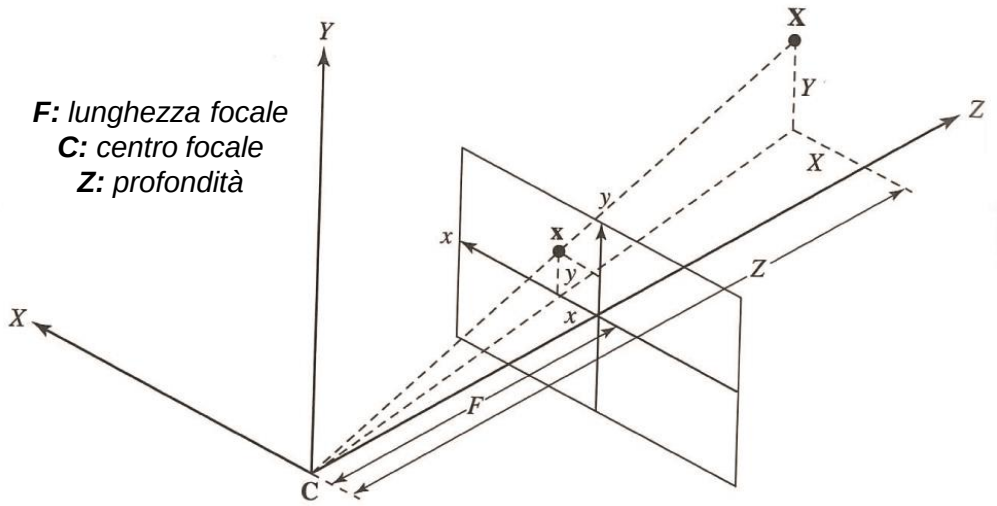
Video – Parte 3

Spazi di riferimento 3D-2D e Proiezioni

Vettori di movimento e Modelli di movimento

Stima del movimento

Telecamera Prospettica



- Assumiamo che
  - L'origine del sistema di coordinate 3D (globale) sia localizzata in ***C***
  - Il piano ***XY*** sia parallelo al piano ***xy*** (nel sistema di coordinate dell'immagine)
  - Il sistema di riferimento globale e quello dell'immagine usino la stessa unità di misura
- Prospettiva → la dimensione di un oggetto sull'immagine è inversamente proporzionale alla sua distanza

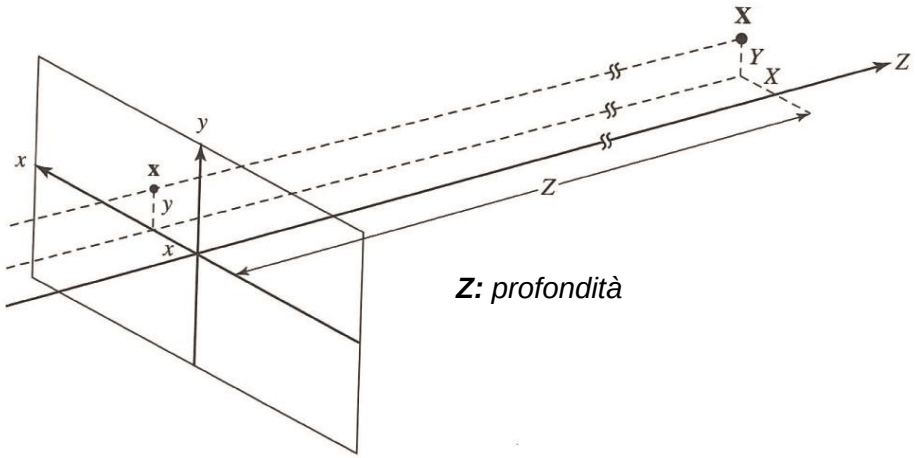
Proiezioni prospettiche:

$$\frac{x}{F} = \frac{X}{Z} \qquad \frac{y}{F} = \frac{Y}{Z}$$

$$x = F\frac{X}{Z} \qquad y = F\frac{Y}{Z}$$

Telecamera Ortografica

- Assumiamo che
  - Gli oggetti da rappresentare nell'immagine siano a distanza (***Z***) molto grande
- Ortografia → la dimensione di un oggetto sull'immagine è indipendente dalla sua distanza



Z: *profondità*

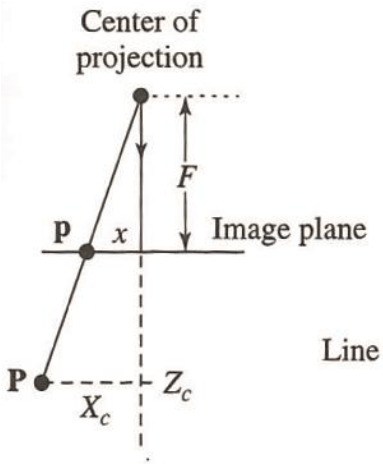
Proiezioni ortografiche:

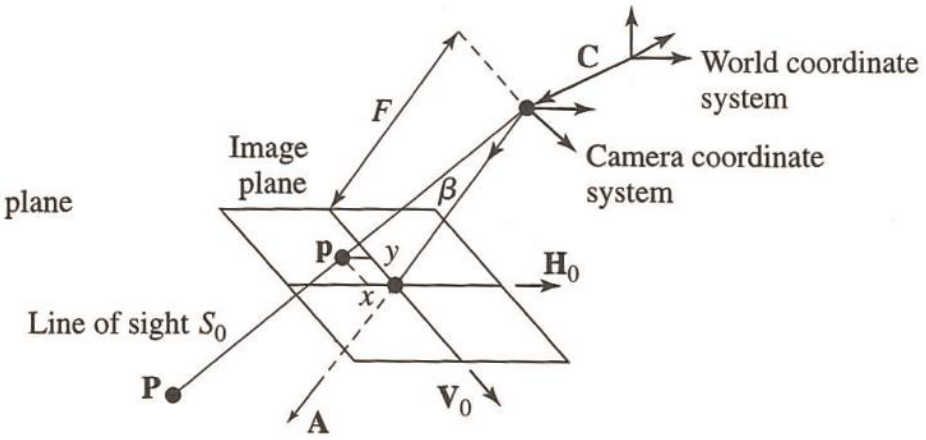
$$x = X \quad y = Y$$

Note sulle proiezioni

- **Invertibilità**: Sia le proiezioni prospettiche che quelle ortografiche stabiliscono una relazione **molti-a-uno**
  - Per invertire il processo e ricostruire una scena 3D partendo da un'immagine 2D ho bisogno di informazioni aggiuntive (z-mapping, stereoscopia, …)
- **Occlusione**: nell'immagine compaiono solo gli oggetti che vengono intercettati per primi dalle **linee di visione** che partono dal centro focale ***C***

