Facultad de Ciencias Exactas, Ingeniería y Agrimensura
Escuela de Ciencias Exactas y Naturales
Departamento de Ciencias de la Computación
R-521 — Robótica Móvil

Trabajo Práctico

Visión por Computadora: Reconstrucción 3D y Estimación de Pose

1. Introducción

El objetivo del trabajo práctico es la realización de los pasos básicos para poder triangular y proyectar puntos con una cámara estéreo. En este trabajo se debe utilizar las librerías OpenCV[1] y software de calibración ampliamente utilizado en el campo de visión por computadora y robótica.

2. Entrega

- Se debe proveer un repositorio git que contenga el código desarrollado y un archivo `README.md` con las instrucciones de compilación y ejecución. Se recomienda hacer una imagen Docker para facilitar la reproducción de los resuldatos.
- Se debe entregar un informe en Lyx o LaTeX explicando el trabajo realizado y analizando los resultados obtenidos.

3. Evaluación

- Haciendo únicamente los ejercicios obligarorios (no opcionales), el trabajo tiene una nota máxima de 8. Los ejercicios opcionales permiten llegar a la nota máxima de 10.
- Entregas Fuera de Término: Si la entrega se realiza durante la primer semana luego del plazo, se decuentan 2 puntos. Si la entrega es más tarde que esto último la nota máxima es de 6.

4. Datos

Utilizar el dataset EuRoC[2]. Debera convertir la secuencia de calibración a formato ROS2. puede utilizar el software `rosbags`[3]. El resto de las secuencias ya se encuentran en formato ROS2 y pueden ser descargadas de `https://docs.openvins.com/gs-datasets.html`

5. Ejercicios

Ejercicio 1. Realizar una calibración de la cámara estéreo del robot. Se debe proveer los parámetros intrínsecos y extrínsecos.

Nota: Para la calibración se puede utilizar cualquiera de las aplicaciones que se detallan a continuación:

- ROS2 camera_calibration: `https://docs.nav2.org/tutorials/docs/camera_calibration.html`
- OpenCV tutorial_camera_calibration: `https://docs.opencv.org/4.x/d4/d94/tutorial_camera_calibration.html`
- Kalibr: `https://github.com/ethz-asl/kalibr/wiki/ROS2-Calibration-Using-Kalibr`

[1] `https://opencv.org/`
[2] `https://projects.asl.ethz.ch/datasets/doku.php?id=kmavvisualinertialdatasets`
[3] https://pypi.org/project/rosbags/

- Basal Calibration: `https://github.com/VladyslavUsenko/basalt/blob/master/doc/Calibration.md`
- BoofCV: `https://boofcv.org/index.php?title=Tutorial_Camera_Calibration`
- Camera Calibration Toolbox for Matlab: `https://www.cs.toronto.edu/pub/psala/VM/cameraCalibrationExample.html`

Ejercicio 2. Desarrollar un programa que lea un par de imágenes estéreo cualquiera y realice los siguientes pasos:

a) Rectificar Imágenes

Rectificar las imágenes utilizando los parámetros íntrinsecos y extrínsecos de la cámara estéreo. Esto se puede realizar utilizando:

- ROS2, por medio del paquete `stereo_image_proc`

```
ros2 launch stereo_image_proc stereo_image_proc.launch.py
```

Para reproducir el rosbag y remapear los tópicos a los que el paquete `stereo_image_proc` requiere puede hacer:

```
ros2 bag play CARPETA_ROSBAG --remap /stereo/left/image_raw:=/left/image_raw \
/stereo/left/camera_info:=/left/camera_info \
/stereo/right/image_raw:=/right/image_raw \
/stereo/right/camera_info:=/right/camera_info
```

- o bien, OpenCV, por medio de las funciones: `cv::stereoRectify()`, `cv::initUndistortRectifyMap()` y `remap()`.

b) Extraer Feaures Visuales: Keypoints y descriptores

Selecionar un detector de keypoints (FAST, ORB, SIFT, SURF, GFTT, BRISK, etc.) y un descriptor (BRIEF, ORB, BRISK, etc.), y extraer features en ambas imágenes. Capturar una imagen izquierda y derecha con los features extraídos. Agregar captura al informe.

c) Buscar correspondencias visuales

Realizar la búsqueda de correspondencias entre los feature de ambas imágenes (*matching*). Para esto se debe utilizar la función `cv::BFMatcher::BFMatcher()`. Visualizar todos los matches. Luego, visualizar todos los matches que tengan una distancia de matching menor a 30. Agregar capturas al informe.

d) Triangular Puntos 3D

Dadas las correspondencias visuales (*matches*) obtenidas en el paso anterior, realizar la triangulación de los features extraídos utilizando la función `cv::sfm::triangulatePoints()`. Para la visualización de la nube de puntos 3D se puede publicar un mensaje de tipo `sensor_msgs/PointCloud2`[4] y hacer uso de RViz. Agregar capturas al informe.

e) Filtrar de Correspondencias Espúreas

Aplicar RANSAC (*Random sample consensus*) para filtrar los matches espúreos y computar la Matríz homográfica que relaciona los puntos. Para esto puede utilizar la función `cv::findHomography`

[4] `http://docs.ros.org/en/melodic/api/sensor_msgs/html/msg/PointCloud2.html`

`()`. Para verificar el impacto del filtrado, visualizar los matches entre las imágenes nuevamente como en la nube de puntos 3D generada. También, visualizar en la imagen derecha los puntos de la imagen izquierda transformados por la matríz homográfica. Para esto último utilizar la función `cv::perspectiveTransform()`. Agregar capturas al informe.

f) (Opcional) **Feature Mapping con Localización *ground-truth***

Mapear el entorno utilizando las poses dada por el *ground-truth* del dataset. Visualizar el mapa reconstruido. Agregar captura al informe.

g) Computar el Mapa de Disparidad

Computar el mapa de disparidad con las librerías utilizando la función `cv::StereoMatcher::compute()`. Opcionalmente para tener mejores resultados puede utilizar la librería LIBELAS[5]. Visualizar el mapa de disparidad. Agregar captura al informe.

h) Reconstruir la Escena 3D de manera Densa

Utilizando el mapa de disparidad obtenido en el paso anterior realizar una reconstrucción densa de la escena observada utilizando la función `cv::reprojectImageTo3D()`. Para esto debe utilizar la matríz de reproyección Q retornada por la función `cv::stereoRectify()`. Visualizar la nube de puntos 3D. Agregar captura al informe.

i) (Opcional) **Dense Mapping con Localización Ground-truth**

Mapear el entorno de manera densa utilizando las poses dada por el *ground-truth* del dataset. Visualizar el mapa reconstruido. Agregar captura al informe.

j) Estimar la Pose utilizando Visión Monocular

Utilizando `cv::recoverPose()` estimar la transformación entre la cámara izquierda y la cámara derecha. Para esto deberá calcular la matríz esencial utilizando la función `cv::findEssentialMat()`. Notar que `cv::recoverPose()` retorna el vector unitario de traslación, por lo tanto deberá multiplicarlo por el factor de escala (en este caso el baseline entre las cámaras) para obtener la traslación estimada. Una vez hecho esto se pide:

- Visualizar la pose estimada de ambas cámaras. Agregar captura al informe.
- Estimar y visualizar la trayectoría realizada por la cámara izquierda utilizando como factor de escala la distancia entre cada par de frames dada por el *ground-truth*. Agregar captura al informe.

[5] `http://www.cvlibs.net/software/libelas/`