Bloque-2.pdf

Daysapro

Fundamentos de Sistemas Inteligentes en Visión

3º Grado en Ingeniería Informática

Escuela Politécnica Superior de Córdoba
Universidad de Córdoba

BLOQUE 2

Las sesiones de las clases resumidas
Después de leer el resumen intentad hacer las preguntas de otros años que algunas están en wuolah
- Daysa -

Tema 5: Introducción a la visión 3D

5.1 Introducción

Cuando la escena 3D se proyecta en el sensor de una cámara surge el problema de que se pierde una propiedad muy importante de la realidad: la profundidad. En este bloque se estudiarán las distintas técnicas que se usan para recuperar esa profundidad perdida, y que una imagen sea lo más afín a la realidad posible.
Todas estas técnicas se basan en buscar la imagen del punto 3D en al menos 2 vistas relacionadas de la misma escena. Sin embargo, en la práctica no todo es tan fácil como podría parecer, ya que tenemos un ruido que nos puede afectar en la estimación del punto 3D y que trataremos.

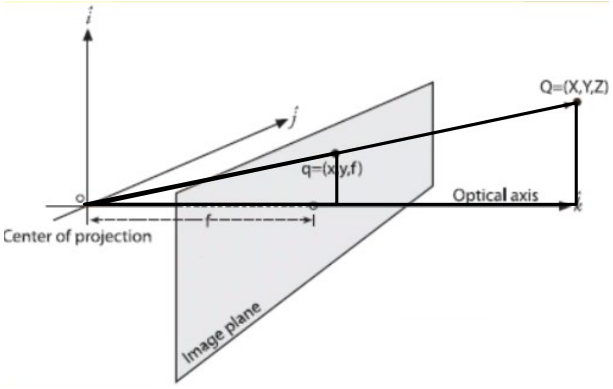
Tema 6: Modelo proyectivo a fondo: pin-hole

6.1 Introducción

El modelo proyectivo recordamos que nos indicaba dónde un punto XYZ de la realidad se proyecta en un punto XY de la fotografía. Nosotros para esto trabajamos con el modelo pin-hole. Los parámetros que explican el modelo se denominan parámetros intrínsecos y los que lo relacionan con el mundo extrínsecos, y aprenderemos a relacionarlos por un proceso llamado calibración de la cámara.

6.2 Parámetros intrínsecos

Podemos plantear y explicar sus parámetros intrínsecos por 2 métodos:
- Semejanza de triángulos. En la imagen podemos ver que:
x/f=X/Z; x=f·(X/Z)
y/f=Y/Z; y=f·(Y/Z)
siendo:
x e y coordenadas de la proyección, f distancia focal y X, Y y Z coordenadas de la realidad.



- Forma matricial. Expresamos las relaciones de manera matricial:

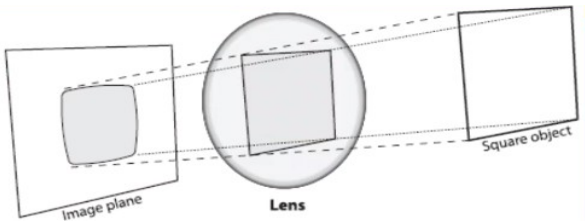
$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \qquad \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} u/w \\ v/w \end{bmatrix}$$

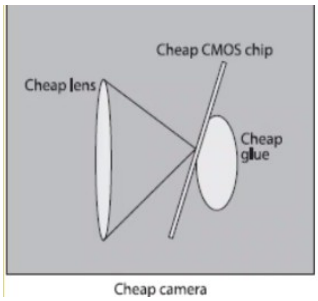
siendo:
u, v y w un vector como x e y con un elemento extra w para hacerlo homogéneo. Para normalizarlo, dividiríamos u y v entre w, de ahí sale x=u/w e y=v/w. Si operaramos la matriz resultaría en:
x/f=X/Z; x=f·(X/Z)
y/f=Y/Z; y=f·(Y/Z)
En la realidad no todo es así, ya que el centro de imagen no coincide con el centro proyectivo, debido al ruido y otras causas. Así, la forma matricial anterior podemos expresarla tal que:

