

L5 – Esercitazione su trasformazioni prospettiche

Corso di Visione Artificiale

Francesco Valenti

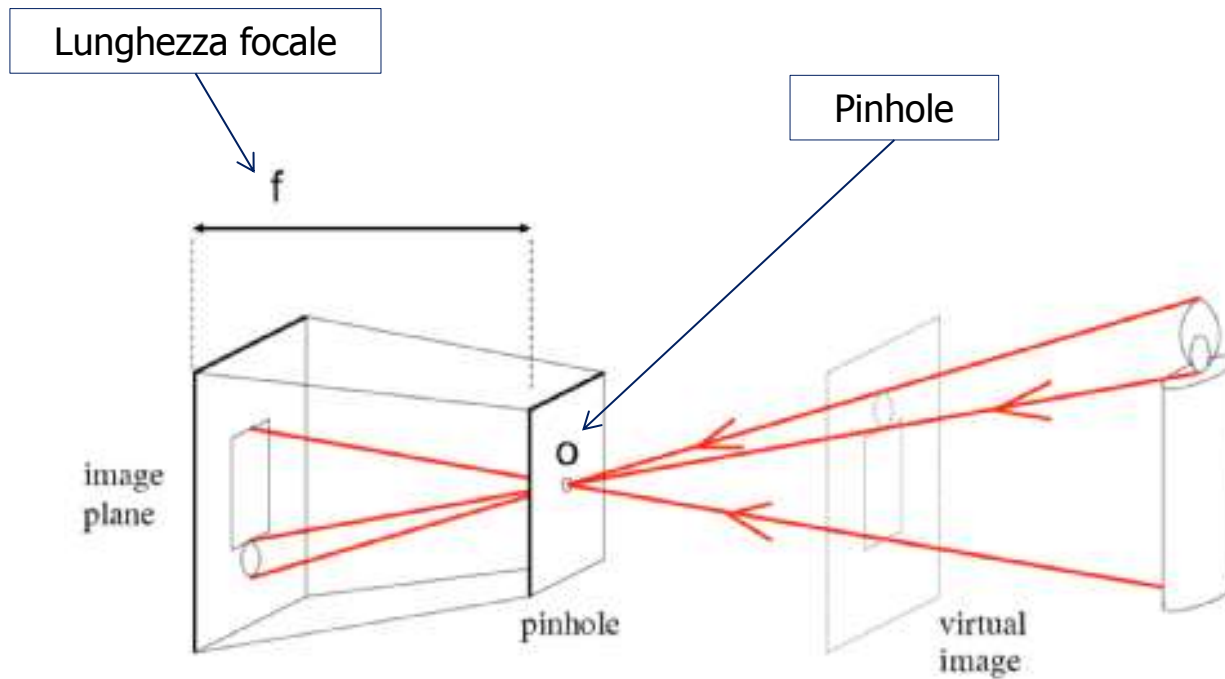
valenti@ce.unipr.it

ANNO ACCADEMICO 2019-2020

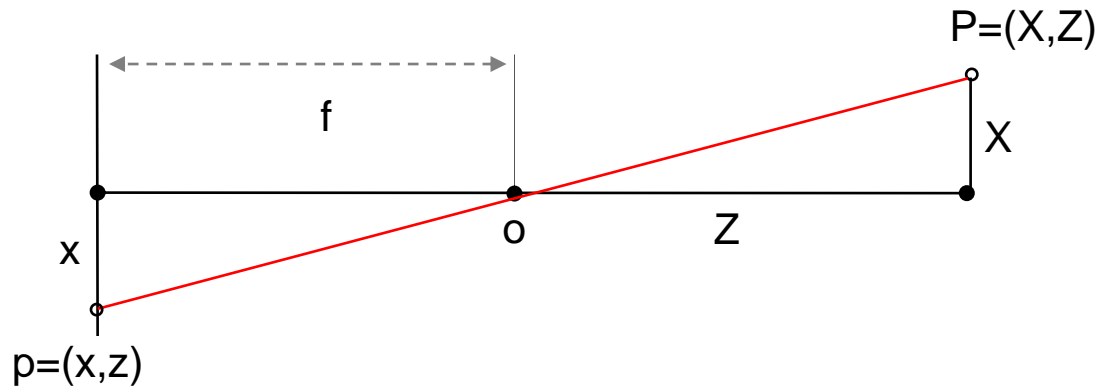
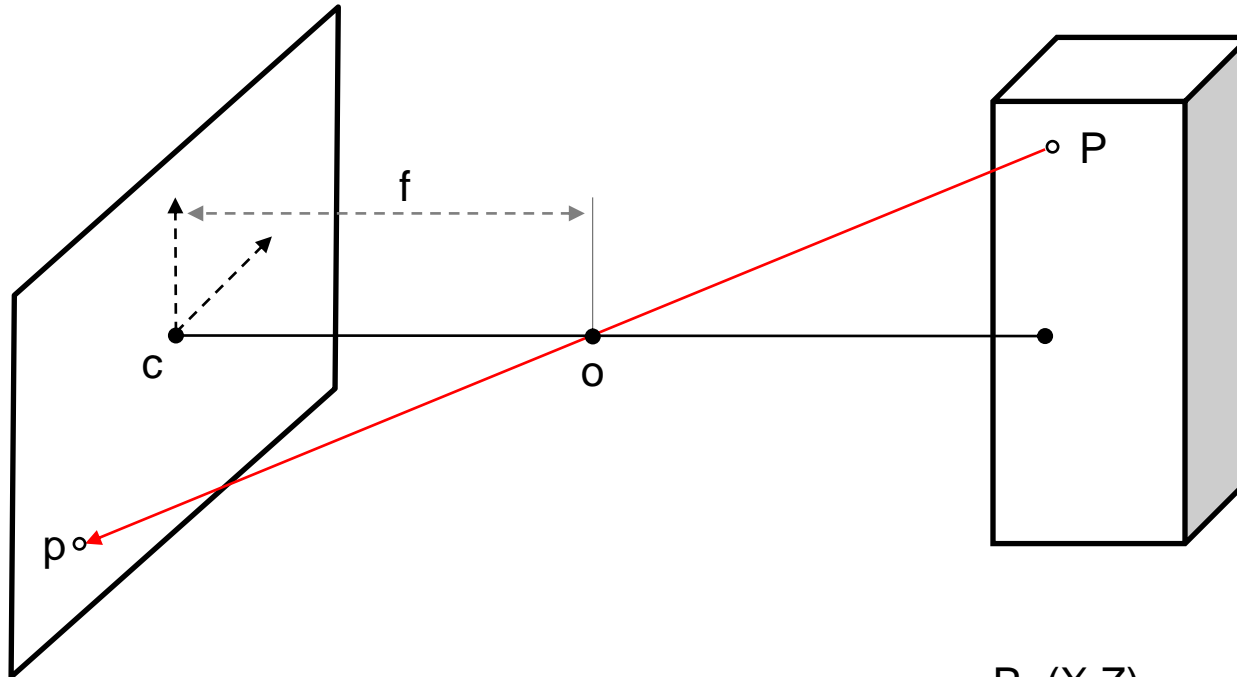
Sommario

