Procesamiento Digital de Imágenes

TP Visión Estéreo y navegación

Objetivos:

- Incorporar conceptos de procesamiento de imágenes estéreo en visión computacional.
- Aplicar herramientas procesamiento de imágenes para acondicionamiento de imagen y extracción de información relevante en sistemas de registro realistas
- Articular los conocimientos adquiridos en la materia Procesamiento Digital de Imágenes con los de Robótica, en una aplicación real.

Materiales.

Para este trabajo práctico se emplearán los siguientes materiales provistos por la cátedra:

- Raspberry Pi 3 B+ (OS Rasbian, Python2.7, OpenCV 4.1)
- Cámara estereo
- LEGO NXT con 2 motores para desplazamiento.

El equipo ya contiene el esqueleto de la implementación para funcionar normalmente. El práctico consistirá en resolver una serie de tareas para completar el código fuente que permitirá al vehículo esquivar obstáculos que se encuentren en la trayectoria. El codigo fuente se encuentra disponible en github.com/lbugnon/pdi-vision-estereo.git.

Calibración y pruebas iniciales.

Se realizó un proceso de calibración previo, mediante el cual se estiman parámetros tales como la matriz de cada cámara (para corrección de aberraciones), las relaciones geométricas entre las cámaras (distancias y matriz de rotación) y parámetros de rectificación vertical. Este procedimiento se tiene que llevar a cabo **siempre** que se trabaje con cámaras de precisión. Los algoritmos utilizados para medir la profundidad de visión se basan en la diferencia entre las imágenes “punto a punto”, por lo que es fundamental que las imágenes esten alineadas.

1. Acceder a la Raspberry y explorar las imágenes utilizadas para la calibración.
2. Correr el script principal (main.py) y verificar las imágenes originales de la cámara.
3. Utilizar los parámetros de calibración provistos para corregir las deformaciones y rectificar la salida. ¿Qué diferencias se observan respecto a las imágenes originales?

Mapa de disparidad

Si comparamos las imágenes registradas por las cámaras, los objetos más cercanos tendrán una distancia relativa mayor, mientras que tendiendo al infinito las imágenes no difieren mucho. Los algoritmos que calculan el mapa de disparidad buscan identificar estas diferencias para asignar valores de “distancia” a cada pixel de la imagen.

4. Evaluar los métodos de OpenCV “SGBM” y “BM” para calcular el mapa de disparidad, utilizando las imágenes rectificadas. ¿El FPS de cada uno es relevante a la hora de implementar soluciones en tiempo real?
5. Colocar un objeto de prueba a ~50 cm de la cámara. Analizar los parámetros del método a configurar. ¿Cuales son los efectos en el mapa de disparidad y como los explica? ¿cuales son críticos y que representa cada uno?

   Parámetros:
   - Min_disp: Disparidad mínima a considerar (por defecto 0)
   - Num_disp: Cantidad de valores de disparidad (max_disp-min_disp=num_disp, por defecto 16)
   - Block_size: Tamaño de bloques en el algoritmo block-matching (por defecto 5)
   - Uniqueness: Umbral de diferencia entre el bloque ganador y el segundo en el matching (valores usuales 5-15).
   - Speckle: Parámetros de filtrado para controlar ruido

6. Proponer e implementar una heurística para identificar un objeto al frente y definir qué acción tomar para evitarlo.

Control del veículo

7. Poner en funcionamiento el robot en trayectoria lineal, graficando la imagen registrada.
8. Incorporar la heurística propuesta y accionar sobre el controlador de movimiento en caso de detectar una posible colisión.