Modello CAHV



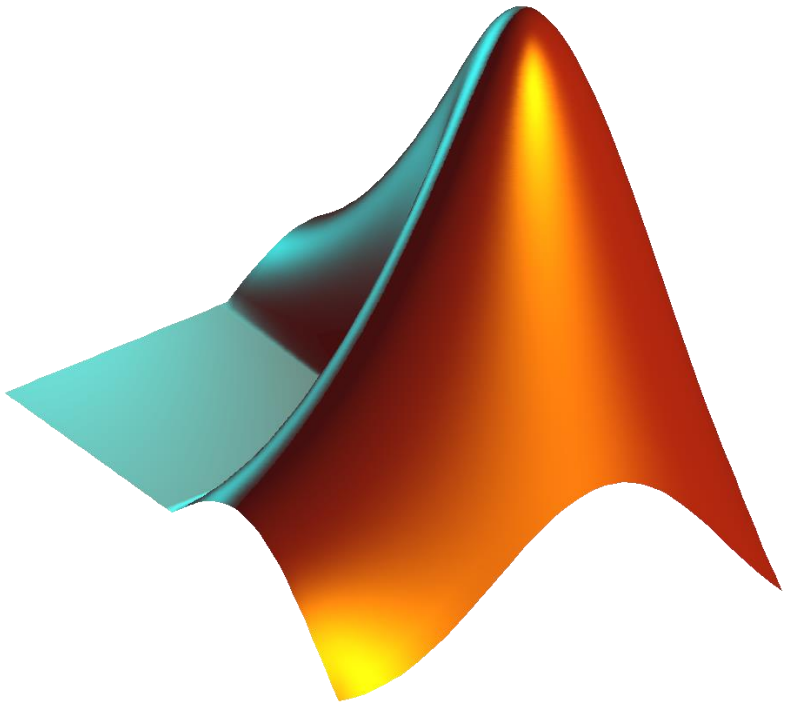


- **C**: Centro focale
- **A**: versore Z
- **H_0**: versore X
- **V_0**: versore Y
- Permette una visualizzazione più comoda del sistema di riferimento della videocamera se questa si muove
- Permette di descrivere un piano dell'immagine non allineato al sistema di riferimento della videocamera (**distorsione del sistema ottico**)

Dalle formule sulle proiezioni prospettiche:

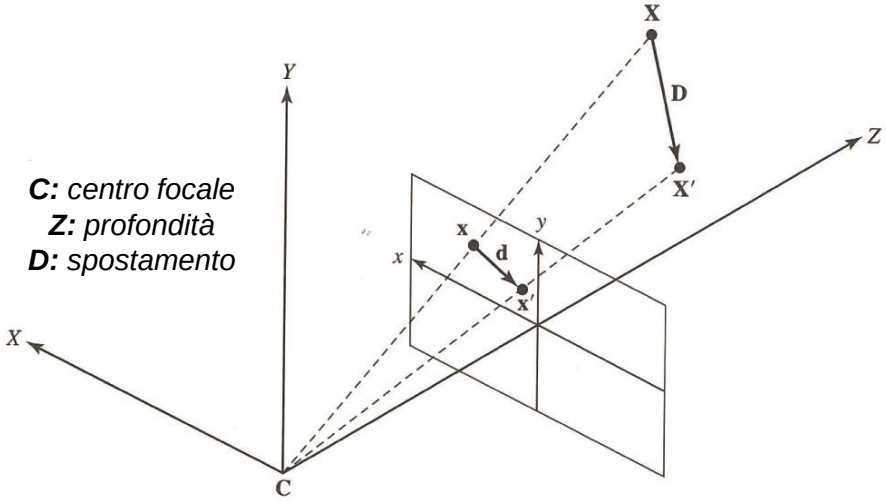
$$\boldsymbol{p} = \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{pmatrix} = \frac{\boldsymbol{F}}{\boldsymbol{A^T \cdot (P - C)}} \cdot \begin{pmatrix} \boldsymbol{H_0}^T \cdot (\boldsymbol{P - C}) \\ \boldsymbol{V_0}^T \cdot (\boldsymbol{P - C}) \end{pmatrix}$$

Proiezioni 3D → 2D



- ProjectionScript.m
  - projectionImage.m
  - projectionOrthoImage.m
  - **projection.m**
- Esempio:
  - Due triangoli nello spazio 3D proiettati in quello 2D, con camera prospettica e ortografica

Spostamenti in 3D e in 2D



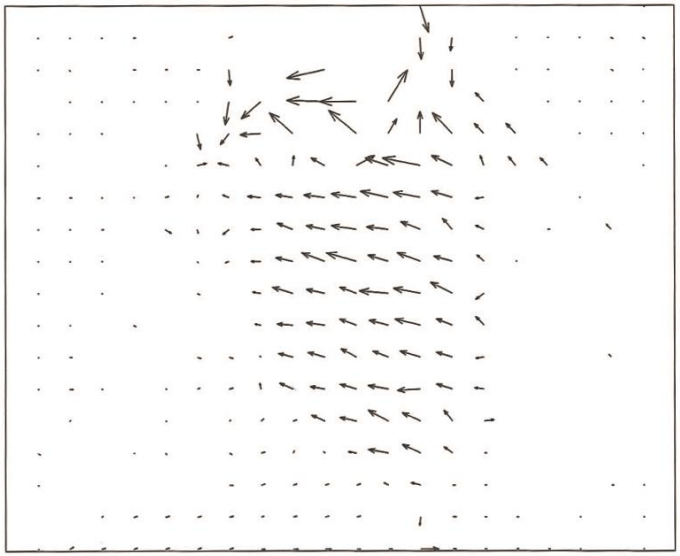
C: *centro focale*
Z: *profondità*
D: *spostamento*

