Sistema de Odometría Visual-Inercial para Aplicaciones XR

1st Bruno Zanotti
Departamento de Ciencias de la Computación
FCEIA-UNR
Rosario, Argentina
brunozanotti96@gmail.com

2nd Mateo De Mayo
Collabora Ltd.
Cambridge, Reino Unido
mateodemayo@gmail.com

3rd Taihú Pire
CIFASIS
CONICET-UNR
Rosario, Argentina
pire@cifasis-conicet.gov.ar

***Resumen*—**
En aplicaciones de realidad extendida (XR) es indispensable que el sistema de localización del dispositivo sea robusto frente a movimientos bruscos y oclusiones en las cámaras. A su vez para que la experiencia de usuario sea fluida e inmersiva debe ser liviano para que pueda correr en tiempo real. Los dispositos XR, como los cascos de realidad virtual (VR), combinan múltiples cámaras e IMU que pueden ser usados por métodos de Odometría Visual-Inercial (VIO) para estimar su localización.

En este paper se propone un sistema VIO capaz de estimar la localización de dispositivos XR. El sistema propuesto tiene como punto de partida al algoritmo de Basalt, con lo cual mantiene una ventana de keyframes que optimiza y marginaliza de forma inteligente para mantener el sistema computacionalmente eficiente, y la vez lo extiende permitiendo trabajar con cascos con más de dos cámaras no paralelas, utiliza la IMU para predecir la localización y es capaz de reidentificar features perdidas por oclusiones o movimientos bruscos.

Se realizaron experimentos sobre un dataset de un casco de XR para validar el sistema VIO desarrollado. Los resultados muestran que el sistema logra una mayor robustez y precisión, y una trayectoria suave, requerimientos necesarios para aplicaciones XR.

***Keywords*—Realidad Extendida, Realidad Virtual, Localización, VIO, SLAM.**

I. Introducción

La Realidad Extendida (XR, *Extended Reality*) es un término que engloba una variedad de tecnologías que combinan elementos del mundo real con elementos virtuales para crear experiencias inmersivas que van más allá de la realidad convencional. Dentro de este campo se puede distinguir la Realidad Virtual (VR, *Virtual Reality*), que sumerge al usuario en un entorno completamente digital, la Realidad Aumentada (AR, Augmented Reality), que superpone elementos virtuales al mundo real, y la Realidad Mixta (MR, Mixed Reality), que permite la interacción entre objetos reales y virtuales [1].

La gran mayoría de los cascos de XR recientes están equipados con múltiples cámaras y una IMU, esto se debe a su portabilidad, bajo costo, bajo consumo y que los sensores visuales e inerciales se complementan muy bien. La cámara, como sensor exteroceptivo, ofrece una amplia cantidad de información sobre el entorno, pero opera a baja frecuencia (generalmente entre 30 Hz a 60 Hz). En contraste, la IMU, sensor propioceptivo y compuesta por un giróscopo y un acelerómetro, provee información de velocidad angular y aceleración lineal a una alta frecuencia (entre 100 Hz a 1000 Hz). Las cámaras desempeñan un papel esencial al permitir mapear el entorno mediante la detección de características visuales, y utilizar dicho mapa para mejorar la localización. Sin embargo, son susceptibles a ruidos externos como la falta de iluminación en el ambiente, movimientos bruscos que generan imágenes difuminadas (*motion blur*) e incluso oclusiones parciales o totales. Por otro lado, la IMU es crucial para detectar movimientos abruptos en un corto período de tiempo, aunque acumula error de posición de manera cuadrática al integrar el movimiento a lo largo del tiempo. La combinación de estos dos tipos de sensores es fundamental para lograr una buena localización en aplicaciones de VR donde el usuario realiza movimientos bruscos y realiza oclusiones con sus manos al interactuar en el mundo virtual.

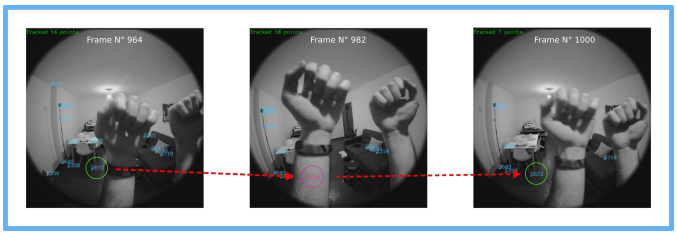

Figura 1. La secuencia de imágenes muestra como con la reidentificación de features se puede recuperar un feature luego de una oclusión. En color rosa se puede ver la proyección del punto 3D, que es matcheado cuando el punto es visible nuevamente.

