll4];
%Wll=[wll1;wll2];
%Xll=[Xll1;Xll2];

%myfopt=[0 1e-4 1e-4 1e-6   0    0     0     0   0....
%        0  0     0    0   0    0   1e-8   0.1  0];
%Xp=fmins('funbestP',Xp0,myfopt,[],Wpp,Xpp,Wll,Xll);

myfopt=optimset('Display','iter','TolX',1e-4,'TolFun',1e-
4,'MaxFunEvals',5000,'MaxIter',5000);
Xp=fminsearch('funbestP',Xp0,myfopt,Wpp,Xpp,Wll,Xll);

P1new=[Xp(1:4)';Xp(5:8)';Xp(9:12)'];
```

```
A=inv(P1new(:,1:3)); [Qa Ra]=qr(A);
K1n=inv(Ra)*[-1 0 0; 0 -1 0; 0 0 1];
R1n=[-1 0 0; 0 -1 0; 0 0 1]*inv(Qa);
t1n=inv(K1n)*P1new(:,4);
t1n=t1n+R1n*ta1_a1; %note don't change order with next statement
R1n=R1n*Ra1_a1;

Xca1n=-inv(R1n)*t1n;

figure(1); colormap(gray(255)); image(img1); hold on; zoom on;
plotonimage(figimg1,K1n,R1n,t1n,Xca1n,Ra1_a1,ta1_a1,bu1_0,h_build1,'Bl
ue');
plotonimage(figimg1,K1n,R1n,t1n,Xca1n,Ra1_a1,ta1_a1,bu2_0,h_build2,'Re
d');
plotonimage(figimg1,K1n,R1n,t1n,Xca1n,Ra1_a1,ta1_a1,bu3h_0,h_build3,'Y
ellow');
plotonimage(figimg1,K1n,R1n,t1n,Xca1n,Ra1_a1,ta1_a1,gr1h_0,h_green1,'G
reen');
plotonimage(figimg1,K1n,R1n,t1n,Xca1n,Ra1_a1,ta1_a1,gr2h_0,h_green2,'G
reen');
plotonimage(figimg1,K1n,R1n,t1n,Xca1n,Ra1_a1,ta1_a1,ga2h_0,h_gates2,'W
hite');
plotonimage(figimg1,K1n,R1n,t1n,Xca1n,Ra1_a1,ta1_a1,ga3h_0,h_gates3,'W
hite');


%-------------------------------------
% Img2 project matrix re-estimation

%point to point correspondences image-map

%wpp1=[; ; 1];  Xpp1=[; 0; 0;1];

wpp1=vp1_2;  Xpp1=[1; 0; 0;0];
wpp2=vp2_2;  Xpp2=[0; 1; 0;0];

Wpp=[wpp1';wpp2'];
Xpp=[Xpp1';Xpp2'];

Xp0=[P2(1,:)'; P2(2,:)';  P2(3,:)'];

%line to line correspondences image-map
% L-> aX+bZ+c=0;
%      Y=h;
wll1=[501 330 281 449]; Xll1=[1 0 -440 180]; % X-140=0 greeen1 lateral
line gates
wll2=[8 123 63 142]; Xll2=[1 0 180 -50];     % build 3 roof
wll3=[501 330 5 301]; Xll3=[0 1 75 180];     % green 1 line near build
1
wll4=[66 136  500 88]; Xll4=[0 1 0 0];       % build 1 roof
wll5=[549 285 526 284]; Xll5=[0 1 -200 150]; % gate 3
wll6=[523 199 525 296]; Xll6=[0 0 440 200]; % build 1 fac2 far h

%hl=line([wll1(1) wll1(3)],[wll1(2)
wll1(4)]);set(hl,'Color','Yellow');

Wll=[wll1;wll2;wll3;wll4;wll5;wll6];
Xll=[Xll1;Xll2;Xll3;Xll4;Xll5;Xll6];


%myfopt=[0 1e-4 1e-4 1e-6   0    0     0     0   0 0  0     0    0   0
0   1e-8   0.1  0];
```

```
%Xp=fmins('funbestP',Xp0,myfopt,[],Wpp,Xpp,Wll,Xll);

myfopt=optimset('Display','iter','TolX',1e-4,'TolFun',1e-
4,'MaxFunEvals',5000,'MaxIter',5000)
Xp=fminsearch('funbestP',Xp0,myfopt,Wpp,Xpp,Wll,Xll);


P2new=[Xp(1:4)';Xp(5:8)';Xp(9:12)'];

