Seguimiento Robusto de Características Visuales para Sistemas SLAM

Hernán Galardi
Universidad Nacional de Rosario
FCEIA-UNR
Rosario, Argentina
hgalardi@fceia.unr.edu.ar

Gastón Castro
Universidad de San Andrés
CONICET-UDESA
Buenos Aires, Argentina
gcastro@udesa.edu.ar

Taihú Pire
Laboratorio de Robótica
CIFASIS (CONICET-UNR)
Rosario, Argentina
pire@cifasis-conicet.gov.ar

***Resumen*—**
En este trabajo se presenta un método de seguimiento robusto de características visuales, el cual utiliza una ventana deslizante de frames, *brute-force matching*, un sistema de votación para gestionar candidatos en reidentificaciones, una estrategia de selección de candidatos que emplea búsqueda de consensos por similitud de descriptores, y una etapa de validación de correspondencias reidentificadas, para construir un *front-end* versátil y adaptable para sistemas SLAM. El método tiene la capacidad de seguir *características visuales* en condiciones desafiantes, como por ejemplo frente a breves oclusiones o movimientos bruscos. Se presentan resultados que validan el método logrando reidentificar más del 50 % de características visuales que KLT (*Kanade–Lucas–Tomasi feature tracker*) pierde en la secuencia de calibración inicial. La capacidad del *front-end* para reidentificar *características visuales* previamente detectados lo convierte en un módulo útil para ser incorporado en aplicaciones futuras que involucren sistemas SLAM.

***Keywords*—Características Visuales, Optical Flow, Localización, SLAM**

I. Introducción

Para que un robot pueda navegar autonomamente por un entorno desconocido este debe poder estimar su localización, al mismo tiempo que reconstruye un mapa del entorno. Este problema se conoce como localización y mapeo simultáneo (*SLAM, Simultaneous Localization and Mapping*), y es un problema fundamental del campo del a robótica móvil. Los sistemas que abordan el problema de SLAM tienen aplicaciones que van desde robots de limpieza, hasta automóviles autónomos y robots de exploración oceánicos. En los últimas tres décadas, la comunidad de robótica ha hecho importantes avances, lo que ha permitido aplicaciones a gran escala en el mundo real y la transición de esta tecnología a la industria [1].

La Figura 1 muestra la arquitectura general de un sistema SLAM. El *front-end* es la parte del sistema SLAM que se encarga de la adquisición y procesamiento de las mediciones crudas de los sensores (como cámaras, IMU, LiDAR y GNSS entre otros) que se utilizan para construir un mapa del entorno, y estimar la localización del robot. El *back-end* utiliza las mediciones procesadas por el *front-end* y es el encargado de estimar la pose del robot y mantener un mapa consistente. De esta manera, el *front-end* abstrae la información cruda proveniente de los sensores, generando representaciones intermedias que serán utilizadas por el *back-end* para resolver el problema de localización y mapeo. El

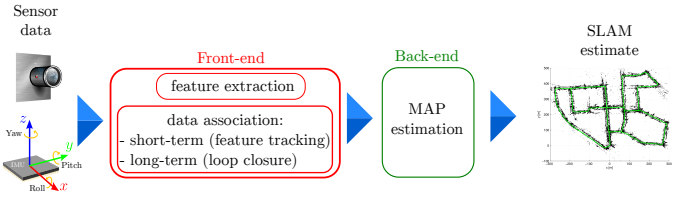


Figura 1. Arquitectura general de un sistema SLAM.

front-end es una parte fundamental de los sistemas SLAM, ya que imprecisiones en el procesamiento de las mediciones provenientes de los sensores condicionan el funcionamiento de todo el sistema SLAM [2], y su diseño está influenciado por las necesidades específicas de la aplicación que se desea abordar. Los algoritmos de procesamiento de mediciones y las estructuras de datos utilizadas durante el procesamiento en el *front-end*, dependerán de los sensores utilizados, el tipo de robot, el entorno de operación, y la aplicación a realizar.

En este contexto, en este trabajo se presenta un *front-end* versátil y funcional para sistemas SLAM visual, partiendo de la implementación del algoritmo KLT [3] para seguimiento de características visuales (*image features*) utilizada en VINS-Fusion [4], e incorporando una ventana deslizante junto a una etapa de reidentificación de características visuales. El *front-end* desarrollado tiene la capacidad de seguimiento de características visuales en casos de oclusiones en un breve período de tiempo.