A pesar de los avances significativos en las tecnologías VIO [2], [3] y VI-SLAM [4], [5], su implementación en entornos de XR presenta desafíos particulares que pueden afectar la experiencia del usuario. Dos problemas comunes en este tipo de aplicaciones es el error de localización de baja frecuencia, conocido como *latencia*, y el error de alta frecuencia, denominado *jitter*. El error de latencia hace referencia al retraso entre la acción del usuario y la respuesta del sistema. Este error se presenta más en sistemas de menor calidad y puede causar cinetosis o mareos debido a la desconexión entre las respuestas esperadas y las reales. Los sistemas modernos de alta calidad generalmente logran una latencia de 20 ms o menos para asegurar una experiencia fluida, empleando técnicas como

la predicción de movimiento y altas tasas de refresco para minimizar este problema [6]–[8]. El error de *jitter* se manifiesta como vibraciones o movimientos oscilatorios de alta frecuencia en la posición de los objetos virtuales en el entorno. Esta fluctuación constante puede perjudicar la experiencia del usuario al causar distracciones visuales y afectar la percepción de la estabilidad del entorno virtual [9], [10].

En este paper se presenta un método VIO desarrollado específicamente para aplicaciones de XR. El método propuesto está basado en el sistema del estado del arte Basalt [3], que es además utilizado por la plataforma XR de código abierto Monado como principal sistema de localización [11]. Basalt es un sistema VIO basado en optimización que emplea FAST [12] para detección de *features* y *optical flow* por KLT [13] para el *tracking*. El *back-end* optimiza una ventana de *keyframes* y utiliza una heurística para marginalizar *keyframes* manteniendo el sistema computacionalmente eficiente. El sistema propuesto lo extiende, implementando múltiples mejoras detalladas en la sección III.

Los resultados muestran que el sistema propuesto mejora Basalt original en términos de precisión y robustez. Por un lado, los resultados muestran un menor error RTE (*Relative Trajectory Error*) lo que implica trayectorias mas suaves, requerimiento particularmente importante en aplicaciones XR. A su vez, el sistema desarrollado es más robusto a breves oclusiones, frecuentes en aplicaciones VR (ver Figura 1.

II. Trabajo Relacionado

En esta sección se describen sistemas VIO y VI-SLAM claves del área y su relación con el problema de XR [1], [14].

Los primeros enfoques concebidos para abordar los desafíos de SLAM se apoyaron en la teoría de filtros de Kalman para estimar la pose de un dispositivo. Un ejemplo es S-MSCKF [15] es un sistema VIO estéreo basado en filtros de kalman que usa como punto de partida el trabajo de Mourikis & Roumeliotis (MSCKF [16]) lo que lo caracteriza por ser una solución computacionalmente eficiente. El estado contiene información de la IMU y una cantidad fija de poses de las cámaras, sin incluir los landmarks. Con el fin de mantener el sistema eficiente, una vez que la cantidad de poses alcanza un límite específico, algunas de ellas son *marginalizadas*. En la misma línea, otro enfoque basado en MSCKF [16] es OpenVINS [17], un sistema VIO que se caracteriza por calibrar los parámetros intrínsecos y extrínsicos en tiempo de ejecución (*online*), y fue diseñado como una plataforma base para que otros proyectos de investigación puedan extenderlo a sus requerimientos específicos.

Soluciones posteriores han logrado mejorar tanto la precisión como la eficiencia computacional al recurrir a métodos de optimización basados en mínimos cuadrados no lineales, también conocidos como *fixed-lag smoothers*, lo que ha permitido que sean más idóneos para aplicaciones en campos como robótica y XR [18].

Una propuesta de este tipo de sistemas es VINS-Fusion [19]–[21], una extensión de VINS-Mono [22] que soporta múltiples sensores y múltiples configuraciones de cámaras e IMU: cámara monocular con IMU, cámara estéreo con IMU e incluso solo cámara estéreo. Además, cuenta con un módulo de relocalización, el cual busca correspondencias entre la imagen actual y el mapa reconstruido y las incorpora a la función de costo de la optimización con el objetivo reducir el *drift*.

Otro ejemplo es Basalt [3], en este caso un sistema VIO que se caracteriza por extraer factores no lineales (*non-linear factors*) al marginalizar *keyframes* que luego utiliza para optimizar un mapa de forma global. Los autores proponen un sistema a dos capas: Por un lado un sistema VIO basado en optimización donde el *front-end* emplea el algoritmo FAST [12] para detección de *keypoints* y KLT para el *tracking* y *matching*. El *back-end* optimiza una ventana de keyframes aplicando *bundle adjustment* y utiliza una heurística para marginalizar keyframes manteniendo el sistema computacionalmente eficiente. Cada vez que un keyframe es removido Basalt calcula un término de marginalización que usa en la optimización del próximo paso. En cuanto a la segunda capa, Basalt procesa los datos marginalizados por el VIO (de manera offline) para crear un mapa global y obtener nuevos emparejamientos de puntos para luego optimizarlo mediante un *Global Bundle Adjustment* ajustando las poses de todos los keyframes y landmarks. Si bien Basalt no es un sistema SLAM porque no genera un mapa en tiempo real, el VIO logra métricas comparables con otros sistemas del estado del arte y el resultado es un mapa globalmente consistente.

