UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

CONSOLIDACIÓN ESPACIAL PARA CAPTURAS DE SISTEMAS ROBÓTICOS

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERA CIVIL EN COMPUTACIÓN

MARÍA ANTONIA HERNÁNDEZ RAMÍREZ

PROFESOR GUÍA:
EDUARDO GRAELLS GARRIDO

MIEMBROS DE LA COMISIÓN:
LUCIANO RADRIGAN FIGUEROA
FEDERICO OLMEDO BERÓN

SANTIAGO DE CHILE
2023

Resumen

El presente trabajo aborda el problema de la consolidación de objetos en el contexto de Zippedi, una empresa de robótica especializada en la digitalización de tiendas de *retail*. La consolidación de objetos es el proceso de identificar distintas instancias de un mismo objeto capturado en una serie de fotos. Esto es fundamental para brindar información precisa y confiable a los clientes. Al reunir y agrupar correctamente las detecciones de los objetos, se garantiza que los datos proporcionados reflejen con precisión la cantidad y la ubicación de los objetos en un entorno de tienda.

El objetivo principal de este trabajo fue desarrollar un algoritmo versátil de consolidación de objetos, capaz de manejar diferentes cantidades de cámaras, mejorar el rendimiento en comparación con la solución actual de la empresa y adaptarse a cualquier tipo de detección. Para lograr este objetivo, se realizó un estudio de técnicas de seguimiento de objetos y descripción de imágenes. Diseñando un algoritmo inicial para la consolidación en una sola cámara y luego lo adaptándolo para que funcionara con múltiples cámaras. Además, se realizaron pruebas para evaluar el desempeño y la eficacia del nuevo algoritmo. Finalmente, se disponibilizó la solución en un microservicio y se documentó de manera detallada para su futuro uso por parte de otros miembros de la empresa.

Se ha observado una mejora en los resultados de la consolidación de objetos para uno de los tipos de cámaras utilizadas en el robot. Sin embargo, para lograrlo fue necesario aumentar el tiempo de procesamiento y los requisitos de recursos del sistema. Gracias a esta solución implementada, ahora es posible consolidar objetos que carecen de lecturas, superando así las limitaciones de la solución anterior de la empresa. Esto amplía la aplicabilidad del algoritmo a diferentes tipos de objetos, abriendo nuevas posibilidades en su utilización.

Se constató que la utilización de información visual proveniente de las imágenes capturadas permite obtener una consolidación más precisa y completa. Asimismo, se encontró que el enfoque basado en *tracks* facilita la consolidación de capturas provenientes de cualquier cantidad de cámaras.

En cuanto a las implicaciones de este proyecto, se logró ofrecer mejores resultados a los clientes en cuanto a la cantidad total de objetos en el pasillo. Esto se traduce en una mejora de la eficiencia para los trabajadores de la tienda, quienes utilizan esta información para llevar a cabo tareas como reposición de productos, gestión del inventario, entre otras.

A todos los que les dije que no estudiaría ingeniería..

Mentí.

Aquí estoy.

Agradecimientos

Agradezco con mucho cariño a mi madre, por hacerme quien soy. Desde donde quiera que estés, espero te sientas orgullosa de mis logros.

A mi hermano, por siempre escucharme y ayudarme tanto en los estudios como en la vida en general.

A todos los amigos que hice durante mi vida universitaria, sin los cuales no habría logrado pasar tantos ramos, o quizás si, pero como menos risas.

Quiero agradecer a Zippedi por darme la oportunidad de hacer mi memoria con ellos, especialmente al Road y a Martin. Gracias por el apoyo durante el proyecto y la confianza.

Tabla de Contenido

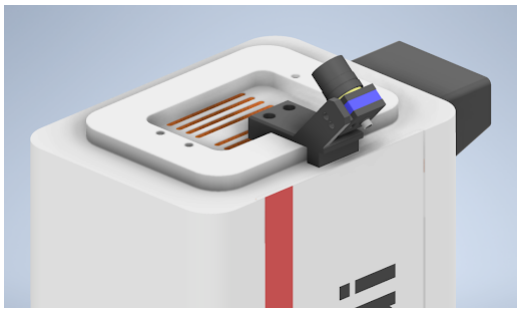
Capítulo 1

Introducción

1.1. Contexto

Zippedi es una empresa chilena de *robot as a service*. Poseen robots (ver Figura 1.1) que permiten digitalizar las tiendas de *retail* y con ello ayudarlos a presentar un mejor servicio para sus clientes, evitando falta de *stock* de productos, precios mal etiquetados y otros problemas relacionados a la disposición física de productos. Esto lo hacen con robots que navegan por las tiendas capturando imágenes y luego con distintos procesos se obtiene la información que el cliente necesita.

(a) Robot.



(b) Cámara overhead.

Figura 1.1: (a) Robot utilizado por la empresa, mide 1,70 metros y está equipado con sensores de proximidad que le permiten navegar de manera autónoma en el entorno de la tienda. Además, cuenta con tres cámaras laterales que capturan imágenes de los objetos en las estanterías. (b) En algunas tiendas, debido a la altura de las estanterías, el robot también está equipado con una cámara adicional en la parte superior, denominada overhead. Esta cámara tiene el propósito de capturar imágenes y brindar información sobre las cajas de reposición ubicadas en la parte superior de las estanterías.

En el proceso de digitalización de las tiendas de *retail*, los robots recorren los pasillos capturando imágenes de las estanterías para identificar la presencia o ausencia de productos, así como sus flejes asociados (ver Glosario). Estas imágenes son luego procesadas mediante

un sistema de análisis desarrollado por la empresa, el cual es capaz de detectar objetos y clasificar la información relevante.

Las capturas de imágenes se realizan en intervalos estratégicos, permitiendo que un mismo objeto aparezca en varias tomas, como se muestra en las Figuras 1.2 y 1.3. Este enfoque brinda múltiples oportunidades para detectar y hacer una lectura correcta de cada objeto. Sin embargo, es esencial evitar los falsos positivos al informar al cliente la cantidad real de errores o faltas de productos en los pasillos, en lugar de simplemente reportar las detecciones realizadas por Zippedi. Por lo tanto, surge la necesidad de desarrollar un algoritmo que, al consolidar las detecciones realizadas en diferentes capturas, sea capaz de discernir si se trata del mismo objeto en distintos momentos, evitando así falsas duplicaciones.

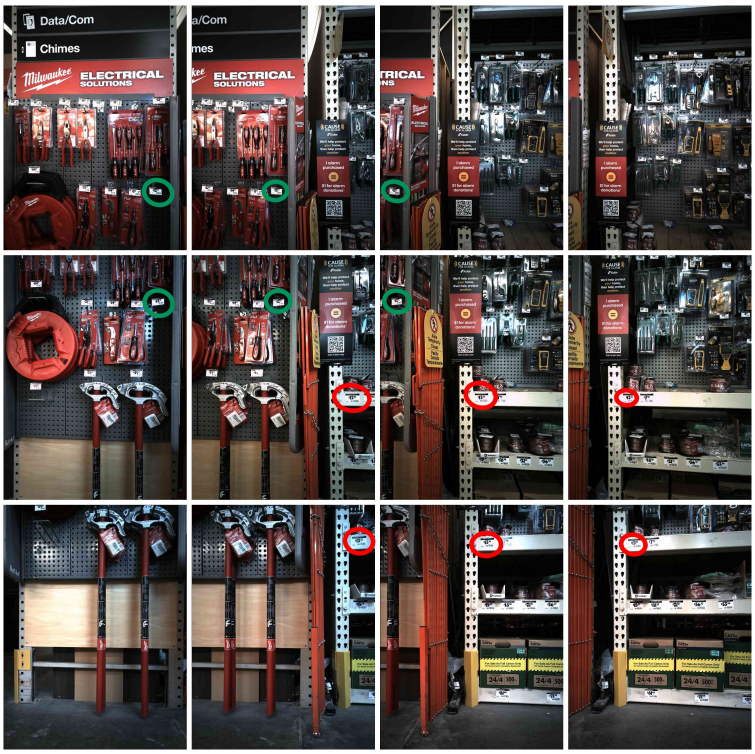


Figura 1.2: Tres corridas de fotos capturadas por el robot, una por cada cámara lateral. Cada fila de imágenes representa una secuencia consecutiva de capturas realizadas por una cámara específica, y las columnas corresponden a los momentos en los que las cámaras 1, 2 y 3 capturaron las fotos desde sus respectivas alturas. En verde se destaca un objeto que se repite en múltiples capturas. Lo mismo en rojo.

1.2. Motivación

La consolidación de objetos detectados en imágenes desempeña un papel esencial en la entrega precisa y confiable de información a los clientes de Zippedi. Es fundamental poder proporcionar información exacta sobre la ubicación, disponibilidad y cantidad de productos en los pasillos, lo que facilita la toma de decisiones informadas en la gestión del inventario y la planificación de reposición.

Figura 1.3: Cuatro imágenes consecutivas tomadas por el robot con la cámara overhead. En rojo se ejemplifica la repetición de objetos entre capturas.

A pesar de contar con una solución actual para la consolidación de objetos, se han identificado limitaciones que requieren mejoras. La solución actual se basa en la lectura de objetos en imágenes, lo que implica que aquellos objetos sin lectura o con lecturas incorrectas no pueden ser adecuadamente consolidados. Para abordar esta situación, es necesario mejorar el proceso de consolidación, permitiendo la inclusión de todo tipo de objetos y ampliando así su capacidad.

En la actualidad, el código de la solución se encuentra separado en dos repositorios distintos: uno para la consolidación de cámaras laterales y otro para la consolidación de la cámara overhead. Esta división puede generar inconsistencias y dificultades al realizar modificaciones en el código, lo cual no es óptimo para lograr una integración y unificación efectiva del proceso de consolidación.

Conforme la empresa experimenta un crecimiento, y sus clientes esperan una mejor calidad de datos, nace la necesidad de unificar el código y mejorar el proceso de consolidación de objetos detectados en las imágenes. Esta mejora permite una mayor escalabilidad del código, evitando conflictos entre los algoritmos de consolidación de cámaras laterales y overhead.

1.3. Solución Actual

La solución actual implementada aborda el desafío de consolidar objetos detectados que poseen lectura. Esta solución se compone de una serie de procesos que se llevan a cabo antes de la consolidación propiamente dicha como se muestra en la Figura 1.4.

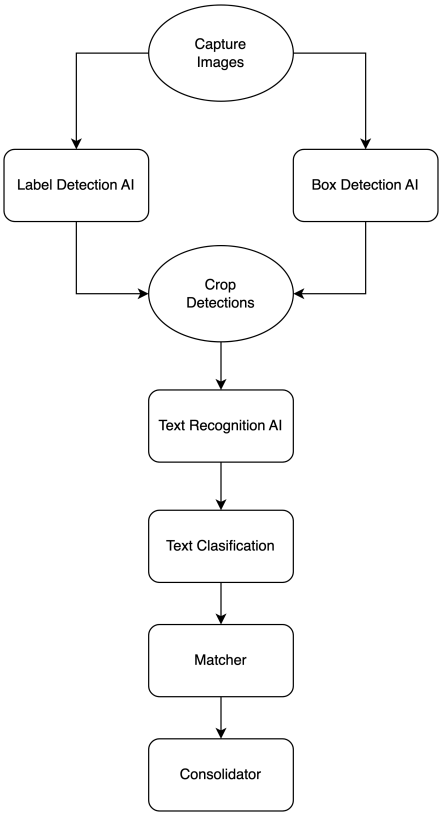


Figura 1.4: Diagrama del *pipeline* actual, desde la captura de las imágenes hasta la consoliadacion de objetos. Los óvalos representan las etapas realizadas por el robot. Los rectángulos representan las operaciones realizadas en la nube.

1.3.1. Pasos previos a la consolidación

Captura de las imágenes

En el proceso de captura de imágenes en un pasillo, el robot opera en dos escenarios distintos (ver Figura 1.5). Estos escenarios son los siguientes:

- **Captura de cámaras laterales:** En este escenario, las cámaras del robot capturan imágenes de las estanterías generales de la tienda. En estas capturas, se observan flejes que contienen información como el precio y el SKU (ver Glosario) del producto.
- **Captura de cámara overhead**: Por otro lado, la cámara overhead del robot se encarga de capturar imágenes de las cajas colocadas sobre las góndolas (ver Glosario). Estas cajas, que se utilizan para la reposición de productos, y pueden contener múltiples tipos de productos en su interior. Por lo tanto, los flejes en estas cajas muestran los SKU de todos los productos que contienen, pero no incluyen los precios asociados. Es importante destacar que estas capturas de imágenes overhead se someten a una transformación para corregir la inclinación y simular una perspectiva frontal. Esto se debe a que el robot se encuentra a una altura inferior a la de las góndolas, lo que puede generar cierta inclinación en las imágenes tomadas.

(a) Foto cámara lateral. (b) Foto cámara overhead.

Figura 1.5: Ejemplos de capturas realizadas por los dos tipos de cámaras utilizadas por el robot. (a) En la cámara lateral se capturan imágenes de productos fácilmente accesibles para los clientes en las estanterías de las tiendas. (b) La cámara overhead captura fotos de cajas que contienen productos destinados a la reposición, ubicados en alturas a las que solo el personal de la tienda puede acceder.

Tras este paso, las imágenes tomadas por el robot son enviadas y almacenadas en Google Cloud Storage. Esta plataforma brinda la infraestructura necesaria para llevar a cabo los procesamientos posteriores sobre las imágenes.

Deteccion de objetos de interés

Después de la captura de imágenes, se procede a detectar los elementos de interés presentes en cada una de ellas. En el caso de las cámaras laterales, se lleva a cabo la detección de flejes que contienen información como el nombre, SKU y precio de los productos. En cambio, en las capturas de la cámara overhead se detectan flejes con uno o más SKUs, así como cajas que pueden o no tener los SKUs escritos a mano. Se puede observar ejemplos de detecciones en la Figura 1.6.

Las detecciones se representan como *bounding boxes* (ver Glosario) y se guardan sus coordenadas respecto a la imagen en el formato $[x_1, y_1, x_2, y_2]$. Estas coordenadas definen las esquinas opuestas de la caja delimitadora que encierra el objeto de interés en la imagen.

(a) Detecciones capturas cámara lateral.

(b) Detecciones capturas cámara overhead.

Figura 1.6: Ejemplos de detecciones realizadas en dos tipos de imágenes. (a) Captura hecha por la cámara lateral, se resaltan en rojo los flejes detectados. (b) Captura hecha por la cámara overhead, se muestra la detección de flejes en rojo y cajas en verde.

Recorte de detecciones

Una vez generadas las detecciones, se envían los datos de las *bounding boxes* al robot, el cual lleva a cabo dos procesos adicionales para mejorar la calidad de los resultados:

- **Generación de recortes de las detecciones:** El robot utiliza las *bounding boxes* para recortar las áreas de interés de las imágenes originales, generando recortes con una mayor resolución (ver Figura 1.7). Estos recortes también se suben a Google Cloud Storage, lo que permite disponer de imágenes de mayor calidad para los siguientes procesos.
- **Proyección de la posición real:** Utilizando la información de las detecciones y las coordenadas relativas del robot, se realiza una proyección de la posición real de cada objeto detectado con respecto al inicio del pasillo. Para ver el formato de los datos mencionados, consultar la Tabla 1.1.

Una vez completados estos procesos de detección y generación de recortes de las detecciones, se da paso a la etapa de lectura y clasificación de la información capturada.

(a) Recorte de la captura inicial. (b) Recorte entregado por el robot.

Figura 1.7: (a) Recorte obtenido utilizando la captura inicial. (b) Recorte entregado por el robot. El recorte proporcionado por el robot presenta una mayor calidad, lo cual es de gran ayuda para el proceso de lectura de la información.

capture_id	detection_id	x_1	x_2	y_1	y_2	location_x	location_y	location_z
814131	814131_001	56.934	92.165	421.615	446.943	1.077	1.306	0.574
814131	814131_000	660.643	714.154	-0.401	28.644	1.889	1.287	1.123
814131	814131_003	27.316	63.286	287.699	313.270	1.039	1.290	0.752

Tabla 1.1: Tabla de ejemplo de la información entregada y proyectada de posición para las detecciones. Los valores de x_1, x_2, y_1 e y_2 representan las coordenadas del *bounding box* de cada detección, mientras que los valores de location_x, location_y y location_z indican las coordenadas proyectadas en metros para el centroide de cada objeto detectado. Esta proyección de posición proporciona datos sobre la ubicación real de los objetos en el pasillo.

Lectura y clasificación de las detecciones

Una vez detectados los elementos de interés en las capturas, se procede a la lectura y clasificación de la información. Para los flejes, se realiza una lectura del contenido, identificando SKU, nombre y precio, y clasificándolos en consecuencia. Por otro lado, en el caso de las cajas, se lee si es que hay contenido escrito a mano y se identifican los SKU correspondientes. La Figura 1.8 muestra un ejemplo de la lectura de un fleje de góndola.

(a)

prediction_type	value	confidence
text	PAPEL IMPRESO 1D	0.620753
item	1776050	0.999969
price	8890.0	1.000000

(b)

Figura 1.8: Ejemplo de lectura y clasificación del texto para fleje de góndola.

Match con la información de la maestra de la tienda

Luego de obtenidas las lecturas de las detecciones, se realiza un proceso de *matching* con la base de datos del cliente. Este proceso se utiliza como método de verificación de las lecturas realizadas. Mediante el *matching*, se busca identificar si una lectura coincide con algún ítem

existente en la base de datos del cliente, lo que indica que la lectura es correcta. En caso de que no se encuentre una coincidencia, se registra esta discrepancia para su posterior análisis y consideración.