Además, el método desarrollado también permite exportar los datos procesados en formato YAML, extendiendo las posibilidades de aplicación del método a sistemas SLAM *online* y *offline*. Estas cualidades convierten el presente módulo en una alternativa útil para su eventual uso posterior en sistemas SLAM.

El resto del trabajo está estructurado de la siguiente forma. En la sección II se analizan algunos de los avances más importantes del estado del arte en a lo que se refiere a mejoras en el *front-end* de sistemas SLAM. En la sección III, se presenta un método de seguimiento robusto de *características visuales* concebido para ser utilizada como *front-end* de sistemas SLAM. En la sección IV, se presentan experimentos que

validan cualitativamente y cuantitativamente la capacidad de seguimiento de características visuales del front-end propuesto ante oclusiones. Finalmente, en la sección V se presentan las conclusiones obtenidas y se exponen algunas líneas de trabajo futuras para profundizar los resultados obtenidos.

II. Trabajo Relacionado

En esta sección se presenta el estado del arte de sistemas de SLAM. En particular, se analizan las técnicas utilizadas por cada sistema para robustecer la estimación de localización.

En [4] se presenta el sistema VINS-Fusion, un sistema VI-SLAM que fusiona información de sensores visuales (cámaras) e inerciales (acelerómetros y giroscopios) para estimar con precisión la posición y la orientación de un robot, y construir un mapa del entorno en tiempo real. En particular, VINS-Fusion utiliza una técnica de optimización basada en grafo para fusionar la información proveniente de sensores locales (cámara, IMU, LiDAR) precisos para calcular la pose en regiones pequeñas, con la información proveniente de sensores globales (GNSS, magnetómetro) para lograr una estimación de pose precisa y libre de *drift* [4]. VINS-Fusion recupera y extiende el *front-end* de VINS-Mono, el cual utiliza *optical flow* basado en KLT (Kanade-Lucas-Tomasi) [3], [5] para realizar el seguimiento de características visuales (features) de cada nueva imagen recibida por la cámara. A la vez, realiza la detección de nuevos puntos salientes (*keypoints*) utilizando el método Shi-Tomasi [3] para mantener un mínimo de 100 a 300 features en cada imagen, y garantiza una distribución uniforme de los mismos, fijando una restricción de separación mínima entre features cercanos [6].

Basalt [7] es un método VIO (*Visual-Inertial Odometry*) del estado del arte que procesa las mediciones que provienen de la cámara y la IMU, pero tomando una cantidad fija de frames anteriores para estimar la pose. A diferencia de VINS-Fusion, Basalt emplea el detector FAST [8], [9] para la identificación de keypoints de las imágenes, y utiliza KLT para realizar el seguimiento de features empleando un enfoque composicional-inverso, con una norma de disimilitud de parche que es invariante a la escala de intensidad [7]. A diferencia de VINS-Fusion, Basalt utiliza *features ORB (Oriented FAST and Rotated BRIEF)* para la optimización de mapas.

ORB-SLAM3 [10] es un sistema VI-SLAM robusto que se caracteriza por su capacidad para poder seguir operando de forma extendida en el tiempo, teniendo información visual de baja calidad. Para lograr esto, ORB-SLAM3 tiene la capacidad de reutilizar en sus distintas etapas toda la información previamente recopilada. El *front-end* de ORB-SLAM3 utiliza las técnicas de detección y descripción de características visuales presentes en el sistema ORB-SLAM, el cual utiliza descriptores ORB. Los descriptores ORB son descriptores binarios basados en BRIEF [11], que se caracterizan por ser invariantes a rotación y resistentes al ruido. Por esta razón, el sistema mantiene una gran robustez aun al ser utilizando en drones, o aplicaciones que involucran AR/RV [12].

El sistema VI-SLAM OKVIS2 [13] aborda los desafíos que involucra ejecuciones extensas y con frecuentes ciclos en la trayectoria. El *front-end* trabaja con múltiples frames y mediciones inerciales para la inicialización de nuevos estados y utiliza descriptores BRISK [14] para la descripción de características visuales y detección de ciclos en la trayectoria. El estimador en tiempo real optimiza, luego, el grafo de factores subyacente y se encarga del mantenimiento del grafo de poses. De forma asíncrona, se lleva a cabo una optimización completa del grafo al detectar ciclos, cuyo resultado será incorporado cuando sea posible.

