Cámaras

Modelo y calibración

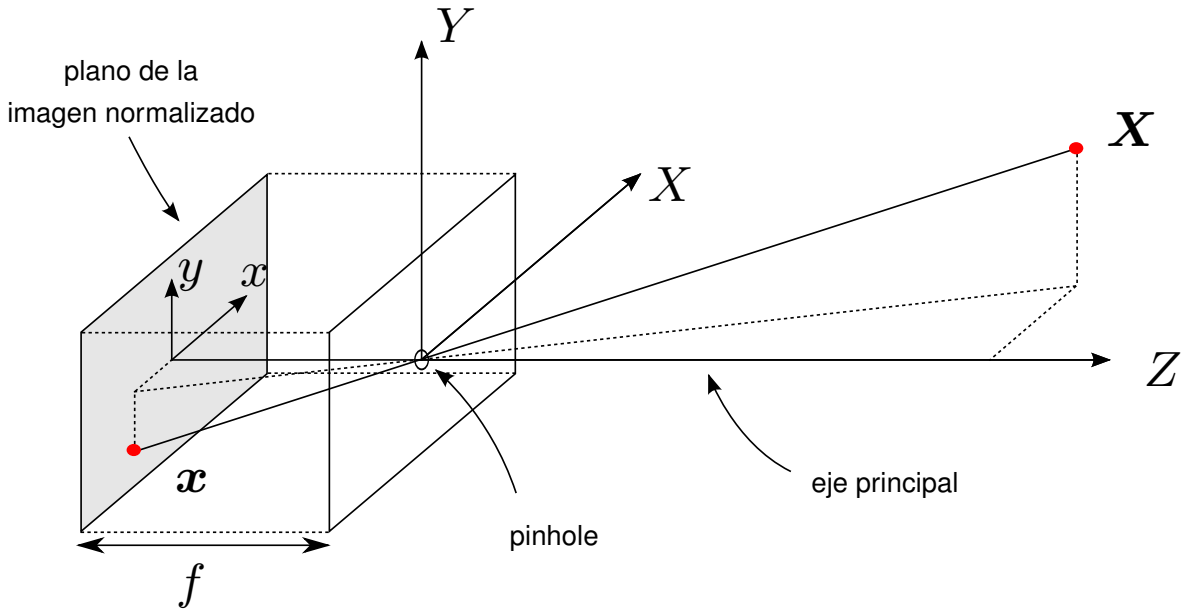
Gastón Araguás

Centro de Investigación en Informática para la Ingeniería
Universidad Tecnológica Nacional, F.R.C.
Córdoba, Argentina
http://ciii.frc.utn.edu.ar

1 de noviembre de 2013

Modelo de cámara

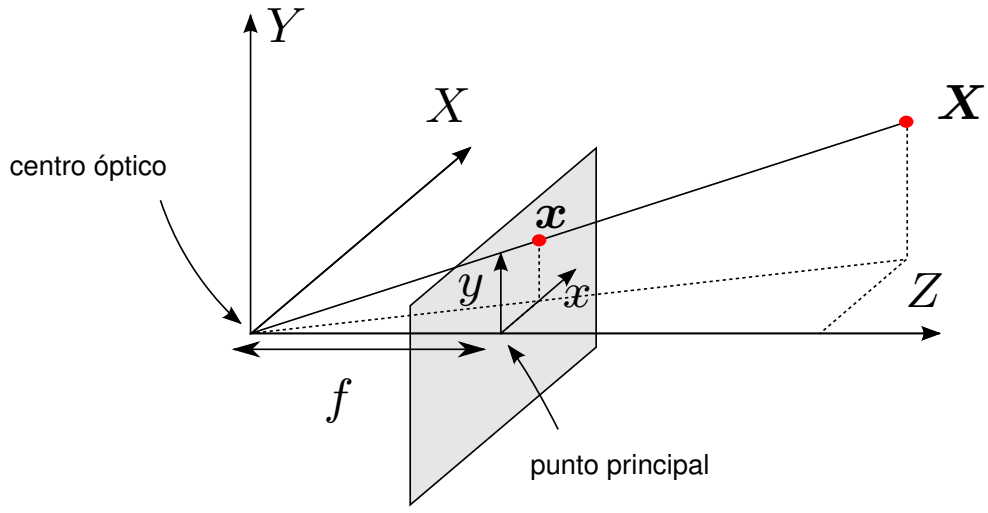
Modelo “pinhole”



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

Modelo de cámara

Proyección central



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

- El plano de la imagen esta en $z = f$, la distancia focal.
- Por triángulos semejantes se cumple $\frac{x}{f} = \frac{X}{Z}$ y $\frac{y}{f} = \frac{Y}{Z}$.
- En coord. homogeneas $\boldsymbol{X} = [X, Y, Z, 1]^T$ y $\boldsymbol{x} = [fX, fY, Z]^T$.
- Mapea $\mathbb{P}^3 \rightarrow \mathbb{P}^2$.