A=inv(P2new(:,1:3)); [Qa Ra]=qr(A);
K2n=inv(Ra)*[-1 0 0; 0 -1 0; 0 0 1];
R2n=[-1 0 0; 0 -1 0; 0 0 1]*inv(Qa);
t2n=inv(K2n)*P2new(:,4);
t2n=t2n+R2n*ta1_a1; %note don't change order with next statement
R2n=R2n*Ra1_a1;

Xca2n=-inv(R2n)*t2n;

figure(2); colormap(gray(255)); image(img2); hold on; zoom on;
plotonimage(figimg2,K2n,R2n,t2n,Xca2n,Ra1_a1,ta1_a1,bu1_0,h_build1,'Bl
ue');
plotonimage(figimg2,K2n,R2n,t2n,Xca2n,Ra1_a1,ta1_a1,bu3h_0,h_build3,'Y
ellow');
plotonimage(figimg2,K2n,R2n,t2n,Xca2n,Ra1_a1,ta1_a1,gr1h_0,h_green1,'G
reen');
plotonimage(figimg2,K2n,R2n,t2n,Xca2n,Ra1_a1,ta1_a1,ga3h_0,h_gates3,'W
hite');



%-------------------------------------
% Img4 project matrix re-estimation

%point to point correspondences image-map

%wpp1=[; ; 1];  Xpp1=[; 0; 0;1];
%wpp2=[; ; 1];  Xpp2=[; 0; 0;1];
%wpp3=[; ; 1];  Xpp3=[0;0; 0;1];
wpp1=vp1_4;  Xpp1=[0; 0; 1;0];
wpp2=vp2_4;  Xpp2=[0; 1; 0;0];

Wpp=[wpp1';wpp2'];
Xpp=[Xpp1';Xpp2'];

%Wpp=[];Xpp=[];
Xp0=[P4(1,:)'; P4(2,:)';  P4(3,:)'];

%line to line correspondences image-map
% L-> aX+bZ+c=0;
%      Y=h;

wll1=[26 276 19 320]; Xll1=[0 0 -80 -400]; % X=-80; Z=-400; build 2
fac 1 left ver
wll2=[2 394 217 421]; Xll2=[1 0 -510 180];   % X-140=0 gate 2 floor
wll3=[33 251 96 256]; Xll3=[1 0 80 0];      % build 2 roof
wll4=[166 114 493 119]; Xll4=[1 0 -440 0];   % build 1 fac 2 roof
wll5=[135 339 166 114]; Xll5=[0 0 440 0];   % build 1 corner fac1 fac
2 ver

Wll=[wll1;wll2;wll3;wll4;wll5];
Xll=[Xll1;Xll2;Xll3;Xll4;Xll5];
```

```
%myfopt=[0 1e-4 1e-4 1e-6   0    0     0     0   0....
%        0  0     0    0   0    0   1e-8   0.1  0];
%Xp=fmins('funbestP',Xp0,myfopt,[],Wpp,Xpp,Wll,Xll);

myfopt=optimset('Display','iter','TolX',1e-4,'TolFun',1e-
4,'MaxFunEvals',5000,'MaxIter',5000);
Xp=fminsearch('funbestP',Xp0,myfopt,Wpp,Xpp,Wll,Xll);

P4new=[Xp(1:4)';Xp(5:8)';Xp(9:12)'];

A=inv(P4new(:,1:3)); [Qa Ra]=qr(A);
K4n=inv(Ra)*[-1 0 0; 0 -1 0; 0 0 1];
R4n=[-1 0 0; 0 -1 0; 0 0 1]*inv(Qa);
t4n=inv(K4n)*P4new(:,4);
t4n=t4n+R4n*ta1_a3; %note don't change order with next statement
R4n=R4n*Ra1_a3;

Xca4n=-inv(R4n)*t4n;

figure(5); colormap(gray(255)); image(img4); hold on; zoom on;
plotonimage(figimg4,K4n,R4n,t4n,Xca4n,Ra1_a3,ta1_a3,bu1_0,h_build1,'Bl
ue');
plotonimage(figimg4,K4n,R4n,t4n,Xca4n,Ra1_a3,ta1_a3,bu2_0,h_build2,'Re
d');
plotonimage(figimg4,K4n,R4n,t4n,Xca4n,Ra1_a3,ta1_a3,ga2h_0,h_gates2,'W
hite');
plotonimage(figimg4,K4n,R4n,t4n,Xca4n,Ra1_a3,ta1_a3,ga3h_0,h_gates3,'W
hite');


%-------------------------------------
% Img5 project matrix re-estimation

%point to point correspondences image-map

%wpp1=[; ; 1];  Xpp1=[-80; 180; -400;1]; %p1
%wpp2=[; ; 1];  Xpp2=[-140; 180; 20;1];  %p2
wpp1=vp1_5;  Xpp6=[0; 0; 1;0];
wpp2=vp2_5;  Xpp7=[0; 1; 0;0];