- Modello pinhole
- La geometria del modello pinhole
 - Intrinseci
 - Estrinseci
- Perspective mapping
 - Esercitazione
- Inverse perspective mapping
 - Esercitazione

Modello pinhole

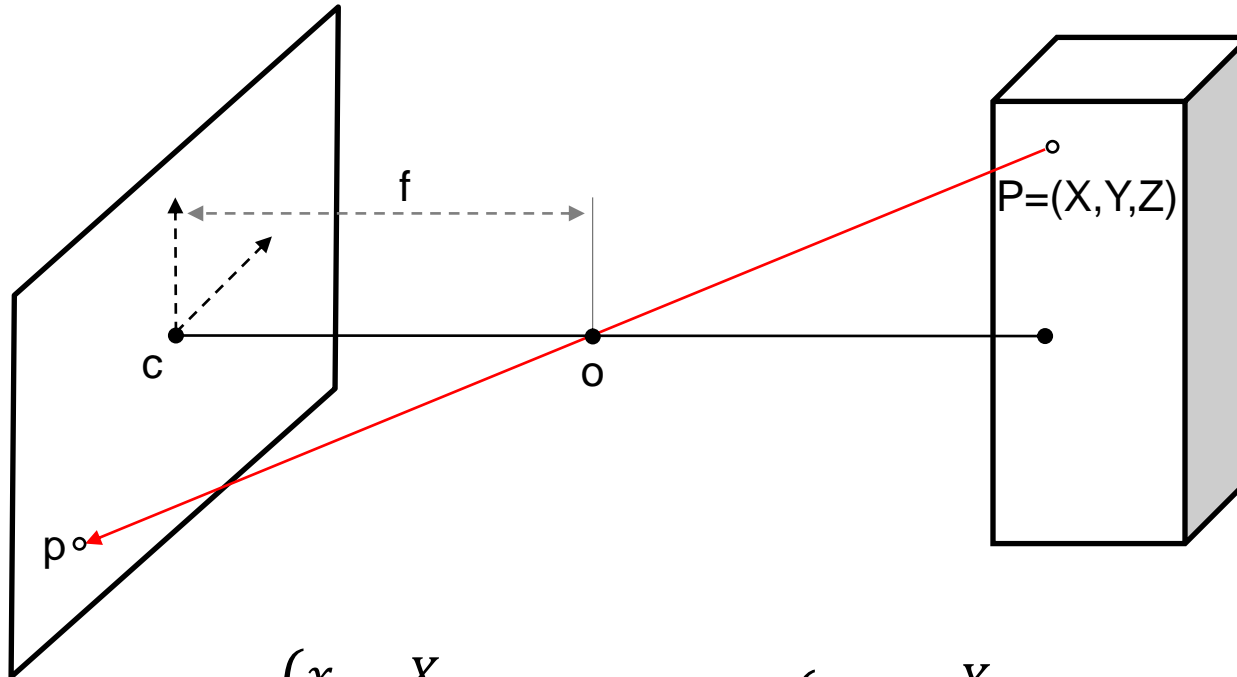


Modello pinhole



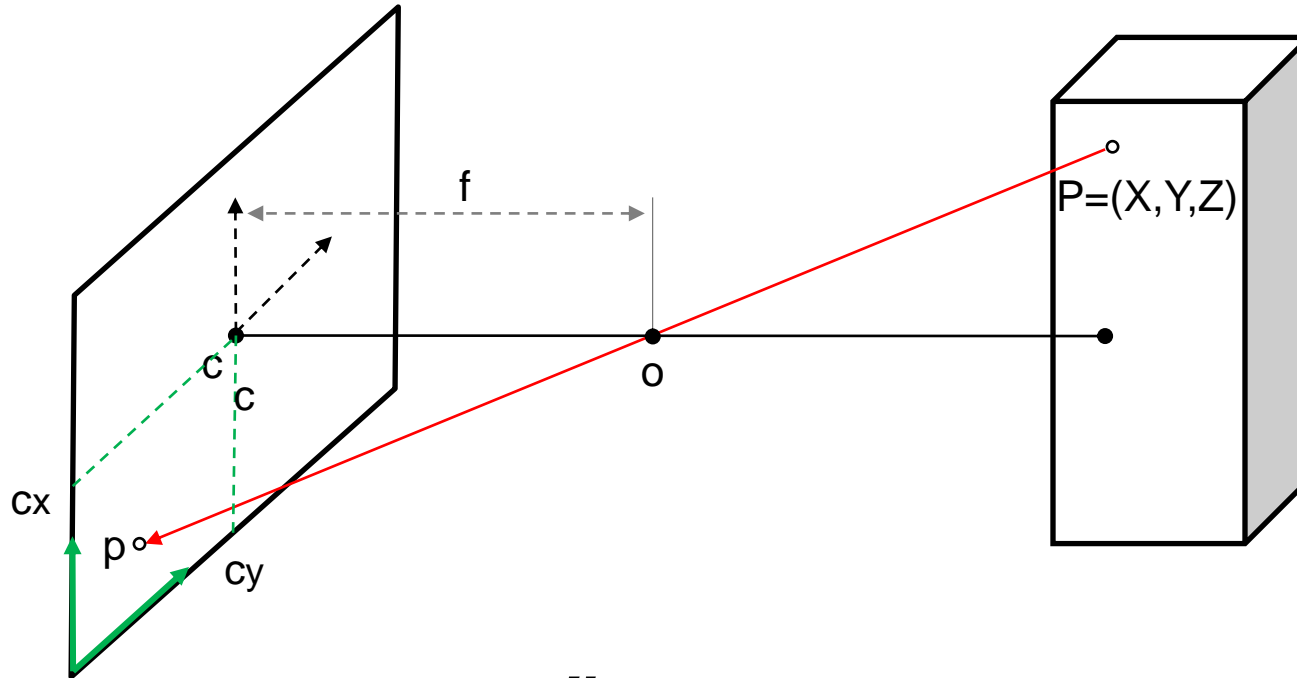
$$\frac{x}{f} = \frac{X}{Z}$$

Modello pinhole



$$\begin{cases} \frac{x}{f} = \frac{X}{Z} \\ \frac{y}{f} = \frac{Y}{Z} \end{cases} \longrightarrow \begin{cases} x = f \frac{X}{Z} \\ y = f \frac{Y}{Z} \end{cases}$$

Modello pinhole

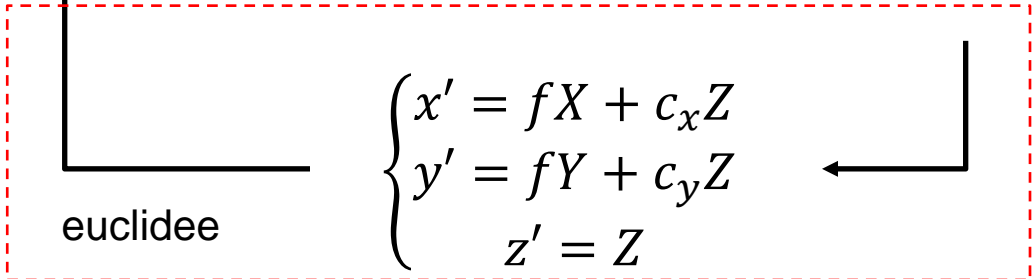


