Trabajo Practico Final
Robótica Móvil

Franco Ignacio Vallejos Vigier
Facundo Emmanuel Messulam

19 de marzo de 2024

1. Introducción

En este trabajo practico replicamos el paper "CCM-SLAM: Robust and efficient centralized collaborative monocular simultaneous localization and mapping for robotic teams" (en adelante llamamos al paper y al programa propuesto como CCM-SLAM) que nos permite hacer un mapeo global colaborativo entre múltiples agentes simultáneamente. Nos pareció novedosa la arquitectura del CCM-SLAM ya que elimina infraestructura externa o mapas preconstruidos para permitir aplicaciones multiagente de navegación.

Este paper propone una arquitectura cliente-servidor, donde los agentes ejecutan odometria visual independientemente (garantizando autonomía del agente), luego esta información se comunica con el servidor remoto (ya sea una PC o en la nube) y este es el encargado, primero de establecer una estimación global de cada agente a partir de los datos aportados usando el reconocimiento de lugares, la optimización global y la eliminación de datos redundantes para garantizar un proceso SLAM eficiente; y segundo de compartir datos con los clientes, permitiendo mapeo compartido eficiente.

En este informe de la replicación del paper vamos a detallar conceptos adquiridos sobre esta arquitectura, así como también como los problemas que afrontamos a la hora del aislamiento del entorno, recolección de datos, ejecución, etc. Haciendo observaciones sobre las soluciones y cuestiones particulares.

Código: https://github.com/EmmanuelMess/Robotica-Movil-TPFinal.

Dataset: A pedido, pesa 10GB.

2. Partes y herramientas

Se uso el iRobot Create 3 junto con una placa Raspberry Pi 4B y una cámara Intel Realsense D435. Sobre la Raspberry se instalo Ubuntu 22.04 server (versión modificada proveída por Raspberry) y sobre el iRobot Create se instalo una distribución de Linux dedicada. El resto de software instalado esta explicado en el cuerpo del este informe.

3. Instalación y puesta a punto

Hubo diversos problemas con el software, en esta sección se explican con detalle.

Instalación de Ubuntu sobre la Raspberry Pi 4B

Al instalar Ubuntu hubo problemas con el arranque. Se tuvo que modificar ligeramente la configuración de Ubuntu 22.04 proveída por Raspberry para que arrancara[1]. Tambien tuvimos problemas con la alimentación de la misma, ya que la adapter board, que a pesar que especifica capacidad de 3A a 5V, no estaba pudiendo soportar la alimentación de la Raspberry; luego con un cargador adaptativo de la serie de laptops ThinkPad (de 65W capacidad máxima) lo solucionamos.

Instalación de software update (SWU) sobre iRobot Create 3

La instalación del ROS 2 Humble Hawksbill en el Create 3 se necesito hacer a través del flasheo directo de las placas. Se uso ROS 2 Humble por problemas de compatibilidad con Galactic Geochelone.

[1] Ver https://askubuntu.com/a/1393089

Instalación del software de Intel Realsense

Para poder usar la cámara Intel Realsense D435 se necesita instalar librealsense[2]. Tuvimos problemas con la transmisión de datos de la cámara, dado que teníamos un cable USB 2.1 y el ancho de banda del cable era inferior al requerido por la cámara, este problema se soluciono cambiando por un cable USB 3.2.

Calibración de la cámara

Para la calibración de la camara usamos el tópico `/device_0/sensor_1/Color_0/info/camera_info` grabado en la rosbag por librealsense. Sacamos esos datos de calibración y los usamos en los parámetros de configuración de CCM-SLAM en el archivo `simulation.yaml`.

4. Replicación del paper

Generación de data

Caminos

Para poder validar el correcto funcionamiento del soporte simultaneo de múltiples drones que el paper propone, se decidió generar una serie de caminos que luego puedan ser reproducidos como múltiples agentes, con la salvedad de que nunca se podrían ver entre ellos. Para los datos de estos agentes se generaron caminos sobre un departamento de tres ambientes, en la figura 1 se muestra el plano original con los caminos (nombrados x_1 a x_5). Para la generación de los caminos x_1 a x_4 se hizo una ida y vuelta, para el camino x_5 (visible en rosa claro en la figura 1) se hizo un camino de solo ida hasta el baño.