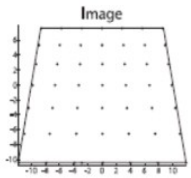
$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}, \quad x = \frac{u}{w}, \quad y = \frac{v}{w}$$

siendo:
c_x y c_y las coordenadas del centro proyectado y f_x y f_y las distancias focales ya que al cambiar el centro, no son iguales.
Recordamos además que las lentes con este modelo producía algunas distorsiones geométricas, como el efecto circular de la lente gran angular. Tenemos un sistema de ecuaciones que nos explica estas distorsiones, y que trabajaremos con ellas en la calibración, para solucionar 2 tipos de distorsiones, la radial y la tangencial.

$$x_{corregida} = x\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) + 2 * p_1 xy + p_2\left(r^2 + 2x^2\right)$$
$$y_{corregida} = y\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) + 2 * p_2 xy + p_1\left(r^2 + 2y^2\right)$$
$$r = x^2 + y^2$$







6.3 Parámetros extrínsecos
Cuando tenemos más de una cámara en la misma escena, todas no pueden ser el sistema de referencia. Tenemos que elegir una. Así, podremos definir para cada una de ellas donde está un punto de la realidad.
Para pasar de un sistema de referencia a otro necesitamos una matriz de traslación y una matriz rotación.

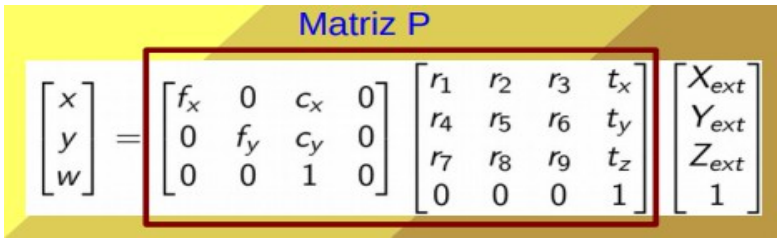
$$P_c = R_x R_y R_z (P_w - T) \rightarrow R P_w - RT \rightarrow \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$R = R_x R_y R_z \quad , \quad R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -sen\,\alpha \\ 0 & sen\,\alpha & \cos\alpha \end{bmatrix}, \ \ldots$$

Donde está una cámara en relación al sistema de referencia global se denomina pose.

6.4 Relación intrínsecos-extrínsecos
Tras entender ambos tipos de parámetros y para prepararnos de cara a la calibración, debemos saber la relación entre ambos. La utilidad de esta relación reside en, al usuario tener un punto, ya sea del mundo real, o de la imagen, conseguir a partir de uno de ellos el otro. Esto se hace por medio de la que se denomina matriz de proyección P.