- Siano:
  - Posizione iniziale a t_1:
    - $\boldsymbol{X} = [X, Y, Z]^T$
    - $\boldsymbol{x} = [x, y]^T$
  - Posizione finale a t_2:
    - $\boldsymbol{X}' = [X', Y', Z']^T = \\ = [X + D_X, Y + D_Y, Z + D_Z]^T$
    - $\boldsymbol{x}' = [x', y']^T = \\ = [x + d_x, y + d_y]^T$

- Spostamento 3D:
  - $\boldsymbol{D}(\boldsymbol{X}; t_1, t_2) = \boldsymbol{X}' - \boldsymbol{X} = \\ = [D_X, D_Y, D_Z]^T$
- Spostamento 2D:
  - $\boldsymbol{d}(\boldsymbol{x}; t_1, t_2) = \boldsymbol{x}' - \boldsymbol{x} = \\ = [d_x, d_y]^T$

Modelli di Movimento 2D

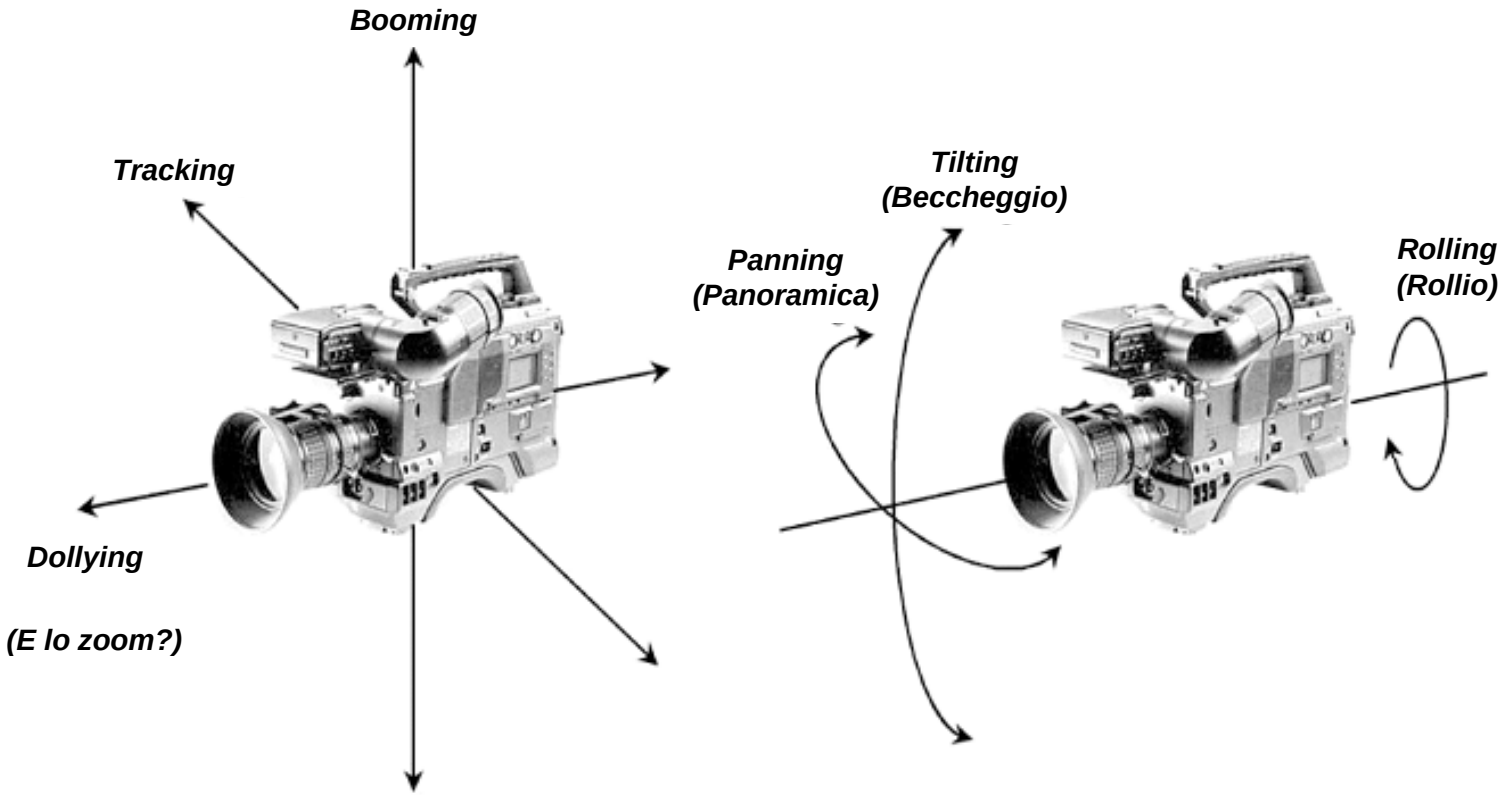
- Se t_1 e t_2 sono chiari si possono omettere e si può scrivere $\boldsymbol{d}(\boldsymbol{x})$
- $\boldsymbol{d}(\boldsymbol{x}) = \boldsymbol{x}' - \boldsymbol{x}$ è detto "*displacement*", "Vettore di movimento 2D" o "*motion vector*" (**MV**)
- L'insieme dei $\boldsymbol{d}(\boldsymbol{x})$ per ogni *x* nell'immagine è detto "Campo di movimento" (*motion field*)