%plot(wpp1(1),wpp1(2),'yx');

Wpp=[wpp1';wpp2'];
Xpp=[Xpp1';Xpp2'];

Xp0=[P5(1,:)'; P5(2,:)';  P5(3,:)'];

%line to line correspondences image-map
% L-> aX+bZ+c=0;
%      Y=h;
wll1=[477 211 480 294]; Xll1=[0 0 0 0];      % build 1 corner left
vert
wll2=[456 283 453 203]; Xll2=[0 0 -180 200];  %build 3 corner right
vert
wll3=[358 190 359 252]; Xll3=[0 0 -80 -100];  %build 2 fac 1 corner
right vert
wll4=[438 151 513 160]; Xll4=[1 0 180 -50]; %l4
%hl=line([wll1(1) wll1(3)],[wll1(2)
wll1(4)]);set(hl,'Color','Yellow');
```

```
Wll=[wll1;wll2;wll3;wll4];
Xll=[Xll1;Xll2;Xll3;Xll4];

myfopt=optimset('Display','iter','TolX',1e-4,'TolFun',1e-
4,'MaxFunEvals',5000,'MaxIter',5000)
Xp=fminsearch('funbestP',Xp0,myfopt,Wpp,Xpp,Wll,Xll);

P5new=[Xp(1:4)';Xp(5:8)';Xp(9:12)'];

A=inv(P5new(:,1:3)); [Qa Ra]=qr(A);
K5n=inv(Ra)*[-1 0 0; 0 -1 0; 0 0 1];
R5n=[-1 0 0; 0 -1 0; 0 0 1]*inv(Qa);
t5n=inv(K5n)*P5new(:,4);
t5n=t5n+R5n*ta1_a5; %note don't change order with next statement
R5n=R5n*Ra1_a5;

Xca5n=-inv(R5n)*t5n;

figure(6); colormap(gray(255)); image(img5); hold on; zoom on;
plotonimage(figimg5,K5n,R5n,t5n,Xca5n,Ra1_a5,ta1_a5,bu1_0,h_build1,'Bl
ue');
plotonimage(figimg5,K5n,R5n,t5n,Xca5n,Ra1_a5,ta1_a5,bu2_0,h_build2,'Re
d');
plotonimage(figimg5,K5n,R5n,t5n,Xca5n,Ra1_a5,ta1_a5,bu3h_0,h_build3,'Y
ellow');


green1b=[0 0 0;490 0 0;490 0 340;0 0 340;0 0 0];
R_green1_O=[1 0 0;0 1 0; 0 0 1]; t_green1_O=[-70;0;-415];
gr1_0b=plotonmap(figmap,green1b,R_green1_O,t_green1_O,1,0,'Green','');
h_green1b=0; gr1h_0b=gr1_0b; gr1h_0b(:,2)=gr1_0b(:,2)-(h_green1b-
h_build1);

plotonimage(figimg5,K5n,R5n,t5n,Xca5n,Ra1_a5,ta1_a5,gr1h_0b,h_green1b,
'Green');

green2b=[0 0 0;270 0 0;270 0 40;0 0 40;0 0 0];
R_green2_O=[1 0 0;0 1 0; 0 0 1]; t_green2_O=[-130;0;-510];
gr2_0b=plotonmap(figmap,green2b,R_green2_O,t_green2_O,1,0,'Green','');
h_green2b=0; gr2h_0b=gr2_0b; gr2h_0b(:,2)=gr2_0b(:,2)-(h_green2b-
h_build1);

plotonimage(figimg5,K5n,R5n,t5n,Xca5n,Ra1_a5,ta1_a5,gr2h_0b,h_green2b,
'Green');

%-----------------------------------
% Img6 project matrix re-estimation

%point to point correspondences image-map

wpp1=[100; 324; 1];  Xpp1=[-80; 180; -400;1]; %p1
wpp2=[466; 318; 1];  Xpp2=[-140; 180; 20;1];  %p2
wpp3=[527; 320; 1];  Xpp3=[0; 180; 0;1];      %p3
wpp4=[494; 310; 1];  Xpp4=[-180; 180; 115;1]; %p4
wpp5=[488; 154; 1];  Xpp5=[-180; -50; 115;1]; %p5
wpp6=vp1_6;  Xpp6=[0; 0; 1;0];
wpp7=vp2_6;  Xpp7=[0; 1; 0;0];



%plot(wpp1(1),wpp1(2),'yx');
```

```
Wpp=[wpp1';wpp2';wpp3';wpp4';wpp5';wpp6';wpp7'];
Xpp=[Xpp1';Xpp2';Xpp3';Xpp4';Xpp5';Xpp6';Xpp7'];