#ESTASREADYCOLACAO
ColaCao®

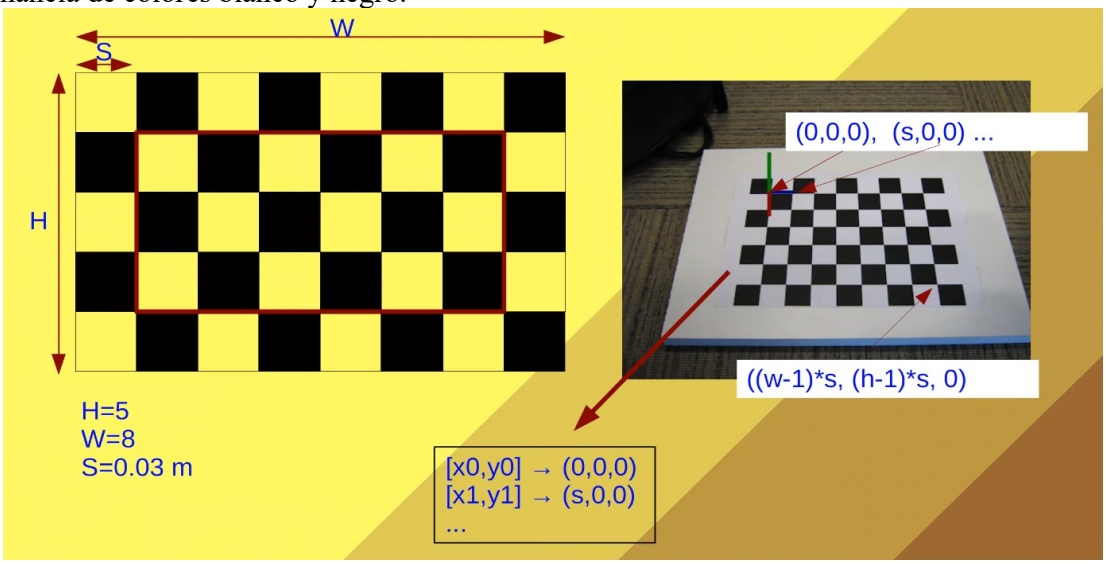


$P=M\cdot[R|t]$

Esta matriz nos proporciona algunos datos:
- $x=P\cdot X$. Esta es su utilidad mayor, un punto en la imagen será el punto de la realidad por la matriz.
- $P^{+}\cdot P=I$. P^{+} es la pseudoinversa (al no ser cuadrada no tiene inversa), y nos sirve para calcular punto de la realidad respecto a la imagen.
- $\vec{v}=P^{+}\cdot x$. $r=C_0+\lambda v$. La pseudoinversa obviamente no nos da el punto exacto de la imagen, porque carece de profundidad, pero si nos da un vector que equivale al rallo de luz en el que sabemos que está el punto 3D.

6.5 Calibración de la cámara

Hay distintas técnicas, pero nosotros estudiaremos la de opencv. Para esta necesitamos lo que se llama un patrón de calibración, y uno básico es un tablero de ajedrez. El motivo principal por el que se usa una forma así es por la facilidad de obtener a nivel computacional las esquinas del mismo y diferenciar a la vez sus formas internas. Lo recomendable es que tenga un número de filas par mientras el de columnas impar o viceversa, para así diferenciar si el tablero está rotado por la alternancia de colores blanco y negro.



Como vemos en la ilustración, no usaremos tanto la primera fila y columna como ambas últimas, porque no sabremos si nuestro software podrá detectar esas primeras en relación a donde coloquemos el patrón en nuestro entorno. Así, como podemos ver, el primer punto será (0, 0, 0), el segundo (s, 0, 0) siendo s el alto y ancho de cada cuadrado y el último((w-1)·s, (h-1)·s, 0).
Con estos datos ya presentados, podemos calcular los parámetros intrínsecos y extrínsecos de la escena para la calibración.
- Calibración intrínseca: necesitamos de de 7 a 9 vistas del patrón. Se calcula un vector de puntos 3D de cada vista. Esto en opencv se hace con findChesboardCorners(). De aquí salen píxeles con decimales pero, con optimización, la función cornerSubPix() refina el píxel que mejor estima ese punto. Ya con todos los vectores se usa calibrateCamera() que devuelve la matriz de los parámetros intrínsecos de la cámara.
- Calibración extrínseca: cuando hay más de una cámara y tenemos los patrones intrínsecos de cada una de ellas, ponemos el patrón a vista de todas y calculamos la pose de todas ellas. De nuevo, usamos findChesboardCorners(), refinamos con cornerSubPix() y ahora con SolvePnP() obtenemos

las matrices de rotación y de traslación. Estas matrices nos da la rotación en un vector y el módulo es el ángulo a rotar, pero con la función rodriguez() obtenemos las matrices en sus formas originales.