Kimera [23] es una librearía compuesta por diferentes módulos. Kimera-VIO es un sistema VIO basado en el trabajo de Forster [24]. El *front-end* usa el detector Shi-Tomasi [25], implementa detección de features, matching estéreo y verificación geométrica para keyframes y realiza KLT tracking para frames intermedios. El *back-end* usa la librería GTSAM [26] para optimizar y preintegrar información de la IMU. El módulo Kimera-RPGO se encarga de detección y cierre de ciclos (*loop closures*) y de recomputar la pose de los *keyframes* para que sean globalmente consistentes. Los módulos *Kimera-Mesher* y *Kimera-Semantics*, implementan diferentes métodos para confeccionar una malla 3D, útil para la evasión de obstáculos.

Otro ejemplo de un sistema SLAM basado en optimización es ORB-SLAM3 [5] que parte de ORB-SLAM2 [27] y ORB-SLAM-VI [28], notable por sus excelentes métricas y su capacidad de generar múltiples mapas y reutilizar información de sesiones anteriores. Es capaz de trabajar solo con datos visuales o con datos visuales e inerciales, en configuración mono, estéreo o con sensores RGB-D. El funcionamiento del sistema se divide en tres hilos principales: el hilo de *tracking* procesa la información de los sensores y computa la pose del frame actual con respecto al mapa activo en tiempo real, a su vez decide cuando un frame se convierte en *keyframe* y si el *tracking* se pierde, intenta relocalizarse en alguno de todos los mapas generados hasta el momento. El hilo del mapa local se encarga de refinar el mapa activo agregando y quitando puntos y *keyframes*, y de optimizarlo aplicando *bunde adjustment* sobre una ventana de *keyframes*. Por último, el hilo encargado de "*map merging*" reconoce zonas comunes entre mapas, realiza detección y cierre de ciclos, y unificación

de mapas. OKVIS 2 [4] es un sistema VI-SLAM con soporte para trabajar en tiempo real, multi-cámara que implementa detección y cierre de ciclos (*loop closure*). El sistema se divide en *front-end* y *real time estimator* que procesan los frames y datos de la IMU de manera sincrónica. El *front-end* se encarga de la inicialización, detección y emparejamiento de keypoints, triangulación de frames sucesivos o provenientes de múltiples cámaras, reconocer y remover objetos dinámicos mediante una red neuronal especializada y detectar zonas ya visitadas. Por otra parte, la función del estimador es optimizar el *factor-graph* construido marginalizando las observaciones más antiguas y fijando estados.

MapLab 2.0 [29] es un sistema SLAM completo, aunque presenta un enfoque diferente a los previamente mencionados ya que es una plataforma *open-source* orientada a la investigación pensada para desarrollar e integrar nuevos módulos. Maplab 2.0 extiende *Maplab* [30] que ya permitía integrar sistemas VIO. En esta nueva versión admite múltiples sesiones, múltiples robots y adaptabilidad a diversas configuraciones de sensores. Además, ofrece una modalidad de mapeo colaborativo online que utiliza un servidor para compilar y distribuir el mapa globalmente y mantenerlo consistente.

Es importante destacar que si bien todos los sistemas VIO y VI-SLAM presentados pertenecen al estado del arte, fueron concebidos para aplicaciones de robótica, y por tanto los desafíos propios de aplicaciones XR, mencionados en la sección I, no son debidamente contemplados. Para el desarrollo del sistema VIO propuesto en este trabajo se toma como punto de partida el sistema Basalt [3] dado que es capaz funcionar en tiempo real y que presenta un bajo error de RTE con respecto al resto de los sistema del estado del arte. Además, basalt tiene una licencia permisiva (BSD de 3 cláusulas) lo que facilita su uso y extensión por la comunidad de XR.

III. Método propuesto

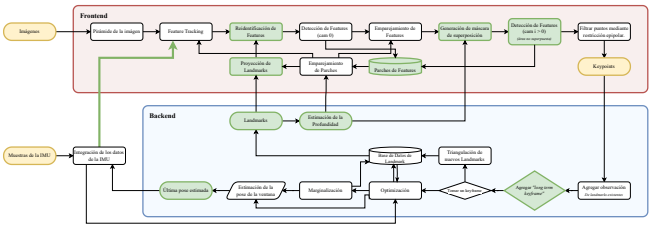

Figura 2. Estructura del sistema VIO propuesto. En verde las adiciones por sobre Basalt original.

En este trabajo, presentamos una serie de nuevas funcionalidades en la implementación principal de Basalt, ver Figura 2. Estas están diseñadas para mejorar su rendimiento en diversas situaciones y escenarios de uso, con enfoque en los casos de XR. Estas mejoras abordan aspectos clave del proceso de localización, desde la predicción inicial para hacer feature matching hasta la gestión eficiente de la ventana de keyframes.

