Trabajo Práctico 3 Robótica

Maximiliano Nielsen - Tomas Pitinari

October 31, 2025

Nota: El código se encuentra en https://github.com/Pitinari/robotica/tree/main/TP3

Ejercicio 2

Para las implementaciones creamos diferentes librerias (feature_extractor, feature_matcher, triangulator, etc) y las vamos utilizando en los nodos para tener diferentes implementaciones.

Ejercicio b)

Extraer Feaures Visuales: Keypoints y descriptores

Consigna

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.

Resolución

Como detector de keypoints y descriptor, utilizamos ORB.

Ejercicio c)

Buscar correspondencias visuales

Consigna

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.

Resolución

En la primera imagen vemos los keypoints que obtuvieron un match entre la imagen izquiera y derecha, como esperabamos hay matches que no tienen mucho sentido. Por lo que en la imagen de abajo agregamos un filtro de los matches que tienen una distancia mayor a 30.

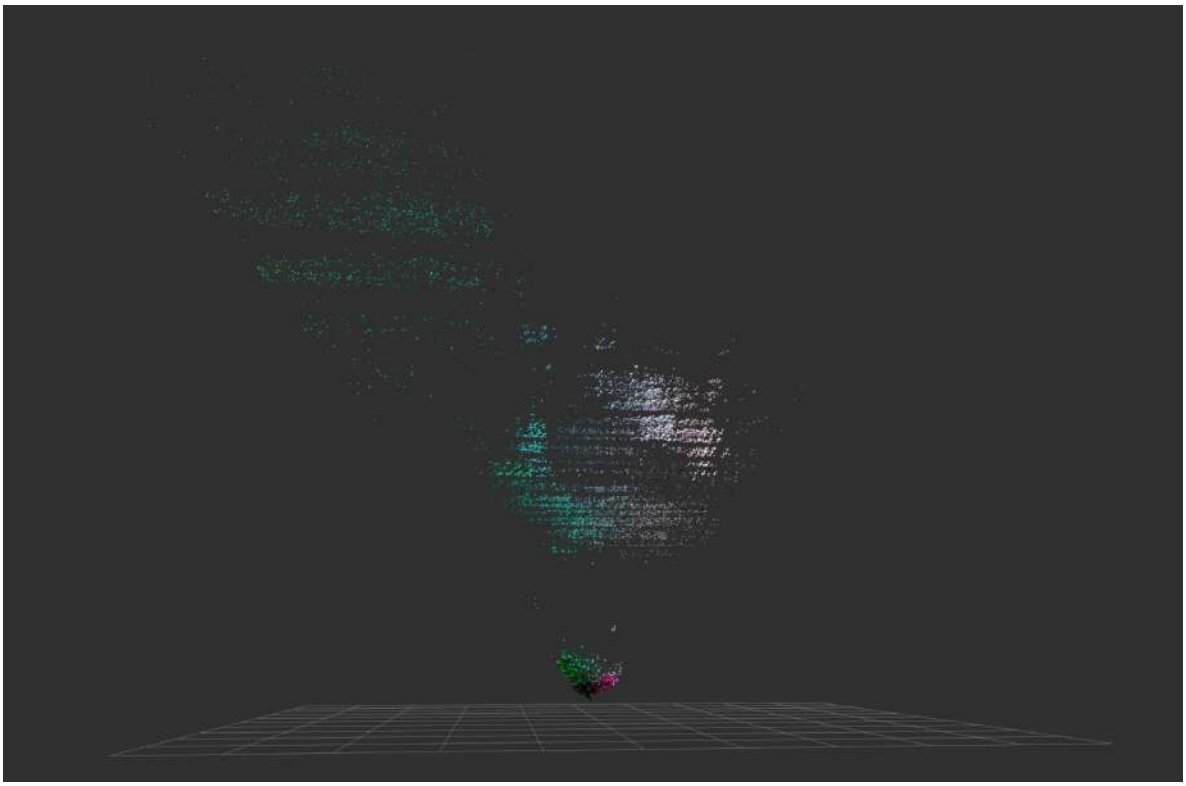
Ejercicio d)

Triangular Puntos 3D

Consigna

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 y hacer uso de RViz. Agregar capturas al informe.

