UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Modelador de objetos 3D para el robot GOLEM III

TESIS

Que para obtener el título de

Ingeniero Mecatrónico

P R E S E N T A

Renato Antonio Romero Fonseca

DIRECTOR DE TESIS

Dr. Gibrán Fuentes Pineda

Ciudad Universitaria, Cd. Mx., 2017

ÍNDICE GENERAL

ÍNDICE DE TABLAS

CAPÍTULO 4 PÁG

ÍNDICE DE FIGURAS

CAPÍTULO 1. PÁG.

CAPÍTULO 2.

CAPÍTULO 3.

CAPÍTULO 4.

RESUMEN.

En el presente trabajo se tiene como objetivo establecer y desarrollar una propuesta de modelado de objetos y estimación de pose de los mismos como utilidad en la investigación de la inteligencia artificial y la robótica de servicios.

El trabajo está dividido en 5 capítulos los cuales son: Introducción, antecedentes, el modelado de objetos 3D con Kinect Fusion y librerías de OpenCV y OpenGL, así como la propuesta planteada como solución de visión para el robot Golem III, experimentos con metodología y resultados y, finalmente, conclusiones y propuestas de mejora.

En la introducción planteé varios de los conceptos básicos que abordó este trabajo, desde la robótica general, robótica de servicios, MOPED[1], y el robot Golem III. Una vez definidos estos conceptos generales el trabajo abordará el planteamiento del problema; las mecánicas de la estimación de pose y modelado de objetos en la actualidad y el problema científico de modelar objetos para el reconocimiento. Seguidamente de los objetivos de la presente tesis: El desarrollar un método para modelar objetos que sirva para el reconocimiento de objetos MOPED de forma rápida y eficiente.

La justificación: La importancia que supone un reconocedor de objetos en la robótica de servicios actual y a su futuro desarrollo. La necesidad de contar con buenos modelos, robustos y que se obtengan de forma rápida y sencilla.

A continuación se piensa en la metodología a seguir, en este caso se utilizó Kinect como medio de censar el medio y obtener información, que después es procesada mediante librerías de visión por computadora, en particular se utilizan los algoritmos de SIFT[2] , y FLANN[3] .

Como último subcapítulo de éste primer capítulo introductorio habrá un resumen de cada capítulo siguiente visto de forma más detallada, los cuales serán: El capítulo 2, donde se abordan los antecedentes del modelado y reconocimiento de objetos, estrategias, diferencias conceptuales así como ventajas y desventajas de las distintas metodologías. En el capítulo 3 se abordará Kinect, específicamente Kinect Fusion, y el sistema propuesto para modelar los objetos.

En el capítulo 4 se presentarán la serie de experimentos, pruebas y maneras en que se evaluó el rendimiento del sistema propuesto, se mostrará una cierta cantidad de modelos, cada uno con condiciones distintas de evaluación acuerdo a las características del modelador. Se reportarán los resultados de estos experimentos.

Finalmente en el capítulo 5 se abordarán las conclusiones de los resultados obtenidos en el capítulo 4, así como los alcances y limitaciones del sistema propuesto.

---

[1] MOPED: Por sus siglas en inglés, Multiple Object Pose Estimation and Detection; Estimación de la posición de los objetos y detección de los mismos, un conjunto de técnicas y estrategias basadas en algoritmos que son capaces de detectar, estimar y cuantificar características visuales.

[2] SIFT: Por sus siglas en inglés: Scale Invariant Feature Transform; Algoritmo que detecta características visuales invariantes de la escala en la que se mira.

[3] FLANN: Por sus siglas en inglés: Fast Library for Approximate Nearest Neighbors; Librería rápida para vecinos aproximados más cercanos. Busca las características similares en los puntos capturados.

CAPÍTULO 1.

1 – Introducción.

1.1 – Contexto del problema.

1.1.1 – Robótica.

¿Qué es la robótica?

A lo largo del tiempo los seres humanos nos hemos sumergido en una búsqueda eterna por comprender y mejorar el entorno en que nos encontramos, saber que hay más allá de los límites existentes, siempre buscamos facilitarnos las cosas o solucionar problemáticas en la rutina del día a día, ya sea alimento, transporte, entretenimiento, industria, inclusive la trascendencia. Es ahí donde entran los avances en la ciencia y la tecnología, a lo largo de todos los ramos industriales, logradas por incontables científicos e investigadores. Este desarrollo involucró procesos optimizados e innovadores que, muchas veces han sido posibles gracias sólo a dos conceptos: La robótica y su ente de aplicación, los robots. [1]

Pero a todo esto, ¿Qué es la robótica?

La robótica es el conjunto de conocimientos de la electrónica, mecánica, la computación y el diseño para el empleo de dispositivos que faciliten y/o sustituyan el trabajo de las personas.

La robótica dentro de la ingeniería, la industria y el entorno doméstico tiene un infinito número de aplicaciones desde las que va la educación, con softwares y aplicaciones inteligentes que fungen como traductores, maestros, guías geográficos, buscadores entre otros, a la industria textil donde tenemos máquinas auto programables como trituradoras de materia prima, hiladores, utilización de láseres para lograr acabados de cortes más precisos y limpios, entre muchas otras, llegando hasta los últimos automóviles capaces de conducirse por sí mismos y evitar accidentes mediante sensores inteligentes, control mediante la voz o las señas del usuario, las aplicaciones de la robótica son ilimitadas en nuestro mundo cada vez más industrializado.