Modelo de cámara

Ecuaciones del modelo “pinhole”

Representación matricial del mapeo

Sea $\boldsymbol{X}$ un punto en el espacio, su proyección es

$$\boldsymbol{x} = \begin{bmatrix} fX \\ fY \\ Z \end{bmatrix} = \begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \boldsymbol{x} = P\boldsymbol{X}$$

con P una matriz de proyección de 3×4 llamada *matriz de la cámara*

Offset en el *punto principal*

La proyección central será $[X, Y, Z]^T \rightarrow [f\frac{X}{Z} + c_x, f\frac{Y}{Z} + c_y]$, luego

$$P = \begin{bmatrix} f & & c_x & 0 \\ & f & c_y & 0 \\ & & 1 & 0 \end{bmatrix} = K[I|\mathbf{0}]$$

K es la matriz de calibración y (c_x, c_y) se llama punto principal.

Modelo de cámara

Ecuaciones del modelo “pinhole”

Rotación y traslación de la cámara

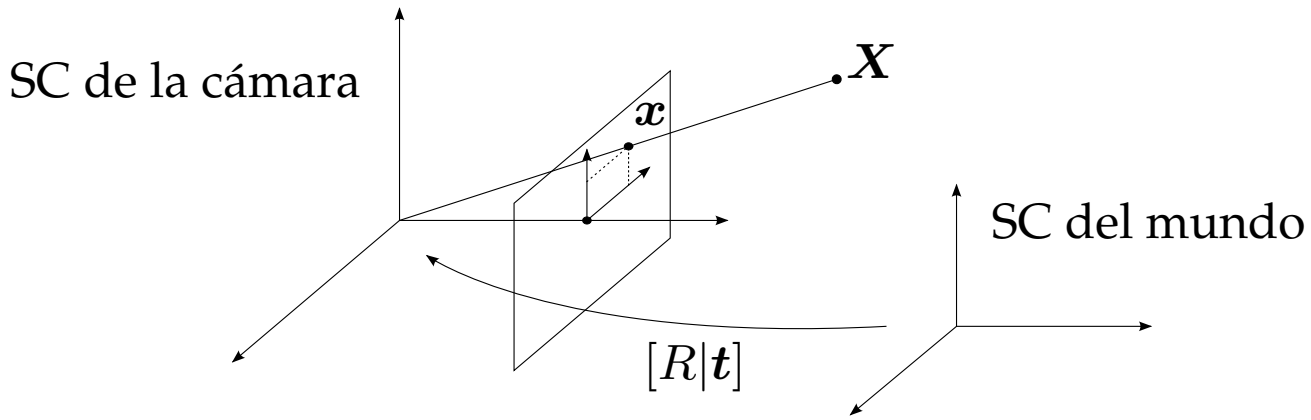
En general...

$$\boldsymbol{X}_{cam} = \begin{bmatrix} R & \boldsymbol{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \boldsymbol{X}$$

la proyección será

$$\boldsymbol{x} = K[I|\mathbf{0}]\boldsymbol{X}_{cam} = K[R|\boldsymbol{t}]\boldsymbol{X} = P\boldsymbol{X}$$

P tiene 9 GDL, 3 de la matriz K, 3 de la rotación (3D) y 3 de la traslación.



Modelo de cámara

Ecuaciones del modelo “pinhole”

Sistema de coordenadas en píxeles (CCD)

Mediante un escalado se pasa al sistema de píxeles

$$\boldsymbol{u} = [u, v, 1]^T = m\boldsymbol{x}$$

Si los píxeles no son cuadrados se escala diferente en x e y

$$u = f\frac{X}{Z}m_x + c_x m_x; \quad v = f\frac{Y}{Z}m_y + c_y m_y$$

esto modifica la matriz de calibración K quedando

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_x & 0 & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} [R|\boldsymbol{t}]\boldsymbol{X} = K[R|\boldsymbol{t}]\boldsymbol{X}$$

con $\alpha_x = fm_x$, $\alpha_y = fm_y$ en unidades de píxeles, y $x_0 = c_x m_x$ y $y_0 = c_y m_y$ el punto principal en el sistema de píxeles.