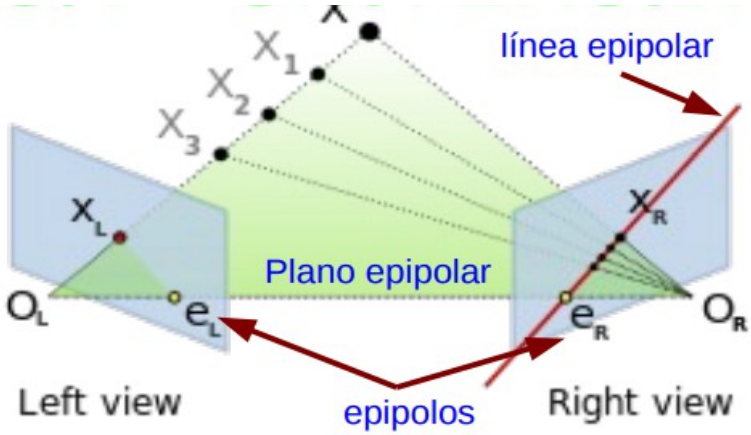
Tema 7: Visión estéreo

7.1 Introducción

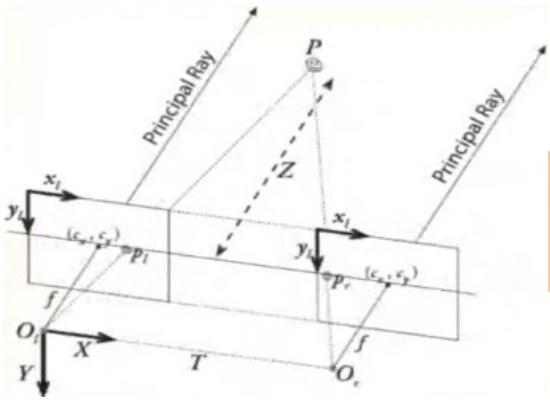
La visión estéreo es la primera técnica para reconstruir la escena 3D. Se basa, como se explicó en la introducción, en 2 vistas relacionadas de la misma escena, en las que se calculan los rayos del punto que queremos estimar y se calcula la intersección. Un ejemplo podría ser una cámara con 2 lentes capturando imágenes muy similares para facilitar la búsqueda del punto. La distancia entre las 2 lentes en este caso se denomina base line y por la naturaleza de esta, esta técnica tiene mayor resolución y efectividad en distancias cortas.

7.2 Geometría epipolar

Es el modelo geométrico por el que se basa esta técnica.
Cogemos los centros proyectivos de ambas vistas, O_L y O_R, los unimos y junto a uno de los puntos de las imágenes se forma lo que se llama el plano epipolar. La intersección de este plano con ambas imágenes son las líneas epipolares, las que siempre pasan por los epipolos, y sabiendo que el punto proyectado de una de ellas está en su línea epipolar el otro desconocido debe estar en la otra, por lo que simplificamos con este método la búsqueda de complejidad de $O(n^2)$ a $O(n)$.



En una situación ideal, tenemos ambas cámaras perfectamente alineadas. Si esto ocurre, por semejanza de triángulos:



$$\frac{T-(x^l-x^r)}{Z-f}=\frac{T-d}{Z-f}=\frac{T}{Z}\rightarrow Z=f\frac{T}{d}$$

siendo:
x^l-x^r la discrepancia entre los puntos.
Ya obtendríamos la Z y con ella sabríamos las coordenadas exactas del punto.

7.3 Rectificación
La situación ideal no ocurre en la realidad. Por esto, necesitamos aplicar un proceso matemático para estimarla lo más afín posible. Tiene 3 pasos:
- Rotar ambas cámaras para que miren perpendicularmente a la línea que une los dos centros proyectivos.
- Rotar los ejes ópticos de tal forma que los ejes horizontales se se alineen.
- Si es necesario, escalar la imagen más pequeña tal que ambas imágenes tengan la misma resolución.

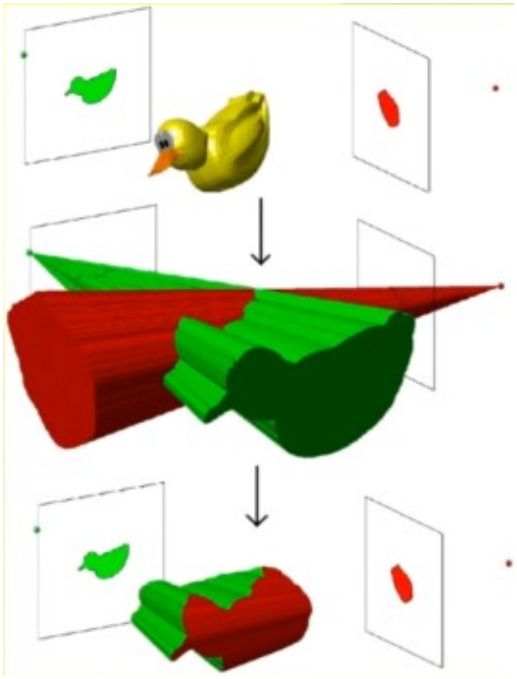
A nivel de programación, por medio de opencv, tomamos los siguientes pasos:
- Usamos stereoCalibrate() para obtener los parámetros intrínsecos de cada cámara y la pose de la cámara derecha respecto a la izquierda.
- Tomadas 2 imágenes usamos stereoRetify() para obtener las versiones rectificadas.