Un método VIO del estado del arte relacionado con los sistemas previamente analizados es *Semi-Direct Visual Odometry* (SVO) [15]. SVO utiliza una técnica de seguimiento de features *semi-directa*, que permite realizar el seguimiento de features empleando la intensidad de los píxeles de la imagen frame a frame, como suelen hacer los métodos directos de estimación de movimiento visual. De esta manera se evita la necesidad de realizar procesos costosos de extracción de features y matching para la estimación de movimiento, y gracias a esto, SVO alcanza velocidades de ejecución que van de 55 frames por segundo en computadoras embebidas, hasta 300 frames por segundo, en una laptop estándar. SVO se destaca al ser utilizado en vehículos aéreos pequeños (MAV, del inglés *Micro Aerial Vehicle*) en entornos sin señal de GNSS [15].

El método propuesto en este trabajo toma como punto de partida el *front-end* del sistema SLAM VINS-Fusion [4]. Este sistema utiliza técnicas de *optical flow* para realizar el seguimiento de features. Sin embargo, dado que la estrategia utilizada en el módulo de seguimiento de features se concentra en el seguimiento de características entre el frame previo y el frame actual, sin contemplar extracción de descriptores u otra estrategia que permita identificar y reconocer features, el mismo no permite realizar un seguimiento de features a mediano plazo, y resulta susceptible a oclusiones y pérdida de features. Por tanto, en este trabajo, se presenta una mejora a este comportamiento creando un nuevo *front-end* que realiza el seguimiento de features a través de la incorporación de una ventana deslizante, una etapa de extracción de descriptores, un sistema de votación, y una estrategia de selección de candidatos que emplea la búsqueda de consensos por similitud de descriptores, junto a una etapa de validación de correspondencias reidentificadas.

III. Método propuesto

En esta sección se describe el método de seguimiento robusto de features concebido para ser utilizado como *front-end* para sistemas SLAM. El mismo, consta de una ventana deslizante de frames sucesivos (de tamaño configurable), *brute-force matching*, un sistema de votación para establecer candidatos a reidentificar features en la ventana, y una estrategia de selección de candidatos que emplea la búsqueda de consensos por similitud de descriptores, junto a una etapa de validación de correspondencias reidentificadas. En particular, se busca que la capacidad de seguimiento de features sea robusta frente a oclusiones, cambios de iluminación o movimientos bruscos. La capacidad del método para reidentificar features

previamente detectados lo convierte en un módulo *front-end* útil para su eventual incorporación a un sistemas SLAM completo.

Si bien KLT es un algoritmo del estado del arte muy utilizado para el seguimiento de features, calcular el *optical flow* de un feature y rastrearlo a lo largo de una secuencia requiere que el mismo se encuentre visible en todos los frames de la secuencia. Si, por ejemplo, en un frame intermedio, dicho feature no es visible, pero en los siguientes frames es visible nuevamente, este será considerado como un nuevo feature. Este comportamiento en un *front-end* de un SLAM Visual, genera una duplicación de map-points y deterioro en la estimación de la localización. Para permitir que el *front-end* pueda mantener el seguimiento de features incluso cuando estos dejen de ser visibles durante un período de tiempo acotado (un número consecutivo de frames acotado), se utiliza una ventana deslizante junto a una serie de estrategias complementarias. El tamaño de la ventana determina el tiempo de oclusión que podrá soporta el método.

Para poder realizar el seguimiento de features aun si los mismos se pierden de vista durante un breve período de tiempo, el método de seguimiento robusto de features debe poder reidentificar aquellos features extraídos en frames previos en la ventana. En la Figura 2 se presenta un esquema del método de seguimiento robusto de features con sus distintas etapas. Primeramente, el método de seguimiento robusto de features