Modelo de cámara

Ecuaciones del modelo “pinhole”

Ejes de la cámara no perpendiculares

$$\boldsymbol{u} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} [R|\boldsymbol{t}]\boldsymbol{X} = K[R|\boldsymbol{t}]\boldsymbol{X} = P\boldsymbol{X}$$

P de una cámara CCD tiene 11 GDL (3×4 menos el factor de escala).

Cámara proyectiva finita

P puede escribirse

$$P = KR[I|R^{-1}\boldsymbol{t}]$$

Una cámara P, donde KR es una matriz de 3×3 no singular como la anterior se llama cámara proyectiva finita

Modelo de cámara

Anatomía de P

Rotación y matriz K

Sea M la submatriz 3×3 de una cámara finita P tal que

$$P = M[I|\boldsymbol{p}_4]$$

Descomponiendo a M por factorización RQ, donde R es una triangular superior y Q una matriz ortogonal, se obtiene K y R

Punto central

C es punto central de P si $PC = 0$, por lo que

$$K[R|\boldsymbol{t}]C = KR\tilde{\boldsymbol{C}} + K\boldsymbol{t} = 0 \Rightarrow \tilde{\boldsymbol{C}} = -R^T\boldsymbol{t}$$

o

$$\tilde{\boldsymbol{C}} = -M^{-1}\boldsymbol{p}_4$$

$\tilde{C}$ punto central en coord. inhomogeneas.

Modelo de cámara

Anatomía de P

Rotación y matriz K

Sea M la submatriz 3×3 de una cámara finita P tal que

$$P = M[I|\boldsymbol{p}_4]$$

Descomponiendo a M por factorización RQ, donde R es una triangular superior y Q una matriz ortogonal, se obtiene K y R

Punto central

$\boldsymbol{C}$ es punto central de P si $P\boldsymbol{C} = 0$, por lo que

$$K[R|\boldsymbol{t}]\boldsymbol{C} = KR\tilde{\boldsymbol{C}} + K\boldsymbol{t} = 0 \Rightarrow \tilde{\boldsymbol{C}} = -R^T\boldsymbol{t}$$

o

$$\tilde{\boldsymbol{C}} = -M^{-1}\boldsymbol{p}_4$$

$\tilde{C}$ punto central en coord. inhomogeneas.

Modelo de cámara

Distorciones ópticas

Sean x y $\hat{x}$ los puntos ideal y con distorsión

Distorsión radial

$$\hat{x} = x + x\left[k_1\left(x^2+y^2\right) + k_2\left(x^2+y^2\right)^2\right]$$
$$\hat{y} = y + y\left[k_1\left(x^2+y^2\right) + k_2\left(x^2+y^2\right)^2\right]$$

donde k_1 y k_2 son los coeficientes de distorsión radial; suponiendo para simplificar $\gamma = 0$ y operando

$$\hat{u} = u + (u-u_0)\left[k_1\left(x^2+y^2\right) + k_2\left(x^2+y^2\right)^2\right]$$
$$\hat{v} = v + (v-v_0)\left[k_1\left(x^2+y^2\right) + k_2\left(x^2+y^2\right)^2\right]$$

Modelos más complejos contemplan k_3, k_4 y k_5, y p_1, p_2 (coeficientes de distorción tangencial).

Calibración

Método de Zhang[1]

Homografía entre plano e imagen

Sea $\boldsymbol{X}^w$ un punto 3D sobre el plano $Z = 0$ en el sistema de coordenadas del mundo, entonces $\boldsymbol{X}^w = [X, Y, 1]^T$

$$s\boldsymbol{x} = K[\boldsymbol{r}_1\, \boldsymbol{r}_2\, \boldsymbol{t}]\boldsymbol{X}^w = H\boldsymbol{X}^w$$

$$H = \lambda K[\boldsymbol{r}_1\, \boldsymbol{r}_2\, \boldsymbol{t}]$$

$\boldsymbol{r}_1$ y $\boldsymbol{r}_2$ columnas de la matriz de rotación R (ortogonales).