7.4 Correspondencia
En todo este tema hemos supuesto que al tener ambas imágenes podemos diferenciar el mismo punto al verlas. Eso en la práctica un ordenador no puede hacerlo. Por eso nos queda el último problema a resolver: la correspondencia. Uno de los algoritmos clásicos para resolver este problema es el Konolige97, y sigue los siguientes pasos:
- Normaliza el brillo de ambas imágenes y resaltar la textura por el posible cambio de ajustes de las cámaras.
- Usamos SAD (suma de diferencias en valor absoluto) para buscar correspondencias. Se pasa una ventana y la función SAD cuando encuentre el punto dará prácticamente 0, y será un mínimo local.
- Filtramos para eliminar malas correspondencias.

Este algoritmo solo funciona de manera óptima en imágenes monocroma.

Tema 8: Otras técnicas de reconstrucción de escena 3D
8.1 Luz estructurada
El principio de la luz estructura se basa en iluminar la escena con un patrón conocido, y a partir de la información visual que obtenemos con esta iluminación obtenemos la profundidad. Esto lo conseguimos por medio de un proyector que proporciona la luz. Lo trataremos como si fuera una cámara, sabiendo de él su pose en relación a la cámara y su calibración, para usar al igual que en la visión estéreo, triangulación. Además, el problema de la correspondencia que teníamos en la visión estéreo aquí está muy simplificado.
- Ejemplo de patrón: código Grey. Se van proyectando varios patrones estructurados en columnas y después de varias imágenes con distintos patrones podemos recuperar cual es el punto en la imagen. Como tenemos columnas de algunos pocos píxeles cogemos los planos verticales incluidos en la columna y se interseccionan con el rayo que ya tenemos del punto de la imagen.
- Ejemplo de patrón: desplazamiento de fase. Se proyectan 3 patrones basados en ondas coseno con un desfase. Por medio de una operación trigonométrica, podemos obtener cual es la fase de un punto, y como la fase la genero el usuario con el ajuste del cañón, obtenemos el punto.

8.2 A partir de siluetas
En base a, por ejemplo, 2 vistas relacionadas de un objeto, podemos formar los conos visuales de todos los puntos proyectados e interseccionándolas obtener una aproximación del objeto 3D. Esta intersección se denomina Visual-Hull. Tiene 2 posibles implementaciones: voxel-set u octree.
Estas técnicas funcionan muy bien en entornos muy controlados, pero en la realidad fuera de un estudio no son demasiado eficaces.



- Voxel-set: usamos por ejemplo 4 cámaras calibradas en una habitación cuadrada con un objeto en el centro. Dividimos esta habitación en numerosos cubos de volumen y capturamos imágenes con las 4 vistas diferentes. Vamos comparando en base a un algoritmo todos los cubos para ver si está el objeto o no, de manera que si una vista observa que más del 50% de los píxeles no son fondo se marca como bloque ocupado, y todas las vistas tienen que coincidir para dar el bloque como bueno en el Visual Hull.
- Octree: es una versión más eficiente del método anterior, ya que en él se hacían recorridos de volumen innecesarios. El volumen a tratar se divide en 8 cubos, y si alguno de ellos está vacío no lo analizamos y seguimos dividiendo en 8 el módulo con el objeto. Es un algoritmo recursivo que va de elementos más grandes a más pequeños.

8.3 A partir del movimiento

Supongamos que se toman 2 vistas de forma manual de un mismo objeto y podemos a simple vista diferenciar algunos puntos equivalentes en la realidad en ambas. En base a esos puntos, podemos definir lo que se denomina matriz fundamental F, la cual nos proporciona a partir de un punto de una imagen la línea epipolar donde está ese punto en la otra.
Conociendo esa matriz y la matriz de la cámara podemos obtener la matriz esencial (E=M'FM), que es equivalente a la fundamental pero desde puntos 3D a rayos en el mundo 3D en una imagen.
Con estas matrices y numerosas vistas relacionadas por los puntos en común ya podemos hacer la reconstrucción del modelo 3D.