1.3.2. Algoritmo de consolidación

La empresa actualmente ha desarrollado dos códigos diferentes para abordar la consolidación de objetos de tipo fleje, los cuales comparten una base común pero se adaptan a las variaciones presentes en los diferentes tipos de cámaras utilizadas por el robot y los formatos de flejes capturados. De esta forma tienen un código para las detecciones provenientes de las cámaras laterales y otro para las de overhead.

Los algoritmos reciben la siguiente información para realizar la consolidación de los objetos:

- Posición real de las detecciones entregada por el robot luego de hacer la proyección de posición.
- Lecturas del contenido de los flejes.
- Puntaje obtenido en el cruce con la maestra del cliente.

Caso detecciones cámaras laterales

Para las detecciones de las cámaras laterales se identifican los flejes cuyo contenido coincida con algún producto en el catálogo del cliente. De aquellas detecciones, dos se consideran el mismo fleje si:

1. El contenido de su lectura coincide.
2. No se encuentran dentro de la misma imagen.
3. Se encuentran cercanos espacialmente (distancia menor a un radio definido), esto es calculado con las posiciones proyectadas.

En el caso de las detecciones que no coinciden con ningún producto en el catálogo del cliente y, por lo tanto, no poseen una lectura confiable, se asignan a uno de los objetos previamente consolidados si cumplen con las siguientes condiciones:

1. No se encuentran dentro de la misma imagen.
2. Se encuentran cercanos espacialmente (mismo radio ya mencionado).

Si una detección no logra consolidarse con ningún otro objeto existente, se considera como un objeto independiente.

Finalmente, se selecciona un representante de cada grupo de objetos consolidados, dándole preferencia a aquellos que coinciden con una entrada del catálogo de productos. Este representante será el objeto que se muestra a los clientes en las plataformas utilizadas.

Caso detecciones cámara overhead

En el escenario de las detecciones provenientes de la cámara overhead, se introduce una modificación al algoritmo debido a la posibilidad de que los flejes detectados contengan múltiples SKU correspondientes a diferentes productos en la caja. Para considerar que dos detecciones son el mismo objeto, se establece como criterio que un porcentaje de los SKU capturados coincidan.

1.3.3. Problemas relacionados a la solución

Algunos problemas asociados a esta solución son:

- **Dependencia de la lectura de contenido:** La solución actual utiliza la lectura del contenido de los flejes como parte del proceso de consolidación. Esto puede ser problemático si la lectura no es confiable o si no se puede obtener para ciertas detecciones. Esto puede resultar en la asignación incorrecta de detecciones a objetos.
- **Error en la detección de distancia:** El robot presenta errores en la estimación de las posiciones. Para abordar esta situación, se utiliza un radio de búsqueda en las detecciones. Sin embargo, esto puede llevar a dos tipos de errores. En primer lugar, puede suceder que dos detecciones de un mismo objeto tengan predicciones de posición muy diferentes y, por lo tanto, no se consoliden juntas. En segundo lugar, puede ocurrir que haya dos objetos idénticos en proximidad cercana y se consoliden como uno solo, a pesar de ser dos objetos separados.
- **Mantenimiento y gestión de repositorios:** La solución actual requiere revisar dos repositorios diferentes en caso de errores o mejoras en el proceso. Esto puede generar una carga adicional de trabajo además de dificultar la gestión y mantenimiento del sistema a largo plazo.

1.4. Objetivo

El objetivo de este trabajo es desarrollar un algoritmo capaz de asociar las detecciones de un mismo objeto a lo largo de capturas contiguas. La solución buscada debe ser escalable para adaptarse a la gran cantidad de clientes que posee la empresa, así como ofrecer un rendimiento superior a los algoritmos existentes. Además, se considerarán variables como el costo de implementación de la solución y el tiempo requerido para obtener resultados, con el fin de proporcionar una solución eficiente y efectiva para la empresa.

1.4.1. Objetivos específicos

Para lograr el objetivo general se necesitará completar los siguientes puntos:

1. Realizar una revisión de la disponibilidad de datos relevantes para la tarea de asociación de detecciones.
2. Evaluar diferentes enfoques y técnicas utilizadas en el campo del *tracking* de objetos, analizando su eficacia y rendimiento, así como su capacidad para cumplir con necesidades específicas de la asociación de detecciones en las imágenes capturadas por los robots.
3. Implementar una solución inicial que sea capaz de asociar las detecciones de un mismo objeto en capturas contiguas para una única cámara.
4. Extender la solución implementada para abordar la asociación de detecciones en escenarios con múltiples cámaras.
5. Disponibilizar la solución implementada en la nube, teniendo en cuenta las capacidades de las máquinas para garantizar su eficiencia en el despliegue y el uso de recursos.
6. Evaluar la implementación realizada en términos de calidad de los resultados obtenidos, costo de implementación y escalabilidad.
7. Documentación del código y repositorio.

1.5. Metodología

La metodología seguida en este trabajo se dividió en los siguientes pasos:

1. **Investigación y revisión de literatura:** Se realizó una investigación y una revisión de la literatura existente sobre el tema. Esto proporcionó una base sólida de conocimientos y mejores prácticas en el área de estudio.
2. **Solución para una cámara:** En esta etapa, se diseñó y desarrolló una solución inicial para el procesamiento de imágenes capturadas por una única cámara.
3. **Solución para múltiples cámaras:** Con base en la solución diseñada para una sola cámara, se amplió el enfoque para abarcar múltiples cámaras. Se implementaron estrategias para la fusión de datos de diferentes cámaras, asegurando la coherencia y consistencia de las detecciones en todo el sistema
4. **Evaluación y análisis de resultados:** Se llevaron a cabo pruebas de la solución implementada utilizando conjuntos de datos reales. Se evaluaron los resultados obtenidos, comparando la precisión, eficiencia y escalabilidad de la solución con los objetivos planteados. Además, se generaron visualizaciones para comprender mejor los resultados del sistema.

5. **Documentación y conclusiones:** Se documentaron todos los pasos del proceso metodológico, incluyendo el diseño de la solución, los resultados obtenidos y las conclusiones derivadas del estudio. Se destacaron los logros alcanzados, los desafíos enfrentados y las posibles áreas de mejora en trabajos futuros.

1.6. Estructura del documento

El resto del documento se organiza de la siguiente manera: En el Capítulo 2 se lleva a cabo una revisión de la literatura existente utilizados en investigaciones anteriores. En el Capítulo 3 se presenta la solución propuesta en detalle, describiendo el algoritmo utilizado en la consolidación de objetos. El Capítulo 4 se centra en la implementación de la solución, detallando los aspectos técnicos y las herramientas utilizadas en el desarrollo del algoritmo. Luego, en el Capítulo 5 se presentan y analizan los resultados obtenidos a partir de la aplicación de la solución propuesta, evaluando su efectividad y rendimiento. El Capítulo 6 se dedica a la discusión de los resultados, donde se realiza una interpretación de los hallazgos, se identifican limitaciones y se proponen posibles trabajos futuros. Finalmente, en el Capítulo 7 se presentan las conclusiones del trabajo realizado, resumiendo los principales hallazgos y destacando la contribución de la investigación al campo de estudio.

Además, se incluye un Apéndice A el cual contiene un glosario con términos específicos utilizados en el trabajo, cuya lectura se recomienda para una mejor comprensión del contenido.

Capítulo 2

Marco Teórico

2.1. Procesamiento de imágenes

El procesamiento de imágenes es una disciplina en el campo de la visión por computadora que se enfoca en el desarrollo de algoritmos y técnicas para transformar, analizar y extraer información valiosa a partir de estas representaciones visuales, permitiendo así la comprensión y el procesamiento automatizado de las imágenes.

En este proyecto, la información visual capturada puede ser de gran utilidad para la consolidación de objetos. Ya que se puede comparar el contenido visual de las detecciones para decidir si se tratan de un mismo objeto.

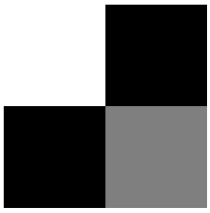
Además, al emplear un nuevo método para comparar detecciones, se podría prescindir de las lecturas de estas. Lo cual expandiría los escenarios en los que puede ser utilizado el algoritmo.

Dado esto, es esencial realizar un estudio del procesamiento de imágenes. Esto implica comprender la naturaleza de una imagen digital, además de explorar operaciones aplicables para mejorar la calidad, extraer información relevante o realizar transformaciones específicas. Estos conceptos y técnicas serán fundamentales para la creación y comprensión de los descriptores utilizados en la consolidación de objetos.

2.1.1. Representación de imágenes

Una imagen se puede describir como una señal bidimensional discretizada, compuesta por una matriz de elementos llamados píxeles. Cada píxel representa un punto en la imagen y contiene información sobre el brillo, el color y otras propiedades visuales.

Una forma común de representar imágenes es en escala de grises, donde cada píxel se asigna a un valor que indica su nivel de intensidad lumínica. En este tipo de imágenes, los valores de los píxeles varían desde el negro (valor mínimo) hasta el blanco (valor máximo), lo que permite representar distintos niveles de brillo y sombras.



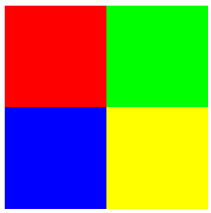
(a)

$$\begin{bmatrix} 255 & 0 \\ 0 & 127 \end{bmatrix}$$

(b)

Figura 2.1: (a) Imagen en escala de grises compuesta por 4 píxeles. (b) Matriz que representa la imagen en escala de grises.

Por otro lado, las imágenes a color contienen información adicional sobre el color de cada punto en la imagen. Existen varios modelos de color utilizados para representar imágenes a color, como el modelo RGB (rojo, verde y azul), el modelo HSV (matiz, saturación y valor), el modelo CMYK (cian, magenta, amarillo y negro) y el modelo CIE (Comisión Internacional de la Iluminación).



(a)

$$\begin{bmatrix} 255 & 0 \\ 0 & 255 \end{bmatrix}$$

(b)

$$\begin{bmatrix} 0 & 255 \\ 0 & 255 \end{bmatrix}$$

(c)

$$\begin{bmatrix} 0 & 255 \\ 255 & 0 \end{bmatrix}$$

(d)

Figura 2.2: (a) Imagen a color compuesta por 4 píxeles. (b) Matriz que representa el componente rojo de la imagen. (c) Matriz que representa el componente verde de la imagen.(d) Matriz que representa el componente azul de la imagen.

2.1.2. Tipos de operaciones

En el procesamiento de imágenes, existen diferentes tipos de operaciones que permiten realizar transformaciones y manipulaciones en los píxeles de una imagen. Estas operaciones se aplican con el objetivo de resaltar características, mejorar la calidad visual, eliminar ruido o realizar otras tareas específicas. A continuación, se describen algunos de los tipos de operaciones más comunes:

Operadores punto a punto

Estas operaciones se aplican individualmente a cada píxel de la imagen sin considerar su contexto espacial. Son operaciones simples que modifican en forma independiente cada pixel

de la imagen de entrada I para generar una nueva imagen G:

$$G(i,j) = h(I(i,j)) \tag{2.1}$$

Ejemplos de operadores punto a punto incluyen el ajuste de brillo y contraste, la inversión de colores o la aplicación de filtros de umbral.

Operadores Lineales

En los operadores lineales, el valor de cada píxel de la imagen resultante G se calcula como la suma ponderada de los valores de una ventana de píxeles en la imagen original I:

$$G(i,j) = \sum_{k,l} I(i+k, j+l)h(k,l) \tag{2.2}$$

Una técnica comúnmente utilizada para realizar este tipo de operaciones es la convolución, que consiste en deslizar una máscara o *kernel* sobre la imagen original. Al hacerlo, se multiplican los valores de los píxeles de la ventana por los correspondientes coeficientes del *kernel* y se suman para obtener el nuevo valor del píxel en la imagen resultante. Este proceso se repite para todos los píxeles de la imagen, generando así una transformación lineal de la imagen original.

En la Figura 2.3, se ilustra el proceso de convolución. La ventana se desliza sobre la imagen original, y en cada posición se realiza la multiplicación y suma de los valores de los píxeles con los coeficientes del *kernel*.

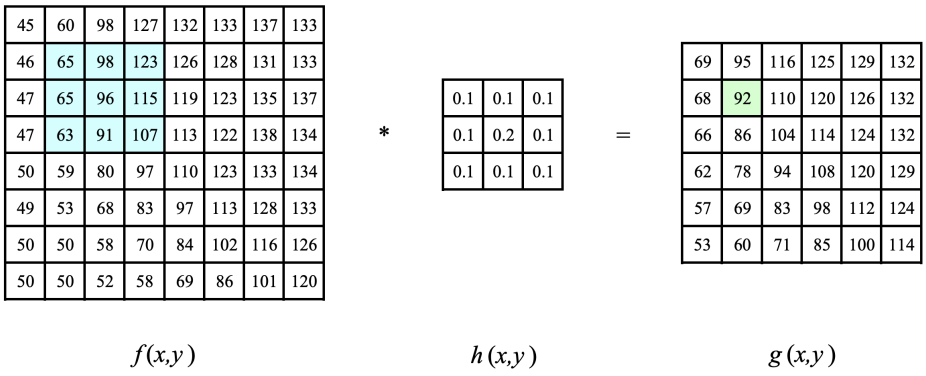


Figura 2.3: Ejemplo de un filtro de convolución. Cada valor de píxel transformado se crea multiplicando su valor actual y los valores de los píxeles a su alrededor contra una matriz de coeficientes.[18]

El uso de diversos *kernels* permite aplicar una variedad de efectos y filtros a una imagen. Algunos ejemplos comunes de operadores lineales incluyen el suavizado, que se logra mediante el uso de un *kernel* de promedio. También está el realce de bordes, el cual utiliza un *kernel* de detección de bordes como el conocido filtro de Sobel [16], ilustrado en la Figura 2.4. Otro efecto es la detección de líneas, para lo cual se emplea un *kernel* de convolución específico.

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

(a)

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

(b)

(c)

(d)

Figura 2.4: (a) *Kernel* para detectar bordes verticales. (b) *Kernel* para detectar bordes horizontales. (c) Imagen original, en escala de grises. (d) Bordes detectados al aplicar el filtro de Sobel.

2.2. Descripción de características

La descripción de características tiene como objetivo principal generar datos estructurados que resuman el contenido de una imagen. En lugar de tratar directamente con la imagen completa, se utiliza un descriptor que captura de manera compacta y representativa una o más características específicas, como el color, los bordes, las texturas, entre otras. Este enfoque permite simplificar y agilizar el procesamiento de imágenes, ya que la similitud entre descriptores se corresponde con la similitud entre los contenidos visuales que representan.

En este proyecto, la elección adecuada del descriptor es crucial para lograr una consolidación precisa de objetos, especialmente cuando se enfrentan variaciones visuales entre distintas detecciones de un mismo objeto. Cada tipo de descriptor tiene sus ventajas y desventajas en términos de robustez ante cambios en la iluminación, escala, rotación y otras variaciones visuales.

2.2.1. Descriptores Globales

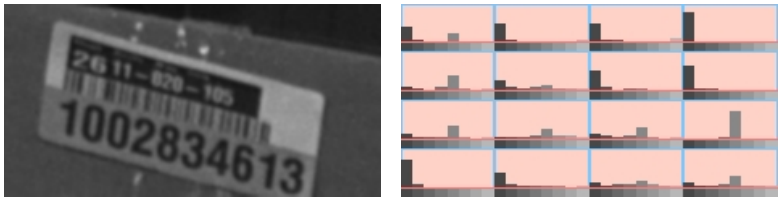
Los descriptores globales son representaciones que consideran la imagen en su totalidad, sin tener en cuenta regiones o detalles específicos. Estos descriptores capturan características globales, como la distribución de colores, la textura general o la forma global de la imagen. Son útiles para evaluar la similitud global entre imágenes y son especialmente adecuados para aplicaciones donde se busca comparar imágenes en su conjunto.

- **Descriptores de intensidades :**
  Los descriptores de intensidades se basan en el análisis de la distribución de los niveles de intensidad en la imagen. Estos descriptores pueden incluir histogramas de intensidad, donde se calcula la frecuencia de aparición de diferentes niveles de intensidad. También pueden utilizarse descriptores estadísticos, como la media y la desviación estándar de los valores de intensidad. Estos descriptores son útiles para capturar información sobre la luminosidad y el contraste de la imagen.

- **Descriptores de bordes :**
  Los descriptores de bordes se centran en la detección y descripción de los bordes y contornos presentes en la imagen. Estos descriptores pueden incluir características como

la magnitud y la orientación de los bordes, la detección de esquinas o la descripción de la estructura de los contornos. Son útiles para capturar información sobre las formas y estructuras presentes en la imagen.

Un ejemplo es el histograma de orientaciones de gradientes (HOG). En este método, primero se calculan los píxeles de borde utilizando un filtro, como el filtro de Sobel. Luego, para cada píxel de borde, se calcula su ángulo de gradiente. A continuación, se crea un histograma que agrupa los ángulos o direcciones de los gradientes en diferentes zonas. Un ejemplo visual se muestra en la Figura 2.5.



(a) (b)

Figura 2.5: (a) Imagen original, en escala de grises. (b) HOG asociados a la imagen, usando 4x4 zonas y 8 bins por zona.

- **Descriptores de texturas :**
  Los descriptores de texturas se utilizan para analizar y representar las características texturales presentes en la imagen. Estos descriptores pueden incluir medidas de la regularidad, la suavidad, la rugosidad o la coherencia textural. Pueden utilizarse técnicas como la matriz de co-ocurrencia de niveles de gris, el espectro de frecuencia o el análisis de texturas basado en filtros. Estos descriptores son útiles para capturar información sobre los patrones y las texturas presentes en la imagen.