Calibración

Método de Zhang

Algoritmo

- Estimar H usando DLT
- Usando varias imágenes maximizar LE minimizando:

$$\sum_{i}^{N}\sum_{j}^{P} d(\boldsymbol{x}_{ij}, \hat{\boldsymbol{x}}(K, k_1, k_2, R_i, \boldsymbol{t}_i, \boldsymbol{X}_j^w))$$

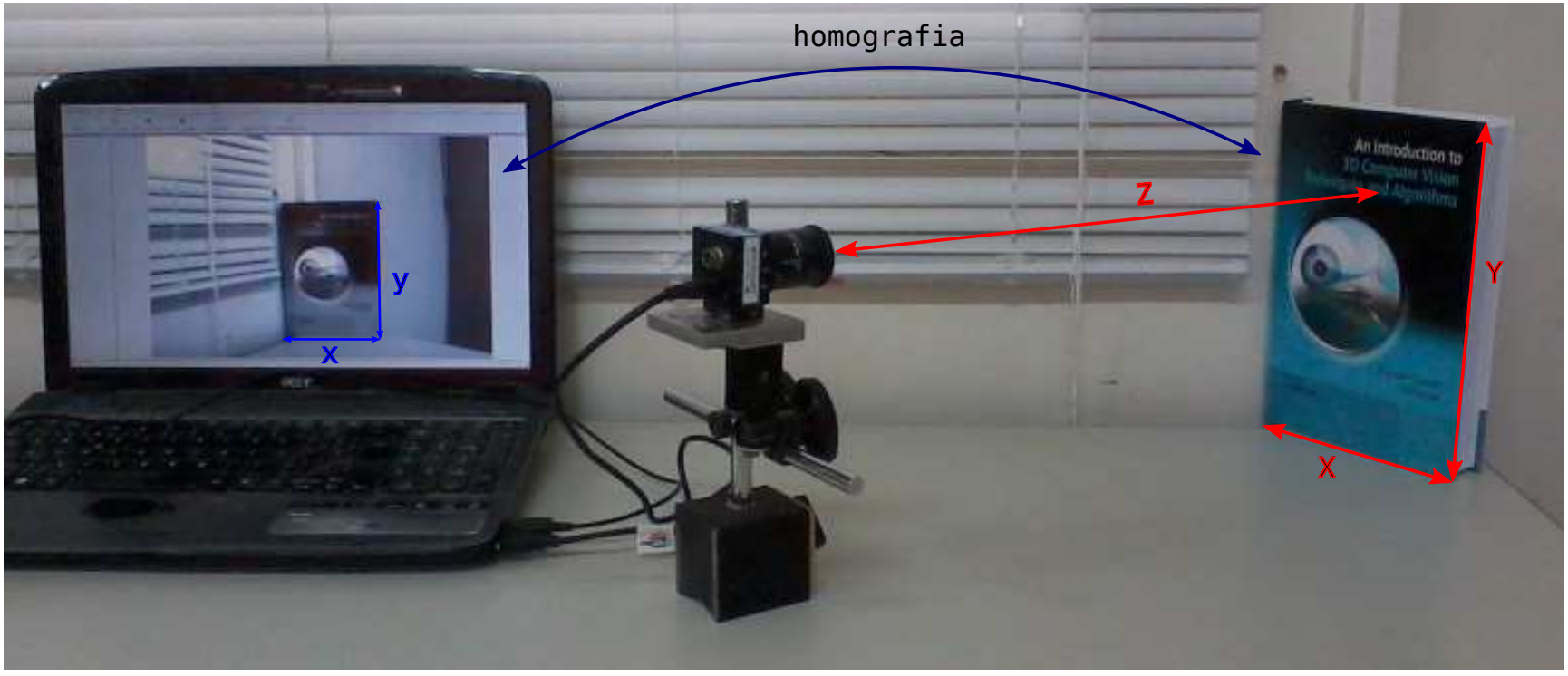
donde $\hat{\boldsymbol{x}}(K, k_1, k_2, R_i, \boldsymbol{t}_i, \boldsymbol{X}_j^w)$ es la proyección del punto 3D $\boldsymbol{X}_j^w$, en la $i-$ésima imagen

Cada imagen aporta 8 GDL, de los cuales 3 son de la rotación y 3 de la traslación, los dos restantes ajustan los parámetros intrínsecos.