Guardado de los datos

Para guardar la mayor cantidad de información sobre las ejecuciones (ya que se tenia acceso al hardware por poco tiempo) se decidió usar la rosbag de ROS, se crearon dos rosbags por camino: una en el Create 3, con ROS 2 y otra en la Raspberry Pi, con ROS 1.

Para el guardado de los datos de cámara, se utilizo el software preparado por Intel, visible en la figura 2. Este software tiene varios modos de utilización de la cámara, pero debido a problemas de rendimiento, finalmente se decidió usar el algoritmo preimplementado de "monocularidad", que toma ambas cámaras y la percepción de profundidad y los fusiona; generando un único tópico de imágenes y otro de parámetros de calibración.

Puesta a punto del software

Para la replicación del paper se decidió usar el software creado originalmente, sin embargo, este software debía ser dockerizado para su correcto funcionamiento y parametrizado para funcionar con el hardware a disposición. Para una validación inicial de la ejecución contra datos reales, se decidió utilizar el dataset EuRoC (Machine Hall 01 a 04), para el cual el paper propone que la herramienta funcionó correctamente; adicionalmente, EuRoC tiene cámara frontal, como nuestro agente, en vez de una cámara inferior.

[2] Ver https://dev.intelrealsense.com/docs/open-source-ethernet-networking-for-intel-realsense-depth-cameras