$$\begin{cases} x = f \frac{X}{Z} + c_x \\ y = f \frac{Y}{Z} + c_y \end{cases}$$

Modello pinhole

- Ma questa è una trasformazione non lineare... per conservare la linearità bisogna introdurre le coordinate omogenee

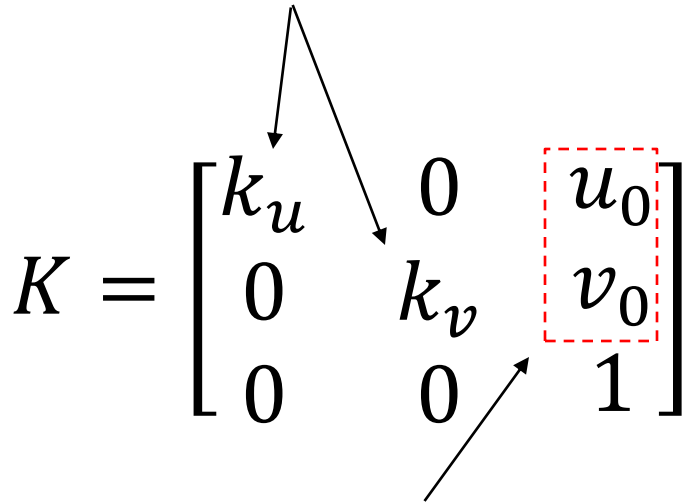
$$\begin{cases} x = f \frac{X}{Z} + c_x \\ y = f \frac{Y}{Z} + c_y \end{cases} \xrightarrow{\text{omogenee}} \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$



Geometria del modello pinhole

- Intrinseci

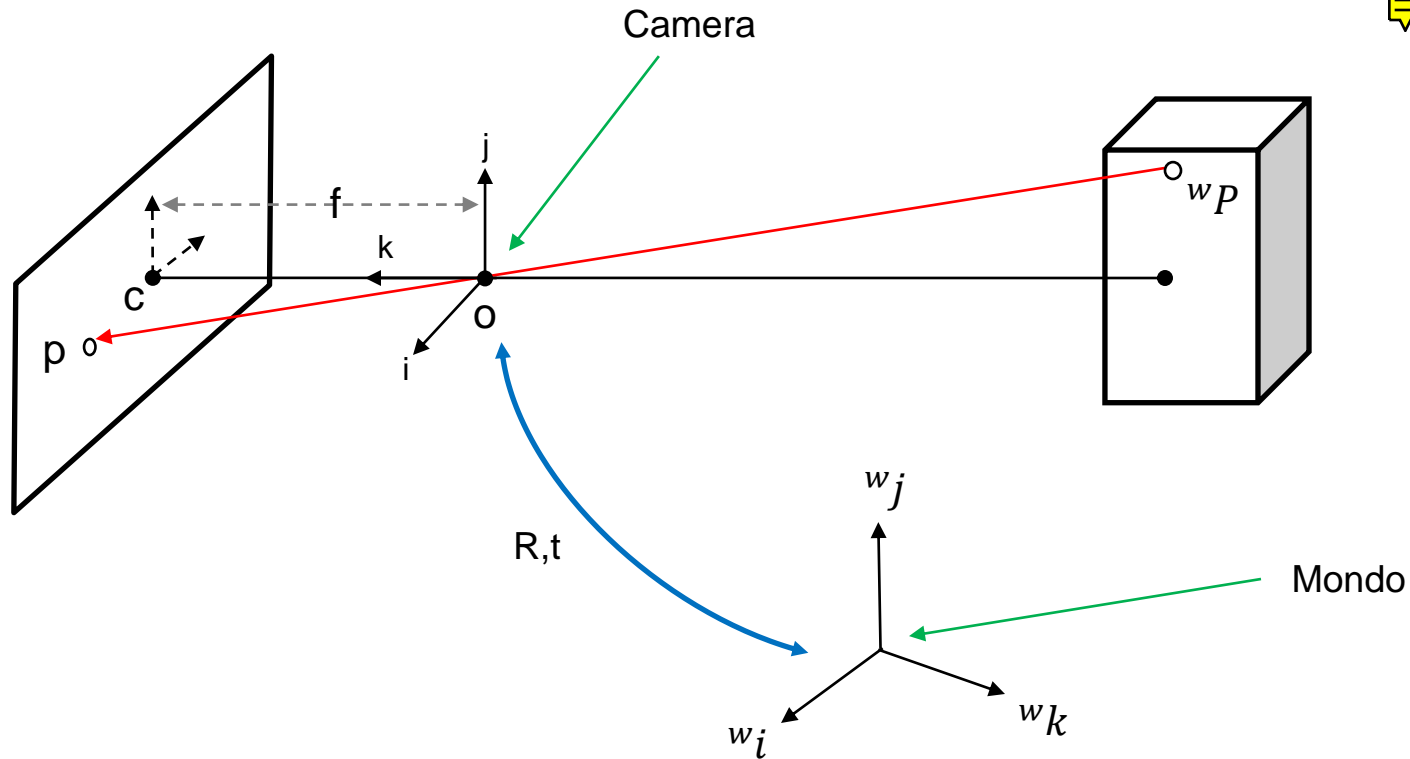
f = lunghezza focale
uguali quando i pixel sono quadrati


