Calibración de cámaras

Visión Computacional 2015-16
Práctica 2
4 de noviembre de 2015

1. OBJETIVOS

Los objetivos de esta práctica son:

- Calibrar una cámara usando el método de calibración de Zhang, que está implementado en *OpenCV*.
- Hacer uso de los resultados de la calibración en un sistema simple de realidad aumentada que proyecte un modelo 3D sintético sobre imágenes reales.
- Calibrar un par de cámaras y deducir información sobre la posición relativa de las mismas.

2. REQUERIMIENTOS

Para ésta y el resto de prácticas de la asignatura, es necesario disponer del siguiente software:

- *Python* 2.6 o 2.7 (aunque otras versiones 2.x pueden funcionar también).
- Un shell avanzado de *Python*: *IPython*.
- Las librerías científicas de *Python*: *NumPy*, *SciPy* y *Matplotlib*.

Además, para esta práctica se empleará también la versión de *Python* de la librería *OpenCV*.

El material necesario para la práctica se puede descargar de la carpeta `MaterialesPractica` del aula virtual. Esta carpeta contiene:

- Dos secuencias de imágenes tomadas con un par de cámaras (izquierda y derecha) en los directorios `left` y `right`.
- Tres modelos tridimensionales: la tetera de Utah (`teapot`), el conejo de Stanford (`bunny`) y un cubo (`cube`).
  La carga de un modelo en *Python* se realiza como si fuera un módulo. Por ejemplo:

  ```
  from models import bunny
  ```

  El módulo cargado contiene dos variables. `bunny.vertices` es una matriz $4 \times N_v$ con las coordenadas homogéneas de los N_v vértices del modelo (en este caso, el conejo de Stanford). Cada columna son las coordenadas de un vértice. `bunny.edges` es una matriz $2 \times N_e$ con los N_e arcos del modelo. Cada columna contiene los índices de los dos vértices que une un arco.
- Algunas funciones auxiliares en el módulo `misc.py`. La descripción de las funciones puede consultarse accediendo a su documentación en un intérprete de *Python* o leyendo su código fuente.
- El archivo `calibration.py`, donde se debe escribir el código de la parte de la práctica dedicada a calibración.

3. CONDICIONES

- La práctica es individual.
- La fecha de entrega límite es el 17 de noviembre a las 23:55.
- La entrega consiste en:
  - Documentación en formato PDF con los resultados de los Ejercicios 3 (sólo una imagen), 5, 6, 11 (una imagen),12 y 13 .
  - El fichero `calibration.py`, que debe contener las funciones pedidas en los ejercicios sobre calibración. Debe llevar a cabo una ejecución de la práctica que incluya calibración (Ejercicios 1, 2, 3, 4 y 5), realidad aumentada con la calibración (Ejercicio 11), calibración de la cámara derecha (Ejercicio 12) y cálculo de la orientación relativa de las cámaras (Ejercicio 13). Se valora que esta función imprima información (con `print`) sobre la tarea que está haciendo en cada momento y algunos resultados.
  - Los ficheros con los resultados de la calibración `calib_left.npz` y `calib_right.npz` de los Ejercicios 5 y 12.

4. CALIBRACIÓN DE UNA CÁMARA

En esta parte se trabajará con la secuencia de imágenes del directorio `left`. Esta secuencia consiste en una serie de imágenes de la plantilla de calibración. Para la calibración se debe tener en cuenta que el tamaño de cada escaque de la plantilla es de 30mm en las direcciones X e Y.

Ejercicio 1. Implementa la función

```
load_images(filenames)
```

que reciba una lista de nombres de archivos de imagen y las cargue como matrices de *NumPy*. Usa la función `scipy.misc.imread` para cargar las imágenes. La función debe devolver una lista de matrices de *NumPy* con las imágenes leídas.

Usa `load_images` para cargar todas las imágenes del directorio `left` por orden alfabético.

Escribe en la función `main` de `calibration.py` el comando utilizado para cargar las imágenes del directorio `left`.

Consejo: La función `glob.glob` permite generar la lista de nombres de archivo, y la función `misc.sort_nicely` ordena alfabéticamente una lista de cadenas de texto.