Esempio di motion field

- **Funzione di mapping**: $\boldsymbol{w}(\boldsymbol{x}) = \boldsymbol{x} + \boldsymbol{d}(\boldsymbol{x})$
  - Ogni pixel contiene un valore diverso da quello iniziale; si noti che, in generale, $\boldsymbol{d}(\boldsymbol{x})$ potrebbe non essere noto

Movimenti della Telecamera



Traslazioni e Rotazioni tipiche della telecamera

Modelli di Movimento della Telecamera – Track (Tx) e Boom (Ty)

Cambiamenti di coordinate nel sistema 3D:

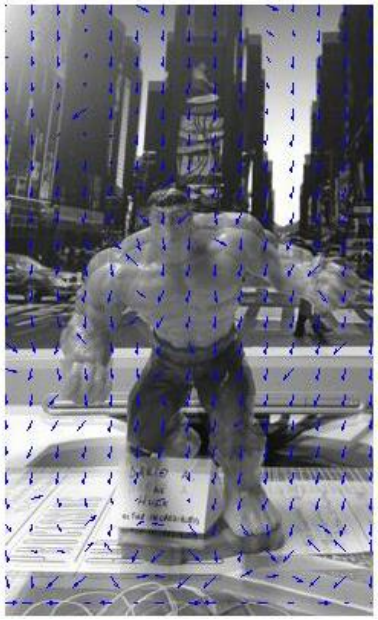
$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} T_x \\ T_y \\ 0 \end{bmatrix}$$

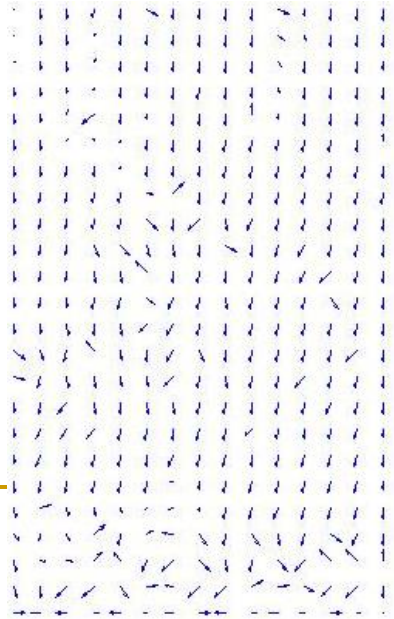
Approssimazione (ortografica) del motion field:

$$\begin{bmatrix} d_x(x,y) \\ d_y(x,y) \end{bmatrix} = \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

Cambiamenti di coordinate nel sistema 2D:

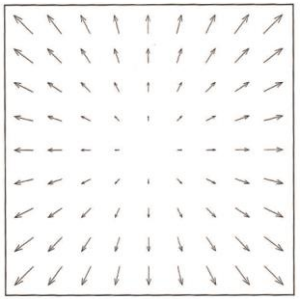
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} FT_x/Z \\ FT_y/Z \end{bmatrix}$$





Un esempio di Motion Field in cui è presente Boom:

Modelli di Movimento della Telecamera – Zoom (ρ)

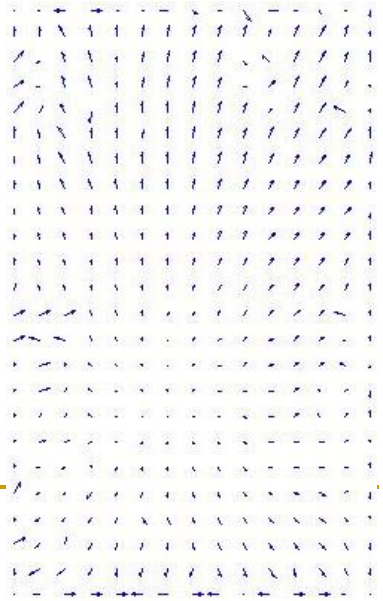


Cambiamenti di coordinate nel sistema 2D:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \rho x \\ \rho y \end{bmatrix}, \qquad \rho = \frac{F'}{F}$$

Approssimazione del motion field:

$$\begin{bmatrix} d_x(x,y) \\ d_y(x,y) \end{bmatrix} = \begin{bmatrix} (\rho - 1)x \\ (\rho - 1)y \end{bmatrix}$$



Zoom (ρ) = Dolly (Tz)???

- NO!
- Nello zoom si conservano le relazioni spaziali

Modelli di Movimento della Telecamera – Tilt (Rx) e Pan (Ry)

Cambiamenti di coordinate nel sistema 3D:

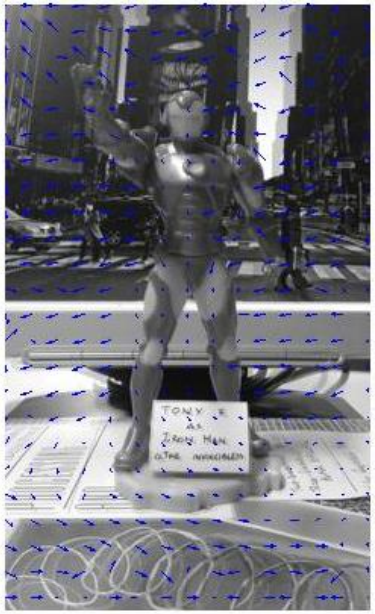
$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} 1 & 0 & \theta_y \\ 0 & 1 & -\theta_x \\ -\theta_y & \theta_x & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

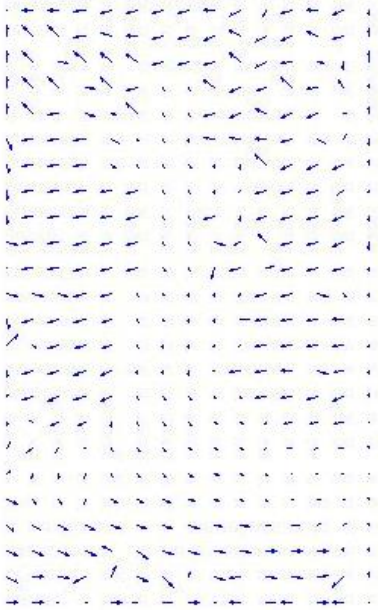
Approssimazione del motion field:

$$\begin{bmatrix} d_x(x,y) \\ d_y(x,y) \end{bmatrix} = \begin{bmatrix} \theta_y F \\ -\theta_x F \end{bmatrix}$$