$$K = \begin{bmatrix} k_u & 0 & u_0 \\ 0 & k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

(cx, cy) = offset sul piano immagine

Geometria del modello pinhole

■ Estrinseci



Geometria del modello pinhole

- Estrinseci

$$T = \begin{bmatrix} R & t \\ \bar{0} & 1 \end{bmatrix} \in R^{4 \times 4}$$

- Matrice proiettiva


$$M = K \cdot \begin{bmatrix} I & 0 \end{bmatrix} \cdot T = \overset{\text{intrinseci}}{\boxed{K}} \cdot \underset{\text{estrinseci}}{\boxed{\begin{bmatrix} R & t \end{bmatrix}}} \in R^{3 \times 4}$$

Perspective mapping

- Dato un punto in coordinate mondo, trovare il corrispondente punto in coordinate immagine

$$p = M \cdot {}^wP = K \cdot [R \quad t] \cdot {}^wP$$

Perspective mapping

- Esercizio: data una nuvola di punti (point cloud) espressa in coordinate mondo e i parametri intrinseci ed estrinseci della camera, generare l'immagine relativa 
- La nuvola di punti è memorizzata sul file "scan.dat"
- Alcuni esempi di parametri della camera in si trovano in "params_front.dat", "params_left.dat", "params_right.dat", "params_back.dat",

NOTA: richiede OpenCv VIZ, quindi la versione 3 o superiore e la libreria VTK!

Perspective mapping

- Per chi non avesse OpenCv VIZ installato:
 1. Commentare la `#define USE_OPENCVVIZ` dentro al file `utils.h`
 2. Utilizzare *gnuplot* (linux&win) per visualizzare la nuvola di punti:

```
gnuplot  
splot "scan_gnuplot.dat"
```

Note su ex1

- Ogni riga del file "scan.dat" contiene le coordinate x,y,z di un punto 3D mondo:

-0.0434742 -4.82982 0.499645

0.0295245 -4.82834 0.541775

0.103245 -4.84538 0.584323

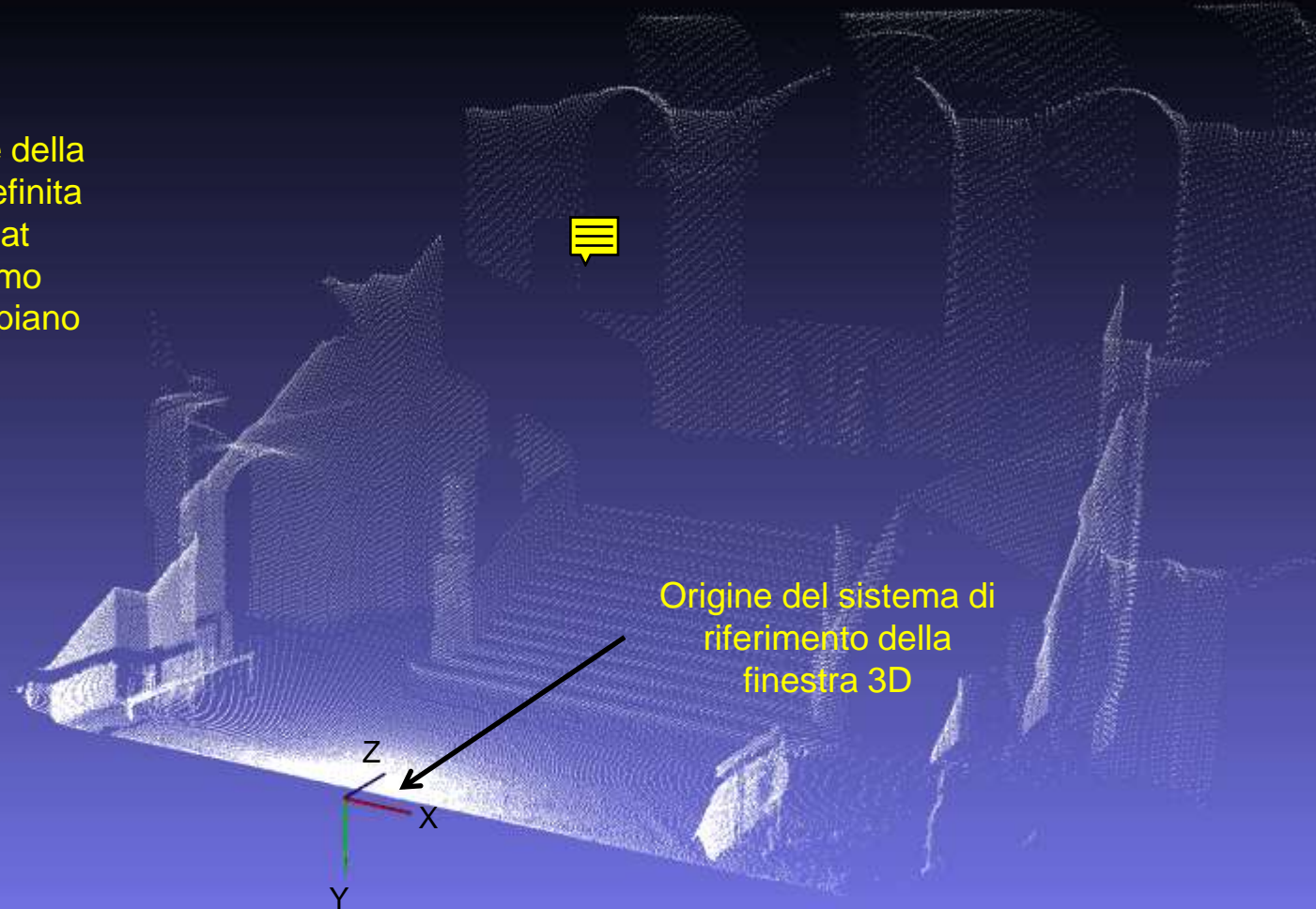
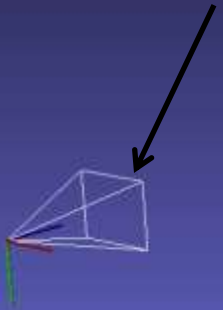
0.176457 -4.84095 0.626577