Resolución



En la imagen vemos con la interfaz de RViz la nube de puntos publicada en el canal triangulation/point_cloud en una vista 3D y coloreado con el color asignado a cada keypoint. A simple vista vemos como hay muchos puntos que no estan siendo bien filtrados.

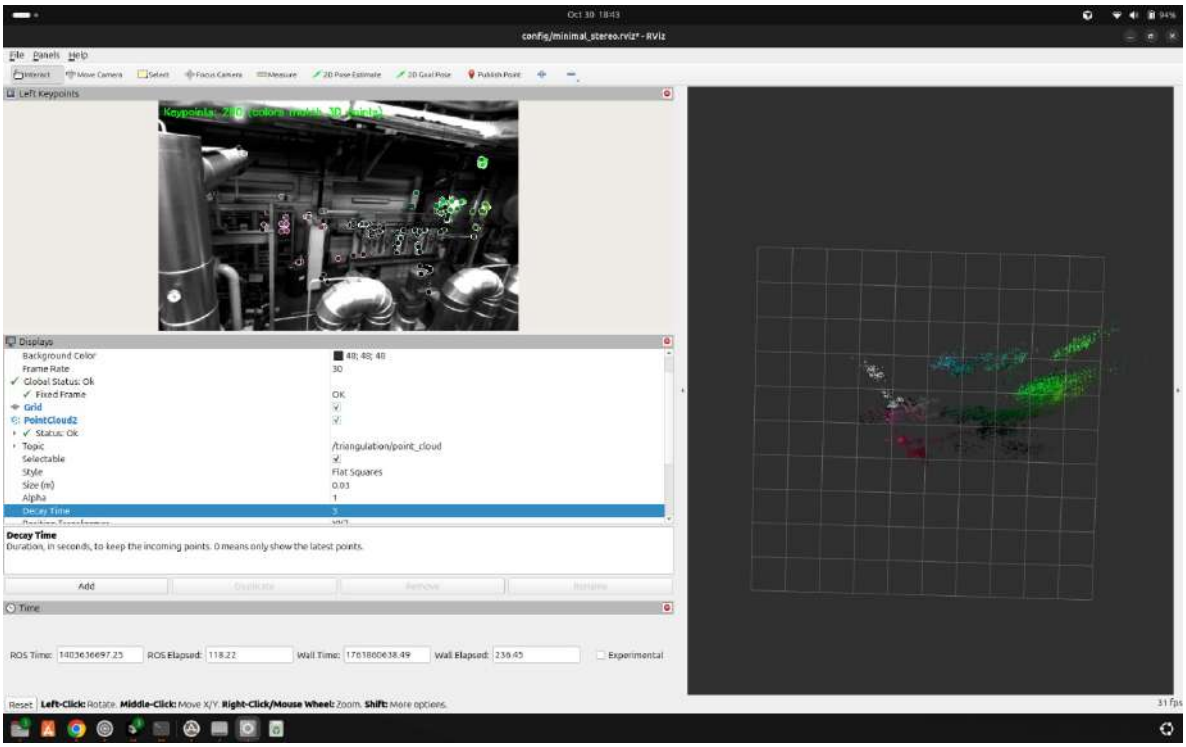
Ejercicio e)

Filtrar de Correspondencias Espúreas

Consigna

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(). 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.

Resolución



En la imagen vemos con la interfaz de RViz la nube de puntos publicada en el canal triangulation/point_cloud en una vista 3D y coloreado con el color asignado a cada keypoint y filtrados aplicando RANSAC.