%Wpp=[wpp6';wpp7'];
%Xpp=[Xpp6';Xpp7'];

Xp0=[P6(1,:)'; P6(2,:)';  P6(3,:)'];

%line to line correspondences image-map
% L-> aX+bZ+c=0;
%      Y=h;
wll1=[1 348 25 327]; Xll1=[0 1 510 180];    %l1
wll2=[201 445 79 327]; Xll2=[0 1 415 180];  %l2
wll3=[562 328 438 325]; Xll3=[0 1 75 180];  %l3
wll4=[438 151 513 160]; Xll4=[1 0 180 -50]; %l4
%hl=line([wll1(1) wll1(3)],[wll1(2)
wll1(4)]);set(hl,'Color','Yellow');

Wll=[wll1;wll2;wll3;wll4];
Xll=[Xll1;Xll2;Xll3;Xll4];
%Wll=[wll1;wll2;wll3];
%Xll=[Xll1;Xll2;Xll3];

%myfopt=[0 1e-4 1e-4 1e-6   0    0     0     0   0....
%        0  0     0    0    0    0   1e-8   0.1  0];
%Xp=fmins('funbestP',Xp0,myfopt,[],Wpp,Xpp,Wll,Xll);

myfopt=optimset('Display','iter','TolX',1e-4,'TolFun',1e-
4,'MaxFunEvals',5000,'MaxIter',5000)
Xp=fminsearch('funbestP',Xp0,myfopt,Wpp,Xpp,Wll,Xll);

P6new=[Xp(1:4)';Xp(5:8)';Xp(9:12)'];

A=inv(P6new(:,1:3)); [Qa Ra]=qr(A);
K6n=inv(Ra)*[-1 0 0; 0 -1 0; 0 0 1];
R6n=[-1 0 0; 0 -1 0; 0 0 1]*inv(Qa);
t6n=inv(K6n)*P6new(:,4);
t6n=t6n+R6n*ta1_a5; %note don't change order with next statement
R6n=R6n*Ra1_a5;

Xca6n=-inv(R6n)*t6n;

figure(7); colormap(gray(255)); image(img6); hold on; zoom on;
plotonimage(figimg6,K6n,R6n,t6n,Xca6n,Ra1_a5,ta1_a5,bu1_0,h_build1,'Bl
ue');
plotonimage(figimg6,K6n,R6n,t6n,Xca6n,Ra1_a5,ta1_a5,bu2_0,h_build2,'Re
d');
plotonimage(figimg6,K6n,R6n,t6n,Xca6n,Ra1_a5,ta1_a5,bu3h_0,h_build3,'Y
ellow');


green1b=[0 0 0;490 0 0;490 0 340;0 0 340;0 0 0];
R_green1_O=[1 0 0;0 1 0; 0 0 1]; t_green1_O=[-70;0;-415];
gr1_0b=plotonmap(figmap,green1b,R_green1_O,t_green1_O,1,0,'Green','');
h_green1b=0; gr1h_0b=gr1_0b; gr1h_0b(:,2)=gr1_0b(:,2)-(h_green1b-
h_build1);

plotonimage(figimg6,K6n,R6n,t6n,Xca6n,Ra1_a5,ta1_a5,gr1h_0b,h_green1b,
'Green');

green2b=[0 0 0;270 0 0;270 0 40;0 0 40;0 0 0];
```

```
R_green2_O=[1 0 0;0 1 0; 0 0 1]; t_green2_O=[-130;0;-510];
gr2_0b=plotonmap(figmap,green2b,R_green2_O,t_green2_O,1,0,'Green','');
h_green2b=0; gr2h_0b=gr2_0b; gr2h_0b(:,2)=gr2_0b(:,2)-(h_green2b-
h_build1);

plotonimage(figimg6,K6n,R6n,t6n,Xca6n,Ra1_a5,ta1_a5,gr2h_0b,h_green2b,
'Green');



%new camera displacemente
figure(8);
plot3(Xca1n(1),Xca1n(3),0,'bx');
%text(Xca1n(1)+10,Xca1n(3)+10,0,'C1');
plot3(Xca2n(1),Xca2n(3),0,'bx');
%text(Xca2n(1)+10,Xca2n(3)+10,0,'C2');
%Xca3_a1=Ra1_a3*Xca3+ta1_a3;plot3(Xca3_a1(1),Xca3_a1(3),0,'rx');
%text(Xca3_a1(1)+10,Xca3_a1(3)+10,0,'C3');
Xca4_a1=Ra1_a3*Xca4n+ta1_a3;plot3(Xca4_a1(1),Xca4_a1(3),0,'rx');
%text(Xca4_a1(1)+10,Xca4_a1(3)+10,0,'C4');
%Xca5_a1=Ra1_a5*Xca5+ta1_a5;plot3(Xca5_a1(1),Xca5_a1(3),0,'gx');
%text(Xca5_a1(1)+10,Xca5_a1(3)+10,0,'C5');
Xca6_a1=Ra1_a5*Xca6n+ta1_a5;plot3(Xca6_a1(1),Xca6_a1(3),0,'gx');
%text(Xca6_a1(1)+10,Xca6_a1(3)+10,0,'C6');
```

File gentexture.m

