RECONSTRUCCIÓN ESTEREO

VALENTINA VUL

ETAPAS DEL PROCESO

CALIBRACIÓN

Calcular parámetros intrínsecos y extrínsecos de la cámara

RECTIFICACIÓN

Rectificar geometría epipolar

DISPARIDAD

Calcular mapas de diapridad y profundidad

RECOSNTRUCCIÓN

Proyectar puntos a 3D y generar la nube de puntos

CALIBRACIÓN

INTRODUCCIÓN TEÓRICA

En un modelo de cámara pinhole, la relación entre un pixel de la imágen x_i y un punto en 3D X, se define según la siguiente proyección:

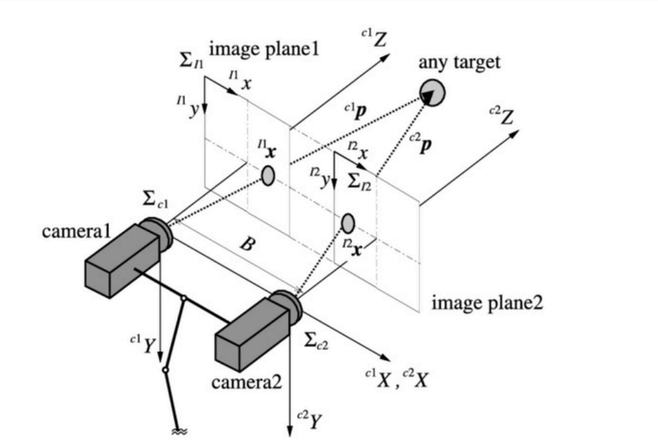
$$x_i = K[R|t]X$$

Donde K es la matriz intrínseca de la cámara, R la matriz de rotación y t el vector de traslación de la cámara con respecto al mundo.

En la vida real, los lentes de la cámara agregan distorsión, por lo que no basta con conocer los parámetros de la cámara para hacer la proyección, sino que también es necesario conocer los parámetros de distorsión

CALIBRACIÓN ESTEREO

Al usar una cámara estereo, la calibración se hace en base a un par de imágenes izquierda y derecha. La matriz de rotación y el vector de traslación calculados son aquellos que definen la relación entre ambas cámaras.



Para proyectar un punto de una imágen a la otra, es necesario conocer la *matriz fundamental* del par. Esta es aquella que cumple que

$$x_2^T F x_1 = 0$$

A partir de un conjunto lo suficientemente grande de correspondencias de puntos entre las imágenes se puede llegar a una solución lineal que define a la matriz F.

La calibración estereo entonces, tiene 3 objetivos:

1. ***Calcular los parámetros intrínsecos de cada cámara.***
   Para esto se necesita un conjunto de correspondencias de puntos con las cuales se estiman los parámetros mediante métodos lineales. Luego estas estimaciones se optimizan usando cuadrados mínimos para minimizar el error reproyectivo.

2. ***Calcular la relación estereo.***
   La rotación y la traslación entre las cámaras también se calculan mediante estimaciones y optimizaciones.

3. ***Calcular la matriz fundamental y esencial.***
   La matriz fundamental se calcula mediante las correspondencias de puntos.
   Una vez calculada, se puede calcular la *matriz esencial*, que define la relación entre puntos normalizados.

$$E = K_2^T F K_1$$

```python
err, left_K, left_dist, right_K, right_dist, R, T, E, F = cv2.stereoCalibrate(
        world_points_mm,
        left_images_points,
        right_images_points,
        left_K,
        left_dist,
        right_K,
        right_dist,
        image_size,
        flags=0
    )
```

Las correspondencias de puntos se hayan teniendo un patrón conocido que puede ser detectado en todas las imágenes.