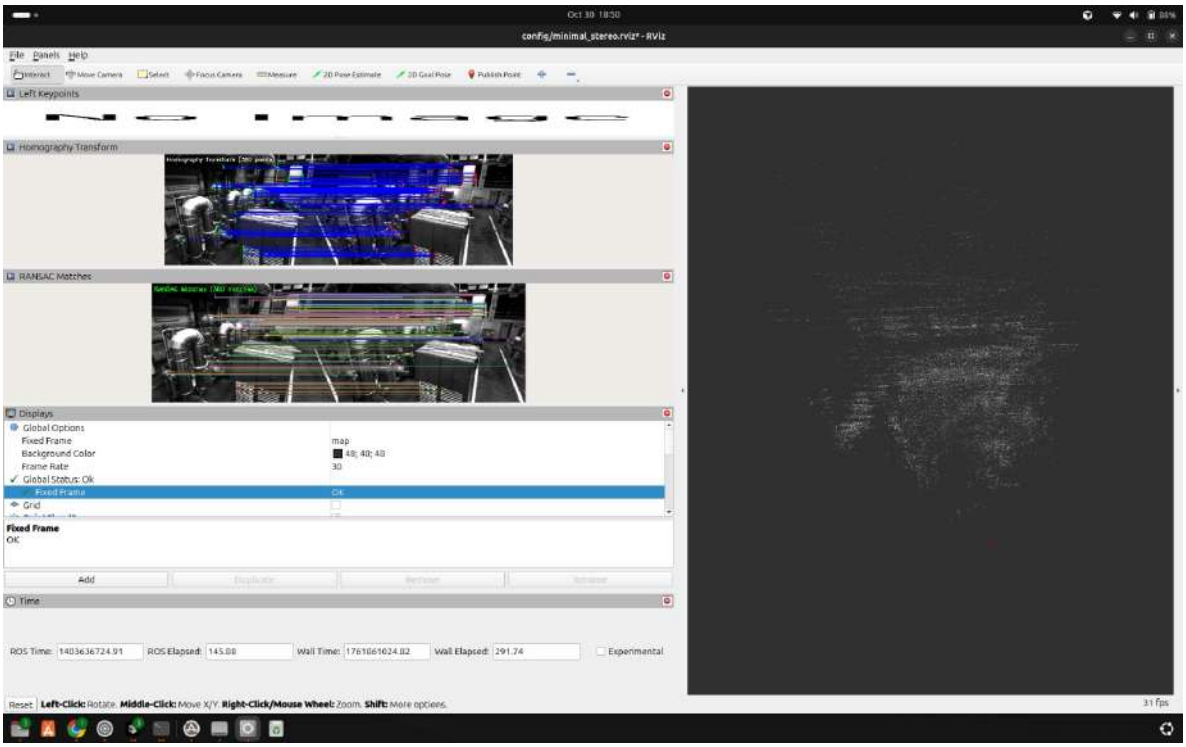
Ejercicio f)

(Opcional) Feature Mapping con Localización ground-truth

Consigna

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

Resolución



En la imagen vemos con la interfaz de RViz la nube de puntos publicada en el canal mapping/point_cloud en una vista 3D mapeando el entorno ajustando la posicion de los puntos con el ground-truth.