Los descriptores globales son útiles para capturar características generales de una imagen, pero pueden tener dificultades para identificar detalles específicos y no son tan robustos ante cambios de perspectiva o iluminación. Para abordar estas limitaciones, se estudiarán los descriptores locales.

2.2.2. Descriptores Locales

Contrario al caso de los descriptores globales, los descriptores locales se enfocan en regiones o puntos de interés específicos dentro de una imagen. Estos descriptores capturan características locales, como patrones texturales, esquinas o bordes. Debido a su enfoque en regiones específicas, los descriptores locales suelen ser más robustos ante cambios en la iluminación, escala y rotación de la imagen.

En lugar de describir la imagen en su totalidad, los descriptores locales buscan puntos de interés que representen la imagen y calculan los descriptores solo de esos puntos. Esto permite calcular la similitud parcial entre imágenes, enfocándose en regiones específicas de

interés. La cantidad de descriptores puede variar según la complejidad y cantidad de regiones destacadas en la imagen.

Detección de puntos de interés

Los puntos de interés, también conocidos como *keypoints*, son píxeles o regiones destacadas en una imagen que se utilizan para describirla y pueden ser usados para encontrar coincidencias entre dos imágenes. Estos puntos son fácilmente distinguibles del resto de la imagen y tienen la propiedad de ser invariantes a cambios en el color, rotación, traslación, escala de la imagen y homografías (proyecciones de perspectiva). La finalidad de los puntos de interés es que sean consistentes y repetibles entre imágenes que contengan los mismos objetos.

Una característica importante de estos puntos es su capacidad para mantener la robustez frente a cambios en la escala del tamaño de la imagen. Esto significa que se pueden comparar las apariciones de objetos idénticos en imágenes de diferentes tamaños. Para lograr esta robustez a escala, se utiliza el concepto de multi-resolución y el espacio de escala, conocido como *scale space*.

El espacio de escala es un volumen tridimensional $L(x, y, \sigma)$ que representa la convolución de la imagen original I con un *kernel* gaussiano G de varianza σ en diferentes escalas:

$$L(x, y, \sigma) = (I * G_\sigma)(x, y), \sigma \in \mathbb{R} \tag{2.3}$$

Al aumentar el valor de σ en el *kernel* gaussiano, la imagen resultante se vuelve más borrosa. De la misma forma cuando σ vale 0, la imagen es equivalente a la original:

$$L(x, y, 0) = I(x, y) \tag{2.4}$$

Además, es importante destacar que una imagen con un *kernel* gaussiano contiene la misma información que una imagen reducida de tamaño. Por lo que las operaciones pueden aplicarse indistintamente para este contexto.

La multi-resolución se logra mediante la construcción de una pirámide de imágenes con diferentes niveles de resolución, donde cada nivel es obtenido mediante la reducción de la imagen original, como se muestra en la Figura 2.6. Cada nivel de la pirámide representa una versión de la imagen con una escala diferente. Esta pirámide de resolución permite aplicar un algoritmo de detección de *keypoints* de tamaño fijo en todos los niveles, lo que proporciona una detección multi-resolución.

Existen dos enfoques comunes para detectar puntos de interés en una imagen: uno basado en esquinas y otro basado en *blobs*. Estos enfoques se utilizan para identificar puntos distintivos y relevantes en la imagen. La elección de uno de estos enfoques nos proporcionará los *keypoints* necesarios para llevar a cabo tareas de procesamiento y análisis de imágenes de manera efectiva.

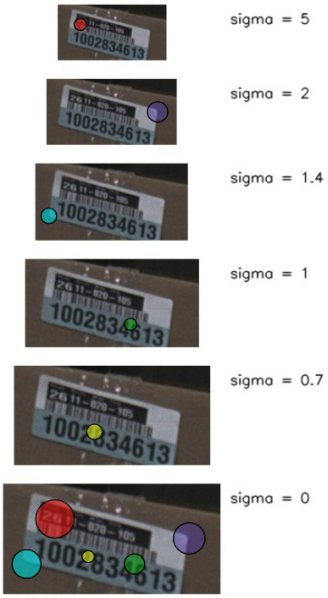


Figura 2.6: La figura muestra una pirámide de Gauss construida con distintos niveles de resolución. En cada nivel, se muestran las detecciones de *keypoints* calculadas para la imagen en esa escala específica. Esta representación multi-resolución es esencial para lograr una detección y descripción precisa de características en diferentes escalas de la imagen original.

- **Detector de esquinas:**

  Existen varios detectores de esquinas, uno de los clásicos es el detector de Moravec [14]. Las esquinas son puntos donde hay cambios significativos en la intensidad de los píxeles en diferentes direcciones. Por ejemplo, una esquina podría tener una intensidad alta en una dirección y baja en las direcciones perpendiculares.

  Otros detectores de esquinas populares incluyen el detector de Harris & Stephens [5] y el detector de Shi-Tomasi [11]. Estos detectores también buscan puntos de interés que representen esquinas en una imagen.

- **Detector de blobs:**

  Los *blobs* son manchas o regiones distintivas en una imagen, cómo los que se muestran en la Figura 2.7. El objetivo del detector de *blobs* es identificar estas regiones.

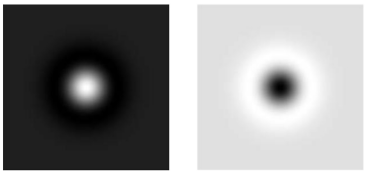


Figura 2.7: Ejemplo visual de un *blob*. El *blob* se caracteriza por ser una región circular con tonalidades de color y textura distintivas en comparación con el fondo de la imagen.

  El detector de *blobs* utiliza el operador Laplaciano del Gaussiano (LoG) para detectar regiones distintivas en una imagen. El LoG consiste en aplicar un suavizado gaussiano seguido de un operador Laplaciano, lo cual resalta los detalles y detecta los bordes.

El enfoque común para detectar *blobs* es construir una pirámide Laplaciana, similar a la pirámide Gaussiana pero con niveles que representan las escalas de los bordes según el LoG. Luego, se compara cada píxel con sus 26 vecinos: 8 vecinos en la misma imagen de borde, 9 píxeles en la imagen anterior y 9 píxeles en la imagen siguiente de la pirámide.

Se buscan píxeles que sean máximos o mínimos, ya que estos representan los píxeles que desaparecen al aplicar la operación gaussiana. Además, se descartan candidatos con bajo contraste (gradiente bajo) o candidatos de borde (gradiente menor en un sentido).

A los *keypoints* seleccionados se les pasa a calcular un descriptor que representa sus características.

Calcular descriptor

Existen varios descriptores, pero uno destacado por su eficacia es SIFT (*Scale-Invariant Feature Transform*)[10]. En este método, cada punto de interés se representa por (x, y, σ, θ). La ubicación espacial (x, y) y el tamaño σ se definen hasta este punto. Para lograr la invariancia a rotaciones, se calcula una orientación θ para cada *keypoint* basándose en el contenido del punto de interés.

Para representar la vecindad de cada *keypoint* en SIFT, se calcula un histograma de orientaciones de gradiente (HOG) en zonas de 4x4 y con 8 bins. Esto genera un descriptor de 128 dimensiones que captura las características locales de la región.

Algunas propiedades del descriptor SIFT son que los gradientes son invariantes a cambios regulares de iluminación, la normalización proporciona invariancia a ajustes globales de brillo y contraste, y los cambios no regulares de iluminación afectan más la magnitud de los gradientes que la orientación.

Los descriptores SIFT tienen una longitud unitaria (norma L2) y se pueden comparar utilizando la distancia euclidiana entre vectores.

2.2.3. Deep Features

Otra forma de describir imágenes es mediante el uso del aprendizaje profundo, también conocido como *deep learning*. En esta técnica, se utilizan redes neuronales profundas para procesar datos multimedia. A diferencia de las redes neuronales tradicionales, como los perceptrones multicapa (MLP), el aprendizaje profundo utiliza directamente los datos multimedia como entrada a la red y entrena la red para calcular un descriptor de contenido.

Las redes neuronales convolucionales (CNN) son una forma avanzada de extraer características de una imagen. Estas redes se entrenan con grandes volúmenes de datos para aprender a reconocer patrones y características en las imágenes. Estas características se utilizan para asociar imágenes similares.

Crear una red neuronal convolucional desde cero es un proceso complejo, por lo que es

común utilizar redes pre-entrenadas en grandes conjuntos de datos. Estas redes ya han aprendido características relevantes de las imágenes, lo que permite aprovechar su conocimiento previo en nuevas tareas.

Detector-Free Local Feature Matching with Transformers (LoFTR)

LoFTR [17] utiliza la arquitectura Transformer para calcular descriptores de características locales sin necesidad de un detector. La arquitectura Transformer consta de capas de autoatención y atención cruzada. Las capas de autoatención aprenden a prestar atención a diferentes partes de los mapas de características de una imagen, mientras que la atención cruzada permite emparejar características entre dos imágenes.

Los descriptores en LoFTR se calculan de la siguiente manera:

1. Los mapas de características de dos imágenes se introducen en la arquitectura Transformer.
2. Las capas de autoatención aprenden a prestar atención a partes relevantes de los mapas de características.
3. La capa de atención cruzada aprende a emparejar características similares en las dos imágenes.
4. La salida de la arquitectura Transformer son descriptores que se utilizan para emparejar las dos imágenes.

Los descriptores en LoFTR son vectores de 128 dimensiones y se calculan mediante un modelo de aprendizaje profundo. Esto permite que los descriptores capturen características relevantes para el emparejamiento de imágenes.

2.3. Similitud y comparación de características

La similitud es un concepto fundamental en el análisis de imágenes, pero su definición y modelado presentan desafíos debido a su naturaleza subjetiva y contextual. La similitud entre imágenes puede ser percibida de manera diferente por diferentes personas, lo que dificulta su cuantificación precisa.

Además, la similitud no siempre cumple con la desigualdad triangular, es decir, dos imágenes pueden ser similares a una tercera imagen, pero no necesariamente ser similares entre sí. Estos desafíos resaltan la necesidad de desarrollar métodos y técnicas robustas para medir y comparar la similitud entre imágenes, que consideren tanto aspectos visuales como semánticos, y sean capaces de capturar la complejidad y subjetividad de la percepción humana.

En este proyecto es importante comparar los descriptores de las detecciones y utilizar esta información para determinar de manera matemática si se refieren a un mismo objeto o no.

2.3.1. Comparar descriptores

En la comparación de descriptores locales, cada descriptor de la primera imagen se compara con cada descriptor local de una segunda. Para esto, se utiliza la distancia euclidiana entre los descriptores, sin considerar las coordenadas espaciales de los *keypoints*.

Una forma común de realizar esta comparación es mediante la búsqueda del vecino más cercano para cada descriptor local. Las coincidencias entre descriptores de dos imágenes se definen como los pares $(v_i, NN(v_i))$, donde v_i es un descriptor local de la imagen 1 y $NN(v_i)$ es el descriptor local de la imagen 2 que tiene la menor distancia a v_i.

Sin embargo, esta medida de distancia por sí sola no es suficiente para determinar si dos descriptores son lo suficientemente similares para considerarlos como coincidentes. Para abordar este problema, se aplican técnicas adicionales, como el filtrado de buenas coincidencias y la coherencia espacial, que permiten mejorar la precisión en la identificación de coincidencias significativas entre descriptores.

Filtrado de buenas coincidencias

En el filtrado de buenas coincidencias, se busca seleccionar aquellas coincidencias que son cercanas o parecidas entre sí bajo algún criterio. Esto se puede lograr aplicando un criterio individual, donde la selección de una coincidencia no afecta al resto. Un ejemplo de esto sería estableciendo una distancia máxima como valor umbral.

También se pueden filtrar las coincidencias basándose en la relación de distancias entre los descriptores. Para esto se seleccionan las coincidencias $(v_i, NN(v_i))$ que cumplen con la siguiente relación:

$$\frac{dist(v_i, NN(v_i))}{dist(v_i, 2NN(v_i))} \leq umbral \tag{2.5}$$

El umbral utilizado debe ser ajustado y permite elegir descriptores donde la distancia del más cercano es mucho menor que la distancia de la segunda opción, lo que representa descriptores distintivos o únicos del objeto y permite filtrar gran parte del ruido.

Coherencia espacial

La coherencia espacial se refiere a la selección de coincidencias que apuntan a un mismo lugar en ambas imágenes. En este caso, se aplica un criterio grupal y se decide en función de un conjunto de coincidencias.

Este proceso implica encontrar una transformación geométrica que sea mayoritaria entre las coincidencias. Para encontrar estas transformaciones, se pueden utilizar algoritmos como RANSAC [4] o mínimos cuadrados [6], que permiten estimar la transformación geométrica

que mejor se ajusta a las coincidencias y descartar aquellas que no cumplen con el patrón mayoritario.

2.4. Seguimiento de objetos

El *tracking* de objetos es una técnica utilizada para seguir y mantener el registro del movimiento de objetos en una secuencia de imágenes o videos. El objetivo principal del *tracking* es estimar la trayectoria y la identidad de los objetos a medida que se desplazan en el espacio y el tiempo.

En el contexto de este proyecto, el *tracking* de objetos se puede utiliza para asociar las detecciones en distintas capturas de imágenes y así determinar qué detecciones corresponden a un mismo objeto a lo largo del tiempo.

El seguimiento de objetos generalmente se basa en dos aspectos principales: el modelado del movimiento de los objetos y, en algunos casos, el modelado de su apariencia. A continuación, se describen ambos enfoques que permiten realizar el seguimiento de los objetos en una secuencia de imágenes o videos.

2.4.1. Seguimiento basado en modelos de movimiento

El enfoque se fundamenta en la predicción del movimiento de los objetos mediante el uso de modelos dinámicos. Estos modelos describen el comportamiento esperado de los objetos a medida que se desplazan en el espacio y en el tiempo, tomando en cuenta información previa sobre su movimiento. Esto permite realizar proyecciones de su ubicación futura con mayor precisión.

Modelos de movimiento

- **Filtro de Kalman :**
  El filtro de Kalman [7] es un método de estimación utilizado en el seguimiento de objetos para predecir la trayectoria y estado de un objeto en función de mediciones y modelos dinámicos.

  Este modelo utiliza un enfoque probabilístico para combinar información previa y mediciones actuales con el objetivo de obtener una estimación óptima del estado actual del objeto. Se basa en dos etapas principales: la etapa de predicción y la etapa de actualización.

  En la etapa de predicción, se utiliza el modelo dinámico del objeto para predecir su estado futuro. El modelo dinámico describe cómo se espera que el objeto se mueva en función de su estado anterior. Se calcula una estimación del estado predicho utilizando

el modelo y se estima la incertidumbre asociada a esta predicción.

En la etapa de actualización, se incorporan las mediciones actuales para corregir la estimación predicha. Las mediciones proporcionan información adicional sobre el estado real del objeto y se utilizan para ajustar la estimación previa. El filtro de Kalman utiliza la estimación predicha y la incertidumbre asociada, junto con las mediciones y su incertidumbre, para obtener una estimación actualizada del estado del objeto.

En general, el filtro de Kalman es una herramienta poderosa en el seguimiento de objetos, ya que permite estimar y predecir el estado de los objetos en movimiento a partir de mediciones ruidosas y modelos dinámicos. Su aplicación en el seguimiento de objetos ha demostrado ser efectiva en una variedad de escenarios, desde el seguimiento de objetos en videos hasta la localización de objetos en sistemas de navegación.

- **Movimiento lineal de velocidad constante**

  El modelo de movimiento lineal de velocidad constante [3] es utilizado para aproximar los desplazamientos entre fotogramas de cada objeto. En este modelo, se asume que el objeto se mueve de manera lineal y constante en términos de velocidad a lo largo del tiempo.

  El estado del objeto objetivo se representa mediante un vector de estado que incluye las siguientes variables:

  - Ubicación horizontal (u): Representa la posición del objeto en el eje horizontal.
  - Ubicación vertical (v): Representa la posición del objeto en el eje vertical.
  - Escala (s): Indica el tamaño o área del objeto.
  - Relación de aspecto (r): Representa la relación entre el ancho y el alto del objeto.

  El modelo de movimiento lineal de velocidad constante considera que las variables de estado evolucionan de acuerdo con una trayectoria lineal y predecible. A partir de las observaciones y mediciones de detección en cada fotograma, se utiliza un filtro de Kalman para estimar y actualizar el estado del objeto objetivo.

  Este modelo es utilizado en aplicaciones de seguimiento de objetos siendo una representación simplificada del movimiento de estos. Sin embargo, es importante tener en cuenta que este modelo asume una velocidad constante y no captura variaciones más complejas en el movimiento de los objetos.

Simple Online and Realtime Tracking (SORT)

Una implementación eficiente de seguimiento de objetos basado en modelos de movimientos es SORT [3]. En esta implementación, los objetos son detectados en cada fotograma y se representan como *bounding boxes*. Una de las características distintivas es su eficiencia, lo

que lo hace adecuado para aplicaciones en tiempo real. Esta implementación utiliza información de momentos previos, en combinación con el fotograma actual, para llevar a cabo el seguimiento de objetos.

El proceso de seguimiento comienza con la detección de objetos en el fotograma actual. Luego, se estima el movimiento de los objetos utilizando un modelo de movimiento lineal de velocidad constante.

A continuación, se lleva a cabo un proceso de asociación de datos para asignar las detecciones a las predicciones de movimiento de las cajas delimitadoras. Se calcula una matriz de costos de asignación que contiene los valores de intersección sobre unión (IOU) entre cada detección y todas las predicciones de movimiento de las *bounding boxes*. Luego, se resuelve de manera óptima la asignación utilizando el algoritmo húngaro [9], que busca la asignación que minimiza los costos totales.