III-A. *Predicción Inicial de Feature Matches*

Basalt emplea optical flow para el feature matching entre cámaras, minimizando el error de un parche de píxeles con un diámetro de 8 píxeles mediante una norma invariante a la iluminación (LSSD). Sin embargo, este enfoque requiere una buena estimación inicial de la posición 2D de la feature para garantizar el éxito del matching. Inicialmente, se utilizaba la misma coordenada del píxel donde se encontraba la feature en una cámara como estimación inicial para buscarla en otra cámara con optical flow. Aunque este método funciona bien para cámaras con orientaciones suficientemente paralelas, no generaliza eficazmente, especialmente en situaciones donde las cámaras no están alineadas. Para abordar esta limitación, se propone un enfoque versátil que prescinde de la curva epipolar, estimando una profundidad promedio en función de las features actuales. Las features 2D a emparejar se reproyectan a 3D utilizando esta profundidad estimada, luego se proyectan a la cámara donde se busca la feature, estableciendo así su posición inicial. Es notable que este método funciona adecuadamente, dado que, por lo general, las features experimentan cambios mínimos en la curva epipolar proyectada en la cámara de destino durante movimientos a cierta profundidad. Con esta mejora, Basalt incrementa significativamente el número de coincidencias exitosas en cámaras no alineadas, lo que le permite funcionar en conjuntos de datos que antes resultaban inaccesibles para la versión original.

III-B. *Utilización de Áreas de Cámara sin Superposición*

Aunque la mejora anterior permite encontrar features en cámaras no alineadas, estas son exclusivamente las presentes en la cámara principal (generalmente la izquierda). Como consecuencia, las áreas sin superposición con la cámara principal en otras cámaras carecen de features identificables. Para poder utilzar el campo de visión completo, se lleva a cabo una redetección de features en la imagen secundaria, tratándola como si fuera la imagen principal. Sin embargo, con el fin de evitar duplicar esfuerzo al detectar en la misma región (dada la costosa naturaleza de la detección) y aprovechar lo ya observado por la cámara principal, se ha desarrollado un sistema de máscaras para ignorar regiones específicas de la imagen. En particular, se ha diseñado un proceso que genera máscaras en las áreas de superposición de las cámaras, permitiendo así la detección exclusivamente en las áreas sin superposición y haciendo uso del campo de visión completo del conjunto de cámaras del dispositivo.

III-C. *Soporte para Más de Dos Cámaras*

La naturaleza flexible de las ecuaciones de optimización en el backend de Basalt facilita su extensión para admitir la optimización de más de dos cámaras. Las limitaciones previas en el frontend que impedían esta funcionalidad se han superado con las mejoras mencionadas anteriormente. Por lo tanto, la expansión del sistema para soportar más de dos cámaras resulta una extensión intuitiva y sencilla. Aunque algunos cascos utilizan solo dos cámaras, como el Samsung Odyssey+ o el Valve Index, la mayoría emplea actualmente 4

(p. ej., Oculus Quest 2, HP Reverb G2) o 5 (p. ej. Oculus Rift S), lo que ofrece una mayor robustez frente a la oclusión de cámaras y un campo de visión más amplio para rastrear más features.

III-D. Uso de la IMU en el Feature Tracking

Otro desafío importante en el seguimiento de features es la capacidad de mantener los feature tracks a lo largo del tiempo. Basalt, en su versión original, se basaba únicamente en información de las imágenes para realizar este seguimiento. La utilización del optical flow, similar al utilizado en la sección III-A, pero aquí para realizar feature matching entre dos imágenes consecutivas temporalmente de una misma cármara, proporcionaba una estimación inicial de la posición de la feature en el cuadro actual, basada en su posición en el cuadro anterior. Aunque esta estimación es efectiva en movimientos de rapidez moderada, los movimientos bruscos resultaban en la pérdida de la mayoría de los feature tracks. Para abordar este problema, se ha incorporado el uso de la IMU en el frontend. A la última pose estimada de la IMU, proporcionada por el backend, se le integran las mediciones de la IMU recopiladas después de la exposición de la imagen que generó la estimación de la pose. Las features se reproyectan sobre la misma cámara utilizando esta nueva pose, y las coordenadas resultantes se utilizan como estimación inicial para continuar el seguimiento de features. Esta funcionalidad ha mejorado significativamente la robustez de Basalt ante movimientos bruscos, lo que ha permitido aumentar considerablemente la longitud promedio de los feature tracks. Aunque esta mejora puede no ser evidente en conjuntos de datos simples como EuRoC [31], en otros conjuntos con movimientos bruscos continuos, puede marcar la diferencia entre el funcionamiento normal del sistema y su interrupción debido a la falta de features.