Calibración

Calibración simple

Asumiendo $\gamma = 0$ y sin considerar la distorción de la lente...



$$\alpha_x = \frac{x}{X}Z \qquad \alpha_y = \frac{y}{Y}Z \qquad x_0 = w/2 \qquad y_0 = h/2$$

Calibración

Calibración simple

Calibración simple

$$K = \begin{bmatrix} 600 & 0 & 372 \\ 0 & 600 & 240 \\ 0 & 0 & 1 \end{bmatrix}$$

$$D = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Calibración completa (8 imágenes)

$$K = \begin{bmatrix} 611{,}47746477 & 0 & 432{,}3113038 \\ 0. & 613{,}15654266 & 208{,}46520386 \\ 0 & 0 & 1 \end{bmatrix}$$

$$D = \begin{bmatrix} -2{,}57e-01 & 1{,}44e-01 & -1{,}70e-04 & 4{,}34e-03 & 3{,}24e-02 \end{bmatrix}$$

Python y OpenCV

Read, show, write

Documentación online: `http://docs.opencv.org`

```
import cv2 as cv

cv.namedWindow("RoMAA", 1)

img = cv.imread('romaa.jpg')

cv.imshow("RoMAA", img)
cv.waitKey()

gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
cv.imshow("RoMAA", gray)
cv.waitKey()

cv.imwrite('romaa_gray.jpg', gray)

cv.destroyAllWindows()
```

Python y OpenCV

Video capture

```
# coding: utf-8
import cv2 as cv

cap = cv.VideoCapture(0)
#cap = cv.VideoCapture('video.avi')

while True:
    ret, im = cap.read()
    cv.imshow('video test', im)

    key = cv.waitKey(33) #milisegundos
    if key == 1048603: #ESC
        break

cv.destroyAllWindows()
```

Calibración

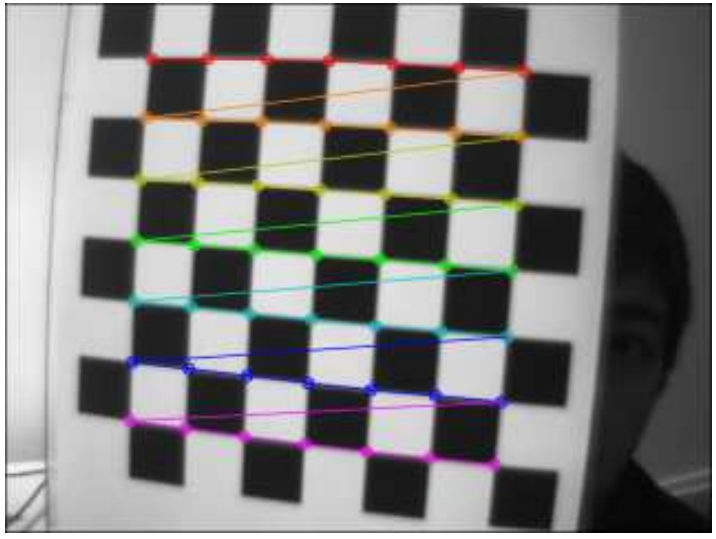
Calibración desde lista de imágenes

```
import cv2 as cv

"""
crear pattern_points con las dimensiones
del patron
pattern_points = [ ... ]
"""
import sys
img_names= sys.argv[1:]
for img_name in img_names:
    img = cv.imread(img_name, 0)
    found, corners = cv.findChessboardCorners(img,pattern_size)
    if found:
      vis = cv.cvtColor(img, cv.COLOR_GRAY2BGR)
      cv.drawChessboardCorners(vis,pattern_size,corners,found)
      img_points.append(corners.reshape(-1, 2))
      obj_points.append(pattern_points)

h, w = img.shape[:2]
rms,camera_matrix,dist_coefs,rvecs,tvecs=\\
        cv.calibrateCamera(obj_points,img_points,(w,h))
```

Calibración

Rectificación de lista de imágenes

```
for fn in img_names:
    print 'undistorting %s ... ' %fn,
    img = cv.imread(fn, 0)
    warped_img = cv.undistort(img, camera_matrix, dist_coefs)
    cv.imshow("undistort", warped_img)
    cv.waitKey()
    print 'ok'
```



Calibración

Propuesta: a partir de `calibrate.py`, calibrar una cámara y mostrar video rectificado.

```
# coding: utf-8
import cv2 as cv

cap = cv.VideoCapture(0)
#cap = cv.VideoCapture('video.avi')

while True:
    ret, im = cap.read()
    cv.imshow('video test', im)

    key = cv.waitKey(33) #milisegundos
    if key == 1048603: #ESC
        break

cv.destroyAllWindows()
```