...

- L'insieme dei punti rappresenta la scansione di un edificio, di cui si riconoscono una scalinata, colonne, archi, ecc.
- La finestra "3D" di OpenCv Viz visualizza questi punti nel suo sistema cartesiano x,y,z

Note su ex1

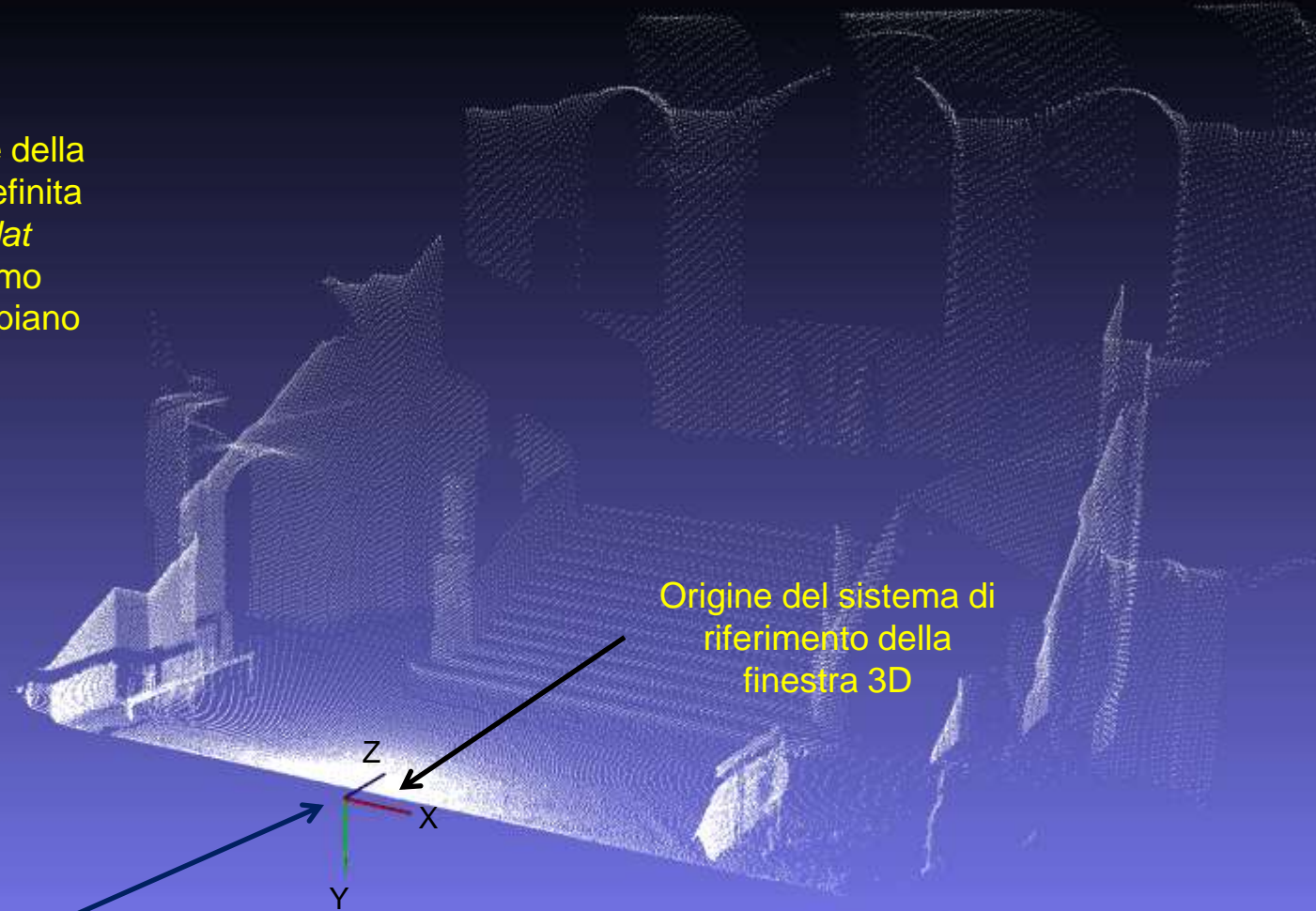
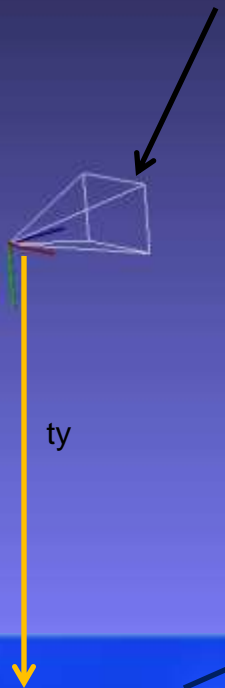
Posizione ed orientazione della nostra camera virtuale, definita dentro params_front.dat
E' quella su cui dobbiamo proiettare in punti 3D sul piano immagine