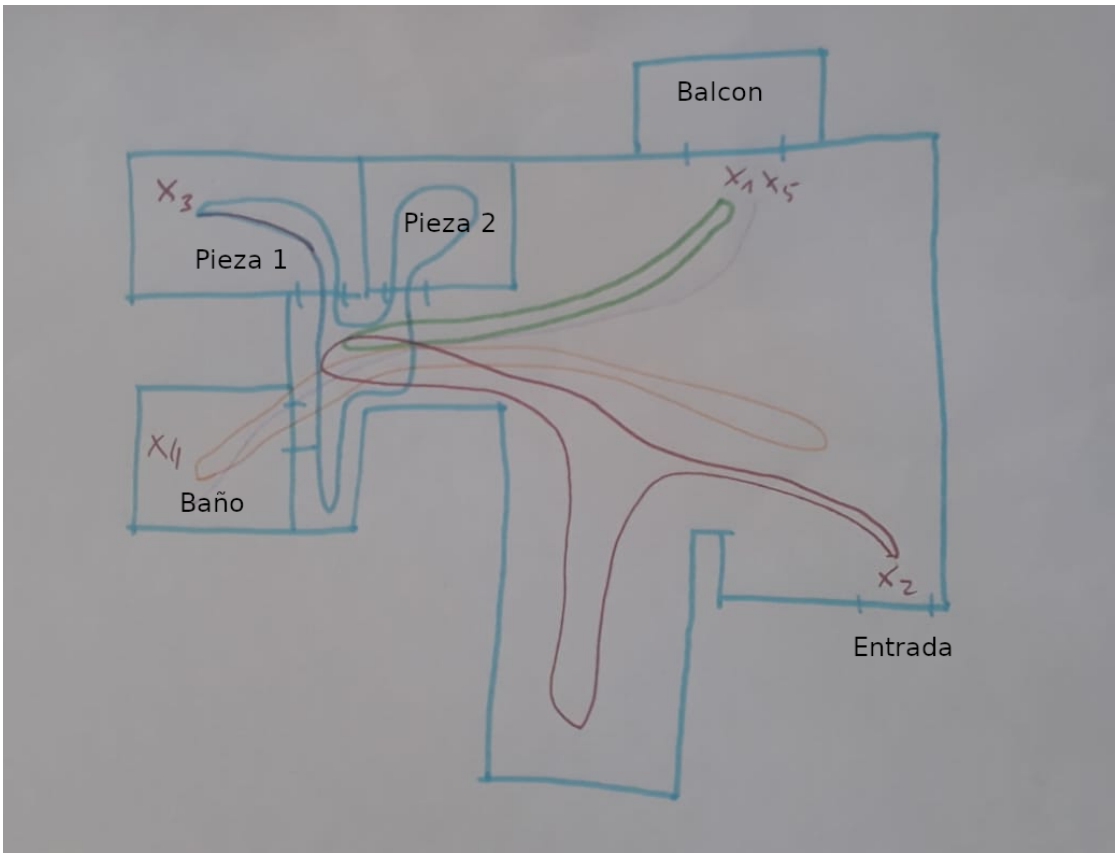

Figura 1: Caminos de los agentes

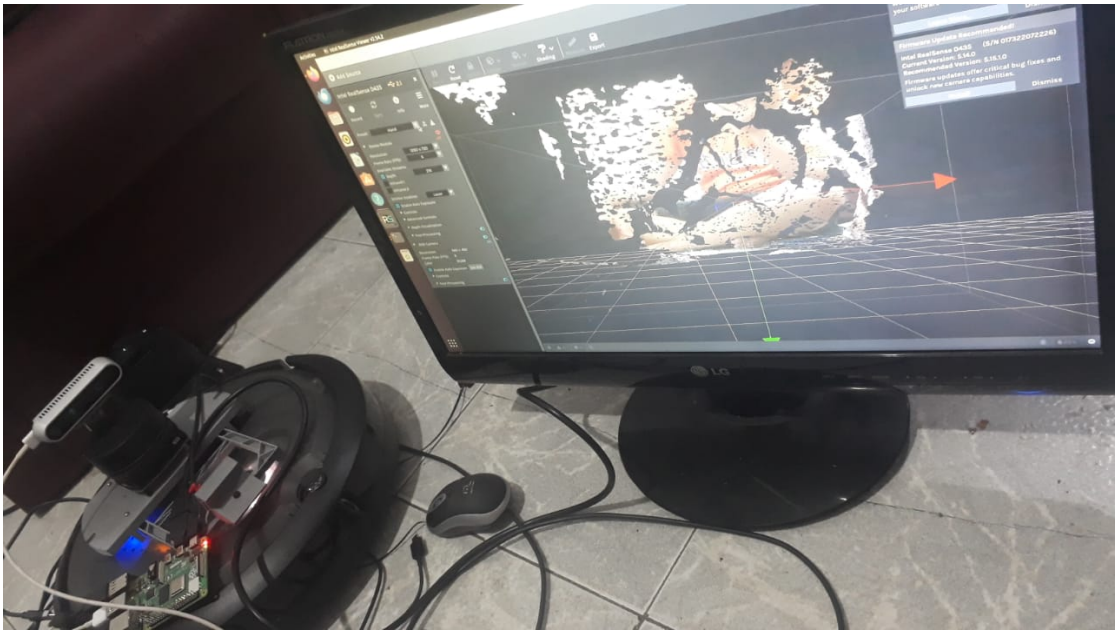

Figura 2: Robot utilizado y visualización de cámara

Dockerización

Para la construcción se uso un makefile muy simple, que permite construir: una imagen docker (con un entorno de ROS Noetic y Ubuntu 20.04), las dependencias de CCM-SLAM y el propio CCM-SLAM.

Luego de la generacion de la imagen docker, se pueden levantar tres docker compose:

- `docker-compose.server.yml`: permite levantar el servidor.
- `docker-compose.client.yml`: permite levantar cualquier cliente.
- `docker-compose.data.yml`: permite reproducir una rosbag, correspondiente a alguno de los agentes.

Todos estos ambientes ya están parametrizados con los datos necesarios, con variables en el archivo `.env`. Adicionalmente, se conectan a la red TCP/IP de la computadora anfitrión y permite ejecutar `rviz -d ccm_slam/cslam/conf/rviz/ccmslam.rviz` en una terminal para abrir una visualización de las trayectorias. Una visualizacion completa de estos sistemas en funcionamiento se puede ver en la figura 4, con terminales para el servidor, el cliente y los datos, y la visualización.