Finalmente, SORT maneja la creación y destrucción de *tracks* para los objetos. Cuando los objetos entran o salen de la imagen, se crean o destruyen identidades únicas en consecuencia. Esto asegura que los objetos sean rastreados correctamente a medida que aparecen y desaparecen en la escena.

Esta implementación es altamente efectiva en aplicaciones en tiempo real, como cámaras de seguridad, donde los objetos generalmente presentan movimientos suaves y cambios sutiles entre fotogramas. Sin embargo, en este proyecto, que trabaja con imágenes individuales en lugar de un video, la cantidad de imágenes es menor, lo que puede llevar a un movimiento más pronunciado entre detecciones. Es esencial considerar esta diferencia en la naturaleza del movimiento al utilizar SORT y evaluar su adecuación para el seguimiento en este contexto específico.

2.4.2. Seguimiento basado en apariencia y características

El enfoque basado en apariencia y características es una estrategia que busca mejorar el seguimiento de objetos al incorporar información sobre la apariencia visual de los mismos.

Simple Online and Realtime Tracking with Deep Associative Metric (DeepSORT)

DeepSORT [19] es una implementación basada en SORT que busca mejorar los resultados utilizando tanto la información de movimiento como la información de apariencia de los objetos.

En el caso de DeepSORT, además de utilizar un modelo de movimiento lineal de velocidad constante, se incorpora un descriptor de apariencia. Este descriptor se obtiene mediante un clasificador entrenado específicamente para reconocer las características visuales de los objetos. Después de entrenar el clasificador, se retira la capa de clasificación final de la red, dejando una capa densa que produce un solo vector de características, conocido como descriptor de apariencia.

Una vez obtenido el descriptor de apariencia, se utiliza el método de vecinos más cercanos (KNN) en la apariencia visual para establecer la asociación de medición a *track* (*Measurement-to-Track Association*). Este proceso consiste en determinar la relación entre una medición y un *track* existente. En el caso de DeepSORT, se utiliza la distancia de Mahalanobis [12] en lugar de la distancia euclidiana para la asociación de medición a *track*, lo que tiene en cuenta las incertidumbres y distribuciones de los datos.

La combinación de información de movimiento y apariencia en DeepSORT proporciona un seguimiento más preciso y robusto de objetos en secuencias de imágenes o videos. Sin embargo, es importante tener en cuenta que los desafíos relacionados con el movimiento aún pueden persistir, especialmente cuando se trabajan con fotos muy separadas en lugar de un video continuo.

2.4.3. Desafíos en el seguimiento

A pesar de los avances en técnicas de *tracking* de objetos, existen desafíos inherentes que pueden dificultar el proceso de asociación de detecciones en distintas capturas de imágenes. Algunos de estos desafíos son:

1. **Oclusiones:** Ocurren cuando un objeto de interés se ve parcial o completamente bloqueado por otro objeto o por el entorno. En tales casos, puede resultar difícil mantener un seguimiento preciso del objeto o incluso identificar si ha cambiado su posición o apariencia. Las oclusiones presentan un desafío significativo en el *tracking* de objetos, ya que es necesario desarrollar estrategias para recuperar la trayectoria y la identidad de los objetos ocultos una vez que vuelven a ser visibles.
2. **Variaciones en la apariencia:** Los objetos pueden experimentar variaciones en su apariencia debido a cambios en la iluminación o perspectiva desde la cual se toma la foto. Estas variaciones pueden dificultar la correspondencia precisa entre las detecciones en diferentes imágenes.
3. **Detecciones falsas y ruido:** Los algoritmos de detección pueden generar falsas detecciones o errores de localización debido a ruido en las imágenes o a la presencia de objetos similares. Estas detecciones falsas pueden interferir con el proceso de *tracking* y dificultar la asociación correcta de las detecciones.

2.5. Evaluación de algoritmos

La evaluación de los algoritmos es una etapa crucial en el proceso de desarrollo, ya que nos permite medir su desempeño y calidad utilizando métricas específicas. Al comparar distintas implementaciones a través de estas métricas, podemos tomar decisiones informadas sobre cuál es la más adecuada para resolver un problema particular.

2.5.1. Métricas de evaluación en procesamiento de imágenes y descripción de contenido

En el procesamiento de imágenes y la comparación de contenido, se utilizan diversas métricas para evaluar el desempeño de los algoritmos. Entre las métricas más comunes para determinar si dos imágenes son iguales, se encuentran:

2.5.2. Precisión y exactitud

Estas métricas miden la calidad de las predicciones o resultados obtenidos por el algoritmo. La precisión se refiere a la proporción de resultados correctos en relación con el total de resultados y se calcula utilizando la siguiente fórmula:

$$Precision = \frac{\text{Verdaderos positivos}}{\text{Verdaderos positivos } + \text{Falsos positivos}} \tag{2.6}$$

El *recall*, también conocido como sensibilidad, es una medida de la proporción de resultados positivos correctamente identificados en relación con el total de resultados reales positivos. Se calcula con la siguiente fórmula:

$$Recall = \frac{\text{Verdaderos positivos}}{\text{Verdaderos positivos } + \text{Falsos negativos}} \tag{2.7}$$

Por otro lado, la exactitud o accuracy se refiere a la cercanía de los resultados obtenidos a los valores reales o verdaderos y se calcula de la siguiente manera:

$$Accuracy = \frac{\text{Verdaderos positivos} + \text{Verdaderos negativos}}{\text{Total muestra}} \tag{2.8}$$

2.5.3. Eficiencia

La eficiencia se refiere a la capacidad del algoritmo para realizar el procesamiento de manera rápida y con un uso eficiente de los recursos disponibles. Se pueden considerar métricas como el tiempo de ejecución, la velocidad de procesamiento y el consumo de recursos (como la memoria o la capacidad de procesamiento).

2.5.4. Costo de implementación y recursos requeridos

Esta métrica evalúa los costos asociados con la implementación y el uso del algoritmo. Puede incluir aspectos como la complejidad del algoritmo, la disponibilidad de recursos computacionales o de *hardware*, y los costos asociados con la adquisición y mantenimiento de los equipos o tecnologías necesarios.

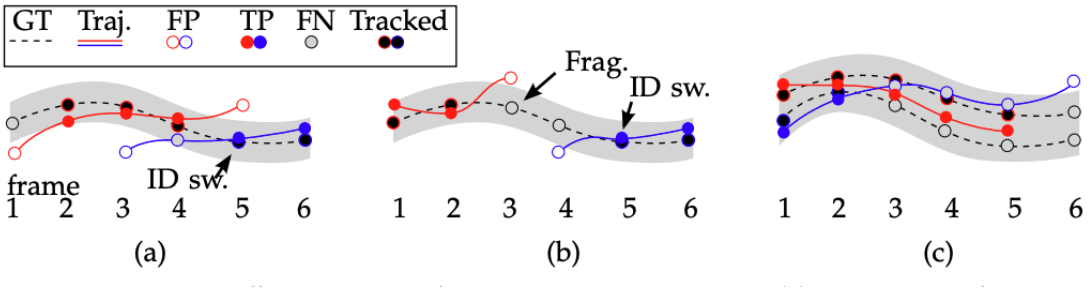


Figura 2.8: Ejemplo de distintos tipos de errores que pueden ocurrir en un algoritmo de seguimiento de objetos. (a) Ejemplifica cómo dos trayectorias pueden pasar por un mismo objeto real. (b) Ilustra un caso de fragmentación en el que una trayectoria se divide y comienza una nueva. (c) Muestra un cambio de ID entre dos trayectorias cercanas [13].

2.5.5. Métricas de algoritmos de seguimiento de objetos

Las métricas utilizadas para evaluar el rendimiento de los algoritmos de seguimiento de objetos deben abordar cinco tipos de errores [8], los cuales se ejemplifican en la Figura 2.8.

1. **Falso negativo:** Estos son objetos que no son detectados por el algoritmo de seguimiento de objetos.
2. **Falso positivo:**Estos son objetos que son detectados por el algoritmo de seguimiento, pero que no están presentes en el *Ground Truth*(ver Glosario).
3. **Fusión o cambio de ID:** se refiere al intercambio de IDs de seguimiento entre dos o más objetos mientras pasan cerca uno del otro, lo que puede llevar a la confusión de las identidades de los objetos rastreados.
4. **Desviación:** Ocurre cuando una trayectoria (*track*) se reinicia con un ID diferente, lo cual puede ocurrir debido a cambios en la apariencia del objeto o a errores en el proceso de asociación de trayectorias.
5. **Fragmentación:** Estos son objetos que se dividen en múltiples trayectorias por el algoritmo de seguimiento.

Algunas métricas comúnmente utilizadas para evaluar el rendimiento de los algoritmos de seguimiento de objetos son las siguientes:

MOTA (Multi-Object Tracking Accuracy) [1] es una métrica integral que tiene en cuenta los aspectos de detección, localización y asociación. Se calcula como una suma ponderada de las tasas de falsos negativos, falsos positivos y desviaciones. Esta métrica proporciona una visión global del rendimiento del algoritmo de seguimiento de múltiples objetos.

MOTP (Multi-Object Tracking Precision) [2] es una métrica de localización que mide la distancia promedio entre las cajas delimitadoras predichas por el algoritmo y el *Ground Truth*. Esta métrica es sensible a los errores en la localización de los objetos y ayuda a evaluar la precisión en la ubicación de los mismos.

IDF1 [15] es una métrica relacionada a la asociación de las detecciones que evalúa la asignación correcta de identidades de objetos a trayectorias. Se calcula como el porcentaje de cuadros en los cuales se asigna la identidad correcta del objeto. Sin embargo, es importante tener en cuenta que un puntaje alto de IDF1 estima el número total de objetos únicos en una escena, pero no proporciona información específica sobre la detección o asociación precisa de los objetos. Además, esta métrica no evalúa la precisión de la localización de los objetos rastreados.

2.6. Entorno de implementación

Los procesos que ocurren entre la captura de imágenes por parte del robot y la entrega de información al cliente se llevan a cabo principalmente en la nube. Zippedi utiliza Google Cloud Platform, que ofrece una amplia gama de servicios y herramientas para el desarrollo y despliegue de aplicaciones en la nube.

Para facilitar el despliegue de sus servicios, la empresa ha desarrollado flujos de Integración Continua/Despliegue Continuo (CI/CD) que permiten una implementación sencilla en plataformas en la nube. Estos flujos incluyen pasos de compilación, *test* y empaquetado de los microservicios en contenedores Docker, lo que garantiza la portabilidad y la independencia de la infraestructura subyacente.

Los *templates* proporcionados permiten una configuración estándar y optimizada para enfocar las energías en la funcionalidad de los microservicios, en lugar de preocuparse por los detalles de cómo desplegar y asegurar su funcionamiento en el entorno en la nube.

En particular, Google Cloud Run es la plataforma utilizada por Zippedi para la ejecución de contenedores. Esta plataforma proporciona un entorno sin servidor que escala automáticamente los recursos en función de la demanda. Esto garantiza un rendimiento óptimo y una alta disponibilidad de las aplicaciones.

Una ventaja significativa de este tipo de entornos es su capacidad de personalización, ya que se puede ajustar la configuración de las máquinas, como la memoria, la cantidad de CPU, el tiempo máximo de respuesta, el número máximo de solicitudes concurrentes, el número de instancias y las bases de datos a las que se pueden conectar.

El costo de estos entornos varía según las características de la máquina y el tiempo de uso. Es importante destacar que cuando no hay solicitudes, las máquinas se apagan para optimizar los recursos y minimizar los costos.

Capítulo 3

Solución propuesta

En este capítulo, se presenta en detalle la solución propuesta para abordar el desafío de consolidación de objetos. La solución utiliza las capturas del pasillo, las detecciones en dichas capturas y la posición proyectada de las detecciones. Aprovechando el orden secuencial de las capturas, se implementa un sistema de *tracks* que permite seguir y consolidar los objetos a lo largo del tiempo. Este capítulo analizará en profundidad los componentes fundamentales del algoritmo, su estructura general y los pasos involucrados en el proceso de seguimiento y consolidación.

3.1. Estructura general

El algoritmo propuesto se basa en la implementación de la técnica SORT, siguiendo una metodología que consta de tres fases esenciales: estimación de movimiento, asociación de detecciones y el manejo de creación y destrucción de *tracks*. Además, en esta propuesta se adopta el enfoque empleado por DeepSORT, donde se realiza la descripción de imágenes, aunque en este caso se ha optado por utilizar el algoritmo SIFT para llevar a cabo dicha descripción.

El procedimiento se lleva a cabo utilizando imágenes capturadas del pasillo, así como las detecciones obtenidas en estas imágenes y proyecciones asociadas. Para ello, se ha desarrollado un sistema de *tracks* que se beneficia del orden secuencial de las capturas del pasillo. Esta estrategia permite prescindir de las lecturas y, por lo tanto, se ha reubicado la posición del algoritmo en el *pipeline* definido anteriormente por la empresa. La Figura 3.1 muestra el diagrama completo de la nueva secuencia.

En el sistema de *tracking*, se realiza una predicción de la posición de las detecciones en la siguiente captura, lo cual proporciona un espacio de búsqueda para comparar las detecciones en términos de apariencia y forma. Esto permite asociar las detecciones con los *tracks* correspondientes. Además, se implementa la creación y destrucción de *tracks*, ya que, dado el movimiento unidireccional del robot en el pasillo, se sabe que los objetos que salen del campo de visión no volverán a aparecer.

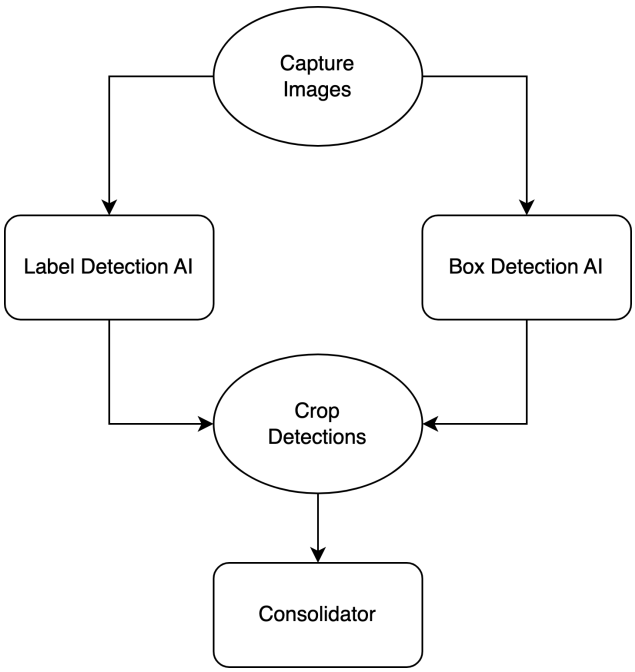


Figura 3.1: Diagrama del nuevo *pipeline*. Los óvalos representan las etapas realizadas por el robot. Los rectángulos representan las operaciones realizadas en la nube.

De esta forma, la vida de un objeto en el algoritmo se puede resumir en los siguientes pasos:

1. **Aparición inicial:** Se detecta el objeto por primera vez.
2. **Sin consolidación:** La detección no se consolida con ningún *track* existente.
3. **Creación del *track*:** Se crea un nuevo *track* para el objeto, se utiliza el detection_id para identificar la detección.
4. **Nueva captura:** En la siguiente captura hay nuevas detecciones.
5. **Predicción y generación de *gate*:** Se predice la posición del *track* en la captura actual y se genera un *gate* de búsqueda.
6. **Comparación y similitud:** Se comparan la apariencia y la forma con las detecciones dentro del *gate* de búsqueda y se calcula un grado de similitud.
7. **Asociación:** La detección con mayor similitud se asocia al *track*, y se agrega el detection_id en una lista.
8. **Seguimiento continuo:** Se continúa observando las capturas sucesivas hasta que el *track* no se actualice dentro de un límite de tiempo máximo establecido.
9. **Eliminación del *track* y consolidación:** Si el *track* no se actualiza, se elimina y se guarda la lista de detection_ids asociados como una consolidación del objeto.

El sistema ha sido desarrollado siguiendo una estructura modularizada que permite una clara separación de cada paso del proceso. Además, los pasos ocurren de forma secuencial,

lo que facilita la sustitución de funciones individuales sin afectar el funcionamiento de los demás.

El código fuente del algoritmo está disponible en el repositorio Gitlab de la empresa, debidamente documentado. Se ha implementado en Python 3, utilizando las bibliotecas "cv2" y "pandas" como las principales herramientas. Cada paso del algoritmo se encuentra en archivos separados, lo que contribuye a mantener un código ordenado y estructurado.

3.2. Entradas del algoritmo

El algoritmo utiliza dos tipos de datos para llevar a cabo la consolidación:

3.2.1. Imágenes

Para utilizar las imágenes, se obtiene la dirección del archivo comprimido (.tar) que las contiene. Estos archivos están almacenados en Google Cloud Storage. Las imágenes se organizan de manera secuencial según el orden en que fueron capturadas, esto gracias al nombre que tienen. Cada cámara envía su propio archivo comprimido con las imágenes correspondientes

Una vez obtenida la dirección del archivo, se procede a descargarlo utilizando la biblioteca "google.cloud" para acceder a los servicios de almacenamiento en la nube. Una vez descargado, se extraen las imágenes y se guardan en un directorio temporal para su posterior uso. Una vez finalizada la consolidación, el directorio es eliminado.

Cuando es necesario utilizar las imágenes, se leen utilizando la biblioteca "cv2" de Python. Esto permite acceder a la información contenida en cada imagen, como sus dimensiones, canales de color y píxeles.