```
filename='KingsParade.wrl';
initvrmlfile(filename);

figure(1); colormap(gray(255)); image(img1); hold on; zoom on;

vis=1;

% punti presi dall'immagine di dimensioni 1/2
W=[53 128 0.5  101 144 0.5; 91 207 0.5  91 252 0.5; ....
   83 268 0.5   66 268 0.5; 50 220 0.5  50 179 0.5];
W=W*2;
genrectif('texturefA1.ppm','tex_bu3.ppm',W,1125,900,vis);
%need convertion ppm to jpg
X=[-180 -50 115 ;-180 -50  205;-180 180 205;-180 180 115];
addvrmlfile(filename,X,'tex_bu3.jpg',1)



% punti presi dall'immagine di dimensioni 1/2
W=[6 278 0.5 528 280 0.5; 528 280 0.5 530 441 0.5;
   530 441 0.5 6 389 0.5; 6 389 0.5 6 278 0.5];
W=W*2;
genrectif('texturefA1.ppm','tex_ga1.ppm',W,1125,900,vis);

X=[-140 150 -520 ;430 150 -520;430 180 -520;-140 180 -520];
addvrmlfile(filename,X,'tex_ga1.jpg',1)


% punti presi dall'immagine di dimensioni 1/2
W=[66 136 0.5 500  89 0.5; 503 139 0.5 506 301 0.5;
   76 302 0.5 428 320 0.5; 57  284 0.5 63  172 0.5];
W=W*2;
genrectif('texturefA3.ppm','tex_bu1a.ppm',W,1125,900,vis);
```

```
X=[0 0 0; 440 0 -0;440 180 0;0 180 0];
addvrmlfile(filename,X,'tex_bu1a.jpg',1)


% punti presi dall'immagine di dimensioni 1/2
% triangle facade A building 1
W=[vp1_2(1) vp1_2(2) 0.5  238 94 0.5; vp2_2(1) vp2_2(2) 0.5 319 129
0.5;
   319 129 0.5 166 144 0.5; 166  144 0.5 vp2_2(1) vp2_2(2) 0.5];
W=W*2;
genrectif('texturefA3.ppm','tex_bu1b.ppm',W,1125,900,vis);

X=[140 -20 0; 300 -20 0;300 20 0;140 20 0];
addvrmlfile(filename,X,'tex_bu1b.jpg',1)


% punti presi dall'immagine di dimensioni 1/2
% green1/green2  texture
W=[205 335 0.5  250 335 0.5; 250 335  0.5 250 380 0.5;
   250 380 0.5  205 380 0.5; 205 380  0.5 205 335 0.5];
W=W*2;
genrectif('texturefA3.ppm','tex_gr1.ppm',W,1125,900,vis);

% green1
X=[-70 180 -415; 440 180 -415;440 180 -75;-70 180 -75];
addvrmlfile(filename,X,'tex_gr1.jpg',100);
% green2
X=[-130 180 -510; 440 180 -510;440 180 -470;-130 180 -470];
addvrmlfile(filename,X,'tex_gr1.jpg',100);




% punti presi dall'immagine di dimensioni 1/2
% facade 2 building1
W=[166 114 0.5  493 119 0.5; 494 224 0.5  493 373 0.5;
   319 412 0.5  288 409 0.5; 153 209 0.5  135 339 0.5];
W=W*2;
genrectif('texturefB2.ppm','tex_bu1c.ppm',W,1125,900,vis);

X=[440 0 0; 440 0 210; 440 180 210; 440 180 0];
addvrmlfile(filename,X,'tex_bu1c.jpg',1)


% punti presi dall'immagine di dimensioni 1/2
% triangle facade 2  building 1
W=[vp1_4(1) vp1_4(2) 0.5  322 80 0.5; vp2_4(1) vp2_4(2) 0.5 458 149
0.5;
   458 149 0.5 188 142 0.5; 188  142 0.5 vp2_4(1) vp2_4(2) 0.5];
W=W*2;
genrectif('texturefB2.ppm','tex_bu1d.ppm',W,1125,900,vis);

X=[440 -20 25; 440 -20 185; 440 20 185;440 20 25];
addvrmlfile(filename,X,'tex_bu1d.jpg',1)