Cambiamenti di coordinate nel sistema 2D:

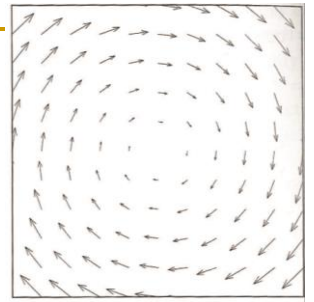
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} \theta_y F \\ -\theta_x F \end{bmatrix}$$





Un esempio di Motion Field in cui è presente Pan:

Modelli di Movimento della Telecamera – Roll (Rz)



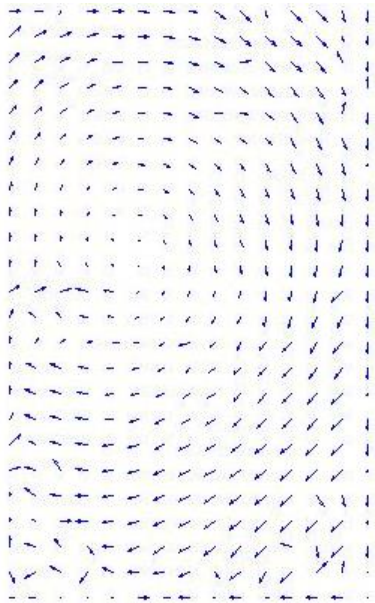
Cambiamenti di coordinate nel sistema 2D:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta_z & -\sin\theta_z \\ \sin\theta_z & \cos\theta_z \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

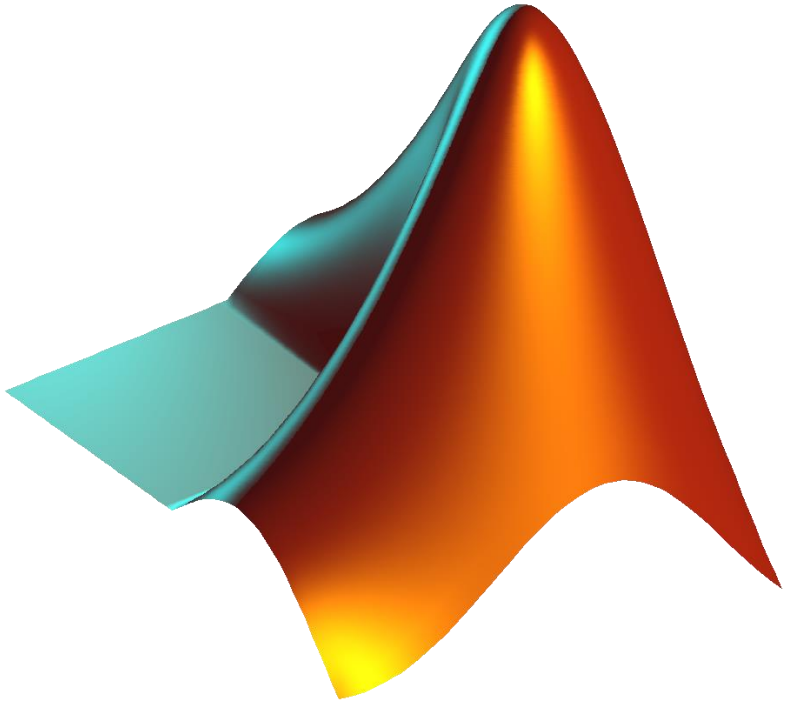
$$\approx \begin{bmatrix} 1 & -\theta_z \\ \theta_z & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}, se\ \theta_z \approx 0$$

Approssimazione del motion field:

$$\begin{bmatrix} d_x(x,y) \\ d_y(x,y) \end{bmatrix} = \begin{bmatrix} -\theta_z y \\ \theta_z x \end{bmatrix}$$



Modelli di Movimento della Telecamera



- MotionFieldScript.m
  - trackBoom.m
  - myZoom.m
  - myRotate.m
  - **myMotionField.m**

Modelli di Movimento della Telecamera – Modello a 4-parametri

- Consideriamo il caso in cui una telecamera effettua in sequenza: traslazione, pan, tilt, zoom e rotazione
- Usando le approssimazioni viste in precedenza è possibile mappare una **funzione a 4-parametri** che riassuma tutte le trasformazioni
- Se l'ordine dovesse cambiare, la formula rimarebbe valida, ma cambierebbero le relazioni fra i 4-parametri e i parametri delle singole trasformazioni

Modelli di Movimento della Telecamera
Modello a 4-parametri

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \rho \begin{bmatrix} \cos\theta_z & -\sin\theta_z \\ \sin\theta_z & \cos\theta_z \end{bmatrix} \begin{bmatrix} x + \theta_y F + t_x \\ y - \theta_x F + t_y \end{bmatrix} =$$
$$= \begin{bmatrix} c_1 & -c_2 \\ c_2 & c_1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} c_3 \\ c_4 \end{bmatrix}$$

- Questa formula è un caso particolare di **trasformazione affine** (*affine mapping*), normalmente a 6-parametri
- Prende il nome di **trasformazione geometrica** (*geometric mapping*), e caratterizza qualsiasi combinazione di scaling, rotazione e traslazione in 2D

Modelli di Movimento della Telecamera
Modello a 6-parametri (1)