3.2.2. Datos de detecciones

El algoritmo también utiliza datos de detecciones almacenados en dos tablas de una base de datos SQL:

- **Tabla de detecciones por captura:** Esta tabla contiene información detallada sobre cada detección, incluyendo su identificador (detection_id) y las coordenadas de su *bounding box* (x_1, y_1, x_2, y_2).
- **Tabla de posición de detecciones en el pasillo:** Esta tabla contiene la información de la posición de cada detección proyectada en el pasillo. Incluye el detection_id, así como las coordenadas de posición (location_x, location_y, location_z).

Estos datos de detecciones en las tablas SQL están vinculados con las imágenes a través de un capture_id, que corresponde al nombre de la imagen asociada.

3.3. Movimiento

Con el objetivo de reducir el espacio de búsqueda de los objetos a consolidar, se estima el desplazamiento de los objetos entre capturas sucesivas. El movimiento se ilustra en la Figura 3.2, donde se observa claramente el movimiento de las detecciones entre dos imágenes consecutivas.

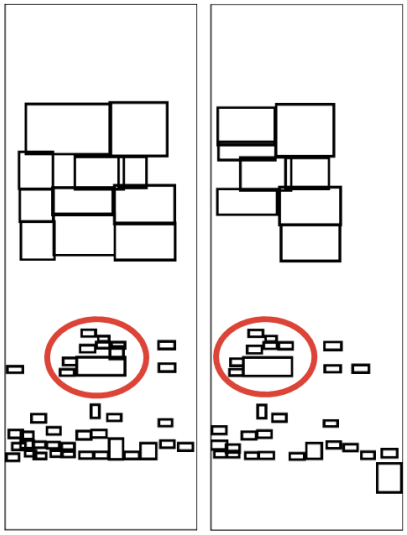


Figura 3.2: Ejemplo de cómo las detecciones se mueven entre dos capturas consecutivas.

A diferencia de SORT, en este algoritmo se calcula el movimiento basándose en las proyecciones de las detecciones en las fotos. Esto con el fin de tener una predicción más precisa del movimiento y abordar casos en los que el desplazamiento no es constante, como puede ocurrir cuando hay obstáculos en el camino del robot.

3.3.1. Calcular posición de las capturas

Para estimar el movimiento entre imágenes, se calcula la posición del centro de la imagen utilizando las detecciones disponibles.

En el caso de imágenes con más de dos detecciones, se utilizan las posiciones de los *bounding boxes* de las detecciones (x_1, y_1, x_2, y_2) y las posiciones reales de las detecciones (location_x, location_y, location_z). Con esto se calcula la relación entre metros y píxeles en la foto. A continuación, se selecciona la detección más cercana al centro de la imagen y se obtiene la distancia en píxeles desde dicha detección hasta el centro de la imagen. Utilizando la relación previamente calculada, se convierten los píxeles a metros, lo que proporciona una estimación de la posición del centro de la imagen.

Para el caso de las imágenes que tienen menos detecciones, se estima su posición en base a las capturas contiguas. Se toma en consideración la información de las imágenes anteriores y posteriores para inferir la ubicación. Este enfoque basado en capturas contiguas proporciona una forma de completar la información faltante en las imágenes con menos detecciones.

Este proceso se lleva a cabo debido a la falta de información sobre la posición exacta de cada imagen capturada por el robot en la implementación actual. Sin embargo, se espera que en el futuro se disponga de esta información. Con el objetivo de asegurar la flexibilidad y adaptabilidad del sistema, los resultados obtenidos en este proceso se almacenan en un *Data Frame* de "pandas", el cual puede ser fácilmente cambiado por información proveniente del robot. De esta manera, el sistema está preparado para integrar la información de posición entregada por el robot en futuras versiones, lo que podría mejorar la precisión y confiabilidad del proceso de consolidación de objetos.

3.3.2. Calcular movimiento

Después de obtener las posiciones de todas las capturas, el cálculo del movimiento en píxeles es sencillo. Se toman las posiciones de las imágenes consecutivas, se calcula la distancia en metros y, utilizando la relación previamente calculada, se convierte a píxeles. Esto nos permite obtener resultados como los mostrados en la Figura 3.3.

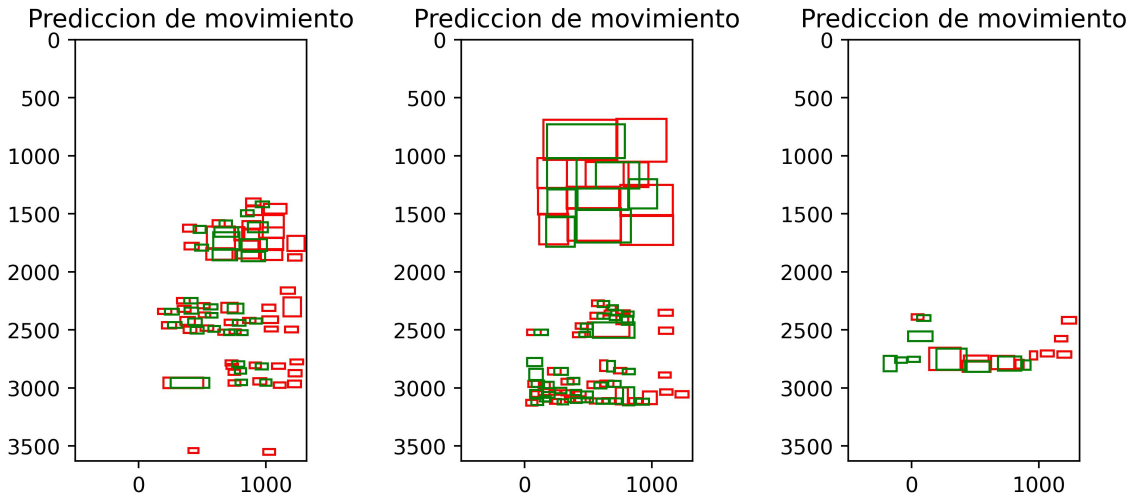


Figura 3.3: Resultados obtenidos para el vector de desplazamiento entre dos capturas consecutivas. En color verde, se representan las detecciones de la primera foto, mientras que en rojo se muestran las detecciones de la foto siguiente, desplazadas con el vector calculado. Se puede observar que a medida que el robot se desplaza, algunas detecciones verdes desaparecen al no tener una contraparte roja. Por otro lado, aparecen nuevas detecciones rojas que no se habían visto en la captura anterior.

3.4. Generación de descriptores

En esta etapa, se procede a resumir la información de las detecciones en descriptores, para lo cual se evaluaron dos métodos: SIFT y LoFTR. Ambos son descriptores locales, lo que les otorga robustez ante cambios de iluminación y perspectiva, condiciones comunes en las capturas tomadas por el robot, como se muestra en las Figuras 3.4 y 3.5.

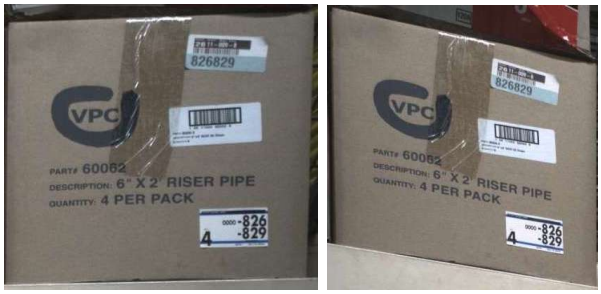


Figura 3.4: Ejemplo del cambio de perspectiva entre detecciones de un mismo objeto en dos capturas.

Figura 3.5: Ejemplo del cambio de iluminación entre detecciones de un mismo objeto en dos capturas.

Ambos métodos fueron evaluados en la búsqueda de objetos similares, y los resultados se presentan en las Figuras 3.6 y 3.7, así como en la Tabla 3.1. Se observa que LoFTR genera una mayor discriminación entre detecciones que corresponden al mismo objeto y aquellas que no, pero, al analizar las métricas asociadas, SIFT terminó por superar a LoFTR. Cabe mencionar que el tiempo de procesamiento de LoFTR fue significativamente mayor que el de SIFT, lo cual es relevante en esta implementación, donde se busca alcanzar un bajo tiempo de ejecución.

Finalmente, se ha decidió utilizar el descriptor SIFT debido a su capacidad para detectar y describir características distintivas en objetos. Además, este descriptor ofrece un tiempo de procesamiento eficiente.

Para generar los descriptores, se realizan recortes de las capturas correspondientes a cada detección. Estos recortes se generan con un margen adicional para considerar posibles características del entorno que puedan ayudar a diferenciar mejor los objetos. Esto es importante ya que dos flejes pueden tener la misma apariencia pero contextos diferentes.

El método SIFT de la biblioteca "cv2" se utiliza para calcular los keypoints y los descriptores SIFT correspondientes a cada recorte. Este algoritmo utiliza una combinación de escalas y orientaciones para detectar y describir puntos clave invariantes a cambios de escala, rotación y cambio de iluminación, entre otros factores, como se ha explicado en el marco teórico.

Los descriptores se asocian a las detecciones y se almacena toda la información en un *Data Frame* de "pandas".

Figura 3.6: Resultados al utilizar el algoritmo SIFT para encontrar similitud entre imágenes. Sobre las imágenes se muestra el detection_id asociado y la cantidad de matches encontrados con la detección original. Las detecciones están ordenadas de la más parecida a la original a la menos similar. Se destaca que no es necesario que las detecciones tengan contenido legible para encontrar matches.

Figura 3.7: Resultados al utilizar el algoritmo LoFTR para encontrar similitud entre imágenes. Sobre las imágenes se muestra el detection_id asociado y la cantidad de matches encontrados con la detección original.

	Precision	Recall	Accuracy
SIFT	1	0.75	0.8909
LoFTR	0.988	0.625	0.818

Tabla 3.1: Métricas asociadas a la descripción y búsqueda de imágenes similares con los métodos SIFT y LoFTR.

3.5. Tracks

Se implementa un sistema de *tracks* de objetos con el objetivo de realizar el seguimiento de los mismos a lo largo de las capturas del pasillo. En esta sección, se presenta la definición de las clases utilizadas para el sistema de *tracks* y se describen las distintas operaciones que se llevan a cabo con los objetos durante el proceso de consolidación.

3.5.1. Definición

Para llevar a cabo el proceso de seguimiento de los objetos, se han creado dos clases: «Object Tracker» y «Mot» (*Multiple Object Tracking*).

La clase «Object Tracker» representa el seguimiento de un objeto específico. Esta clase tiene los siguientes atributos:

- **id:** Identificador único asignado al *track* para distinguirlo de otros *tracks*.
- **time_since_update:** Número de capturas transcurridos desde la última detección asociada.
- **bbox:** Lista que contiene las coordenadas del *bounding box* en el paso de tiempo actual.
- **prediction:** Lista que contiene las coordenadas del *bounding box* predicho en el paso de tiempo actual.
- **detections_id:** Lista que contiene todos los detection_id asociados al *track*.
- **descriptors:** Lista que contiene todos los descriptores asociados al *track*.

La consolidación de un objeto se refiere a la recopilación de todas las detecciones asociadas a ese objeto a lo largo del tiempo. En el caso del sistema de *tracks*, la consolidación se logra mediante la lista de detections_id asociados a cada *track*. Esta lista contiene todos los IDs de detecciones que han sido asociados al objeto durante el proceso de seguimiento.

Además, se utiliza la clase «Mot» que representa todo el *pipeline* de seguimiento. Esta clase cuenta con los siguientes atributos:

- **max_age:** Número máximo de *frames* para mantener vivo un *track* sin detecciones asociadas.
- **trackers:** Lista que contiene todos los objetos de tipo «Object Tracker» activos.
- **consolidation:** Lista que contiene las consolidaciones obtenidas de los *tracks* finalizados.

El método "update" de la clase «Mot» es llamado por cada captura. Este método recibe como entrada un *Data Frame* con la información de las detecciones (x_1, x_2, y_1, y_2, descriptor), así como el movimiento existente entre la captura actual y la captura anterior.

3.5.2. Asociación de detecciones a tracks

En el método "update", se lleva a cabo la asociación de las detecciones con los *tracks* activos. Este proceso se realiza mediante varios pasos.

Movimiento de los tracks y gate de búsqueda

Primero, se realiza la actualización de la posición de los *tracks* activos utilizando el movimiento predicho en pasos anteriores. Esto se hace para tener en cuenta el desplazamiento esperado de los objetos.

Posteriormente, se genera un área rectangular conocida como "*gate* de búsqueda". Esta ventana se crea alrededor de la posición prevista de los *tracks* y sirve como una región de interés para realizar comparaciones de similitud. El tamaño del *gate* se ajusta en función de la dirección y magnitud esperadas del movimiento de los objetos. Se establece un tamaño mayor en la dirección horizontal (eje x) debido a que se espera un mayor desplazamiento en esa dirección.

La implementación del *gate* de búsqueda tiene dos objetivos principales. En primer lugar, permite reducir la cantidad de operaciones de comparación necesarias al limitar el espacio de búsqueda a un área específica. Esto mejora la eficiencia computacional del algoritmo de asociación. En segundo lugar, ayuda a evitar la consolidación errónea de objetos que puedan parecerse pero estén muy alejados espacialmente. Al centrarse en un área de interés cercana a los *tracks*, se favorece la correspondencia entre detecciones y *tracks* con una similitud espacial significativa.

Detecciones dentro del gate

A continuación, se realiza el cálculo de la matriz de IoU (Intersección sobre Unión). Esta matriz nos permite determinar cuáles detecciones se superponen con el *gate* de búsqueda.

El IoU se obtiene dividiendo el área de intersección entre el área de unión de los *bounding boxes*. Un valor de IoU cercano a 1 indica una alta superposición, mientras que un valor cercano a 0 indica poca o ninguna superposición. De esta forma solo se calculará el indice de similitud entre aquellas detecciones y *tracks* que tengan un valor mayor a cero.

Indice de similitud

A continuación, se utiliza la función "find_matches" para calcular el índice de similitud de apariencia entre las detecciones que se encuentran dentro del *gate* de búsqueda. Para este propósito, se emplea el algoritmo `BForce Matcher` de "cv2". Los resultados de esta comparación se ilustran en la Figura 3.8, donde se muestran las detecciones y su similitud.

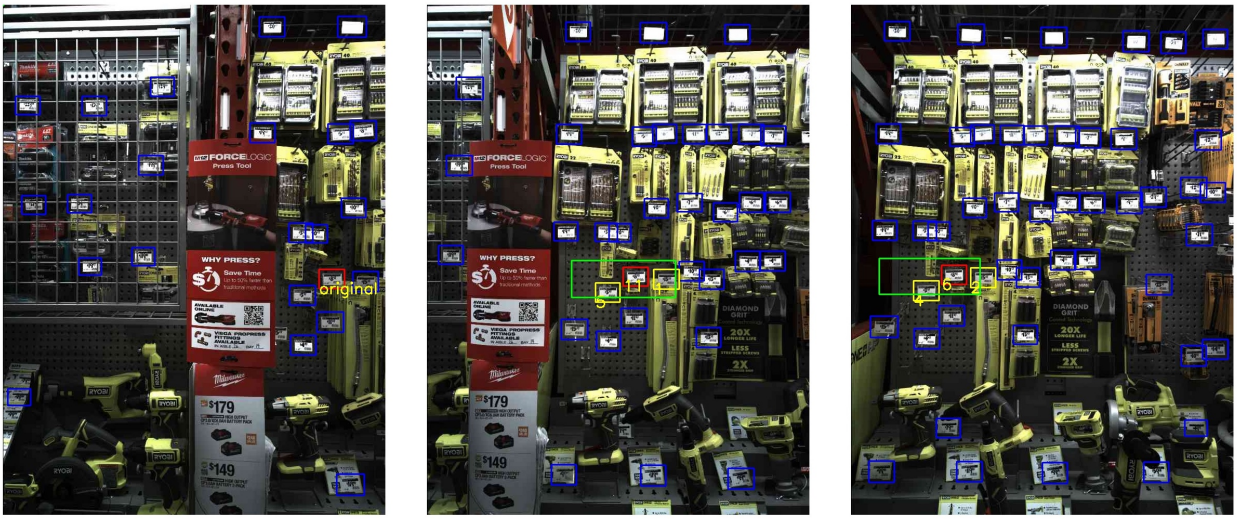


Figura 3.8: Visualización del proceso de asociación de detecciones utilizando el índice de similitud de apariencia. En la secuencia de imágenes mostrada, se resalta la detección inicial del *track* en la primera imagen. En las siguientes imágenes, se muestra en verde el *gate* de búsqueda, donde se realizan las comparaciones de apariencia. Las detecciones se marcan en tres colores: azul para las detecciones fuera del *gate*, amarillo para aquellas dentro del *gate* pero con menor similitud, y rojo para la detección seleccionada para la consolidación debido a su mayor similitud. Además, cada detección comparada se etiqueta con un número que indica el grado de similitud.

El algoritmo `BFMatcher` implementa el algoritmo de coincidencia de fuerza bruta. Este algoritmo compara cada descriptor en un conjunto con todos los descriptores en otro conjunto y encuentra las mejores coincidencias basadas en una medida de distancia. En este caso caso, se utiliza la distancia euclidiana como medida de distancia para calcular la similitud de apariencia entre los descriptores SIFT asociados a las detecciones y los *tracks*.