Origine del sistema di riferimento della finestra 3D

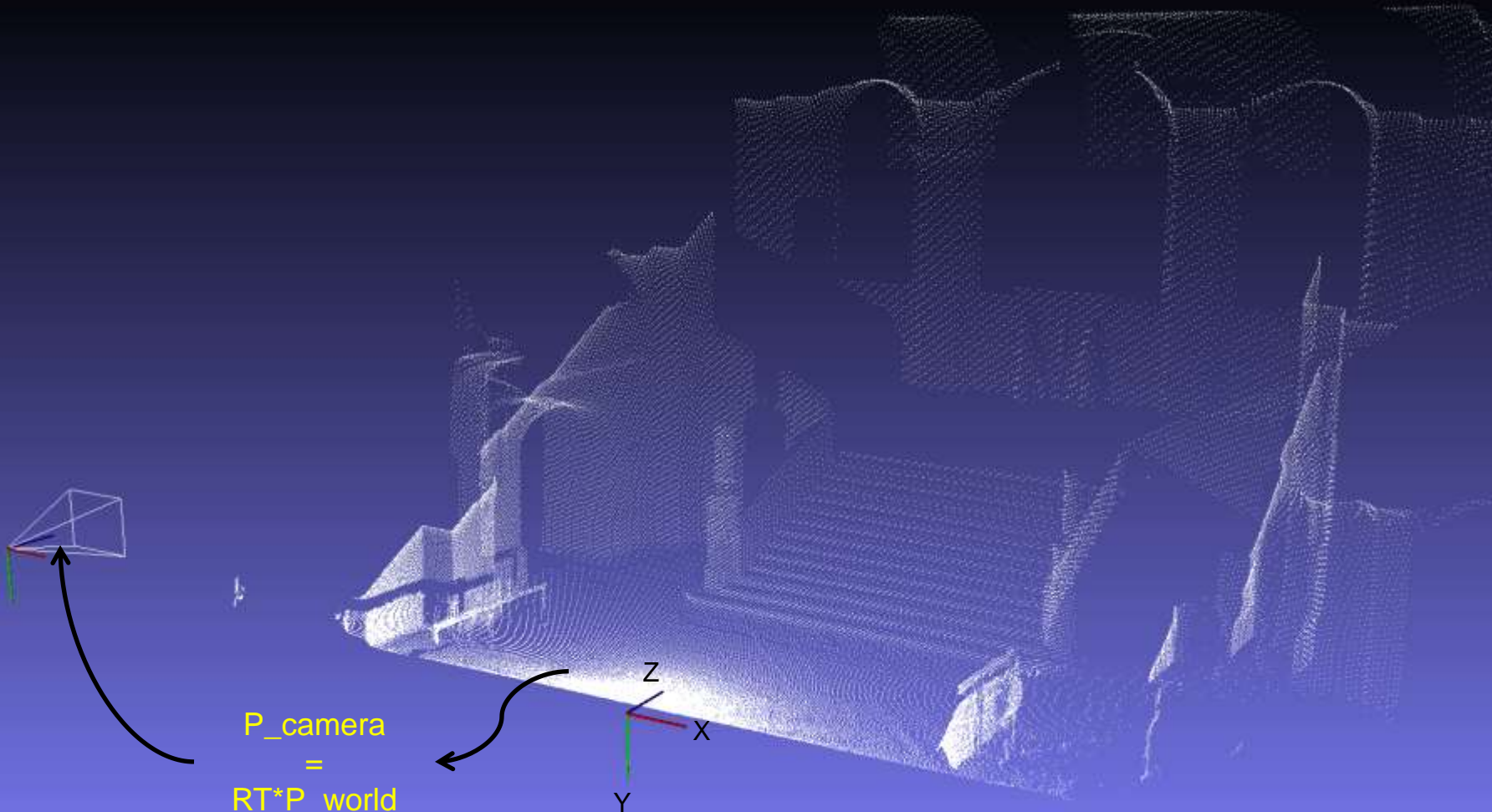
Note su ex1

Posizione ed orientazione della nostra camera virtuale, definita dentro *params_front.dat*
E' quella su cui dobbiamo proiettare in punti 3D sul piano immagine