Fácil, 0.5 puntos ■

La función `cv2.findChessboardCorners` de *OpenCV* busca la plantilla de calibración en una imagen y devuelve una tupla de dos elementos. El primer elemento es 0 si no consiguió detectar correctamente la plantilla, y es 1 en caso contrario. El segundo elemento contiene las coordenadas de las esquinas de la plantilla de calibración, que sólo son válidas si la detección fue exitosa (es decir, si el primer elemento de la tupla es 1).

Ejercicio 2. Usa la función `cv2.findChessboardCorners` para detectar automáticamente el patrón de calibración y sus esquinas en todas las imágenes cargadas.

El tamaño de la plantilla de calibración en las imágenes de la práctica es $(8, 6)$, esto es, 8 filas y 6 columnas.

Almacena los resultados de las múltiples llamadas en una lista, de modo que el elemento i de dicha lista corresponda al resultado de `cv2.findChessboardCorners` para la imagen i cargada en el ejercicio anterior.

Fácil, 0.5 puntos ■

El siguiente ejercicio consiste en dibujar sobre las imágenes los puntos detectados por `cv.FindChessboard-Corners`. Por motivos de eficiencia, la función empleada para hacerlo modifica directamente las imagen pasadas por parámetro en lugar de hacer una copia. Para evitar perder las imágenes originales es mejor realizar una copia de las mismas con antelación. La mejor forma de hacerlo es

```
imgs2 = copy.deepcopy(imgs)
```

donde `imgs` es la lista de imágenes cargadas en el Ejercicio 1. Utiliza estas imágenes copiadas en lugar de las originales en el siguiente ejercicio.

Ejercicio 3. Usa `cv2.drawChessboardCorners` para dibujar las esquinas detectadas en el ejercicio anterior. Aplícalo a todas las imágenes que fueron correctamente detectadas. Ignora el resto.

Dibuja alguna de las imágenes resultantes con el comando `imshow` y añádela a la documentación.

Fácil, 0.5 puntos ■

Para calibrar la cámara, además de las coordenadas de las esquinas en cada una de las imágenes, se necesitan las coordenadas tridimensionales de las esquinas en el sistema de referencia de la escena. Para esta práctica, consideraremos que el centro del sistema de referencia, esto es, el punto de coordenadas $[0, 0, 0]^T$, es la primera esquina de la plantilla de calibración detectada en todas las imágenes. También consideraremos que el eje X corresponde al lado corto de la plantilla de calibración, y el eje Y al lado largo. Esta disposición implica que el eje Z apunta en la dirección normal hacia arriba del plano de calibración.

Para el siguiente ejercicio es muy importante tener en cuenta que las coordenadas de las esquinas en el sistema de referencia de la escena deben darse en el mismo orden que en el que fueron detectadas en cada una de las imágenes.

Ejercicio 4. Implementa la función

```
get_chessboard_points(chessboard_shape,
                      dx, dy)
```

que genere una matriz de *NumPy* (es decir, un `ndarray`) de tamaño $N \times 3$ con las coordenadas (x,y,z) de las esquinas de la plantilla de calibración en el sistema de referencia de la escena. N es el número de esquinas de la plantilla.

`chessboard_shape` es el número de puntos por filas y por columnas de la plantilla de calibración. Al igual que en el Ejercicio 2, debe ser $(8, 6)$. `dx` (resp. `dy`) es el ancho (resp. alto) de un escaque de la plantilla de calibración. Para la plantilla utilizada en esta práctica, ambos valores son $30mm$.

Fácil, 1 punto ■

Ejercicio 5. Calibra la cámara izquierda usando la lista de resultados de `cv2.findChessboardCorners` y el conjunto de puntos del modelo dados por `get_chessboard_points`, del ejercicio anterior.

Para ello usa la función `calibrate` que se distribuye con el material de la práctica.

Guarda el resultado de la calibración, matriz de intrínsecos y matrices de extrínsecos, con el comando

```
np.savez('calib_left',
intrinsic=intrinsic, extrinsic=extrinsic)
```

Fácil, 0.5 puntos ■

4.1. Intrínsecos de la cámara

Una de las características intrínsecas de una cámara más fácilmente comprensible es su *ángulo de visión* o *campo de visión* (FOV). El FOV es la amplitud angular de una determinada escena que es visible por la cámara, y se suele expresar en grados. Como se puede ver en la Figura 1, el FOV es diferente según como se mida: horizontal, vertical o diagonalmente.