% punti presi dall'immagine di dimensioni 1/2
% gate 3
W=[351 369 0.5  479 377 0.5; 510 448 0.5  511 426 0.5;
   510 448 0.5  347 445 0.5; 323 374 0.5  319 444 0.5];
W=W*2;
```

```
genrectif('texturefB2.ppm','tex_ga3.ppm',W,1125,900,vis);

X=[510 150 120; 510 150 210; 510 180 210; 510 180 120];
addvrmlfile(filename,X,'tex_ga3.jpg',1)


% punti presi dall'immagine di dimensioni 1/2
% gate 2
W=[8 328 0.5  331 343 0.5; 370 381 0.5  370 353 0.5;
   291 405 0.5  38 411 0.5; 8 328  0.5  8 379 0.5];
W=W*2;
genrectif('texturefB1.ppm','tex_ga2.ppm',W,1125,900,vis);

X=[510 150 -420; 510 150 80; 510 180 80; 510 180 -420];
addvrmlfile(filename,X,'tex_ga2.jpg',1)



% punti presi dall'immagine di dimensioni 1/2
% facade 1 building2
W=[112 109 0.5  344 148 0.5; 358 193 0.5  359 250 0.5;
   361 315 0.5  240 320 0.5; 89 178 0.5  88 230 0.5];
W=W*2;
genrectif('texturefC1.ppm','tex_bu2a.ppm',W,1125,900,vis);

X=[-80 0 -400; -80 0 -100; -80 180 -100;-80 180 -400];
addvrmlfile(filename,X,'tex_bu2a.jpg',1)


% punti presi dall'immagine di dimensioni 1/2
% facades b/e building2
W=[77 119 0.5  91 108 0.5; 89 178 0.5  88 230 0.5;
   88 325 0.5  67 319 0.5; 71 186 0.5  70 234 0.5];
W=W*2;
genrectif('texturefC1.ppm','tex_bu2e.ppm',W,1125,900,vis);

%fac b
X=[-140 0 -100; -80 0 -100; -80 180 -100;-140 180 -100];
addvrmlfile(filename,X,'tex_bu2e.jpg',1)
%fac e
X=[-140 0 -400; -80 0 -400; -80 180 -400;-140 180 -400];
addvrmlfile(filename,X,'tex_bu2e.jpg',1)


% punti presi dall'immagine di dimensioni 1/2
% facades c building2
W=[366 165 0.5  400 170 0.5; 403 216 0.5  404 250 0.5;
   406 310 0.5  371 312 0.5; 362 316 0.5  361 261 0.5];
W=W*2;
genrectif('texturefC1.ppm','tex_bu2c.ppm',W,1125,900,vis);

X=[-140 0 -40; -140 0 20; -140 180 20;-140 180 -40];
addvrmlfile(filename,X,'tex_bu2c.jpg',1);


% punti presi dall'immagine di dimensioni 1/2
% facades d building2
W=[ 1 136 0.5  99 143 0.5; 100 201 0.5 99 245 0.5;
   92 322 0.5  13 321 0.5;   9 212 0.5  8 240 0.5];
W=W*2;
genrectif('texturefC3.ppm','tex_bu2d.ppm',W,1125,900,vis);
```

```
X=[-140 0 -505; -140 0 -400; -140 180 -400;-140 180 -505];
addvrmlfile(filename,X,'tex_bu2d.jpg',1);
```

File initvrmlfile.m

```
function initvrmlfile(filename);

fid = fopen(filename,'w');
fprintf(fid,'#VRML V1.0 ascii\n');
fprintf(fid,'#My Example\n');

fclose(fid);

return;
```

File genrectif.m

```
function  genrectif(filein,fileout,W,w,h,vis);

% lines from pairs of points
l=zeros(5,3);
for i=1:4;
 Asys=[W(i,1:3);W(i,4:6)];
 [U D V]=svd(Asys);
 l(i+1,1:3)=V(1:3,3)';
end;
l(1,1:3)= l(5,1:3);

% line intersection
W2=zeros(4,3);
for i=1:4;
 Asys=[l(i,1:3);l(i+1,1:3)];
 [U D V]=svd(Asys);
 W2(i,1:3)=V(1:3,3)';
 W2(i,1:3)=W2(i,1:3)./W2(i,3);
end;

Wrect=[1 1 1; w 1 1; w h 1 ; 1 h 1];

% homgraphy recovery

Asys=zeros(12,13);

for i=1:4,
 j=(i-1)*3+1;
 Asys(j:j+2,1:9)=[W2(i,1:3)  0 0 0     0 0 0;....
                  0 0 0   W2(i,1:3)   0 0 0;....
                  0 0 0     0 0 0   W2(i,1:3)];
 Asys(j:j+2,10+i-1)=-Wrect(i,1:3)';
end;