Para llevar a cabo la comparación, se emplea el método `knnMatch` proporcionado por `BFMatcher`. Este método busca los vecinos más cercanos entre los descriptores de las detecciones que se desean comparar. Utilizando los dos vecinos más cercanos, se calcula el índice de similitud de apariencia. La comparación se realiza evaluando la relación entre la distancia del mejor match y la distancia del segundo mejor match. Si esta relación supera un umbral predefinido de 0.3, se considera que el par de descriptores forma un "good match", lo que indica una alta similitud de apariencia entre la detección y el *track*.

Además de la similitud de apariencia, también se tiene en cuenta la similitud de las formas de los *bounding boxes* para asegurarse de que tengan dimensiones similares. Esta métrica establece una relación entre la altura y el ancho de la detección, lo que permite verificar si los objetos detectados tienen proporciones similares.

La similitud de apariencia y forma se combinan en un factor de similitud, que se calcula como la multiplicación de estos dos factores. Esto permite cuantificar la similitud total entre una detección y un *track*.

Asociar

Finalmente, se procede a la etapa de asociación, donde se seleccionan las mejores asociaciones basadas en la matriz de similitud obtenida previamente. El objetivo es determinar qué detecciones corresponden a qué *tracks* activos.

Es importante destacar que la asociación debe ser de tipo "uno a uno", es decir, cada *track* solo puede asociarse con una detección y viceversa. Esto asegura que se establezca una correspondencia única y coherente entre los objetos detectados y los *tracks* existentes.

Para realizar esta asociación, se toman los máximos por fila y por columna en la matriz de similitud. Esto implica que se selecciona la detección con la similitud más alta para cada *track*, y a su vez, se selecciona el *track* con la similitud más alta para cada detección. Estos máximos representan las mejores asociaciones posibles bajo la restricción de asociación uno a uno.

Es importante tener en cuenta que algunos *tracks* pueden no tener una asociación exitosa con ninguna detección en este paso, lo que indica que no se encontró una detección adecuada que se corresponda con ese *track* en particular. En esos casos, estos *tracks* se actualizan incrementando su valor de "time_since_update", lo que indica que ha pasado un captura desde su última actualización.

En la siguiente sección, se abordará cómo se manejan las detecciones que no tienen una asociación exitosa con ningún *track* y cómo se incorporan nuevas detecciones al sistema de seguimiento.

3.5.3. Creación y destrucción de tracks

La creación y destrucción de *tracks* ocurre en cada llamada a la función "update". En primer lugar, se realiza el movimiento de los *tracks* existentes de acuerdo al desplazamiento entre capturas. Si un *track* se encuentra fuera del plano de captura, se elimina, ya que se estima que no volverá a aparecer. La consolidacion del objeto, equivalente a la lista de detections_ids se guarda en la lista "consolidation" del objeto «Mot».

Además, los *tracks* que no han sido detectados durante un período de tiempo mayor al valor de "max_age" también son eliminados. En esta implementación, se utiliza un valor de "max_age" igual a 2. Esto se debe a que puede haber errores en la detección de objetos, como se muestra en la Figura 3.9, y es importante darles la oportunidad de volver a aparecer en futuras capturas.

Por otro lado, las detecciones que no se han asociado con ningún *track* existente se convierten en nuevos *tracks*. Estos nuevos *tracks* tienen un "time_since_update" igual a 0 y contienen la información asociada a esa detección.

De esta manera, la creación y destrucción de *tracks* garantiza que solo se mantengan aquellos *tracks* que están activos y tienen una correspondencia significativa con las detecciones. Los *tracks* eliminados se conservan en la lista de consolidación.

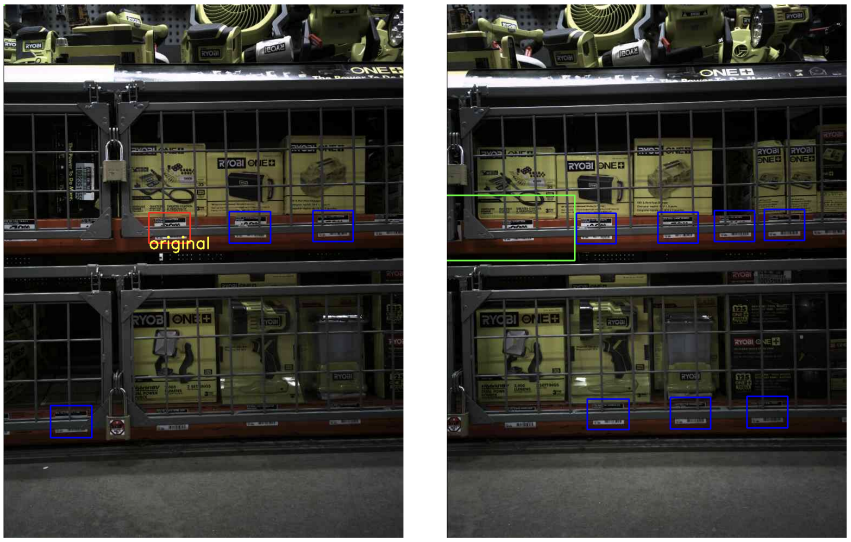


Figura 3.9: Ejemplo de falta de detección en la asociación de *tracks*. En la imagen se muestra cómo, al intentar realizar la asociación, el *gate* de búsqueda acierta en el lugar donde se encuentra el objeto, pero no se asocia debido a un fallo en el algoritmo de detección.

3.6. Consolidación entre múltiples cámaras

La consolidación de objetos en una secuencia de imágenes ha sido abordada exitosamente. Con esto se pueden consolidar los objetos para el caso de la cámara overhead. Sin embargo, en el contexto de las cámaras laterales, que actualmente son tres, es necesario enfrentar el desafío de consolidar los objetos a lo largo de las tres secuencias de imágenes capturadas.

Inicialmente, se planteó una solución que ampliaba el algoritmo existente para manejar múltiples cámaras y se adaptaba el proceso de asociación de detecciones a *tracks*. Sin embargo, surgieron complicaciones relacionadas con el formato de los datos de entrada, lo que dificultó la implementación de esta idea.

Como alternativa, se encontró una solución más viable que utiliza las posiciones proyectadas de los objetos en lugar de los datos de detección originales.

Es importante resaltar que esta solución basada en las posiciones proyectadas no requiere de una mayor información por parte del robot. Sin embargo, en el futuro, si se dispone de información adicional relevante del entorno, se podría explorar la idea originalmente propuesta para lograr una consolidación aún más precisa de los objetos entre las cámaras laterales.

A continuación se describen las dos soluciones.

3.6.1. Primera idea propuesta

Para abordar la consolidación de imágenes provenientes de tres cámaras diferentes, se planteó una solución que involucra una consolidación vertical seguida de la consolidación de

las columnas laterales, como se muestra en la Figura 3.10. Este enfoque se basa en el algoritmo previamente explicado, que utiliza el método "update" para procesar las detecciones de una nueva captura y un vector de movimiento. El algoritmo se aplica tanto al movimiento vertical entre las capturas correspondientes a una misma instancia, como al movimiento horizontal causado por el desplazamiento del robot.

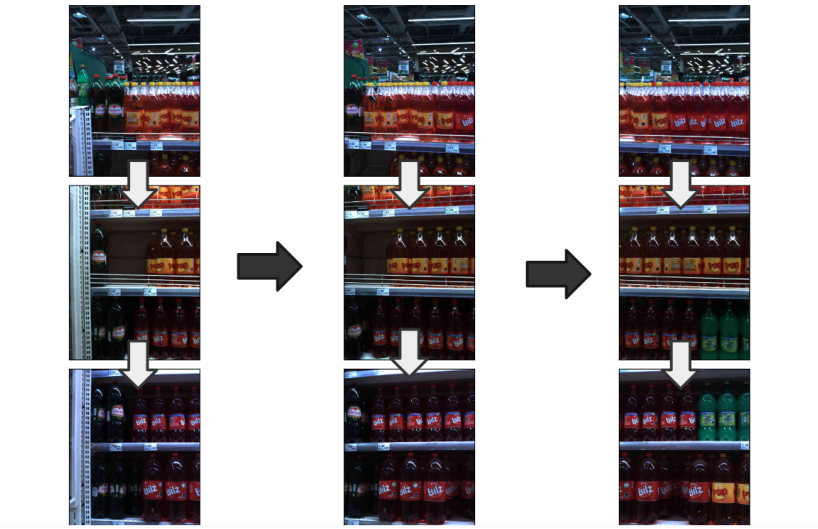


Figura 3.10: Ilustración de la consolidación de imágenes provenientes de tres cámaras diferentes. En la figura se presentan tres filas de imágenes correspondientes a las capturas realizadas por cada una de las cámaras. Las flechas indican el proceso de consolidación, donde cada grupo de tres imágenes se consolida utilizando el algoritmo descrito anteriormente. Posteriormente, se realiza una consolidación adicional para considerar el movimiento del robot y obtener una consolidación final de los objetos detectados en las tres cámaras.

La idea principal de esta solución es obtener una única instancia de cada objeto al consolidar verticalmente las imágenes correspondientes. Luego, se combinan estas consolidaciones verticales con las consolidaciones de las filas laterales. Dado que es el robot el que se mueve, se espera que el vector de movimiento entre las tres imágenes en una instancia y las siguientes sea muy similar. Por lo tanto, se calcula el promedio de los tres vectores de movimiento para obtener un vector representativo.

De esta manera, al consolidar las imágenes verticalmente y luego combinar las filas laterales utilizando el vector de movimiento promedio, se logra una consolidación de las tres fotos de las cámaras.

Problema

Surgieron dificultades al abordar la consolidación entre las cámaras laterales debido al formato en el que se presentan las capturas. Cada cámara proporciona su propio archivo comprimido (.tar) de imágenes, lo que complica el proceso de asociación entre las imágenes capturadas por las diferentes cámaras.

Inicialmente, se asumió que las fotos en la misma posición correspondían a imágenes

tomadas simultáneamente por cada cámara. Además, se suponía que todas las cámaras capturaban la misma cantidad de imágenes. Sin embargo, por razones desconocidas por el equipo de *Software* y *Cloud*, con el que la estudiante trabajó, estas suposiciones iniciales eran incorrectas.

Se encontraron casos en los que una cámara no capturaba una imagen mientras que las otras sí lo hacían, lo que generaba problemas en la consolidación correcta de los objetos. Este problema se ejemplifica en la figura 3.11.

Figura 3.11: Problema de falta de captura en una cámara. La primera imagen muestra el orden incorrecto en el que se leen las imágenes, lo que resulta en una falta de congruencia temporal. La segunda imagen muestra el orden correcto para asegurar la congruencia de las imágenes capturadas por las tres cámaras.

Ante esta situación, se implementó una solución alternativa que se basa en la información disponible actualmente para lograr la consolidación entre las cámaras laterales. No obstante, es importante destacar que en el futuro, si se obtiene información adicional podría volver a revisarse esta implementación.

3.6.2. Consolidación utilizando información proyectada

Se adoptó un enfoque alternativo para abordar el desafío de la consolidación de objetos en múltiples cámaras. En esta nueva implementación, primero se realiza la consolidación de cada cámara con el algoritmo ya descrito. Luego, se combinan los resultados obtenidos de las 3 consolidaciones para obtener una consolidación final.

Para llevar a cabo esta estrategia, se asume que las consolidaciones individuales de cada cámara están correctamente realizadas, lo que implica tener como máximo una instancia de cada objeto por cámara. A partir de estas consolidaciones individuales, se procede a fusionar la información de las tres cámaras.

El proceso de fusión implica iterar a través de las consolidaciones de una cámara y buscar detecciones cercanas en la siguiente cámara utilizando la información proyectada, como las coordenadas de ubicación (location_x, location_y, location_z).

Se establece un radio máximo y se seleccionan las detecciones que se encuentren dentro de este rango. En la figura 3.12 se muestra un ejemplo de detecciones cercanas espacialmente en las secuencias de capturas de dos cámaras.

Las detecciones cercanas se someten a una comparación basada en apariencia utilizando las imágenes de la consolidación. El objetivo es encontrar la detección que presente la mayor cantidad de coincidencias con las imágenes consolidadas. Una vez identificada esta detección, se busca su correspondiente consolidación y se unen las detecciones correspondientes.

Es importante destacar que las detecciones utilizadas se eliminan de las opciones posibles para futuras consolidaciones, evitando así la duplicación de objetos en la consolidación final.

Este enfoque de consolidación basado en la información proyectada ha demostrado ser efectivo para superar la falta de imágenes en algunas cámaras y lograr una consolidación de los objetos entre las cámaras laterales.

Figura 3.12: Ejemplo de detecciones cercanas espacialmente en la secuencia de capturas de dos cámaras. En rojo se visualiza una consolidación de objetos en una secuencia de fotos capturadas por una cámara. En amarillo se marcan las detecciones cercanas a dicho objeto consolidado.

Capítulo 4

Implementación

En este capítulo se describe el proceso de implementación del algoritmo de consolidación de objetos en múltiples cámaras. Se presentan visualizaciones que permiten comprender y analizar los resultados obtenidos. Además, se describe el entorno utilizado para el despliegue del algoritmo, destacando las herramientas y tecnologías empleadas. Por último, se menciona el trabajo realizado en la documentación del algoritmo, asegurando una comprensión clara y completa de su funcionamiento y uso.

4.1. Visualizaciones

Durante el proceso de desarrollo del algoritmo, se utilizaron diversas técnicas de visualización para analizar y evaluar los resultados obtenidos. Estas visualizaciones proporcionaron una representación gráfica de la consolidación de objetos en las secuencias de imágenes.

Una de las visualizaciones más útiles fue el seguimiento visual de un objeto a lo largo de las capturas de una cámara. Esto permitió observar el desplazamiento y la evolución de la detección a medida que se mueve a través de las secuencias de imágenes. Para mejorar la visualización, se agregó un acercamiento que permite distinguir los detalles de la detección. Un ejemplo de este tipo de visualización se muestra en la Figura 4.1.

La visualización anterior se amplió para abarcar el caso de tres cámaras, sin embargo, debido a la gran cantidad de imágenes involucradas, se volvió complicado mostrar los acercamientos individuales para cada detección. Un ejemplo de esta visualización se muestra en la Figura 4.2.

Además, se desarrolló una visualización que permite observar todas las consolidaciones simultáneamente en un determinado sector del pasillo. Esta visualización utiliza colores y números para representar las consolidaciones, lo cual facilita la identificación de las detecciones correspondientes a un mismo objeto. Se optó por utilizar tanto colores como números para representar la misma variable, ya que esto simplifica la identificación visual de las consolidaciones en las imágenes. Un ejemplo de esta visualización se muestra en la Figura 4.3.

Figura 4.1: Visualización del seguimiento y consolidación de un objeto a través de una secuencia de imágenes. En el título de las imágenes se muestra el nombre de la captura y la detección correspondiente marcada. Las imágenes de la segunda fila muestran un acercamiento de la detección para permitir una mejor visualización de los detalles.

Además, la empresa tiene un algoritmo que genera una imagen única para las cámaras laterales, denominada "Muro", mediante un *stitching* (ver Glosario) de las capturas de las tres cámaras. Esta imagen compuesta proporciona una vista panorámica de todo el pasillo y se puede utiliza como referencia visual para el análisis y la visualización de las consolidaciones. En la Figura 4.4, se muestra un ejemplo de la visualización de las consolidaciones superpuestas en la imagen del "Muro".

Las visualizaciones desempeñaron un papel fundamental en el proceso de validación y análisis de los resultados durante el proceso de implementación, proporcionando una comprensión intuitiva y visual de la consolidación de objetos.

4.2. Despliegue

Para el despliegue del algoritmo de consolidación de objetos se utilizó Google Cloud Run. Con el objetivo de disponibilizar el servicio como una API accesible para procesar los datos de los pasillos. A continuación se detallan los aspectos clave de esta implementación:

El algoritmo fue integrado en una API desarrollada con Flask, utilizando el *endpoint* "job" para recibir las solicitudes con los datos del pasillo a procesar.

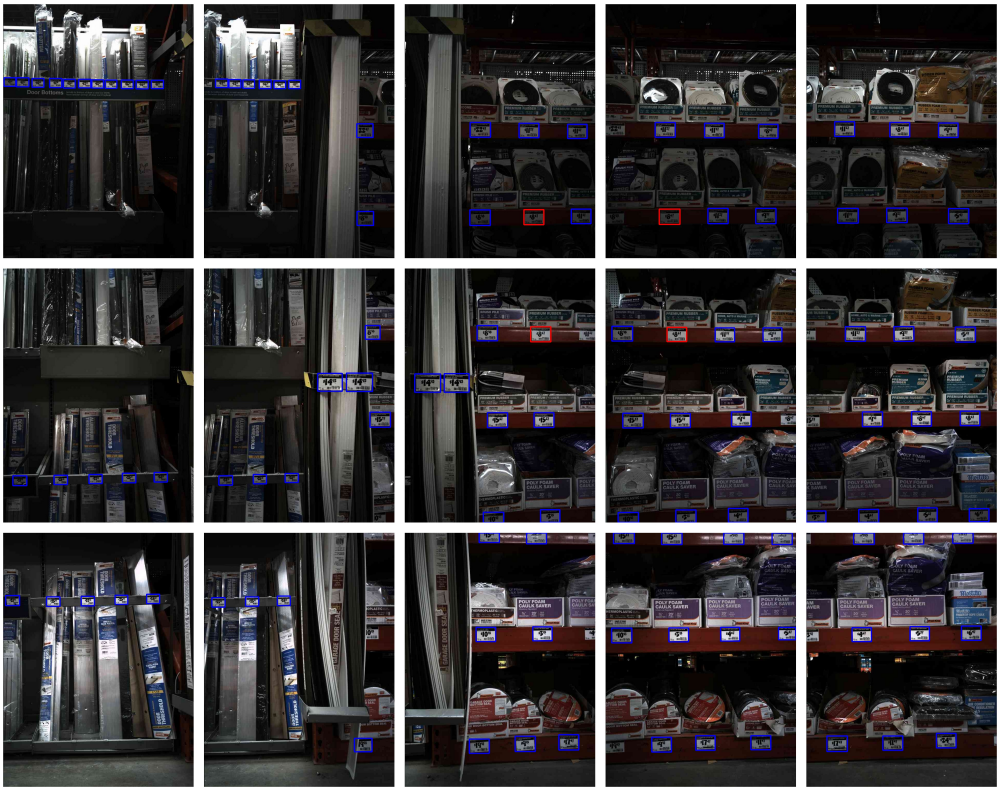


Figura 4.2: Visualización del seguimiento y consolidación de un objeto a través de tres secuencias de imágenes capturadas por cámaras paralelas. Las detecciones de cada captura se muestran en azul, mientras que el seguimiento del objeto de estudio se destaca en rojo. Además, se incluyen imágenes adyacentes para proporcionar contexto adicional del pasillo.