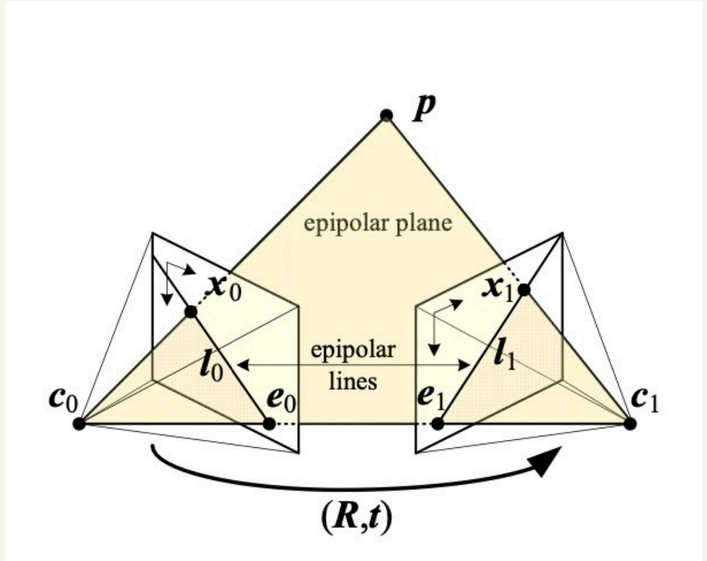
```python
ret, corners = cv2.findChessboardCorners(
        gray,
        CHECKERBOARD,
        cv2.CALIB_CB_ADAPTIVE_THRESH + cv2.CALIB_CB_FAST_CHECK + cv2.CALIB_CB_NORMALIZE_IMAGE
    )
```

RECTIFICACIÓN

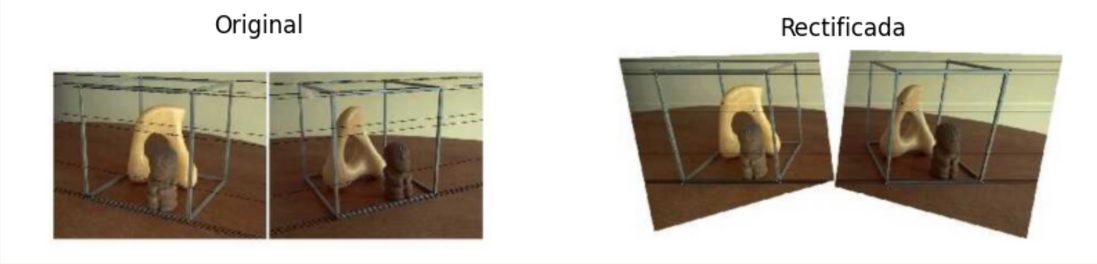
INTRODUCCIÓN TEÓRICA

La geometría epipolar describe el modelo de las cámaras estereo. Este proporciona información extra sobre las imágenes que facilita el proceso de reconstrucción.
A la hora de encontrar puntos que se corresponden entre las imágenes esta geometría limita el espacio de búsqueda haciendo el proceso más rápido y eficiente.

Un punto en una imagen, se proyecta a una linea en la otra, llamada *linea epipolar*. Es sobre esta que se va a encontrar el punto correspondiente en la segunda imagen.
Para realizar esta busqueda, es conveniente que estas lineas de proyección sean paralelas. Es por esto que se hace un proceso de rectificación como primer paso de la reconstrucción de un objeto.



Usando los parámetros intrínsecos y extrínsecos de las cámaras calculados en la calibración, se aplican transformaciones a las matrices de proyección de las cámaras para conseguir que las lineas epipolares en ambas imágenes queden alinedas.



El objetivo es conseguir que las correspondencias estén alineadas. Si se pone una imagen encima de la otra, la única diferenecia que debería haber entre un punto y su correspondiente en la otra imagen es un desplazamiento horizontal

Este proceso, además de calcular los mapas de rectificación, calcula la matriz de reproyección a 3D, Q, que permite pasar de coordenadas en la imágen rectificada a coordenadas en el mundo.