[U D V]=svd(Asys);
Hrect=(reshape(V(1:9,13),3,3))';
Hrectinv=inv(Hrect);

Hrectinv=Hrectinv;
command=sprintf('rectification %s %f %f %f %f %f %f %f %f %f %s',
filein, ....
 Hrectinv(1,1),Hrectinv(1,2),Hrectinv(1,3),...
```

```
 Hrectinv(2,1),Hrectinv(2,2),Hrectinv(2,3),...
 Hrectinv(3,1),Hrectinv(3,2),Hrectinv(3,3),fileout);
unix(command);

if vis~=0,
 command2=sprintf('xv %s &',fileout);
 unix(command2);
end;

return;
```

File addvrmlfile.m

```
function addvrmlfile(filename,Xrec,texfilename,k);

fid = fopen(filename,'a');

fprintf(fid,'#VRML V1.0 ascii\n');
fprintf(fid,'#My Example\n');

triangles=[1 2 3; 3 4 1];
corr=[0 1*k; 1*k 1*k; 1*k 0; 0 0];

for i=1:2,
 i1=triangles(i,1); i2=triangles(i,2); i3=triangles(i,3);
 Tvect(i,1:9)=[Xrec(i1,1) Xrec(i1,2)  Xrec(i1,3) ....
               Xrec(i2,1) Xrec(i2,2)  Xrec(i2,3) ....
               Xrec(i3,1) Xrec(i3,2)  Xrec(i3,3) ];
 textvect=[corr(i1,1) corr(i1,2)  ....
           corr(i2,1) corr(i2,2)  ....
           corr(i3,1) corr(i3,2)];

 fprintf(fid,'\n');
 fprintf(fid,'Separator\n');
 fprintf(fid,'{\n');
 fprintf(fid,'   Coordinate3 {\n');
 fprintf(fid,'      point [\n');
 fprintf(fid,'        %f %f %f,\n',Tvect(i,1),Tvect(i,2),Tvect(i,3));
 fprintf(fid,'        %f %f %f,\n',Tvect(i,4),Tvect(i,5),Tvect(i,6));
 fprintf(fid,'        %f %f %f \n',Tvect(i,7),Tvect(i,8),Tvect(i,9));
 fprintf(fid,'      ]\n');
 fprintf(fid,'    }\n');
 fprintf(fid,'\n');
 fprintf(fid,'  Texture2 {\n');
 fprintf(fid,'    filename %s\n',texfilename);
 fprintf(fid,'  }\n');
 fprintf(fid,'\n');
 fprintf(fid,'  TextureCoordinate2 {\n');
 fprintf(fid,'    point [\n');


 fprintf(fid,'      %f %f,\n',textvect(1),textvect(2));
 fprintf(fid,'      %f %f,\n',textvect(3),textvect(4));
 fprintf(fid,'      %f %f,\n',textvect(5),textvect(6));


 fprintf(fid,'    ]\n');
 fprintf(fid,'  }\n');
 fprintf(fid,'\n');
 fprintf(fid,'  IndexedFaceSet {\n');
 fprintf(fid,'    coordIndex [2, 1, 0, -1,]\n');
```

```
 fprintf(fid,'    }\n');