Validación con EuRoC

La validación inicial del sistema se genero con los datos provistos por el dataset EuRoC[3], con los parámetros provistos por el paper (para renombrado de tópicos y saltear la calibración de IMU inicial). La ejecución de las rutas `Machine Hall 01` (en blanco) y `Machine Hall 02` (en verde) no tiene problemas, cierra ciclos seguido y logra juntar los mapas en un tiempo aceptable.

[3] https://projects.asl.ethz.ch/datasets/doku.php?id=kmavvisualinertialdatasets

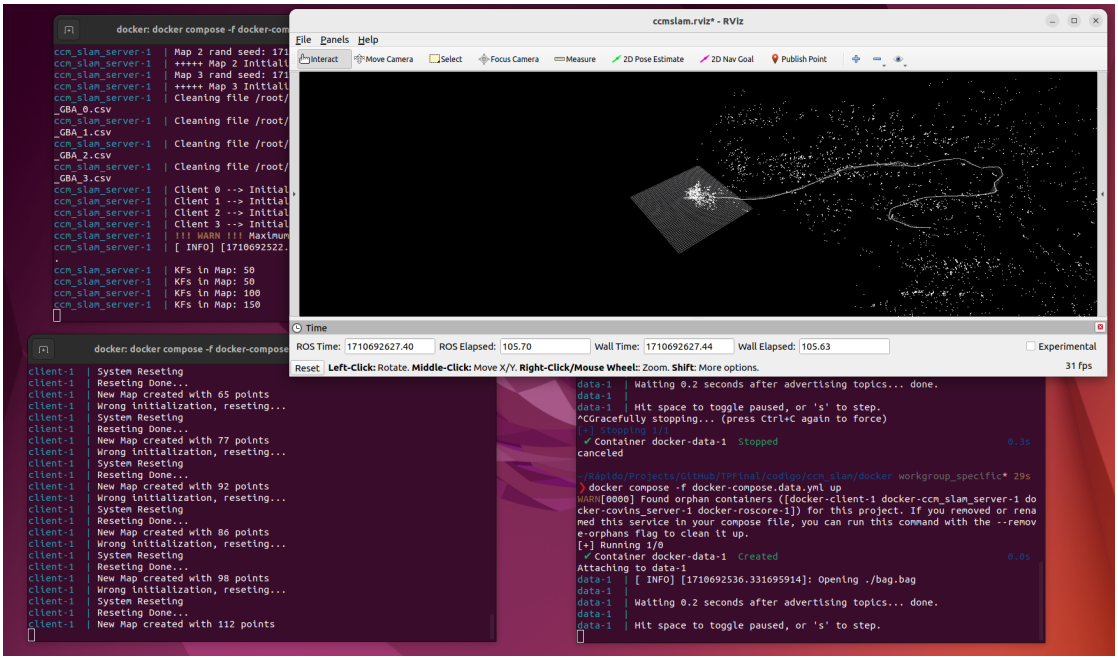

Figura 3: Ejecucion del sistema

Durante la ejecución de `Machine Hall 03` (en azul) se perdió el tracking, evento del cual (por admision del paper) el sistema no se puede recuperar, sin embargo genero una ruta. La ruta generada para Machine Hall 03 esta mal posicionada[4], pero parece estar razonablemente bien trackeada en el mapa propio del dron.

Durante la ejecución de `Machine Hall 04` (estaría en rojo, pero no aparece) se perdió el tracking, y luego el servidor borró la trayectoria en el mapa. En este caso se intento reiniciar el cliente y reproducir los datos nuevamente, pero no se consiguió que el sistema pueda reencontrar los puntos, cerrar algún ciclo o juntar mapas. El `Machine Hall 05` (etiquetado "Dificil" por Michael Burri el al, 2016), no se ejecutó.