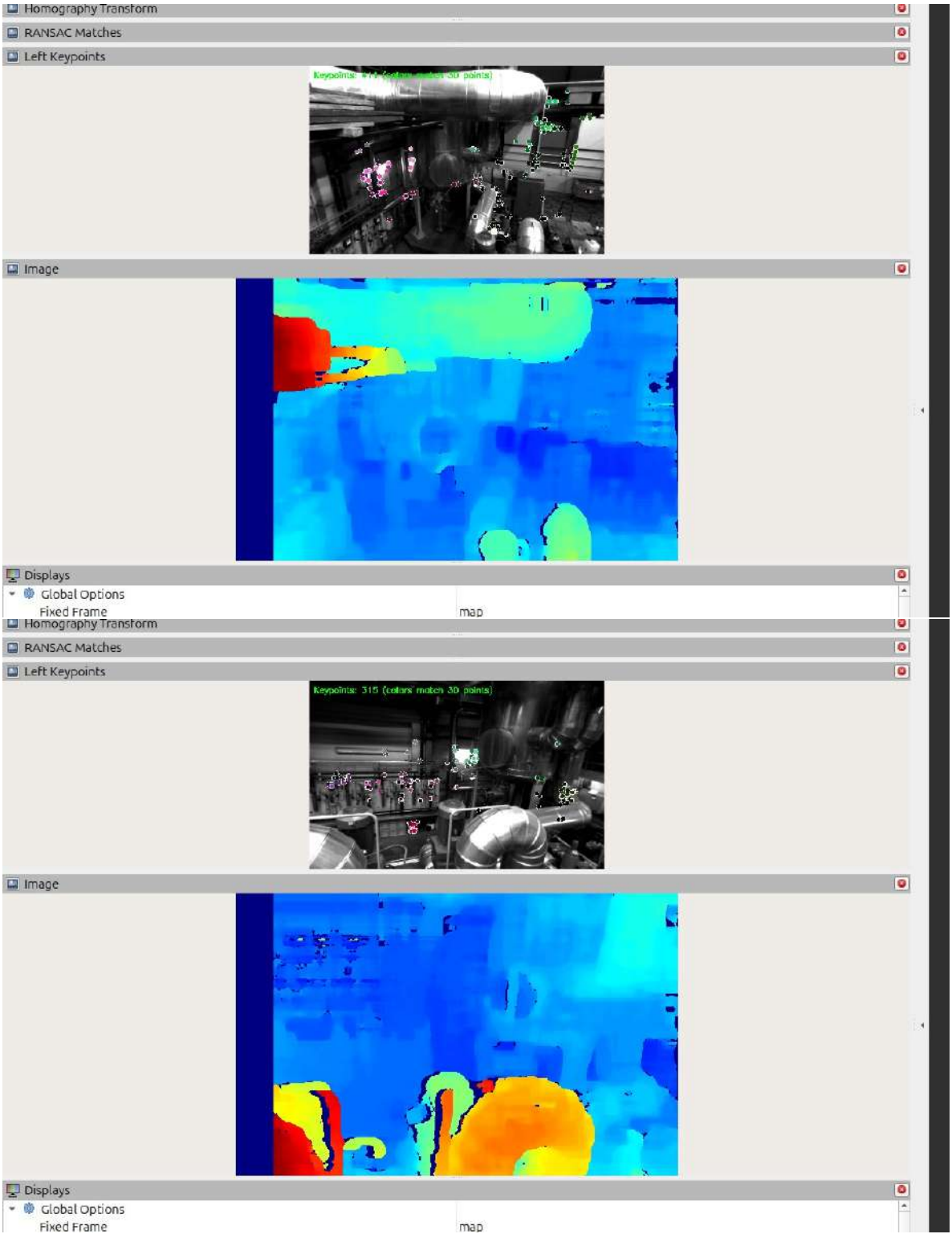
Ejercicio g)

Computar el Mapa de Disparidad

Consigna

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

Resolución



Vemos 2 imagenes que visualizan el mapa de disparidad, mostrando en rojo los objetos mas cercanos y en azul los mas lejanos.

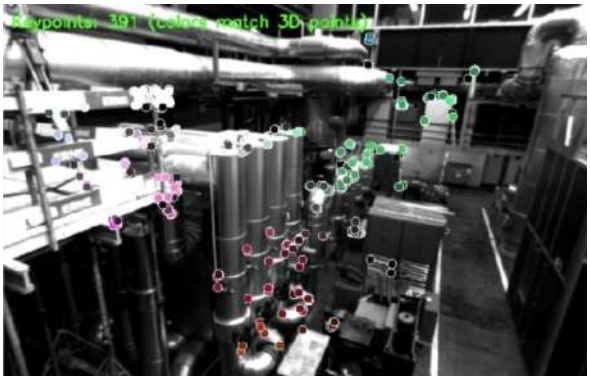
Ejercicio h)