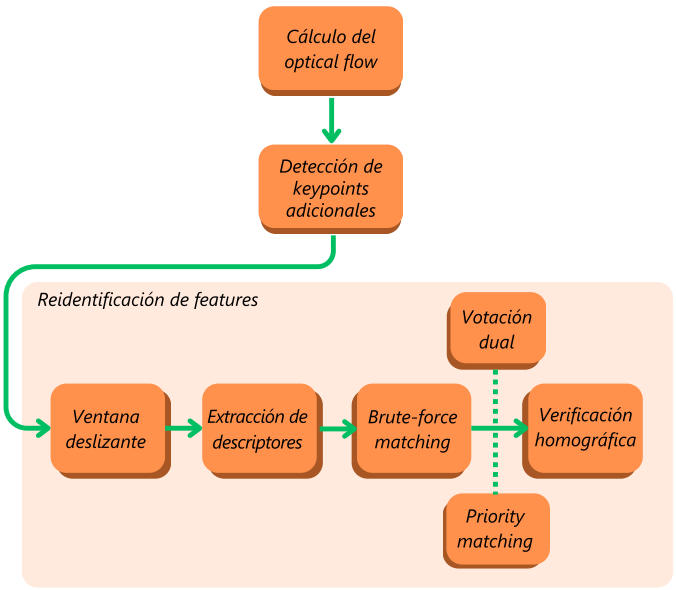


Figura 2. Esquema del método de seguimiento robusto de features.

calcula el *optical flow* entre los frames previo y actual, y detecta *keypoints* adicionales en el frame actual hasta alcanzar un valor preestablecido. Luego, inicia un proceso de análisis sobre los nuevos *keypoints* detectados, en el cual el *front-end* evalúa si un feature detectado por *primera vez* en el frame actual puede corresponder a otro visto previamente. Este procedimiento se realiza de la siguiente manera:

1. Inicialmente se extraen los descriptores de features del frame actual, utilizando el algoritmo BRIEF [11], y se recuperan los datos de features observados en frames previos de la ventana.
2. Posteriormente se utiliza *brute-force matching* para comparar el descriptor de cada nuevo feature que pudiera constituir una reidentificación (*feature en análisis*), con los descriptores de los features previamente vistos en frames de la ventana. Al momento de realizar esta búsqueda de similitudes, no se tienen en cuenta los features que ya se encuentran bajo seguimiento, es decir, aquellos para los que KLT pudo hallar el *optical flow* entre el frame previo y el frame actual.
3. El *brute-force matching* calcula la distancia de *Hamming* entre el descriptor del feature en análisis y los descriptores de features de la ventana, y selecciona el candidato de mayor similitud (menor distancia de *Hamming)* para cada frame de la ventana. Previo a esto se verifica que cada feature candidato también supere un umbral de similitud mínimo prefijado.
4. Se realiza una votación entre los candidatos propuestos en cada frame de la ventana, determinando que el feature en análisis corresponde a una posible reidentificación del feature que se repite con mayor frecuencia entre los candidatos propuestos en cada frame de la ventana deslizante.
5. Una vez seleccionado el mejor candidato de la ventana, se asigna temporalmente su *ID* (número entero no negativo) para reidentificar al feature en análisis. Llamémos a este *ID*, x.
6. A continuación, se calcula la homografía existente entre los frames de la ventana que presentan apariciones de x, y el frame actual. Esto se realiza con la función de OpenCV `findHomography()`. Esta función recibe cada par de frames con sus correspondientes features, y devuelve la matriz de transformación de perspectiva H entre los planos de origen y de destino. Además, `findHomography()` permite utilizar RANSAC para identificar si la aparición de un feature entre el frame inicial y el frame final, se ajusta al modelo que describe la mayoría de features entre ambos. Notar que esta característica, en conjunto con la posibilidad de fijar un parámetro de error de reproyección máximo, permiten identificar los matches *outliers* presentes entre ambos frames.
7. Una vez calculada la matriz H, se evalúa específicamente si RANSAC consideró *inlier* o *outlier* el par de puntos correspondiente a las apariciones del feature x entre el frame inicial, y el frame final. En caso de que RANSAC catalogue el feature correspondiente a tales apariciones como *inlier*, se confirma la reidentificación propuesta, caso contrario se descarta. Una vez confirmada o descartada la reidentificación, se continúa la evaluación de homografías entre los restantes frames que presenten apariciones del feature x, y el frame actual.
8. Si en todas las homografías, el feature x es catalogado

como *outlier*, se rechaza la reidentificación propuesta, se quita el *ID* previamente asignado al feature en análisis, se considera que la observación corresponde a un feature realmente nuevo, y se le asigna un nuevo *ID* global.

9. En el marco de este proceso, se incorporan las estrategias de *votación dual* y *priority matching* para mejorar la capacidad de reconocimiento de features del sistema.

Un resumen de las etapas más destacadas del proceso de reidentificación de features puede observarse en la Figura 3.