- La rotazione di un oggetto attorno all'origine dello spazio 3D è data dalla matrice di rotazione

$$[\boldsymbol{R}] = [\boldsymbol{R_x}] \cdot [\boldsymbol{R_y}] \cdot [\boldsymbol{R_z}]$$

$$[R_x] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x \\ 0 & \sin\theta_x & \cos\theta_x \end{bmatrix} \qquad [R_y] = \begin{bmatrix} \cos\theta_y & 0 & \sin\theta_y \\ 0 & 1 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y \end{bmatrix}$$

$$[R_z] = \begin{bmatrix} \cos\theta_z & -\sin\theta_z & 0 \\ \sin\theta_z & \cos\theta_z & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Modelli di Movimento della Telecamera
Modello a 6-parametri (2)

- Pertanto, combinando insieme R_x, R_y, R_z si ottiene:

$$[R] = \begin{bmatrix} \cos\theta_y \cos\theta_z & \sin\theta_x \sin\theta_y \cos\theta_z - \cos\theta_x \sin\theta_z & \cos\theta_x \sin\theta_y \cos\theta_z + \sin\theta_x \sin\theta_z \\ \cos\theta_y \sin\theta_z & \sin\theta_x \sin\theta_y \sin\theta_z + \cos\theta_x \cos\theta_z & \cos\theta_x \sin\theta_y \sin\theta_z - \sin\theta_x \cos\theta_z \\ -\sin\theta_y & \sin\theta_x \cos\theta_y & \cos\theta_x \cos\theta_y \end{bmatrix}$$

- Assumendo piccole rotazioni si può porre:

$$[\boldsymbol{R}] \approx [\boldsymbol{R'}] = \begin{bmatrix} \boldsymbol{1} & \boldsymbol{-\theta_z} & \boldsymbol{\theta_y} \\ \boldsymbol{\theta_z} & \boldsymbol{1} & \boldsymbol{-\theta_x} \\ \boldsymbol{-\theta_y} & \boldsymbol{\theta_x} & \boldsymbol{1} \end{bmatrix}$$

Modelli di Movimento della Telecamera
Modello a 6-parametri (3)

- Il moto di un punto può essere descritto come:

$$X' = [R] \cdot X + [T]$$

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix}$$

- Per quanto visto, sebbene la matrice di rotazione R ha 9 elementi, essa è determinata soltanto da **3** angoli di rotazione

Modelli di Movimento della Telecamera
Modello a 6-parametri (4)

- Infine, utilizzando le equazioni delle proiezioni prospettiche, si ottiene:

$$x' = F\frac{(r_1x + r_2y + r_3F)Z + T_xF}{(r_7x + r_8y + r_9F)Z + T_zF}$$

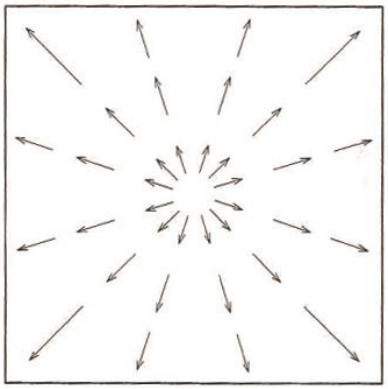
$$y' = F\frac{(r_4x + r_5y + r_6F)Z + T_yF}{(r_7x + r_8y + r_9F)Z + T_zF}$$

Modelli di Movimento della Telecamera
Modello a 6-parametri (5)

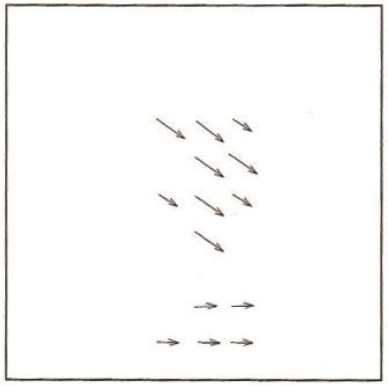
- Le relazioni che abbiamo ottenuto sono note come il **caso generale** del mapping di traslazioni e rotazioni arbitrarie da spazio 3D a spazio 2D
- Rispetto al *geometric mapping* (4-parametri) in questo **modello a 6-parametri** è possibile considerare anche **traslazioni lungo l'asse Z** e **rotazioni attorno ad assi arbitrari**

Rappresentazione del Movimento

- Come parametrizzare i *motion field*?
- **Rappresentazione globale**
  - Non funziona bene se nella scena sono presenti più oggetti che si muovono in maniera differente
- **Rappresentazione basata su pixel**
  - Richiede la stima di troppi vettori di movimento



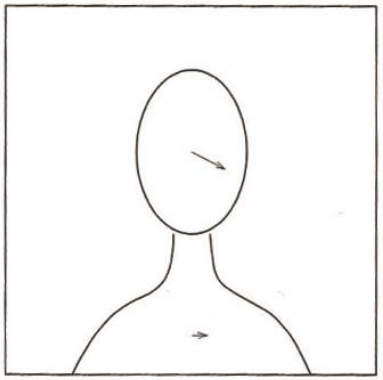
Rappresentazione globale



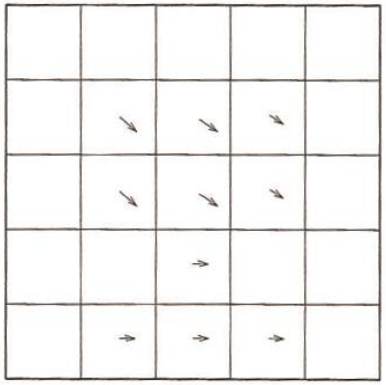
Rappresentazione su pixel

Rappresentazione del Movimento

- **Rappresentazione basata su regioni**
  - Segmentare le regioni e stimarne il movimento è computazionalmente oneroso
- **Rappresentazione basata su blocchi**
  - Buon compromesso fra accuratezze e complessità
  - Problemi di discontinuità lungo i bordi dei blocchi



Rappresentazione su regioni



Rappresentazione su blocchi

Criterio di Stima del Movimento Displaced Frame Difference (DFD)

- Quale criterio usare per **stimare i parametri** del modello di movimento?
  - Ne esistono vari, ci concentreremo sul **criterio basato sulla differenza fra frame discostati** (*displaced frame difference* – **DFD**)
  - Siano:
    - "Anchor frame I_1 *a* t_1" il frame iniziale
    - "Target frame I_2 *a* t_2" il frame su cui stimare il relativo movimento:
      - $t_1 > t_2$: *backward motion estimation*
      - $t_1 < t_2$: *forward motion estimation*