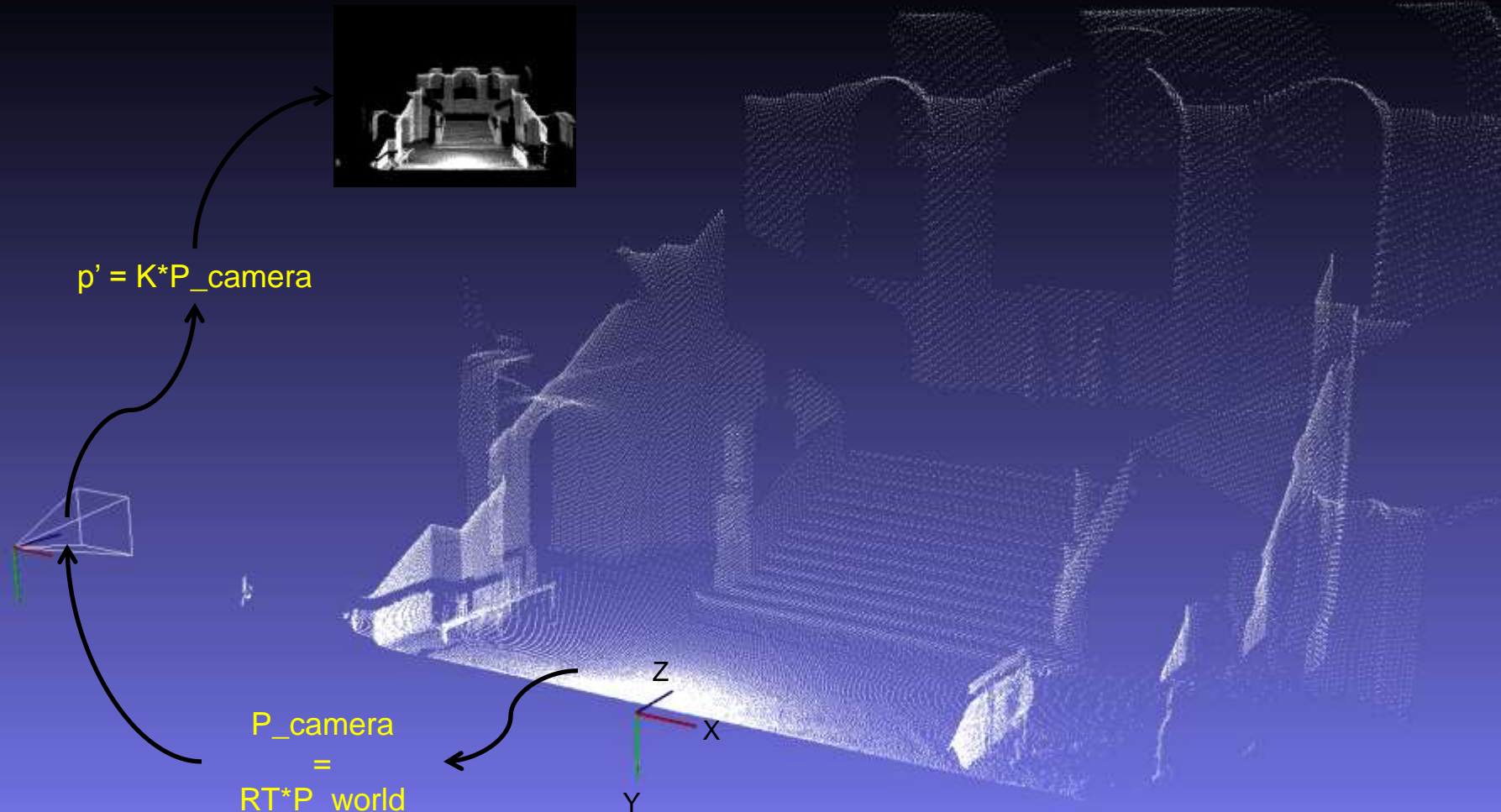


Origine del sistema di riferimento della finestra 3D

Note su ex1



Note su ex1



Note su ex1

- I file tipo "params_front.dat" contengono la calibrazione della camera:

```
640 480 //larghezza e altezza
400 400 //lunghezza focale in pixel
320 240 //centri ottici u0, v0
0.0 0.0 0.0 //orientazione rispetto a x,y,z
0.0 -5.0 -10.0 //posizione x,y,z
```

Perspective mapping

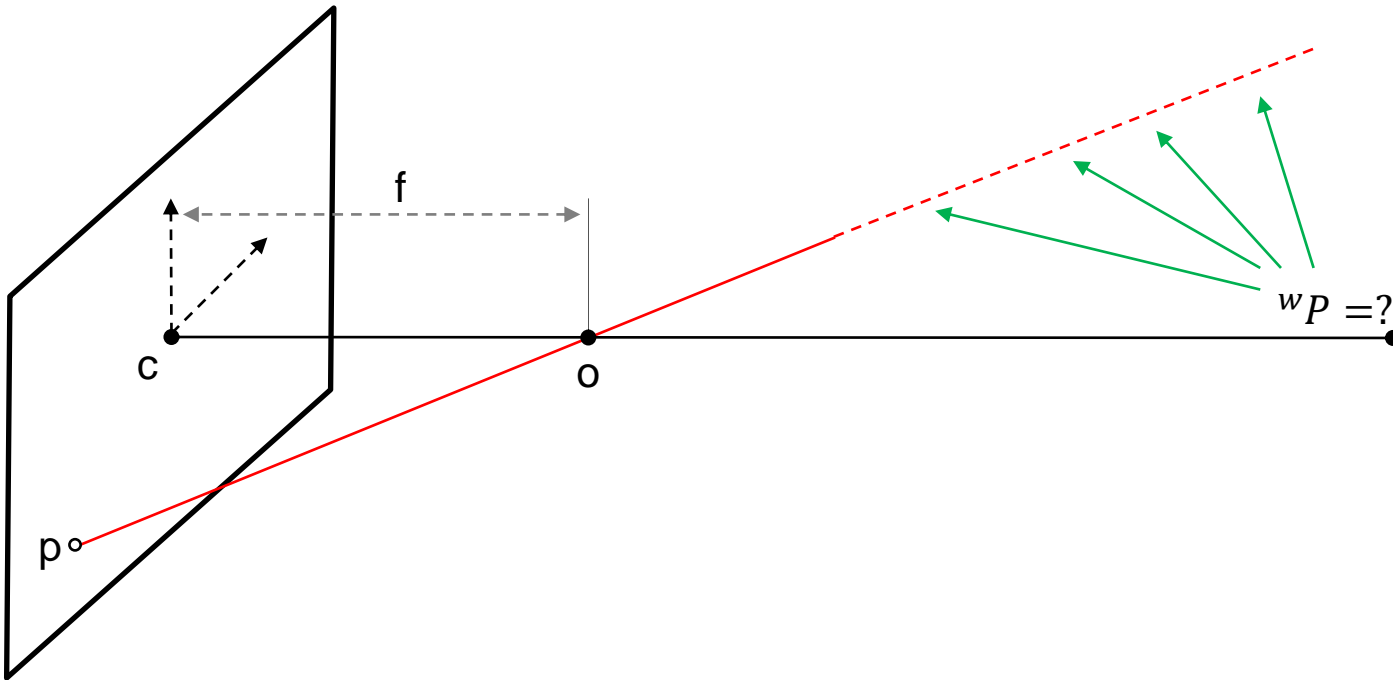
- Esercizio opzionale: creare una *sequenza* di viste lungo una circonferenza centrata sul baricento della nuvola di punti, orientate in modo che l'edificio sia sempre *visibile*