III-A. Sistema de votación

Es posible que al analizar un feature perdido por KLT en búsqueda de reidentificaciones, su descriptor haya sufrido modificaciones, y el *front-end* no logre hallar apariciones previas. En dicho caso, el feature se cataloga como nuevo, y esto ocasiona que en la misma ventana existan dos features con distinto ID que corresponden al mismo objeto en el mundo real (*landmark*). Para mitigar esto, se incorpora al *front-end* un sistema de votación. El sistema de votación considera los candidatos seleccionados en cada frame por el *brute-force matching* como posibles reidentificaciones del feature en análisis. Aquel candidato cuyo *ID* posea un mayor número de apariciones entre los candidatos propuestos en cada frame de la ventana, se selecciona como el *mejor* candidato a reidentificar al feature en análisis, y se le asigna al mismo su *ID*.

Si se diera el caso de un empate, es decir que coincidiera el número de frames donde es propuesto un candidato, con el número de frames donde es propuesto otro, el sistema de votación implementa una estrategia de *votación dual*. Para cada feature nuevo, se contemplan como candidatos posibles de reidentificación, los dos features más votados entre los propuestos por cada frame de la ventana (obtenidos por *brute-force matching*). Posteriormente, a la hora de realizar la verificación con RANSAC de los candidatos reidentificados, si el feature más votado en la ventana es considerado *outlier*, es decir, no coincide con la homografía de puntos de los demás features, se descarta, se asigna como *ID* de reidentificación al feature en análisis el *ID* del segundo candidato más votado, y se vuelve a validar esta asignación con RANSAC. Si esta reidentificación es considerada *inlier*, es decir, si coincide con la homografía de puntos, el feature se considera correctamente reidentificado. Caso contrario termina el proceso, y el feature es considerado como un feature nuevo, visto por primera vez.

En la Figura 4 ilustra un ejemplo de cómo el sistema de votación seleciona el feature candidato más votado en la ventana.

Observar que el sistema de votación dual cubre el caso en que un feature pueda descartarse erróneamente por ser parte de un *empate* entre dos candidatos propuestos por un número exactamente igual de frames.

A diferencia de un sistema de votación simple, el sistema de votación dual, presenta un aumento del costo computacional debido a las llamadas adicionales a RANSAC que conlleva repetir la verificación homográfica con el segundo candidato más votado, en aquellos casos donde candidato principal sea catalogado como *outlier* al realizar la primera verificación.