Criterio di Stima del Movimento Displaced Frame Difference (DFD)

- Siano inoltre:
  - Funzione di mapping: $w(x;a) = x + d(x;a)$
    - dove $a = [a_1, a_2, \dots, a_L]^T$ parametri di movimento (da stimare)

- Definiamo **Errore DFD**:
  - $E_{DFD}(a) = \sum_{x \epsilon \Lambda} \left| I_2\big(w(x;a)\big) - I_1(x) \right|^p$
    - Dove:
      - Λ : insieme dei pixel nell'anchor frame I_1
      - p : intero positivo; casi particolari:
        - $p = 1 \rightarrow E_{DFD}$ è detto ***mean absolute difference*** (**MAD**)
        - $p = 2 \rightarrow E_{DFD}$ è detto ***mean squared error*** (**MSE**)

Criterio di Stima del Movimento Displaced Frame Difference (DFD)

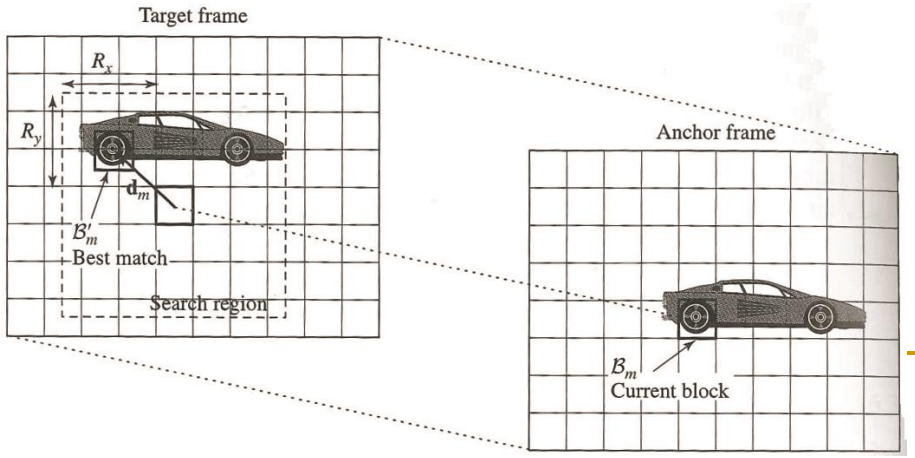
- Affinché la stima dei parametri a sia la migliore, si dovrà **minimizzare E_{DFD}**
  - Essendo a un vettore, dovremo in generale imporre che il gradiente di E_{DFD} valga 0
  - Ad esempio, per $p = 2$ il gradiente sarà:
    - $\frac{\delta E_{DFD}}{\delta a} = 2\sum_{x\epsilon\Lambda}\left(I_2\left(w(x;a)\right) - I_1(x)\right)\frac{\delta d(x)}{\delta a}\nabla I_2\left(w(x;a)\right)$
      - dove $\frac{\delta d}{\delta a} = \begin{bmatrix} \frac{\delta d_x}{\delta a_1} & \frac{\delta d_x}{\delta a_2} & \dots & \frac{\delta d_x}{\delta a_L} \\ \frac{\delta d_y}{\delta a_1} & \frac{\delta d_y}{\delta a_2} & \dots & \frac{\delta d_y}{\delta a_L} \end{bmatrix}^T$

Stima del Movimento
Algoritmi di Block-Matching (**BMA**)

- Consideriamo la rappresentazione del movimento basata su blocchi
  - I blocchi possono avere qualsiasi forma geometrica (solitamente quadrata), purché:
    - $\cup_{m \in M} B_m = \Lambda$, e $B_m \cap B_n = \emptyset, m \neq n$
  - Assumiamo che tutti i pixel di un blocco B_m si muovano nella stessa direzione (*modello traslazionale blockwise*): un MV per blocco
- Dato l'anchor frame B_m vogliamo trovare il target frame $B_m{}'$ che minimizza l'errore E_{DFD}

Stima del Movimento
Algoritmo di Block-Matching Esaustivo

- Vogliamo minimizzare
  - $E_m(d_m) = \sum_{x \epsilon B_m} |I_2(x + d_m) - I_1(x)|^p$
- BMA Esaustivo (**EBMA**):
  - Per trovare il migliore d_m confrontiamo (dopo averli calcolati) tutti i d_m fra l'anchor frame B_m e tutti i possibili target frame $B_m{}'$ all'interno di una regione di ricerca



Algoritmo di Block-Matching Esaustivo Complessità computazionale (1)

- Per ridurre il carico si può usare il *MAD error*
- Solitamente si usa un passo di ricerca pari a 1 pixel (*integer-pel accuracy search*)
- Sia $N \times N$ la dimensione del blocco e il raggio della regione di ricerca pari a $\pm R$, allora:
  - Dovendo fare una sottrazione, un valore assoluto e un addizione: per ogni blocco N^2 operazioni
  - Numero di operazioni per stimare un MV per un blocco: $\boldsymbol{(2R+1)^2 N^2}$

Algoritmo di Block-Matching Esaustivo Complessità computazionale (2)

- Siano $M \times M$ le dimensioni dell'immagine
  - Si avranno $(M/N)^2$ blocchi
- Il numero totale di operazioni sarà quindi:
  - $\boldsymbol{M^2(2R+1)^2}$
- Questo risultato è interessante… perché?

$\rightarrow\rightarrow\rightarrow$

Algoritmo di Block-Matching Esaustivo Complessità computazionale (3)

- Carico computazionale: $\boldsymbol{M^2(2R+1)^2}$
  - E' indipendente dalla dimensione dei blocchi!