La matriz de intrínsecos $\mathbf{K}$ de la cámara no contiene explícitamente el valor de este parámetro, pero puede ser calculado a partir de ella.

Ejercicio 6. Conociendo los intrínsecos $\mathbf{K}$ y que el tamaño de las imágenes es 320×240, indica el ángulo de visión diagonal que abarca la cámara.

Medio, 1 punto ■

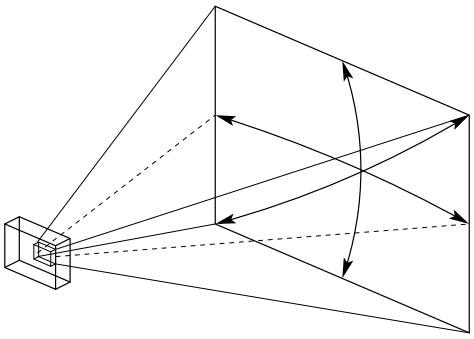

Figura 1. El ángulo de visión (FOV) de la cámara en tres orientaciones diferentes: horizontal, vertical y diagonal.

5. REALIDAD AUMENTADA (OPCIONAL)

En visión computacional, el término *realidad aumentada* hace referencia al conjunto de técnicas que permiten representar información sintética no existente en el mundo real sobre imágenes reales. En nuestro caso, la *información sintética* son modelos tridimensionales. Los siguientes ejercicios proponen una serie de pasos para implementar un pequeño sistema de realidad aumentada, para lo cual serán necesarios los parámetros obtenidos durante la calibración.

Ejercicio 7. Implementa la función

```
m = proj(K, T, verts)
```

que, dada la matriz de intrínsecos `K`, extrínsecos `T` y una matriz de vértices expresados en coordenadas homogéneas `verts`, calcule la proyección de los vértices 3D a puntos 2D de la imagen. Las coordenadas 2D resultantes deben ser homogéneas.

Es decir, este ejercicio consiste en implementar la ecuación de proyección vista en clase.

Fácil, 0.5 puntos ■

Ejercicio 8. Implementa una función

```
plothom(points)
```

que dibuje un conjunto de puntos 2D de entrada expresados en coordenadas homogéneas.

Fácil, 0.5 puntos ■

Ejercicio 9. Usa las funciones implementadas en los ejercicios anteriores para proyectar un modelo sobre las imágenes de la secuencia. Para ello, modifica la función `play_ar`, que se distribuye con la práctica, completando las partes marcadas con `TODO`:

- Proyecta los vértices del modelo con `proj` usando los intrínsecos y los extrínsecos de la imagen que corresponda.
- Dibuja los vértices proyectados o los arcos correspondientes con `plothom`.

Prueba la función `play_ar` una vez terminada. Añade el código necesario en la función `main`.

Fácil, 0.5 puntos ■

Ejercicio 10. Modifica el código de la función `play_ar` del ejercicio anterior para que el modelo se represente en el centro de la plantilla de calibración, y no en la esquina donde se encuentra el origen de coordenadas. Rota además el modelo 90 grados sobre el eje Z. Para ello, usa la función `misc.ang2rotmatrix`, que se distribuye con la práctica.

Añade a la documentación de la práctica una imagen del modelo proyectado.

Fácil, 1 punto ■

6. PAR DE CÁMARAS

Ejercicio 11. Siguiendo el procedimiento de la primera parte de la práctica, calibra la cámara derecha usando la secuencia de imágenes del directorio `right`.

Guarda el resultado de la calibración con el comando

```
np.savez('calib_right',
 intrinsic=intrinsic2,
 extrinsic=extrinsic2)
```

Escribe en la función `main` el código necesario para este ejercicio.

Fácil, 0.5 punto ■

Ejercicio 12. Tomando las dos primeras imágenes de la secuencia, expresa la posición de la cámara derecha en el sistema de referencia de la cámara izquierda.

¿Cuál es la distancia, en milímetros, entre la cámara derecha y la izquierda?

Medio, 0.5 punto ■