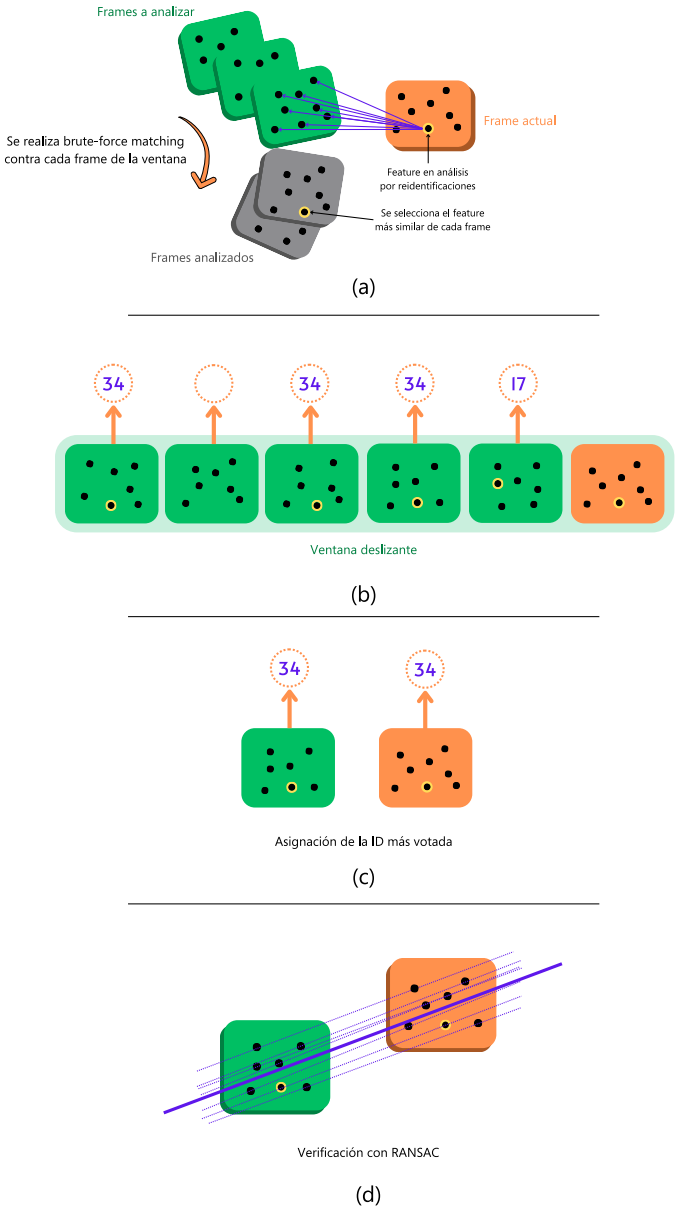


Figura 3. Ejemplo del proceso de reidentificación de features para una ventana de 6 frames. Primero, para cada feature nuevo detectado en el frame actual se realiza *brute-force matching* contra los features de frames previos de la ventana (a). Luego, se selecciona el mejor candidato de cada frame para reidentificar al feature en análisis (b). El *ID* del feature que posea más apariciones entre los candidatos propuestos, se asigna temporalmente al feature en análisis (c). Finalmente, se utiliza RANSAC para verificar la asignación realizada (d). Si el feature es catalogado como *inlier*, se confirma la reidentificación con el feature más votado. Caso contrario, se evalúa la homografía con los frames restantes que contengan apariciones del feature más votado. Agotados los mismos, termina infructuosamente el proceso de reidentificación, y el feature es considerado verdaderamente nuevo.

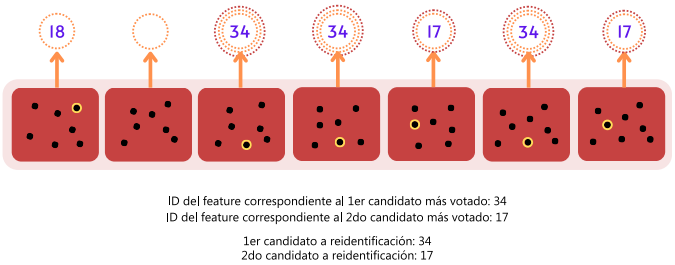


Figura 4. Ejemplo: votación entre los candidatos seleccionados por el *brute-force matching* en cada frame de la ventana. Se seleccionan el 1er y 2do candidato más votados como features posibles a reidentificar el feature en análisis. El identificador de feature que aparece con mayor frecuencia entre los propuestos, se asigna primero como *ID* al feature en análisis. Si el mismo no supera la verificación homográfica con RANSAC, se repite el procedimiento con el 2do candidato más votado.

III-B. Priotity matching

Este enfoque contempla que debido a las condiciones adversas del entorno, es posible que el *brute-force matching* no pueda seleccionar el candidato correcto al comparar el descriptor del feature en análisis, con los descriptores de features en frames de la ventana. Para responder a esto, se modificó el funcionamiento del *brute-force matching* a fin de proponer, por cada frame, no solo un único candidato cuyo descriptor sea el más parecido al descriptor del feature en análisis, sino los dos candidatos más similares. De esta manera se prioriza la posibilidad de que el *brute-force matching* encuentre al candidato correcto entre los dos features del frame cuyos descriptores sean más cercanos al feature en análisis.

En la Figura 5 puede observarse un ejemplo de cómo el *priority matching* modifica el mecanismo de votación en la ventana.

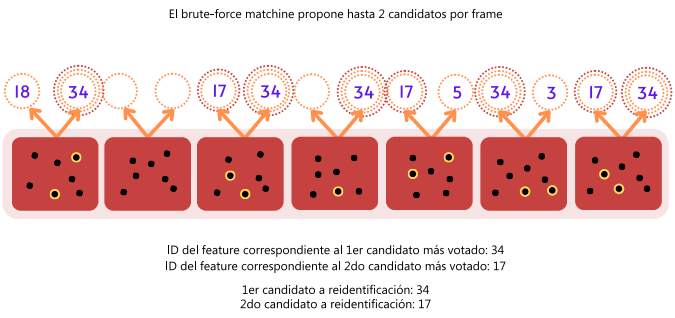
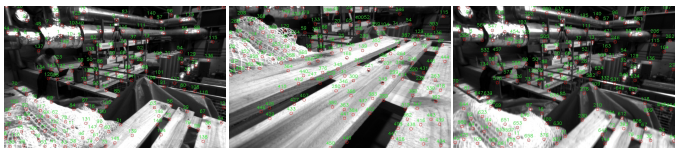


Figura 5. Ejemplo: votación entre los candidatos de la ventana, utilizando *priority matching*. El *brute-force matching* ahora propone hasta 2 candidatos por cada frame, siempre que superen el umbral mínimo de similitud. Posteriormente, se realiza la votación entre todos los candidatos propuestos, resultando elegidos aquellos dos que presenten más apariciones, debido a la votación dual.