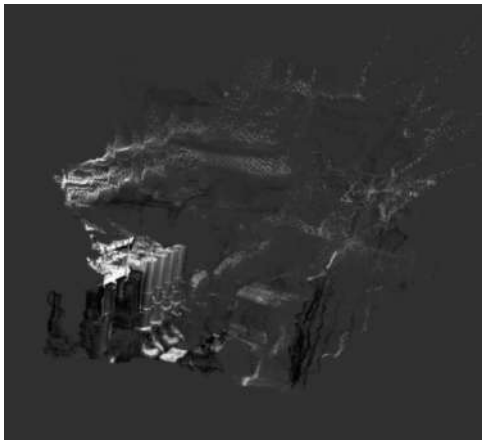
Reconstruir la Escena 3D de manera Densa

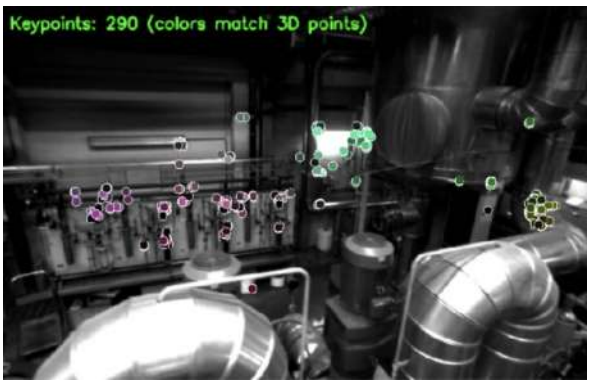
Consigna

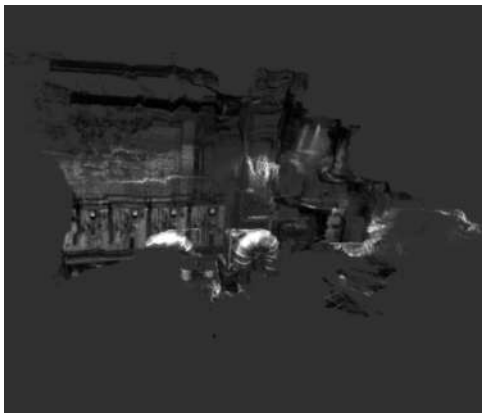
Utilizando el mapa de disparidad obtenido en el paso anterior realizar una reconstrucción densa de la escena observada utilizando la función cv::reprojectImageTo3D(). Para esto debe utilizar la matríz de reproyección Q retornada por la función cv::stereoRectify(). Visualizar la nube de puntos 3D. Agregar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

Resolución









En los 2 dos pares de fotos conseguimos la representacion de la nube de puntos densa generada mediante sus respectivas imagenes.