Figura 4.3: Visualización de un sector del pasillo que muestra el seguimiento simultáneo de varios objetos. Los objetos consolidados se identifican mediante el uso de números y colores correspondientes. Los objetos con el mismo número y color están consolidados, lo que indica que corresponden a una misma entidad a lo largo de las secuencias de imágenes capturadas.

Figura 4.4: Visualización de la consolidación en una imagen de muro. En rectángulos de color verde se muestran todas las detecciones proyectadas en el muro, mientras que en círculos de color azul se representa el objeto consolidado, cuya posición es el promedio de las posiciones de las detecciones consolidadas. Además, se incluyen estrellas rojas que representan la consolidación antigua para efectos de comparación de resultados.

En cuanto al despliegue, se aprovechó el uso de los *templates* de CI/CD proporcionados por la empresa, lo que facilitó en gran medida el proceso. Se realizaron las configuraciones necesarias para adaptar el algoritmo y utilizar las herramientas y servicios proporcionados por el *template*.

La implementación en Google Cloud Run permitió aprovechar la escalabilidad y flexibilidad de este servicio de contenedores sin servidor. Esto garantiza que las instancias se creen automáticamente para evitar sobrecargas en la máquina. Se asignaron recursos apropiados al microservicio, incluyendo 4GiB de memoria y 2 CPU, para asegurar un procesamiento eficiente de las imágenes y una respuesta rápida a las solicitudes. Además, se realizaron ajustes en la configuración basados en registros y análisis de los *logs* de Cloud Run, asegurándose de asignar la memoria necesaria para el correcto funcionamiento del algoritmo.

El manejo de la concurrencia fue otro aspecto importante a considerar. Se estableció una configuración con 80 *requests* concurrentes por instancia, lo que permitió un procesamiento eficiente y un mayor rendimiento del servicio. Además, se implementó una cola para reintentar la consolidación en caso de que ocurrieran fallas, asegurando que no se perdieran datos y se brindara una mayor robustez al sistema.

4.3. Documentación del código

La documentación del código se llevó a cabo siguiendo los estándares establecidos por la empresa. Se prestó especial atención a asegurar que el código estuviera completamente documentado en inglés.

Además, se proporcionó documentación detallada para cada una de las funciones y clases utilizadas, describiendo sus entradas, salidas y propósito. Esto facilita la comprensión y el mantenimiento del código, permitiendo una colaboración eficiente entre los miembros del

equipo de desarrollo.

También se incluyó la especificación de los tipos de datos para cada uno de los parámetros de entrada y el valor de retorno de las funciones. Un ejemplo del formato utilizado en la documentación del código se muestra en la Figura 4.5.

```python
def cut_detection(img: np.ndarray, x1: float, x2: float, y1: float, y2: float,
                  WIDTH_IMAGE: int, HEIGHT_IMAGE: int) -> np.ndarray:
    """Cut the detection from the image based on the coordinates and a margin.

    Args:
        img : An image in the form of a numpy array (read with cv2.imread).
        x1, x2, y1, y2 : Coordinates of the detection.
        WIDTH_IMAGE : Width of the image.
        HEIGHT_IMAGE : Height of the image.

    Returns:
        np.ndarray: Cropped image.
    """

    margin = settings.DETECTION_MARGIN

    x1 = int(x1 - margin) if (x1 - margin > 0) else 0
    x2 = int(x2 + margin) if (x2 + margin < WIDTH_IMAGE) else WIDTH_IMAGE
    y1 = int(y1 - margin) if (y1 - margin > 0) else 0
    y2 = int(y2 + margin) if (y2 + margin < HEIGHT_IMAGE) else HEIGHT_IMAGE

    return img[y1:y2, x1:x2]
```

Figura 4.5: Ejemplo del formato utilizado en la documentación del código. El código se encuentra debidamente documentado, especificando las entradas, salidas y propósito de la función.

4.4. Pruebas unitarias

Con el objetivo de garantizar que el código funcione correctamente y se mantenga libre de errores en el futuro, se llevó a cabo un proceso de *testing*. Se utilizó la biblioteca "pytest" junto con "coverage" para realizar las pruebas y evaluar la cobertura del código. Además, se utilizaron datos simulados, conocidos como "mockdata", para simular diferentes escenarios y casos de uso. Este enfoque permitió validar el comportamiento del código en diversas situaciones y asegurar su correcto funcionamiento.

El resultado obtenido fue un *coverage* del 80 % para el repositorio, lo que indica que la mayoría de las líneas de código fueron ejecutadas y evaluadas durante el proceso de *testing*. Este alto nivel de cobertura es un indicador positivo de la calidad del código y proporciona confianza en su rendimiento y fiabilidad.

4.5. Documentación del repositorio

La documentación del repositorio se realizó pensando en facilitar el entendimiento y la colaboración de futuros miembros del equipo. Se creó un archivo README en formato

Markdown, que proporciona una descripción detallada del funcionamiento del repositorio y sus componentes.

En el README se explica en detalle el algoritmo implementado, incluyendo información sobre las entradas que requiere y los salidas que produce. Además, se proporciona una descripción de los servicios con los que interactúa, detallando cómo se integra en el *pipeline* de microservicios de la empresa. También se mencionan las bases de datos a las que accede y cómo se gestionan los datos.

Para facilitar la ejecución de pruebas en entornos locales, se incluyen instrucciones claras sobre cómo ejecutar los *tests* y verificar el correcto funcionamiento del código. Por último, se mencionan las librerías y dependencias utilizadas en el proyecto, lo que permite tener una visión general de las tecnologías empleadas.

Capítulo 5

Resultados y Análisis

En este capítulo se presentarán los resultados obtenidos por el algoritmo de consolidación de objetos. Se abordarán dos aspectos fundamentales: la efectividad del algoritmo en términos de la precisión de la consolidación y su capacidad para manejar diferentes escenarios, y la eficiencia en cuanto al tiempo de procesamiento y los recursos computacionales utilizados.

5.1. Efectividad

En esta sección se evaluará la efectividad del algoritmo de consolidación de objetos. Se abordarán dos enfoques principales: la evaluación utilizando *Ground Truth* y el análisis de los "items" únicos y las detecciones únicas.

5.1.1. Evaluación utilizando Ground Truth

Una forma de evaluar las consolidaciones fue creando un *Ground Truth* específico para un sector del pasillo en cuestión. Para ello, se desarrolló un programa en Python que utilizó las consolidaciones generadas por el algoritmo como punto de partida. Esto permitió crear un *Ground Truth* con la información precisa y real sobre cómo deberían ser las consolidaciones en ese sector.

Generación del Ground Truth

El *Ground Truth* se creó mediante un proceso de comparación y asignación de las detecciones del algoritmo con las detecciones reales. Se evaluó un pasillo de la cadena Home Depot, específicamente las capturas de la 15 a la 45 tomadas por la cámara overhead. El número real de objetos presentes en ese sector, según el *Ground Truth*, era de 166.

Comparación de Resultados

Para evaluar las consolidaciones generadas, se eligió el objeto del *Ground Truth* que tenga la mayor cantidad de detecciones iguales con la consolidación. A partir de esta información, se procedió a comparar los resultados y clasificar las detecciones de la siguiente manera:

- Si la detección está presente tanto en la consolidación como en el *Ground Truth* → Verdaderos Positivos (TP).
- Si la detección está presente en la consolidación pero no en el *Ground Truth* → Falsos Positivos (FP).
- Si la detección está presente en el *Ground Truth* pero no en la consolidación → Falsos Negativos (FN).

Las consolidaciones generadas por el antiguo consolidador y el nuevo consolidador fueron comparadas con el *Ground Truth*. Los resultados obtenidos se presentan en la tabla 5.1.

Algoritmo	Accuracy	Misclassification	Precision	Recall
Consolidation	0.3	0.69	0.98	0.3
New Consolidation	0.86	0.13	0.99	0.87
Variación porcentual	186.67 %	-81.16 %	1.02 %	190.00 %

Tabla 5.1: Métricas obtenidas al comparar el *Ground Truth* con el antiguo y nuevo consolidador para el caso de la cámara overhead.

Análisis

La evaluación utilizando el *Ground Truth* permitió analizar y comparar la efectividad de los consolidadores. Se observa que el nuevo consolidador logra una mayor precisión y una menor clasificación errónea en comparación con el antiguo consolidador. Además, se identificó un mayor número de consolidaciones perfectas con el nuevo consolidador, lo que indica una mejora en la calidad de las consolidaciones.

Cabe destacar que el nuevo consolidador no utiliza las lecturas para la consolidación, lo que le permite consolidar un mayor número de detecciones en comparación con el antiguo consolidador. En este sector del pasillo, el nuevo consolidador logró consolidar 207 detecciones adicionales en comparación con el antiguo consolidador.

Estos resultados respaldan la efectividad y la mejora significativa en la calidad de las consolidaciones obtenidas con el nuevo algoritmo.

5.1.2. Items únicos

Una métrica adicional utilizada para evaluar la efectividad del algoritmo es el análisis de los "items" únicos y las detecciones únicas antes y después de implementar el algoritmo en producción. Esta métrica es similar a IDF1, ya que mide la cantidad de objetos únicos presentes en una escena.

Los "items" se refieren a los códigos de productos leídos en las detecciones. Durante la consolidación, las detecciones que corresponden al mismo producto son agrupadas en una única entidad. Por lo tanto, al utilizar el nuevo algoritmo de consolidación, la cantidad de "items" no debería cambiar significativamente. Sin embargo, se espera que haya una disminución en la cantidad de detecciones únicas, lo que indica que se está logrando una mayor consolidación de objetos.

Para evaluar esta métrica, se realizó un análisis comparativo antes y después de la implementación del algoritmo, como se muestra en la Figura 5.1. Esta comparación revela la disminución en la cantidad de detecciones únicas después de aplicar el nuevo algoritmo de consolidación. Esto indica que el algoritmo está logrando agrupar de manera efectiva las detecciones que corresponden al mismo producto, reduciendo la cantidad de detecciones aisladas y mejorando la consolidación en general.

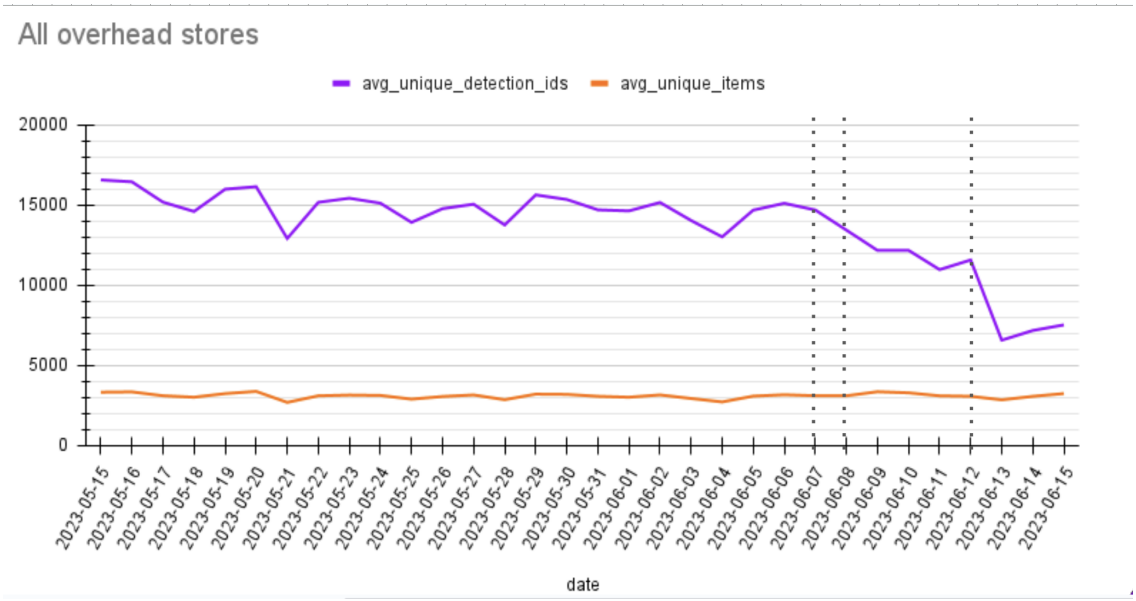


Figura 5.1: Comparación de la cantidad promedio de detecciones únicas (en morado) y la cantidad promedio de items únicos (en naranja) antes y después de la implementación del nuevo algoritmo de consolidación. Las líneas punteadas verticales marcan las fechas en las que se realizaron los despliegues del nuevo algoritmo, que fueron gradualmente para diferentes tiendas. La disminución en la cantidad de detecciones únicas después de la implementación indica una mejora en la consolidación de objetos, al agrupar de manera efectiva las detecciones que corresponden al mismo producto.

5.2. Eficiencia

Para evaluar la eficiencia del algoritmo, se consideraron dos aspectos fundamentales: el tiempo de ejecución y el costo asociado. Estos dos factores son críticos para determinar la viabilidad del algoritmo en un entorno práctico. El tiempo de ejecución refleja la velocidad con la que se realiza la consolidación de objetos, mientras que el costo está relacionado tanto con el tiempo como con los recursos computacionales utilizados durante el proceso. Ambos aspectos son importantes para garantizar un rendimiento óptimo y una utilización eficiente de los recursos disponibles.

Para el cálculo de estas métricas, se tomó en cuenta el tiempo promedio que tomó al algoritmo consolidar cada pasillo. Este cálculo se realizó enviando solicitudes al microservicio en modo *Shadow-mode*, lo que implica que se enviaron solicitudes como si estuviera en producción, pero los resultados no se guardaron en las bases de datos productivas de la empresa. De esta manera, también fue posible calcular el costo del algoritmo en Google Cloud Run, ya que se utilizó de la misma manera que en un escenario real. Esta metodología garantizó una evaluación precisa del tiempo de ejecución y el costo asociado, reflejando las condiciones y los recursos reales utilizados en la implementación práctica del algoritmo.

5.2.1. Tiempo de procesamiento

En la figura 5.2 se muestra una comparación del tiempo de ejecución entre la implementación anterior y la nueva implementación del algoritmo de consolidación de objetos. Los resultados evidencian un incremento en el tiempo de procesamiento con la nueva implementación. Esto se debe a que el proceso de consolidación se realiza de manera más exhaustiva y se realizan más cálculos y comparaciones en cada paso. Sin embargo, a pesar de este aumento en el tiempo de ejecución, al haber movido el microservicio en el *pipeline*, es viable tener este tiempo de procesamiento.

5.2.2. Costo del algoritmo

En la figura 5.3 se presenta una comparación del costo asociado a la nueva implementación del algoritmo. El costo está relacionado con los recursos computacionales utilizados durante el proceso de consolidación, como el consumo de CPU y memoria, y el tiempo de procesamiento del algoritmo. Los resultados muestran un incremento en el costo en comparación con la implementación anterior. Esto se debe a que la nueva implementación requiere más recursos para realizar las operaciones de consolidación de manera más completa y precisa. Además, como vimos en la sección de tiempo de ejecución, el algoritmo toma más tiempo en obtener la consolidación.

Finalmente, tras analizar los datos a lo largo de una semana, se evidenció un aumento del $118,11\,\%$ en los costos y un incremento del $95,85\,\%$ en los tiempos. Esta evolución se tradujo en una notable mejora del algoritmo, con un incremento del $186,67\,\%$ en la precisión y una disminución del $81,15\,\%$ en la tasa de clasificación incorrecta.

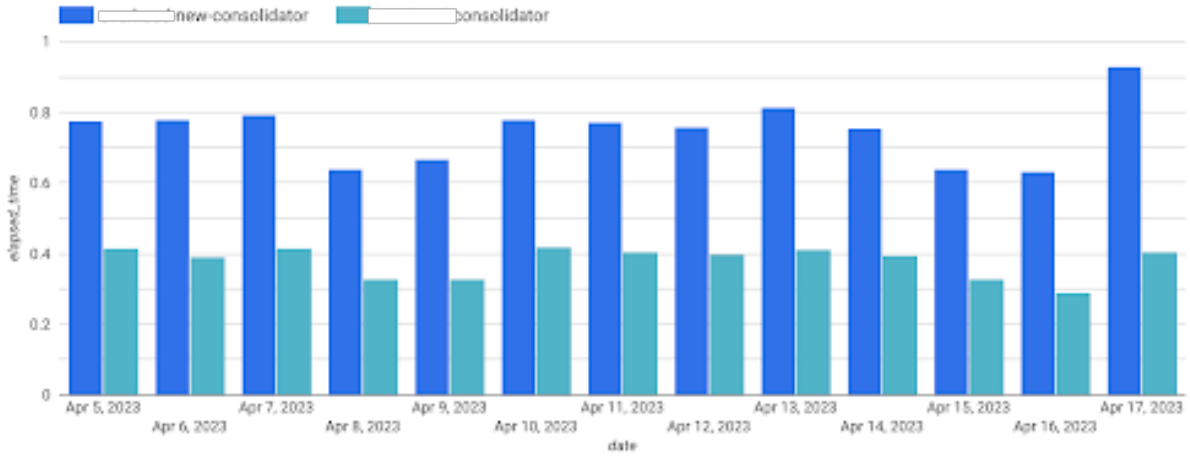


Figura 5.2: Comparación del tiempo de ejecución entre la implementación anterior y la nueva implementación del algoritmo de consolidación de objetos. En azul se muestra el tiempo de la nueva implementación, y en calipso se muestra el tiempo de la implementación anterior.