IV. Experimentos

En esta sección se presentan los experimentos llevados adelante para validar el desempeño del método propuesto, sobre el *dataset Machine Hall*, de *EuRoC MAV Dataset* [16], utilizando imágenes desdistorsionadas. En particular, el estudio se centra en analizar la robustez y la performance del sistema para oclusiones presentes en la secuencia inicial de movimientos de un drone *AscTec Firefly*, un UAV (*Unmanned Aerial Vehicle*), dotado con una unidad de sensores (cámara-IMU) visual-inercial.

En lo siguiente, se propone utilizar la secuencia de frames 1–29 para realizar una evaluación cualitativa y cuantitativa del método de tracking propuesto. La misma forma parte de la secuencia *Machine Hall 01* del dataset, y presenta una oclusión producto del ascenso y descenso vertical del drone, motivo por el cual se vuelve de especial interés para este trabajo. Analizando la evolución de esta sucesión de frames, se identificó que la mayor oclusión de features se produce al llegar al frame nº17, perdiendo de vista la gran mayoría de puntos seguidos en la mitad superior de la imagen, que venían siendo seguidos desde el frame inicial. En la Figura 6 se presentan tres capturas que permiten apreciar el efecto de la oclusión sobre la visión de la cámara. El vídeo[1] muestra el método de seguimiento robusto de características visuales aquí propuesto sobre la secuencia analizada.



(a) frame inicial (nº1) (b) frame con oclusión (nº17) (c) frame final (nº29)

Figura 6. Detección de features en el frame final (nº29). En rojo se identifica la posición de cada feature seguido por el sistema en el frame actual. En verde se muestra el ID global asociado a cada feature señalado.

En la Figura 7 se visualizan los matches de features detectados entre el frame inicial (nº1) y el frame final (nº29) de la cámara izquierda, diferenciando aquellas correspondencias que fueron seguidas por KLT durante toda la secuencia en análisis, dibujadas en color magenta, de aquellas correspondencias identificadas a partir de features que fueron perdidos por KLT durante la oclusión, y luego el sistema pudo reidentificar a partir del análisis de los frames previos de la ventana, que se dibujan en color verde. En la Tabla I se muestra la cantidad de features trackeados por KLT, número y porcentaje de puntos reindetificados y tiempo de procesamiendo por frame. Teniendo en cuenta que el sistema se encuentra configurado para detectar 150 features por frame, para el caso que contempla una ventana deslizante de 30 frames, el 51.33 % (77) de los features presentes en el primer frame, fueron nuevamente detectados en el último frame de la secuencia, de los cuales el 61.03 % (47) habían sido perdidos en medio de la secuencia por la oclusión ocurrida. También puede destacarse la casi nula presencia de matches *outliers* (1, lo que representa el 2.1 % del total de features reidentificados a partir de apariciones previas en la ventana), producto de la etapa de filtrado con RANSAC que forma parte del sistema. Notar que el indicador del porcentaje de reidentificación de features en la ventana se calcula respecto al total de features del frame, y es posible

[1] https://youtu.be/1-PDPU15bYU

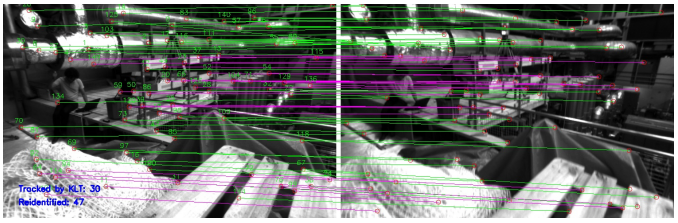


Figura 7. Matches entre frames nº1 (izquierda) y nº29 (derecha) - Longitud de ventana: 30 frames. Las líneas rosas unen apariciones de un mismo feature entre los frames inicial y final de la secuencia, donde el mismo ha sido seguido por KLT durante todo el intervalo. Las líneas verdes unen apariciones de un mismo feature entre los frames inicial y final de la secuencia, donde el feature ha sido perdido en algún momento del intervalo, y posteriormente ha sido reidentificado por el sistema a partir de una aparición previa en la ventana. En la leyenda azul se listan el número de features que KLT ha seguido durante la secuencia, y el número de features que el sistema ha reidentificado a partir de apariciones previas.

que luego de la oclusión, se hayan detectado nuevos features que no fueron previamente vistos. Por tanto, no podríamos considerar que todos los features no reidentificados fueron realmente vistos y luego no reconocidos.