 fprintf(fid,'  }\n');
end;
```

Bibliografia

[Beardsley96]
P. Beardsley, P. Torr and A. Zisserman, "3D model acquisition from extended image sequences", in Proc. 4th European Conf. on Computer Vision, Cambridge (April 96); LNCS 1065, vol. II, pp. 683-695, Springer-Verlag, 1996.

[Caprile90]
B. Caprile and V. Torre, "Using vanishing points for camera calibration," IJCV, pp. 127-140, 1990.

[Chella00]
Chella, A., Cossentino, M., and Lo Faso, U. 'Designing agent-based systems with UML' in Proc. of ISRA'2000 (Monterrey, Mexico, Nov. 2000).

[Cipolla99]
R. Cipolla, T. Drummond and D. Robertson, "Camera calibration from vanishing points in images of architectural scenes," in Proc. British Machine Vision Conference, Nottingham, vol. 2, pp. 382-391, 1999.

[Cossentino02]
M. Cossentino, C. Potts - "A CASE tool supported methodology for the design of multi-agent systems" - The 2002 International Conference on Software Engineering Research and Practice (SERP'02) , June 24 - 27, 2002 , Las Vegas (NV), USA

[Criminisi99]
A. Criminisi, I. Reid, and A. Zisserman, "Single View Metrology," in Proc. 7th International Conference on Computer Vision, pp.434-442, 1999.

[Debevec96]
P.E. Debevec, C.J. Taylor, and J. Malik, "Modelling and rendering architecture from photographs: A Hybrid Geometry-and Image-Base Approach," in ACM Computer Graphics (proceedings SIGGRAPH), pp. 11-20, 1996.

[Faugeras88]
O. Faugeras and F. Lustman, ''Motion and structure from motion in a piecewise planar environment," Int. Journal of Pattern Recognition and Artificial Intelligence, 2(3):485--508, 1988.

[Faugeras98]
O. Faugeras, S. Laveau, L. Robert, G. Csurka, and C. Zeller, "3-D reconstruction of urban scenes from sequences of images,"
Computer Vision and Image Understanding, n.69, vol.3, pp: 292-309,1998.

[Gortler96]
S.J. Gortler, R. Grzeszczuk, R. Szeliski, and M.F. Cohen, "The Lumigraph," in ACM Computer Graphics (Proc. SIGGRAPH), pp: 31-42,1996.

[Hartley92]
R. I. Hartley, "Estimation of relative camera positions for uncalibrated cameras," European Conf. on Computer Vision, pp. 579--587, Santa Margherita Ligure, Italy, May 1992.

[Hartley96]
R. I. Hartley, "Lines and points in three views and the trifocal tensor,"
International Journal of Computer Vision, 22(2):125-140, 1996.

[Infantino02] I. Infantino, M. Cossentino, A. Chella – “An Agent BasedMultilevel Architecture for Robotic vision Systems”, The 2002 International Conference on Artificial Intelligence (IC-AI'02) , June 2002 , Las Vegas (NV), USA

[Kanatami00]
K. Kanatami, N. Ohta and Y. Kanazawa, "Optimal homography computation with a reliabilty measure," in IEICE Trans. Inf. I\& Syst.,vol. E83-D,n.7, pp. 1369-1374, July 2000.

[Kang96]
S.B. Kang and R. Szeliski, "3-D scene data recovery using omni-directional multibaseline stereo," in CVPR'96, pp. 364-370, June 1997.

[Liebowitz98]
D. Liebowitz and A. Zisserman, "Metric Rectification for Perspective Images of Planes," in Proc. IEEE Conference on Computer Vision and Pattern Recognition, pp.482-488, 1998.

[Liebowitz99]
D. Liebowitz,A. Criminisi and A. Zisserman, "Creating architectural models from images," Eurographics'99, vol.18, n.3, 1999.

[Luong96]
Q.-T. Luong and T. Vieville, "Canonical Representations for the Geometries of Multiple Projective Views," Computer Vision and Image Understanding, 64(2):193-229, 1996.

[Narayanan98]
P.J. Narayanan,P.W. Rander and T. Kanade, "Constructing virtual worlds using dense stereo," in Proc. of Sixth IEEE Intl. Conf. on Computer Vision, Bombay (India), pp. 3-10, January 1998.

[Ogawa00]
Y. Ogawa,K. Iwamura and S. Kakumoto, "Exracting object information from arial images: a map-based approach," in IEICE Trans. Inf. I\& Syst.,vol. E83-D,n.7, pp. 1450-1457, July 2000.

[Pollefeyes98]
M. Pollefeyes,R. Koch and L. Van Gool, "Self calibration and metric reconstruction inspite of varying an unknown internal camera parameters," in Proc. of Sixth IEEE Intl. Conf. on Computer Vision, Bombay (India), pp. 90-95, January 1998.

[Seitz96]
S.M. Seitz, and C.R. Dyer,
"Toward image - based scene representation using view morphing," in Proc. of Intl. Conf. IEEE Conf. on Pattern Recognition, Vienna (Austria), January 1996.

[Shum98]
H-Y Shum, M. Han and R. Szeliski, "Interactive construction of 3-D models from panoramic mosaics," in Proc IEEE Conf. On Computer Vision and Pattern Recognition, pp. 427-433, Santa Barbara (USA), June 1998.

[Szeliski97]
R. Szeliski and S. Heung-Yeung, "Creating full view panoramic image mosaics and environment maps," in SIGGRAPH, 1997.

[Szeliski98]
R. Szeliski and P. Torr, "Geometrically constrained structure from motion: Points on planes," in European Workshop on 3D Structure from Multiple Images of Large-Scale Environments (SMILE), pp.171-186, June 1998.

[Tomasi90]
C. Tomasi, and T. Kanade, "Shape and motion from image streams under orthography: a factorisation method," International Journal of Computer vision, 9(2):137-154, 1990.

[Tveit01]
Amund Tveit - “A survey of Agent-Oriented Software Engineering”, May 8, 2001

[Wooldridge95]
M. Wooldridge, N.R Jennings (1995) – “Agents Theories, Architectures, and Languages: a Survey”