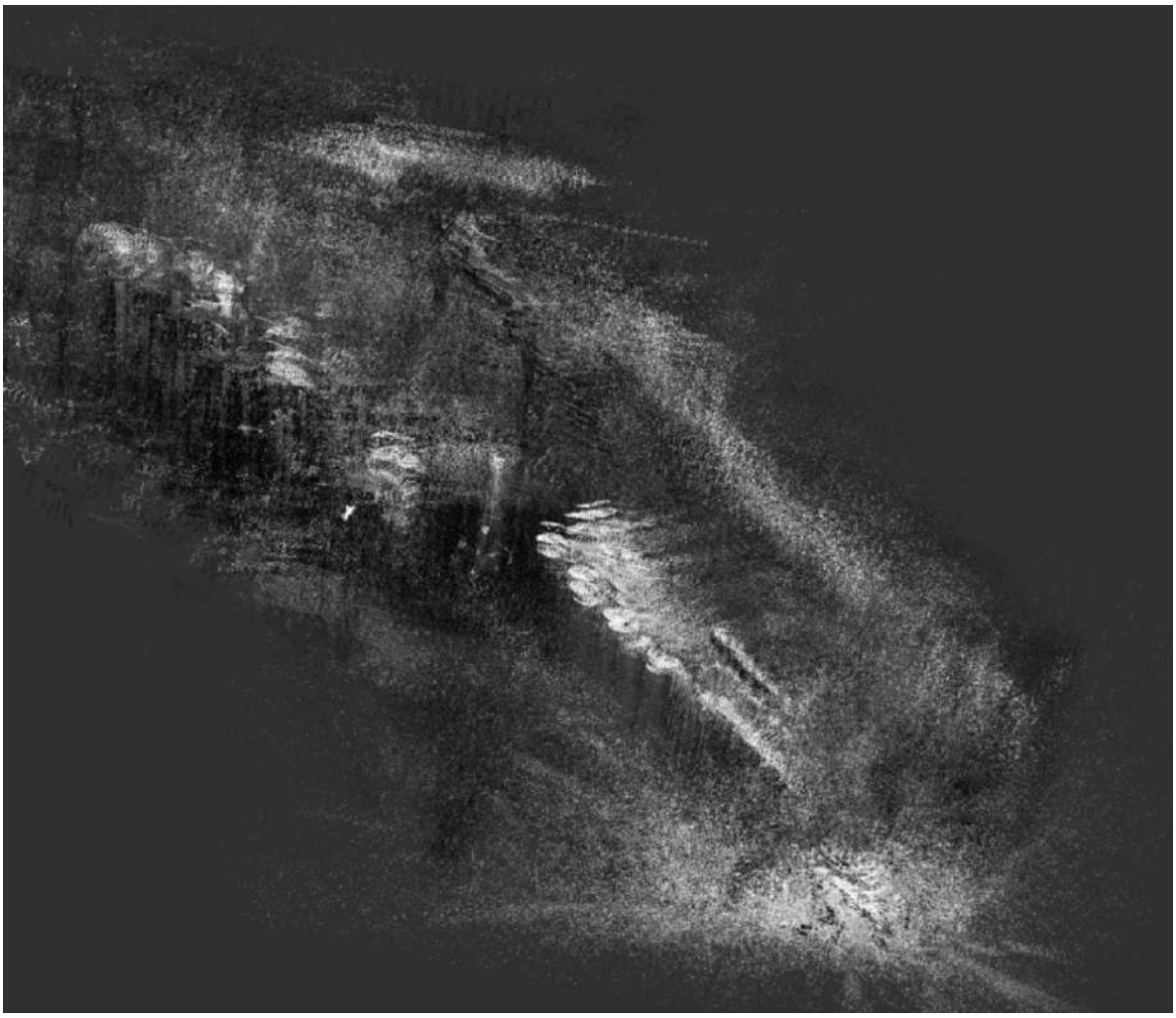
Ejercicio i)

(Opcional) Dense Mapping con Localización Ground-truth

Consigna

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

Resolución



La imagen es la construccion del mapa utilizando la nube de puntos densa y la pose de la camara en cada instante, vimos que hay lugares que logró mapear con exito, pero resultaba muy pesado para poder verlo completo, por lo que dejamos el avance logrado.

Ejercicio j)

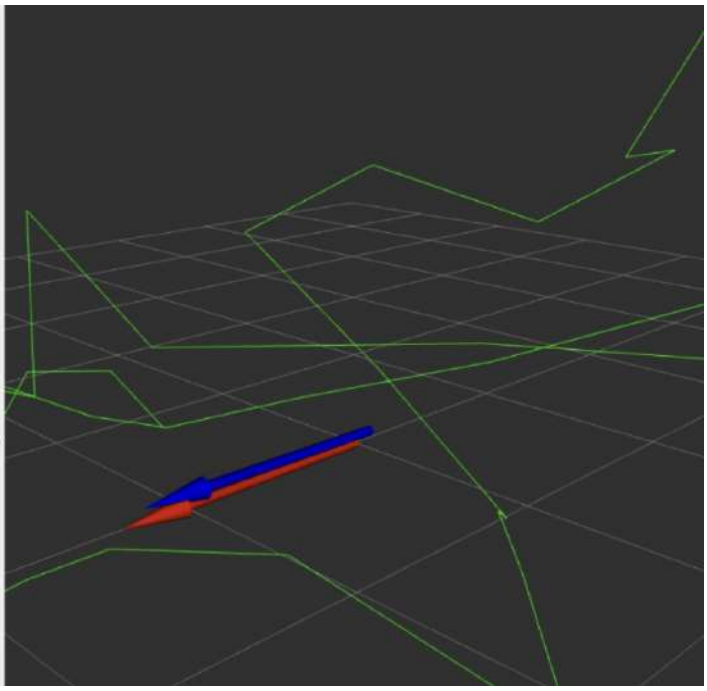
Estimar la Pose utilizando Visión Monocular