Tabla I
DESEMPEÑO DEL SISTEMA.

Frame	KLT	# Reident.	% Reident.	Tiempo de proc. [ms]
21	111	20	51.28	21.794
22	118	18	56.25	23.397
23	121	15	51.72	25.440
24	125	12	48.00	29.684
25	127	17	73.91	30.631
26	129	12	57.14	31.426
27	134	9	56.25	29.782
28	126	12	50.00	33.643
29	129	7	33.33	29.805
30	135	5	33.33	29.929
Promedio	**125.5**	**12.7**	**51.12**	**28.5531**

V. CONCLUSIONES Y TRABAJO FUTURO

En este trabajo se presenta un método de seguimiento features robusto a oclusiones temporales. El mismo está basado en el uso de una ventana deslizante junto con técnicas de *brute-force matching*, un sistema de votación para gestionar candidatos en reidentificaciones, una estrategia de selección de candidatos que emplea búsqueda de consensos por similitud de descriptores, y una etapa de validación de correspondencias reidentificadas, para construir un *front-end* versátil y adaptable para sistemas SLAM. Los experimentos realizados muestran que el método es capaz de reidentificar un 51.12 % de los features perdidos por KLT durante una oclusión. La posibilidad de poder reidentificar features perdidos permite mejorar la precisión y robustez de los sistemas de SLAM que utilizan información visual. Como trabajo futuro se planea integrar el método de seguimiento robusto de features como *front-end* del sistema de SLAM VINS-Fusion [6].

AGRADECIMIENTOS

Este trabajo fue apoyado por CONICET (PIBAA Nº 0042), AGENCIA I+D+i (PICT 2021-570) y Universidad Nacional de Rosario (PCCT-UNR 80020220600072UR).

REFERENCIAS

[1] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. Leonard, "Simultaneous Localization And Mapping: Present, Future, and the Robust-Perception Age," *IEEE Trans. Robotics*, vol. 32, 06 2016.

[2] M. Khan, D. Hussain, K. Naveed, U. Khan, I. Mundial, and A. Aqeel, "Investigation of Widely Used SLAM Sensors Using Analytical Hierarchy Process," *Journal of Sensors*, vol. 2022, pp. 1–15, 01 2022.

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

[4] T. Qin, S. Cao, J. Pan, and S. Shen, "A General Optimization-based Framework for Global Pose Estimation with Multiple Sensors," in *Computing Research Repository*, 01 2019.

[5] B. Lucas and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision (IJCAI)," in *DARPA Image Understanding Workshop (IUW)*, vol. 81, 04 1981.

[6] 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.

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

[8] E. Rosten and T. Drummond, "Machine Learning for High-Speed Corner Detection," in *ECCV*, vol. 3951, 07 2006.

[9] E. Rosten, R. Porter, and T. Drummond, "Faster and Better: A Machine Learning Approach to Corner Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 1, pp. 105–119, 2010.

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

[11] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF: Binary Robust Independent Elementary Features," in *Eur. Conf. on Computer Vision (ECCV)*. Springer Berlin Heidelberg, 2010, pp. 778–792.

[12] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *2011 International Conference on Computer Vision*, 2011, pp. 2564–2571.

[13] S. Leutenegger, "OKVIS2: Realtime Scalable Visual-Inertial SLAM with Loop Closure," in *arXiv preprint arXiv:2202.09199*, 02 2022.

[14] S. Leutenegger, M. Chli, and R. Y. Siegwart, "BRISK: Binary Robust invariant scalable keypoints," in *2011 International Conference on Computer Vision*, 2011, pp. 2548–2555.

[15] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2014, pp. 15–22.

[16] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. Achtelik, and R. Siegwart, "The EuRoC micro aerial vehicle datasets," *The International Journal of Robotics Research*, vol. 35, 01 2016.