III-E. Reidentificación de Features

La versión original de Basalt carecía de la capacidad de reidentificar features una vez que estas salían del campo de visión. De desarrolló esta funcionalidad, lo que constituye un requisito fundamental para la implementación futura de un mapa persistente. Aunque actualmente esta mejora ofrece beneficios sustanciales en la capacidad de seguir rastreando features ante oclusiones temporales, como se ilustra en la Figura 3, también proporciona mejoras en la consistencia de la trayectoria al volver a ver features que aún no están en el campo de visión pero siguen presentes en la ventana de optimización. Por ejemplo, durante movimientos de ida y vuelta, la reidentificación de features vistas previamente ayuda a mantener la consistencia de la trayectoria y reducir el error acumulado. En algunos casos, esta funcionalidad, combinada con la mejoras explicada a continuación, ha mejorado sustancialmente las métricas de ciertos conjuntos de datos en comparación con Basalt original. Sin embargo, se requieren más métricas y pruebas futuras para determinar la configuración óptima para esta funcionalidad.

III-F. Extensiones de la Ventana de Keyframes

Por último, se han realizado varias mejoras para ampliar la versatilidad de la ventana de optimización, adaptandonos al método novedoso de optimización implementado por Basalt que se se basa en almacenar la raíz cuadrada del Hessiano en lugar del Hessiano directamente [32], [33]. Esto incluye:

Mantenimiento de Keyframes: Ahora es posible marcar keyframes para que sean mantenidos en la ventana y no sean marginalizados, lo que resulta útil para diversos propósitos, como mapear puntos de vista clave del entorno del casco. Esto representa un primer paso para reducir significativamente el error acumulado en la operación del sistema VIO.

Fijación de Keyframes Mantenidos: Se puede ahora optar por fijar los keyframes marcados en la optimización, creando así partes de un mapa estático. Esto es especialmente útil para poder utilizar mapeos preliminares que se saben de alta calidad de la habitación donde se ejecutará la experiencia de realidad extendida.

(De)Serialización de Keyframes y Puntos de Referencia: Se ha agregado la capacidad de guardar y cargar keyframes marcados y sus correspondientes landmarks asociadas desde el disco. Esto sucede mediante la capacidad de agregar y quitar keyframes de la ventana de forma arbitraria. Esta funcionalidad sería especialmente útil para integrar un hilo de mapeo que complemente el VIO con un mapa globalmente optimizado.

Heurística de Marginalización: Se ha implementado una heurística alternativa de marginalización que resulta particularmente útil cuando se combina con la funcionalidad de reidentificación de features detallada en III-E. Esta mejora ha demostrado, con una capacidad de cómputo similar, mejorar abruptamente las métricas en los conjuntos de datos *room* del conjunto TUM-VI [34]. En particular se mejoro la ATE de los conjuntos room1-5 de 0.108, 0.078, 0.124, 0.058, 0.173, 0.021 respectivamente con un promedio de **0.094** a errores de 0.031, 0.037, 0.062, 0.035, 0.030, 0.021 con un promedio de **0.036**.

IV. Experimentos

En esta sección se presentan los resultados obtenidos por el método propuesto sobre las secuencias del Monado SLAM Dataset [35]. Estas secuencias fueron grabadas mientras un usuario usaba una aplicación XR. Todas las secuencias utilizadas están compuestas por datos grabados durante una sesión utilizando el casco de VR Valve Index con tres sensores externos lighthouses que proveen una ground-truth de buena calidad [36] para nuestra trayectoria. La computadora utilizada para la experimentación cuenta con un procesador AMD Ryzen 5 7600X 6-Core x12 con 32 GB de RAM.

IV-A. Basalt original vs Reidentificación de features

La reidentificacion de features busca mejorar la robustez general del sistema frente a oclusiones o movimientos bruscos típicos de aplicaciones de XR. En la Figura 3 se puede apreciar un ejemplo del funcionamiento de la reidentificación de features en donde se logra identificar un feature luego de que