Replicación del paper con datos propios

El limite de CCM-SLAM es de 4 agentes máximo de forma independiente por lo que tuvimos que prescindir del quinto agente. Luego al momento de levantar el servidor y los clientes no presentaron mayores dificultades. Se corrió cada agente en un cliente distinto, reproducimos los datos y verificamos con rviz que los tópicos se vean reflejados correctamente. Inicialmente se testearon los agentes uno a uno, luego se testeo todo en simultaneo, no hubo diferencias entre estos dos casos, así que se habla de ambos indistintamente.

En el proceso tuvimos problemas de robustez más graves que en momento de la prueba con el dataset EuRoC, todos los agentes tuvieron problemas de "tracking lost" en algún momento del camino, generalmente en el momento de los grandes giros. Para intentar mitigar los "tracking lost", siguiendo las recomendaciones del paper, se testeo con 5000 keyframes guardados en el cliente, incluso así no hay mejoras de posicionamiento, ver figura 5.

[4]Ver "The EuRoC micro aerial vehicle datasets" figura 4 (Michael Burri el al, 2016)

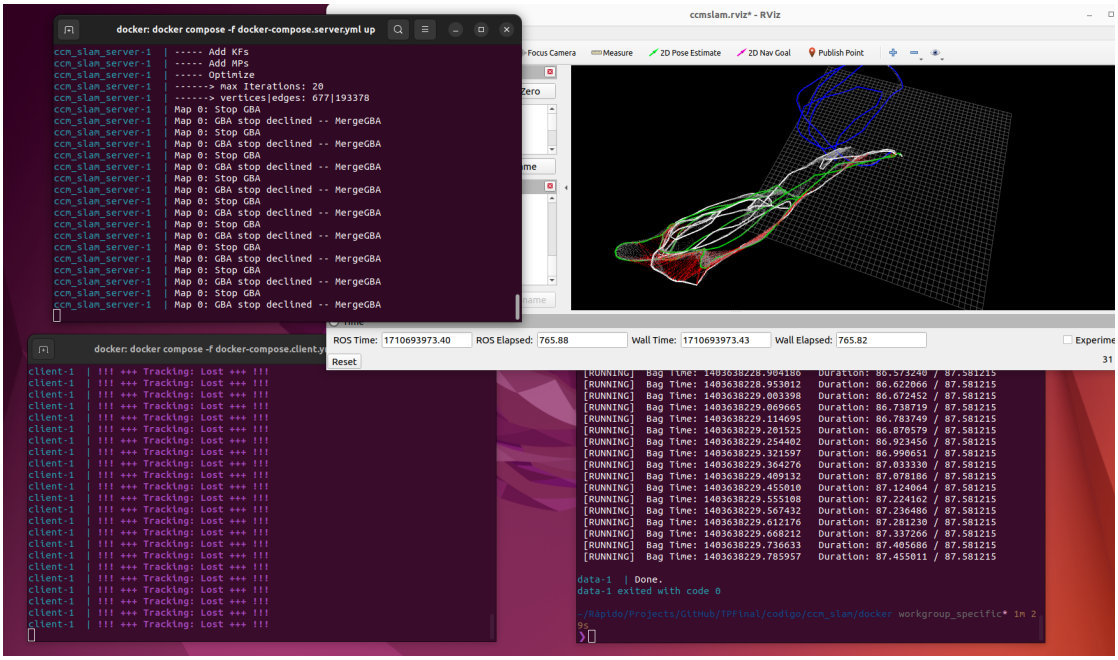

Figura 4: EuRoC 01 a 03

El cambio de los parámetros en los archivos de configuración no quedo excluido de la replicacion del paper. Con la intención de reducir los problemas en el trackeo de los sucesivos frames, aumentamos el numero de iteraciones para forzar a que RANSAC pueda converger a un modelo que se ajuste a nuestros datos. Sin dejar de aclarar que también bajamos el umbral de matcheo de un 0.95 a un 0.80 para facilitar la probabilidad de los matches entre frames. Aun así con estos cambios no logramos hacer que disminuyera el error de “tracking lost” usando nuestro dataset. Queremos destacar la falta de información de los parámetros para calibrar el modelo interno de ORB-SLAM, DBoW2 o CCM-SLAM.