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = Q \begin{bmatrix} u \\ v \\ d \\ 1 \end{bmatrix}$$

```
R1, R2, P1, P2, Q, validRoi1, validRoi2 = cv2.stereoRectify(
        left_K, left_dist, right_K, right_dist, image_size, R, T, alpha=0
    )
```

```
left_map_x, left_map_y=cv2.initUndistortRectifyMap(left_K, left_dist, R1, P1, image_size, cv2.CV_32FC1)
right_map_x, right_map_y=cv2.initUndistortRectifyMap(right_K, right_dist, R2, P2, image_size, cv2.CV_32FC1)
```

```
left_image_rectified = cv2.remap(left_image, left_map_x, left_map_y, cv2.INTER_LINEAR)
right_image_rectified = cv2.remap(right_image, right_map_x, right_map_y, cv2.INTER_LINEAR)
```

DISPARIDAD

La disparidad mide la diferencia en la posición horizontal en la proyección de un punto en las cámaras estereo. Esta esta inversamente relacionada con las profundidad de dicho punto.

$$Z = \frac{f \cdot b}{d}$$

Con f la distancia focal de la cámara, b el *baseline*, o distancia entre las dos cámaras y d la disparidad. Como las imágenes estan rectificadas, comparten una única distancia focal.

Una vez calculada la profundidad se pueden calcular las demás coordenadas del sistema con las ecuaciones

$$X = \frac{(x - c_x) \cdot Z}{f}, Y = \frac{(y - c_y) \cdot Z}{f}$$

Se usa el método CREStereo para estimar la disparidad de cada punto. Es una red neuronal espcializada para encontrar la disparidad

```
method, calibration = disparity_method
pair = InputPair(left_image_rectified, right_image_rectified, calibration)
disparity = method.compute_disparity(pair)
map = disparity_pixels
```

RECONSTRUCCIÓN

INTRODUCCIÓN TEÓRICA

Una vez obtenido el mapa de disparidad de los puntos y la matriz Q, se pueden reproyectar los puntos a 3D según la transformación antes definida.

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = Q \begin{bmatrix} u \\ v \\ d \\ 1 \end{bmatrix}$$

De esta forma se consiguen los puntos 3D dados en coordenadas de la cámara. Es necesario pasarlos a coordenadas del mundo tal que las proyecciones de cada par queden en el mismo espacio. Esto se consigue mediante la matriz de transfromación homogénea,

$$T_{o \to c} = [R|t]$$

Esta matriz toma la información de rotación y traslación de la cámara con respecto al mundo y convierte los puntos dados en coordenadas del mundo a coordenadas de la cámara. Calculando su inversa se puede hacer la transformación opuesta.

Se define una de estas matrices por cámara involucrada en la reconstrucción.

Cada proyección de puntos se almacena en una nube de puntos. Una vez terminado el proceso para todas las imágenes se tiene la reconstrucción deseada del objeto.

```
ret, rvec, tvec = cv2.solvePnP(
            object_points_mm,
            left_corners,
            left_K,
            left_dist,
            flags=cv2.SOLVEPNP_IPPE
        )
```

```
c_R_o = cv2.Rodrigues(rvec)
c_T_o = np.column_stack((c_R_o[0], tvec))
c_T_o = np.vstack((c_T_o, [0, 0, 0, 1]))
o_T_c = np.linalg.inv(c_T_o)
```

```
points_3d = cv2.reprojectImageTo3D(disparity, Q)
point_cloud = points_3d.reshape(-1, points_3d.shape[-1])
```

```
point_cloud = o_T_c @ np.vstack((point_cloud.T, np.ones(point_cloud.shape[0])))
```

REFERENCIAS

- Hartley, R., & Zisserman, A. (2003). Multiple View Geometry in Computer Vision (2nd ed.). Cambridge University Press.
- Szeliski, R. (2022). Computer Vision: Algorithms and Applications (2nd ed.). Springer.
- OpenCV Development Team. (2024). OpenCV Documentation (versión 4.5.3). Recuperado el 4 de diciembre de 2024, de https://docs.opencv.org/4.5.3/