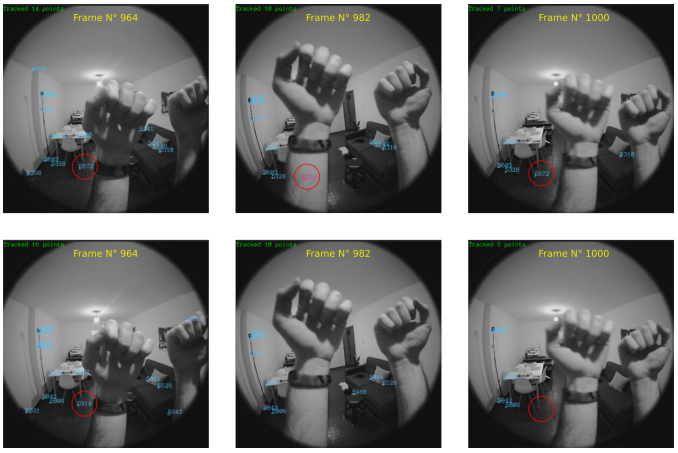

Figura 3. Comparativa del método con y sin reidentificación en la secuencia MIO02 [35]. Las primeras tres imágenes muestran frames de la secuencia con la reidentificación activada, mientras que las imagenes inferiores muestran los mismos frames sin reidentificación. En ambos casos se puede observar como en el frame N°964 se detecta una feature en la pata de la silla (ID 5672 y 5914 respectivamente), luego el tracking nativo de Basalt la pierde debido a una oclusión (frame N°982), con la diferencia de que con reidentificación activada se proyecta la misma sobre la imagen (feature ID 5672 en color rosa) intentando recuperarla. Por último en el frame N°1000, luego de la oclusión, se puede observar que en el primer caso se logró matchear y recuperar la feature mientras que en el segundo caso se pierde.

la cámara sufre una oclusión parcial. La secuencia utilizada es MIO02 en donde en promedio el 10.97 % de los features detectados por el front-end del sistema son provenientes de la reidentificación. Esto permite poder detectar un número menor de nuevos features en el frame actual. Además, la información (restricciones en el grafo) que aportan los features reidentificados es mayor a la que proverían si en lugar de ellos se detectaran nuevos features. Como consecuencia esta mejora tiene un impacto directo en las métricas de error, en este caso el error de trayectoria absoluto de Basalt original es de $70{,}885 \pm 50{,}509\,\mathrm{m}$ comparado con $1{,}252 \pm 0{,}653\,\mathrm{m}$ que se obtiene con la reidentificación activada.

IV-B. Métricas de datasets públicos

Para validar las mejoras presentadas en este trabajo se comparan las métricas ATE y RTE entre Basalt original y el método propuesto sobre las secuencias de Monado SLAM Datasets [35]. En la Tabla I se muestran los resultados obtenidos. A diferencia de otros trabajos en el área de SLAM, nos enfocamos también en el RTE que tiende a ser un buen primer indicador del nivel de cinetosis que el estimador puede producir en el usuario, ya que indica la precisión segmentos de la trayectoria. Como se puede observar en la Tabla I el sistema propuesto obtiene mejores resultados que Basalt en todos los datasets, especialmente vemos una notable reducción del error en datasets como MIO01, MIO02, MIO03 y MIO04 que presentan múltiples oclusiones de cámaras propias de aplicaciones de XR.

V. Conclusiones y trabajo futuro

En este trabajo se presenta un sistema VIO para la estimación de la trayectoria para dispositivos XR. El sistema VIO extiende Basalt incorporando predicción de la pose mediante IMU, la capacidad de funcionar con más de dos cámaras no paralelas e implementando una funcionalidad de reidentificación de features mediante la cual los puntos 3D vistos por keyframes de la ventana son proyectados al nuevo frame en el front-end y se los intenta matchear mediante un parche de píxeles.

Se realizaron experimentos utilizando secuencias capturadas por un casco VR, dichas secuencias presentan desafíos para sistemas VIO/VI-SLAM ya que contienen momentos de oclusión parcial y total de las cámaras, movimientos bruscos, zonas con poca textura y cambios de iluminación. El sistema VIO desarrollados es capaz de localizar al casco VR con precisión y delineando una trayectoria suave (reduciendo el jitter). Además, el sistema es capaz de recuperarse ante oclusiones parcial o total de la cámara.

Como trabajo futuro se planea, transformar el sistema VIO en un sistema VI-SLAM, incorporando la capacidad de mantener un mapa persistente de la escena para la reducción del error de localización acumulado *drift*, implementando técnicas de detección y cierre de ciclos similar a las utilizada en [4], [5], [37].

Agradecimientos

Este trabajo fue apoyado por CONICET (PIBAA Nº 0042), AGENCIA I+D+i (PICT 2021-570), Universidad Nacional de Rosario (PCCT-UNR 80020220600072UR) y Collabora Ltd. con el Servicio Técnico de Alto Nivel con el Laboratorio de Robótica del CIFASIS (CONICET-UNR).

Referencias

[1] S. M. LaValle, *Virtual Reality*. Cambridge ; New York: Cambridge University Press. [Online]. Available: http://lavalle.pl/vr/

[2] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual–inertial odometry using nonlinear optimization," *Intl. J. of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015.

[3] V. Usenko, N. Demmel, D. Schubert, J. Stueckler, and D. Cremers, "Visual-Inertial Mapping with Non-Linear Factor Recovery," *(IEEE) Robotics and Automation Letters*, vol. 5, no. 2, pp. 422–429, 2020.

[4] S. Leutenegger, "OKVIS2: Realtime Scalable Visual-Inertial SLAM with Loop Closure," *arXiv e-prints*, 2022.

[5] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual–Inertial, and Multimap SLAM," *IEEE Trans. Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.