5. Conclusiones

Marco teórico

El paper provee incrementos considerables a la estabilidad de traqueo de múltiples drones independientes, este área de investigación es importante tanto para el control y posicionamiento de drones de todo tipo, como para la integración de sensores. Si bien se usan ORB-SLAM y DBOW2, el sistema provee un tejido conectivo muy versátil y sorprendentemente extensible para la información que estos sistema proveen. La forma en que se propone la separación de tareas entre clientes y servidor es novedosa y robusta, permitiendo al servidor operar independientemente para generar y distribuir datos, mientras los clientes recolectan datos de una forma que requiere poca computación. Además, al no hacer uso de IMU, se simplifica el proceso de calibración y levantado.

Si bien el avance con respecto a la investigación anterior es sustancial, el sistema tiene problemas para adaptarse a diferentes entornos, rápidamente perdiendo la capacidad de mapear cuando el entorno varia rápidamente, u ocurren errores o perdidas de los sensores del lado del

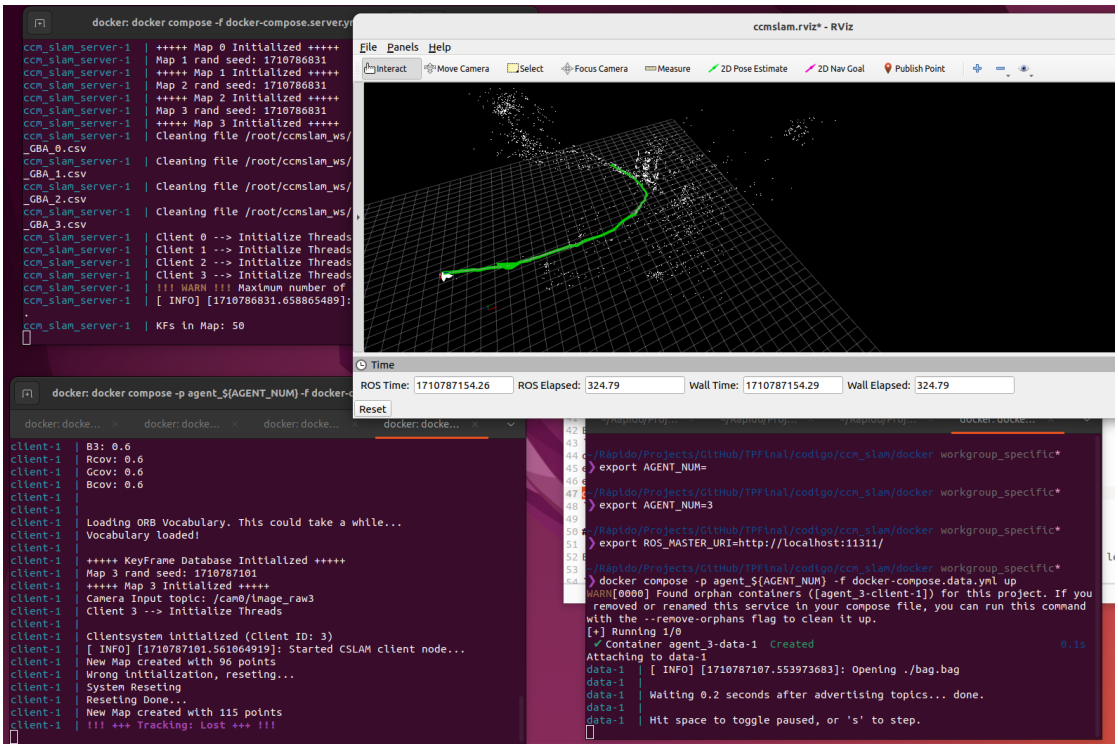

Figura 5: Testeo con 5000 KFs

cliente. El enfoque que se hace sobre la cantidad de mensajes y el peso (en MB) de la transmisión en la red es sustancial, sin embargo los casos de uso que se presentan son de pocos drones, donde la capacidad de transmisión del hardware actual sobrepasa sustancialmente la capacidad necesitada por los (cuatro) clientes propuestos en las pruebas.