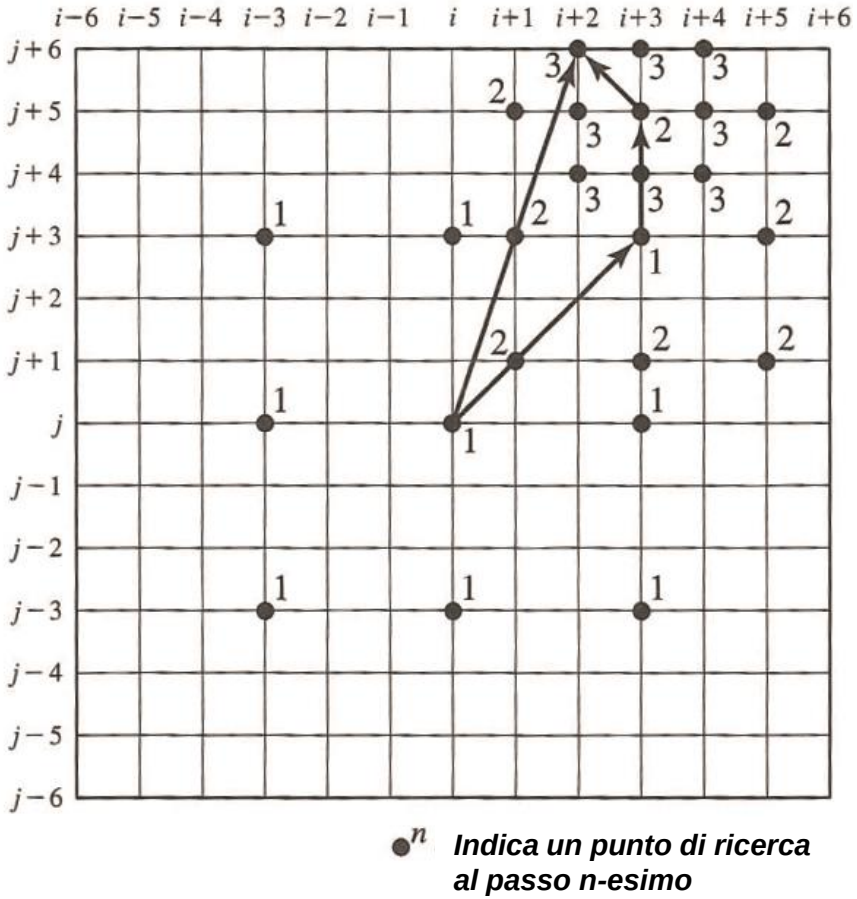
- Un esempio:
  - $M = 512,\ N = 16,\ R = 16$
    - $512^2(2 \cdot 16 + 1)^2 \cong 2.85 \times 10^8\ operazioni\ per\ frame$
  - In un video a 30 fps
    - $2.85 \times 10^8 \cdot 30 \cong 8.55 \times 10^9 operazioni\ al\ secondo$
- Serve un metodo più veloce!

Algoritmi di Block-Matching
Metodo Three-Step Search (1)

- Modifica all'EBMA:
  - Per trovare il migliore d_m confrontiamo (dopo averli calcolati) i d_m fra l'anchor frame B_m **e un sottoinsieme** dei possibili target frame $B_m{}'$ all'interno di una regione di ricerca
- Si inizia con un passo di ricerca R_0 uguale (o leggermente maggiore) della metà del raggio di ricerca R
- Ad ogni passo dell'algoritmo si riduce il passo di ricerca della metà, finché non sarà pari a 1

Algoritmi di Block-Matching
Metodo Three-Step Search (2)

- Al primo passo:
  - Si calcola il miglior MV su 9 punti di ricerca entro un raggio di ricerca R_0
- Dal secondo passo:
  - Si pone $R_{i+1} = R_i/2$
  - Si calcola il miglior MV su 8 punti di ricerca entro un raggio di ricerca R_{i+1}
  - Si reitera finché $R_n = 1$



Algoritmi di Block-Matching
Metodo Three-Step Search (3)

- Nota: a dispetto del nome, potrebbero esserci più di 3 passi di ricerca!
- Fissato R_0 si avranno al più $L = \lceil (\log_2 R_0) + 1 \rceil$ passi di ricerca
  - Ricavato L, si avranno $8L + 1$ punti di ricerca
- Confronto con EBMA:
  - Con $R = 32,\ R_0 = 16$
    - EBMA ha $(2 \cdot 32 + 1)^2 = 4225$ punti di ricerca
    - Con three-step search si hanno $8 \cdot \lceil \log_2 16 + 1 \rceil + 1 = 41$ punti di ricerca (~100 volte di meno!)

Stima del Movimento
Algoritmi basati su Features

- Algoritmo "***F**eatures from **A**ccelerated **S**egment **T**est* (**FAST**)":
  - Individuare dei punti salienti (**corner**) nel frame da usare come features per "tracciare" il movimento
    - Si analizza ad ogni passo un insieme di 16 punti con configurazione a cerchio di raggio 3 per classificare se un punto p sia o meno di corner
    - Data l'intensità l_p del punto p, se per almeno 12 punti contigui con intensità l_x si ottiene che $|l_x - l_p| > t$, con t soglia, allora p sarà considerato corner

Algoritmi basati su Features
Algoritmo FAST

- Si può velocizzare il test di corner detection con l'"**high-speed test**":
  - Verificare $|l_x - l_p| > t$ per i punti del cerchio in posizione 1 e 9, e successivamente in posizione 5 e 13
    - Se per almeno 2 di questi 4 punti $|l_x - l_p| \leq t$ il test fallisce e il punto p non può essere corner
- FAST è computazionalmente efficiente e più veloce di altri metodi (come SIFT), sebbene meno preciso
  - Questo lo rende adatto per applicazioni real-time!

Perché stimare il movimento?

- L'informazione sul movimento nella scena può tornare utile in vari modi, ad esempio:
  - Quality enhancement del video tramite **stabilizzazione**
  - La stabilizzazione aumenta anche il fattore di **compressione** del video
  - Ricostruzione della scena 3D a partire dalla proiezione 2D
  - …