[6] M. S. Elbamby, C. Perfecto, M. Bennis, and K. Doppler, "Toward Low-Latency and Ultra-Reliable Virtual Reality," *IEEE Network*, vol. 32, no. 2, pp. 78–84, 2018.

[7] Y. H. P. Iskandar, L. Gilbert, and G. B. Wills, "Reducing latency when using Virtual Reality for teaching in sport," in *2008 International Symposium on Information Technology*, vol. 3, 2008, pp. 1–5.

[8] J.-P. Stauffert, K. Korwisi, F. Niebling, and M. E. Latoschik, "Ka-Boom!!! Visually Exploring Latency Measurements for XR," in *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*. New York, NY, USA: Association for Computing Machinery, May 2021, no. 20.

[9] M. H. Mughrabi, A. K. Mutasim, W. Stuerzlinger, and A. U. Batmaz, "My Eyes Hurt: Effects of Jitter in 3D Gaze Tracking," in *IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops*, 2022, pp. 310–315.

Tabla I
ERRORES DE TRAYECTORIA ABSOLUTOS Y RELATIVOS.

	Basalt		**Método propuesto**	
	ATE [m]	RTE [m]	ATE [m]	RTE [m]
MIO01	638.616 ± 419.956	3.472079 ± 2.872924	**0.547 ± 0.293**	**0.032950 ± 0.030879**
MIO02	70.885 ± 50.509	0.971657 ± 0.840440	**1.252 ± 0.653**	**0.076237 ± 0.072524**
MIO03	0.141 ± 0.074	0.017960 ± 0.015747	**0.072 ± 0.028**	**0.006790 ± 0.004552**
MIO04	9.025 ± 4.486	0.079876 ± 0.069436	**0.146 ± 0.075**	**0.015719 ± 0.012333**
MIO05	0.076 ± 0.043	0.009715 ± 0.008960	**0.026 ± 0.011**	**0.003022 ± 0.001855**
MIO06	0.137 ± 0.078	0.022517 ± 0.020026	**0.044 ± 0.021**	**0.011110 ± 0.008883**
MIO07	0.052 ± 0.036	0.017062 ± 0.016538	**0.018 ± 0.007**	**0.003627 ± 0.002288**
MIO08	0.093 ± 0.065	0.016872 ± 0.013768	**0.035 ± 0.012**	**0.009127 ± 0.005315**
MIO09	0.018 ± 0.013	0.012796 ± 0.011451	**0.006 ± 0.003**	**0.004647 ± 0.003280**
MIO10	0.047 ± 0.029	0.029644 ± 0.026315	**0.014 ± 0.007**	**0.005494 ± 0.003604**
MIO11	0.067 ± 0.034	0.022290 ± 0.019256	**0.029 ± 0.011**	**0.008820 ± 0.005085**
MIPB01	0.694 ± 0.230	0.009045 ± 0.007417	**0.028 ± 0.011**	**0.006151 ± 0.003973**
MIPB02	0.295 ± 0.119	0.009954 ± 0.008812	**0.044 ± 0.016**	**0.004942 ± 0.003358**
MIPB03	0.252 ± 0.147	0.015367 ± 0.014132	**0.088 ± 0.054**	**0.006359 ± 0.004691**
MIPB04	0.224 ± 0.182	0.010405 ± 0.008910	**0.028 ± 0.017**	**0.004853 ± 0.003356**
MIPB05	0.100 ± 0.062	0.006339 ± 0.004660	**0.019 ± 0.008**	**0.004169 ± 0.002530**
MIPB06	0.241 ± 0.186	0.011956 ± 0.010356	**0.033 ± 0.013**	**0.004153 ± 0.002625**
MIPB07	0.147 ± 0.073	0.005890 ± 0.004102	**0.017 ± 0.009**	**0.004867 ± 0.003031**
[AVG]	40.062 ± 26.462	0.263412 ± 0.220736	**0.136 ± 0.069**	**0.011835 ± 0.009676**

[10] J. P. Wilmott, I. M. Erkelens, T. S. Murdison, and K. W. Rio, "Perceptibility of Jitter in Augmented Reality Head-Mounted Displays," in *2022 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2022, pp. 470–478.

[11] M. de Mayo, "Localización visual inercial en tiempo real para aplicaciones de xr." [Online]. Available: https://rdu.unc.edu.ar/handle/11086/24898

[12] E. Rosten and T. Drummond, "Machine Learning for High-Speed Corner Detection," in *Eur. Conf. on Computer Vision (ECCV)*, A. Leonardis, H. Bischof, and A. Pinz, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 430–443.

[13] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, ser. IJCAI'81. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., Aug. 1981, pp. 674–679.

[14] G. Welch and E. Foxlin, "Motion tracking: No silver bullet, but a respectable arsenal," *IEEE Computer Graphics and Applications*, vol. 22, no. 6, Nov. 2002.