- Generare quindi una sequenza di parametri camera in cui gli intrinseci *non* cambiano, mentre gli estrinseci cambiano per rappresentare una rotazione intorno alla nuvola di punti

Inverse perspective mapping

- Dato un punto in coordinate immagine, trovare il corrispondente punto mondo



Inverse perspective mapping

- Le soluzioni del problema sono infinite, infatti come risultato otteniamo una retta
- Da un punto di vista matriciale dovremmo fare

$$p = M \cdot {}^wP \xrightarrow{\text{inversione}} {}^wP = M^{-1} \cdot p$$

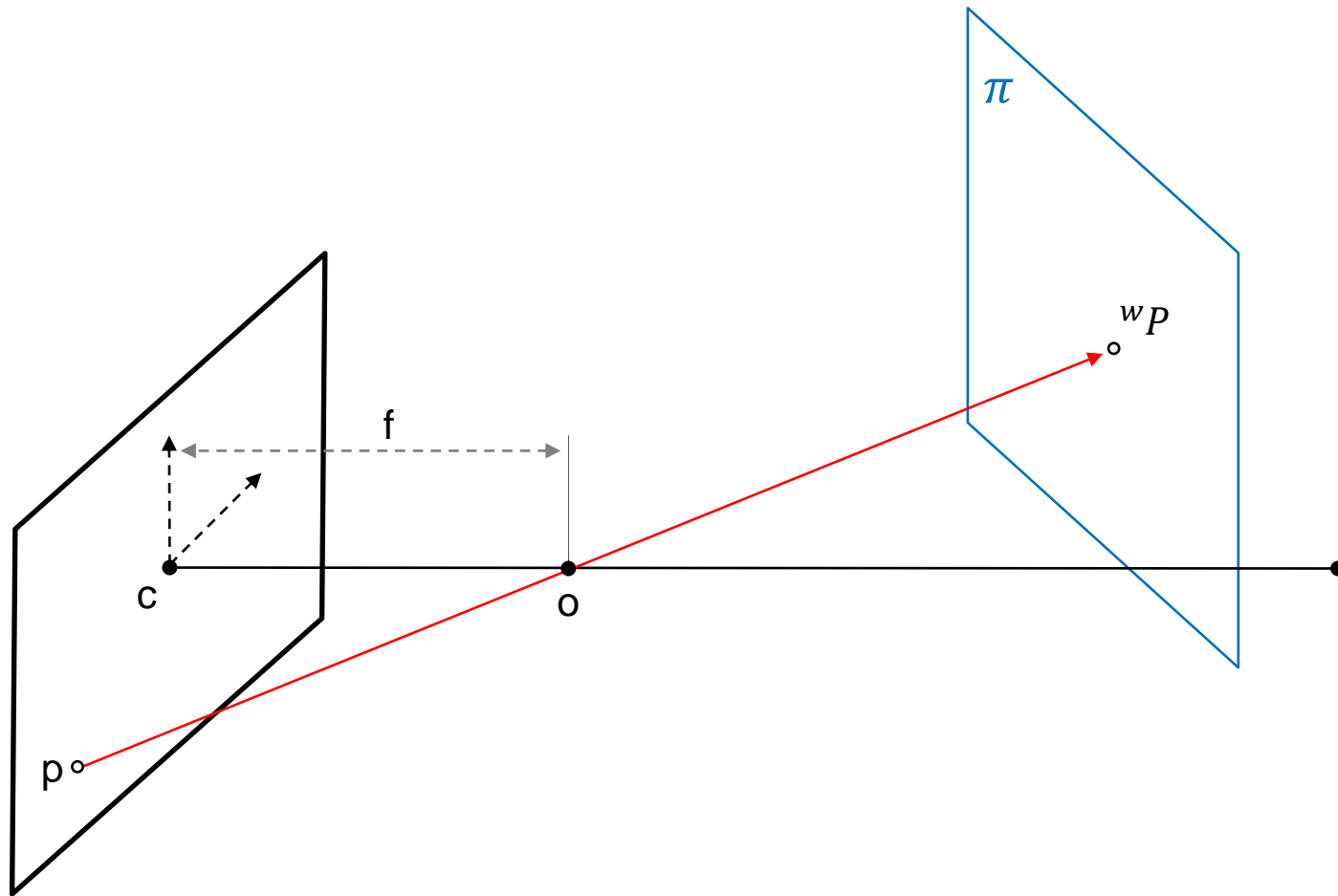
- ma non è possibile

Inverse perspective mapping

- Dobbiamo aggiungere un vincolo, ad esempio che il punto mondo appartenga ad un piano:

$$\pi: aX + bY + cZ + d = 0$$

Inverse perspective mapping



Inverse perspective mapping

- Da un punto di vista matriciale inserire questo vincolo **equivale ad aggiungere una riga a M**

$$p = \begin{bmatrix} x \\ y \\ z \\ 0 \end{bmatrix} = \begin{bmatrix} & M & \\ a & b & c & d \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Inverse perspective mapping

- La matrice ora è invertibile, quindi

$${}^wP = \begin{bmatrix} & M & \\ a & b & c & d \end{bmatrix}^{-1} \cdot p$$

Inverse perspective mapping

- Esercizio: data una immagine ed i parametri intrinseci ed estrinseci della camera, realizzare una IPM planare sul piano $y=0$



Inverse perspective mapping

- Utilizzare il piano $y=0$ ($a,b,c,d)=(0,1,0,0)$ per invertire M ed ottenere le rimanenti coordinate (X,Z) e partire da una coppia di pixel (u,v)
- Le (X,Z) ottenute rappresentano i punti sul piano del terreno delle immagini fornite