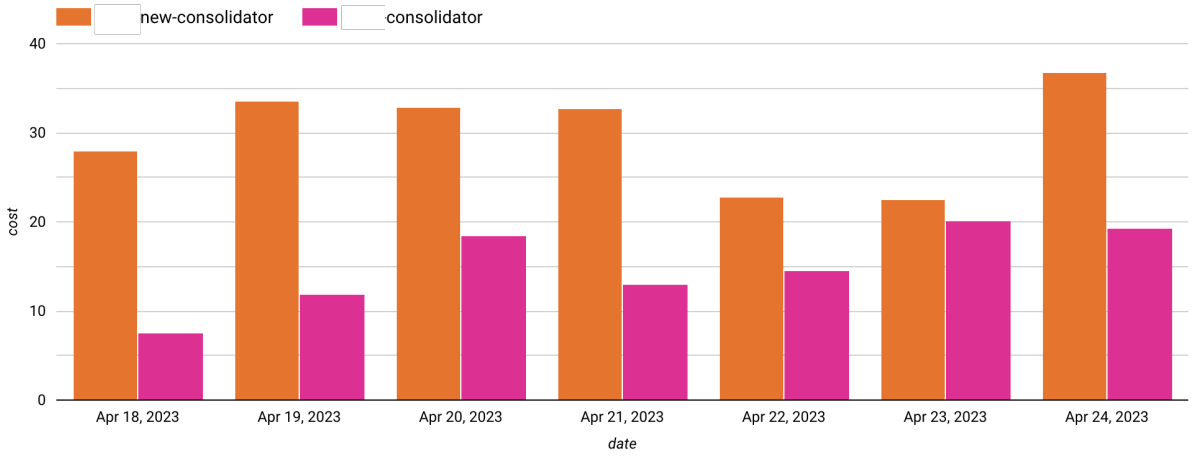


Figura 5.3: Comparación del costo asociado a la nueva implementación del algoritmo de consolidación de objetos. En naranja se muestra el costo de la nueva implementación, y en rosa se muestra el costo de la implementación anterior.

Capítulo 6

Discusión

En esta sección, se analizarán y discutirán los resultados obtenidos en el estudio del algoritmo de consolidación de objetos. Se explorarán las implicaciones de los resultados, se identificarán las limitaciones del estudio y se propondrán posibles direcciones para futuras investigaciones.

6.1. Interpretación de los resultados

En términos de eficiencia, se observó un aumento en los tiempos de procesamiento al implementar el nuevo algoritmo de consolidación. Sin embargo, este incremento en el tiempo se justifica por la naturaleza más exhaustiva y precisa del algoritmo, así como por su movimiento de posición en el *pipeline* de ejecución. Al estar ejecutándose en paralelo con la lectura de los flejes, es aceptable que el algoritmo tome más tiempo, ya que se logra un equilibrio entre el rendimiento y la calidad de los resultados.

En cuanto al costo asociado, se reconoce que el nuevo algoritmo implica un mayor costo debido a la mayor cantidad de recursos computacionales requeridos para las operaciones de consolidación, en conjunto con el mayor tiempo de ejecución ya mencionado.

Sin embargo, tanto el aumento en tiempo como en el costo están justificado por la mejora en la calidad y precisión de las consolidaciones obtenidas. La empresa valora la calidad del servicio y la satisfacción del cliente, por lo que considera que este aumento en el costo es una inversión que ayuda a mantener la excelencia en sus operaciones.

Además, en comparación con el algoritmo anteriormente desarrollado, el nuevo algoritmo es más flexible para consolidar detecciones sin la necesidad de lecturas. Esto lo hace aplicable a una variedad de casos, como la consolidación de productos, bandejas u otros objetos detectados en diferentes capturas. Esto amplía el alcance de utilidad del algoritmo y lo hace potencialmente aplicable a otros problemas de consolidación en el futuro.

Dado que se calcularon métricas y se anotó un *Ground Truth* solo el caso de la cámara overhead, se disponibilizó el algoritmo como un microservicio en producción específicamente

para ser usado en ese caso. A pesar de los costos adicionales, la mejora en el rendimiento y los beneficios obtenidos hacen que la decisión de implementar este nuevo enfoque sea justificada y valiosa para la empresa.

Sin embargo, en el caso de las cámaras laterales, no se obtuvieron métricas que demostraran un mejor funcionamiento del nuevo algoritmo. Además, debido a los altos costos asociados, la empresa decidió no implementar esta nueva solución en producción para las cámaras laterales.

6.2. Limitaciones identificadas

Durante el desarrollo del algoritmo, se identificaron algunas limitaciones y desafíos significativos. Uno de ellos fue la falta de información precisa sobre la posición de las imágenes capturadas. Esta limitación complicó el proceso de consolidación entre múltiples cámaras, ya que no se contaba con datos exactos sobre la ubicación de las capturas. En su lugar, fue necesario realizar estimaciones, lo que llevó a la presencia de errores asociados. Esta situación resalta la complejidad de trabajar con múltiples equipos y la necesidad de adaptarse a las limitaciones y tiempos establecidos para cada tarea.

El uso de la información proyectada para consolidar las detecciones de las cámaras laterales se ha presentado como una limitación en zonas de alta densidad de flejes. En estas áreas, las proyecciones suelen ser de menor calidad, lo que lleva a realizar una mayor cantidad de comparaciones de apariencia. Además, se debe elegir un radio generalizado para estas comparaciones. Las Figuras 6.1 y 6.2 muestra una zona con una alta densidad de flejes, donde se requirió comparar numerosas detecciones.

Además, el nuevo algoritmo se implementó en producción exclusivamente para la cámara overhead, ya que no se pudo completar su evaluación para las tres cámaras laterales. Esto ha resultado en la necesidad de mantener dos códigos distintos y deja un área pendiente de investigación y desarrollo futuro.

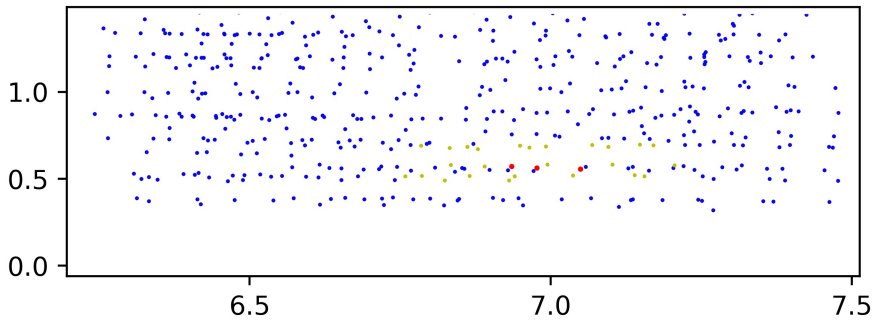


Figura 6.1: Los ejes X e Y representan las posiciones en la góndola en metros. Se muestran las detecciones indicadas en la Figura 6.2. Se observa que las posiciones proyectadas son bastante dispersas para objetos que son los mismos. Esta dispersión en las proyecciones resalta la necesidad de investigar enfoques más precisos y adaptables para mejorar la consolidación en áreas de alta densidad de objetos.

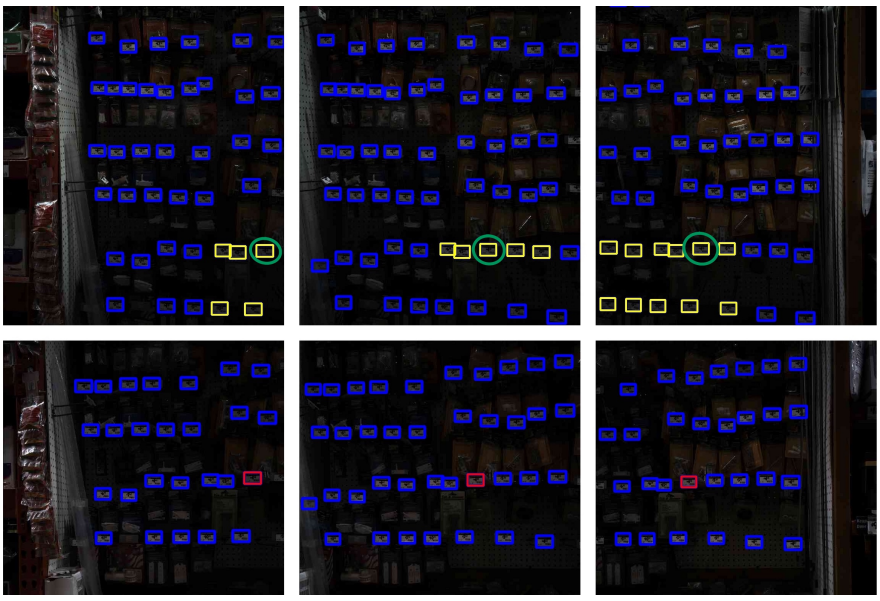


Figura 6.2: Ejemplo de un sector de pasillo con alta densidad de flejes. Se muestra la consolidación realizada por la cámara inferior en color rojo, las detecciones cercanas espacialmente a la consolidación en color amarillo, y las detecciones que se deberían consolidar en color verde. Es evidente que las detecciones cercanas son numerosas, lo que implica una gran cantidad de comparaciones necesarias. Además, se observa que las proyecciones no son siempre iguales para un mismo objeto, ya que en la captura central las detecciones que están más abajo del objeto correcto no quedan dentro del radio de búsqueda, mientras que en las otras capturas sí lo hacen.

6.3. Trabajo futuro

Como trabajo futuro, se propone evaluar el desempeño y la efectividad de la solución de consolidación de objetos utilizando tres cámaras laterales. Actualmente, los resultados disponibles son solo a través de visualizaciones y no se han calculado métricas asociadas. Sería interesante llevar a cabo un análisis cuantitativo y comparativo con el algoritmo actual implementado.

Para mejorar aún más el algoritmo, se sugiere explorar la coherencia espacial en la consolidación. Esto implica verificar si la relación espacial entre las detecciones se mantiene en la consolidación, lo que podría ayudar a mejorar la precisión y calidad de los resultados.

Además de utilizar otros algoritmos de detección de *keypoints*, una posible línea de exploración sería analizar cómo cambian los resultados si se utilizan las imágenes recortadas por el robot, que presentan una mayor resolución. Sin embargo, se debe tener en cuenta que actualmente es posible seleccionar cuánto margen dar a las detecciones para disponer de información visual sobre los alrededores de la captura. Utilizando la información recortada del robot, se perdería esta funcionalidad. Por lo tanto, es importante considerar cuidadosamente los *trade-offs* entre la resolución de las imágenes y la información contextual proporcionada por el margen en las detecciones.

Por otro lado, es importante reconocer que aunque este estudio ha estado enfocado en

la consolidación de detecciones de flejes y cajas, su aplicabilidad no se encuentra limitada exclusivamente a estas categorías. Por lo que se propone como trabajo futuro evaluar el algoritmo con diferentes tipos de detecciones, como pueden ser de productos. Este análisis permitirá discernir si el enfoque desarrollado conserva su efectividad en diferentes escenarios, abriendo así una oportunidad para expandir su utilidad en variados contextos.

En resumen, el nuevo algoritmo de consolidación de objetos ha demostrado ser efectivo y mejorar la calidad de las consolidaciones obtenidas, especialmente en el contexto de overhead. A pesar de algunas limitaciones y desafíos identificados, existen oportunidades prometedoras para continuar mejorando y ampliando el alcance de aplicación del algoritmo en el futuro.

Capítulo 7

Conclusión

Tras finalizar este proyecto, se ha logrado un avance significativo en la consolidación de objetos en una secuencia de imágenes, así como en la consolidación de información entre múltiples secuencias de imágenes paralelas. Esto ha permitido determinar qué objetos corresponden a una misma entidad en cada secuencia, mejorando la calidad y utilidad de los datos obtenidos.

Aunque el tiempo de ejecución se ha incrementado, este aumento se puede tolerar al ubicar el proceso en una etapa adecuada del *pipeline* de microservicios de la empresa. En términos de costos, si bien se ha observado un incremento, se debe destacar la mejora significativa en los resultados obtenidos. Ahora, se logra consolidar detecciones incluso en ausencia de lecturas, lo que brinda mayor versatilidad y aplicabilidad en diversos contextos. Además, se ha logrado una consolidación más efectiva, reduciendo el número total de objetos por pasillo y generando resultados más precisos y confiables.

Este enfoque innovador permite que las máquinas complementen el trabajo logístico humano, brindando un apoyo eficiente en el entorno de la tienda. La reducción de duplicados en la información proporcionada promueve una mayor confiabilidad y eficacia en las tareas de los trabajadores, optimizando su tiempo y mejorando la eficiencia general del proceso.

Bibliografía

[1] K. Bernardin and K. Mikolajczyk. Evaluation of multiple object tracking performance: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(10):1938–1945, 2008.

[2] K. Bernardin and K. Mikolajczyk. Multiple object tracking (mot) benchmark, 2009.

[3] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3464–3468. IEEE, 2016.

[4] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[5] Chris Harris, Mike Stephens, et al. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Citeseer, 1988.

[6] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003.

[7] Rudolf E. Kalman. A new approach to linear filtering and prediction problems. *Journal of basic engineering*, 82(1):35–45, 1960.

[8] Renu Khandelwal. Evaluation metrics for multiple object tracking. *Medium*, 2021.

[9] Harold W. Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[10] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[11] Tao Luo, Zaifeng Shi, and Pumeng Wang. Robust and efficient corner detector using non-corners exclusion. *Applied Sciences*, 10(2):443, 2020.

[12] P. C. Mahalanobis. On the generalized distance in statistics. *Proceedings of the National Institute of Sciences of India*, 2(1):49–55, 1936.

[13] Anton Milan, Laura Leal-Taixé, Ian Reid, Stefan Roth, and Konrad Schindler. Mot16: A benchmark for multi-object tracking. *arXiv preprint arXiv:1603.00831*, 2016.

[14] Hans Moravec. Obstacle avoidance and navigation in the real world by a seeing robot rover. *Machine vision for three-dimensional scenes*, 8(2):156–165, 1980.

[15] Ergys Ristani, Francesco Solera, Roger Zou, Rita Cucchiara, and Carlo Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. In *European Conference on Computer Vision (ECCV)*, 2016.

[16] Irwin E. Sobel. Gradient magnitude thinning by the sobel method. *Machine vision for three-dimensional scenes*, 28(2):136–148, 1973.

[17] Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. Loftr: Detector-free local feature matching with transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8922–8931, 2021.

[18] Richard Szeliski. Computer vision: algorithms and applications. *Springer Nature*, 2nd ed.(3.10):103, 2009.

[19] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 3645–3649. IEEE, 2017.

Anexo

Glosario

- **Bounding-box** : Rectángulo que representa la ubicación que ocupa una detección.

- **Fleje**: Etiquetas que contienen el nombre, SKU (Unidad de Control de Stock, por sus siglas en inglés) y precio de un producto. Los flejes son capturados por las cámaras laterales y overhead de los robots de Zippedi en las tiendas de retail.
  - Los flejes capturados por las cámaras laterales (ver Figura 1) muestran el nombre del producto, SKU y precio asociado. Estas cámaras capturan imágenes de las estanterías desde una perspectiva lateral.
  - Los flejes capturados por la cámara overhead (ver Figura 2) están principalmente relacionados con cajas de reabastecimiento de productos. Estos flejes pueden contener un solo SKU o múltiples SKU asociados a los productos almacenados en las cajas. La cámara overhead toma imágenes desde una perspectiva inferior, ya que el robot es mas bajo que las estanterías.

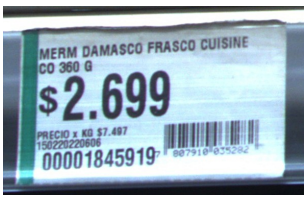
- **Ground Truth** : Referencia confiable utilizada para evaluar y validar los resultados de un algoritmo o modelo en el procesamiento de datos o imágenes.

- **Góndolas** : Estanterías dispuestas por toda la tienda, las cuales albergan los productos en exhibición.

- **SKU** : *Stock Keeping Unit*, número de identificación de un producto.

- **Stitching** : Técnica de combinar múltiples imágenes superpuestas para formar una imagen panorámica más amplia y continua.

(a) Fleje The Home Depot

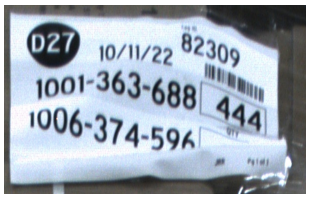


(b) Fleje Jumbo

Figura 1: Ejemplos de flejes capturados por cámaras laterales. Se caracterizan por tener nombre del producto, SKU y precio asociado.

(a) Fleje con un SKU

(b) Fleje con SKU dividido en 2 líneas



(c) Fleje con múltiples SKU

Figura 2: Ejemplos de flejes capturados por cámara overhead. Se caracterizan por tener uno o más SKUs asociados a los productos que se encuentran al interior de las cajas.