Replicación

En el proceso de replicación del paper a partir de nuestros datos nos dimos cuenta que los problemas de robustez en parte son consecuencia del tipo de datos en dataset. El entorno a mapear, es altamente cambiante captura a captura por lo que tenemos pocas coincidencias entre capturas consecutivos haciendo que se pierda el posicionamiento de la trayectoria. Generalmente este problema lo tuvimos en partes del recorrido donde el Create 3 realizaba giros sobre su propio eje, posiblemente los frames captados en los giros sean altamente cambiantes entre si; en contraposición, si hubiesemos realizado giros muy abiertos con el Create 3 los problemas se hubiesen mitigado. En la representación de rviz podemos ver los trayectos realizados por los agentes ejecutados hasta el momento en el que el cliente pierde la posición.

Otro problema es debido a los parámetros de optimización. Sabiendo que la identificación de bucles y el reconocimiento de lugares tienen su base en ORB-SLAM y que tiene limitadas iteraciones puede ser que no este ajustando óptimamente con nuestros frames. Y es posible que en el proceso de reconocimiento de lugares cuando se busquen KF similares a uno recién capturado, no se tengan guardados candidatos suficientemente similares.

Otra cosa que detectamos es que el servidor no pudo ubicar la posición inicial de los agentes (que se lograria en el proceso de fusión de mapas), el servidor al no poder detectar map points (MP) similares entre partes de los recorridos de los clientes no puede transformar el marco de coordenadas entre ellos. Lo anterior es sumamente importante a la hora del mapeo ya que esta transformación de coordenadas queda disponible para todos los clientes.

La inicialización es inestable, tanto con nuestra data como con EuRoC. En el inicio de los clientes cuando se reproducen las rosbags correspondientes se puede llegar a dar problemas de posicionamiento. Lo que nos lleva a concluir sobre la imposibilidad de CCM-SLAM a recomponerse ante errores de "tracking lost"[5]. Si bien el paper afirma que un agente cuya comunicación se perdió se puede llegar a recuperar, cuando tira el error "tracking lost" ya no se recupera. Lo anterior lo podemos simular cortando la reproduccion de un rosbag y reproduciendo otro o la misma. Este caso de prueba emula el hecho de que un agente pierde la conexión en una localización A y luego vuelve a conectarse en una localización B, teniendo que readaptarse rápidamente el mapeo de dicho agente. ¿Por que es la conexión recuperable y el error de trackeo no? Porque el error "tracking lost" impide al cliente seguir funcionando.

Comparativa y discrepancias

Claramente CCM-SLAM tiene problemas en su capacidad de mantener o recuperar la posición relativa de un dron (relativa a si mismo). Cuando se compara el resultado obtenido en el ejemplo del video[6] y en Machine Hall easy contra nuestro caso y Machine Hall medium, en los casos donde la escena cambia "de a poco" CCM-SLAM es capaz de mantener la posición; pero en los casos donde hay cambios bruscos, no se logran crear suficientes puntos comunes que permitan al cliente continuar con el proceso de generacion de mapa individual.

Se detectaron problemas de perdida de los datos graficados e inestabilidad al juntar mapas (global bundle adjustment), esto implica que el servidor (por completo) se vuelve inestable

[5] Para mas detalle vease: https://github.com/VIS4ROB-lab/ccm_slam/issues/59.

[6] Ver https://youtu.be/P3b7UiTlmbQ

cuando detecta que mapas relativos se pueden unir, potencialmente afectando otros clientes y comprometiendo el objetivo general.

Analizando los parámetros propuestos en el paper, parece se que se eligieron los mejores parámetros ejecutando una y otra vez datasets como el EuRoC, sin embargo, reutilizar estos parámetros no parece ser muy factible. Esto implica que cuando se quiere empezar una nueva prueba, se debería hacer una elección de parámetros ad-hoc "al vuelo", a esto se le agrega que para cambiar los parámetros se debe reiniciar el servidor; lo que puede implicar elección de parámetros suboptimos e inestabilidad incrementada en casos de prueba a campo.