Consigna

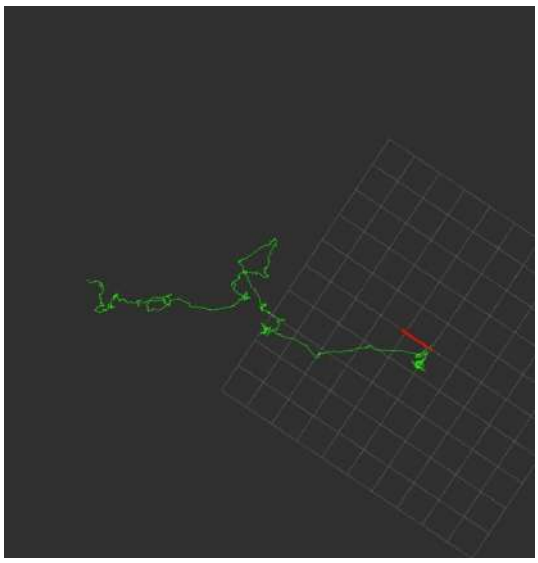
Utilizando cv::recoverPose() estimar la transformación entre la cámara izquierda y la cámara derecha. Para esto deberá calcular la matríz esencial utilizando la función cv::findEssentialMat (). Notar que cv::recoverPose() retorna el vector unitario de traslación, por lo tanto deberá multiplicarlo por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

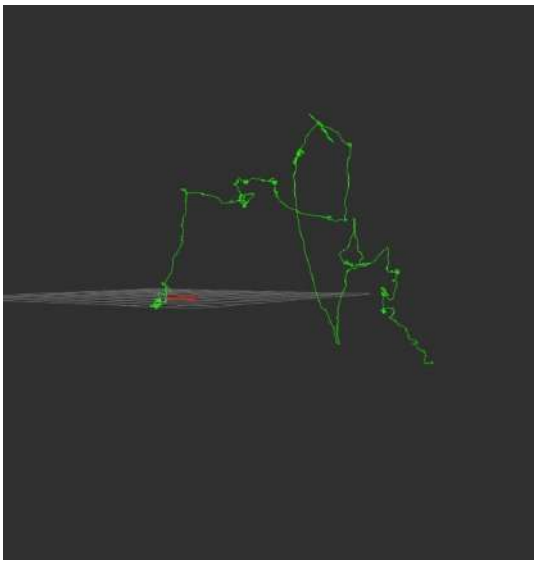
- Visualizar la pose estimada de ambas cámaras. Agregar captura al informe.
- Estimar y visualizar la trayectoría realizada por la cámara izquierda utilizando como factor de escala la distancia entre cada par de frames dada por el ground-truth. Agregar captura al informe.

Resolución



En esta primera imagen visualizamos como estimamos la pose y orientacion de las camaras en los diferentes frames.





En estas imagenes vemos la estimacion del camino del dron utilizando solo las imagenes de las camaras. Podemos ver que esta estimacion acarrea mucho error ya que el dron empieza y termina en el mismo lugar. Si bien la estimacion del movimiento no es tan mala, ya que identifica varios patrones de movimiento, no es una fuente muy fiable de localizacion.