[15] K. Sun, K. Mohta, B. Pfrommer, M. Watterson, S. Liu, Y. Mulgaonkar, C. J. Taylor, and V. Kumar, "Robust Stereo Visual Inertial Odometry for Fast Autonomous Flight," *(IEEE) Robotics and Automation Letters*, vol. 3, no. 2, pp. 965–972, 2018.

[16] A. I. Mourikis and S. I. Roumeliotis, "A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, 2007, pp. 3565–3572.

[17] P. Geneva, K. Eckenhoff, W. Lee, Y. Yang, and G. Huang, "OpenVINS: A Research Platform for Visual-Inertial Estimation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 4666–4672.

[18] H. Strasdat, J. Montiel, and A. J. Davison, "Visual SLAM: Why filter?" *Image and Vision Computing*, vol. 30, no. 2, pp. 65–77, 2012.

[19] T. Qin, J. Pan, S. Cao, and S. Shen, "A General Optimization-based Framework for Local Odometry Estimation with Multiple Sensors," 2019.

[20] T. Qin, S. Cao, J. Pan, and S. Shen, "A General Optimization-based Framework for Global Pose Estimation with Multiple Sensors," 2019.

[21] T. Qin and S. Shen, "Online Temporal Calibration for Monocular Visual-Inertial Systems," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 3662–3669.

[22] T. Qin, P. Li, and S. Shen, "VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator," *IEEE Trans. Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[23] A. Rosinol, M. Abate, Y. Chang, and L. Carlone, "Kimera: an open-source library for real-time metric-semantic localization and mapping," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2020, pp. 1689–1696.

[24] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-Manifold Preintegration for Real-Time Visual–Inertial Odometry," *IEEE Trans. Robotics*, vol. 33, no. 1, pp. 1–21, 2017.

[25] J. Shi and Tomasi, "Good features to track," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 1994, pp. 593–600.

[26] F. Dellaert, "Factor graphs and gtsam: A hands-on introduction," *Georgia Institute of Technology, Tech. Rep*, vol. 2, p. 4, 2012.

[27] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras," *IEEE Trans. Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[28] ——, "Visual-Inertial Monocular SLAM With Map Reuse," *(IEEE) Robotics and Automation Letters*, vol. 2, no. 2, pp. 796–803, 2017.

[29] A. Cramariuc, L. Bernreiter, F. Tschopp, M. Fehr, V. Reijgwart, J. Nieto, R. Siegwart, and C. Cadena, "Maplab 2.0 – A Modular and Multi-Modal Mapping Framework," *(IEEE) Robotics and Automation Letters*, vol. 8, no. 2, pp. 520–527, 2023.

[30] T. Schneider, M. Dymczyk, M. Fehr, K. Egger, S. Lynen, I. Gilitschenski, and R. Siegwart, "Maplab: An Open Framework for Research in Visual-Inertial Mapping and Localization," *(IEEE) Robotics and Automation Letters*, vol. 3, no. 3, pp. 1418–1425, 2018.

[31] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The EuRoC micro aerial vehicle datasets," *Intl. J. of Robotics Research*, 2016. [Online]. Available: http://ijr.sagepub.com/content/early/2016/01/21/0278364915620033.abstract

[32] N. Demmel, C. Sommer, D. Cremers, and V. Usenko, "[Basalt#6] Square Root Bundle Adjustment for Large-Scale Reconstruction," pp. 11 718–11 727. [Online]. Available: http://arxiv.org/abs/2103.01843

[33] N. Demmel, D. Schubert, C. Sommer, D. Cremers, and V. Usenko, "[Basalt#5] Square Root Marginalization for Sliding-Window Bundle Adjustment." [Online]. Available: http://arxiv.org/abs/2109.02182

[34] D. Schubert, T. Goll, N. Demmel, V. Usenko, J. Stuckler, and D. Cremers, "The TUM VI Benchmark for Evaluating Visual-Inertial Odometry," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*. IEEE, Oct. 2018. [Online]. Available: http://dx.doi.org/10.1109/IROS.2018.8593419

[35] Collabora, "Monado SLAM Datasets," 2023. [Online]. Available: https://huggingface.co/datasets/collabora/monado-slam-datasets

[36] V. Holzwarth, J. Gisler, C. Hirt, and A. Kunz, "Comparing the Accuracy and Precision of SteamVR Tracking 2.0 and Oculus Quest 2 in a Room Scale Setup," in *2021 the 5th International Conference on Virtual and Augmented Reality Simulations*, ser. ICVARS 2021. Association for Computing Machinery, pp. 42–46.

[37] T. Pire, T. Fischer, G. Castro, P. De Cristóforis, J. Civera, and J. Jacobo Berlles, "S-PTAM: Stereo Parallel Tracking and Mapping," *Journal of Robotics and Autonomous Systems*, vol. 93, pp. 27–42, 2017.