De algo que podemos estar seguros es que la robótica es una importante técnica que nos ayudará como especie a alcanzar un nuevo nivel de calidad de vida, a mejorar los procesos de producción, abastecimiento, transporte y servicios, lo cual, nos lleva a nuestra siguiente definición; la robótica de servicios.

1.1.2 – Robótica de servicios.

La robótica de servicios valga la redundancia, es la robótica que está enfocada a brindar alguna clase de servicio; la federación internacional de robótica (IFR), ha propuesto una definición tentativa: Todo robot de servicio es aquel que ayuda a los seres humanos a realizar tareas o trabajos de manera parcial o completamente autónoma. Y excluye a todo robot destinado a tareas de tipo industrial. Muchos robots de servicio se encargan de tareas domésticas o rutinarias como limpieza, mantenimiento, inspección, seguridad, recepcionistas, cuidadores, defensa, mayordomos, medicina, bomberos, logística, rescate, relaciones públicas, construcción y demolición e incluso aspectos formativos como la enseñanza y la investigación, entre otros.

Dentro de los exponentes de la robótica de servicio están: DIY Drones, Aldebaran Robotics, Beijing Li Pu Electric Co., Ltd., BiOM Personal Bionics, Bionik Labs, Cyberworks Robotics Inc, Exact Dynamics, Freedom Innovations, Intelligent Motion GmbH, G2 Inventions, Fluidra, entre muchos otros sólo por mencionar unos pocos.

Fig. 1.1 - Robot móvil de vigilancia para interiores. Tomado de: [16]

Fig. 1.2 - Robot cortador de césped. Tomado de: [16]

Fig. 1.3 - Robot de supervisión doméstica. Tomado de: [16]

1.1.3 – Robot Golem III.

Dentro de Grupo Golem en el Instituto de Investigación en Matemáticas Aplicadas y en Sistemas de la UNAM hay una investigación que se centra en el modelado cognitivo de la interacción entre humanos y sistemas computacionales, se trabaja con aplicaciones fijas, sistemas de diálogo y móviles, y entre éstos, también, en robots de servicio.

En la práctica dentro de algunos módulos especializados en Golem se encuentran: Navegación, manipulación, audio, procesamiento del lenguaje y el discurso, control y procesamiento de señales, planeación y coordinación, aprendizaje máquina, interacción humano-robot, inteligencia artificial y probablemente el que más le atañe al presente trabajo; Visión.

En el área de visión en Golem actualmente se tienen algoritmos en inteligencia artificial y otros que utilizan librerías de visión por computadora con el objetivo de modelar, reconocer y manipular objetos, mediante información de características visuales aportadas mediante el escaneo del medio. Uno de los principales problemas que presenta la visión por computadora es el problema de la estimación de pose y el modelado de objetos mediante uno o múltiples puntos de vista.

Fig. 1.4 - Robot Golem en Torneo Mexicano de Robótica 2013. Tomado de: [12]

1.1.4 – MOPED.

MOPED, Multiple Object Pose Estimation and Detection. Por sus siglas en inglés detección y estimación de pose de múltiples objetos, es un aglomerado que integra reconocimiento y estimación de posición de objetos ya sea en una imagen singular o múltiples imágenes, en un único, optimizado, robusto y escalable marco de referencia. En la visión por computadora se tienen dos problemáticas principales: Tener un rendimiento robusto en escenas complejas, y un bajo estado latente para operaciones en tiempo real. MOPED alcanza un rendimiento robusto mediante iterativamente hacer una estimación por agrupamiento (ICE), un algoritmo que iterativamente combina agrupación de características visuales con una robusta estimación de la posición de las mismas. El agrupamiento de características rápidamente hace una partición de la escena y produce una hipótesis del objeto. La hipótesis se utiliza para afinar las agrupaciones de características, y así, los dos pasos iteran una y otra vez hasta alcanzar la convergencia. ICE es fácil de utilizar en paralelo con otros procesos y fácilmente integra estimación de la posición junto con reconocimiento de objetos desde una o múltiples cámaras.

Con MOPED se alcanza una escalabilidad y bajo estado latente con un mejorado algoritmo de emparejamiento de características para grandes bases de datos. El emparejamiento se logra buscando características que sean similares a pesar de ser vistas desde distintas perspectivas o posiciones de cámara.

La meta que se fija con MOPED es el reconocimiento de objetos de ciertas imágenes dadas, contando con una base de datos de modelos de objetos. De este conjunto de imágenes dadas al reconocedor, se obtienen sus características, donde se obtiene un vector de coordenadas en el espacio tridimensional de cada punto en cuestión anexo a un vector de descriptores correspondiente, a la unión de todos estos puntos (coordenadas y descriptores) se le hace un emparejamiento, también llamado "*matching*" donde se busca que las características obtenidas de estas imágenes coincidan con aquellas del modelo previamente guardado en la base de datos.

De esta forma mediante un proceso iterativo de convergencia se determina el reconocimiento del objeto en cuestión.

1.2 – Planteamiento del problema.

El problema científico del reconocimiento de objetos surge por la necesidad de ubicar y cuantificar las características visuales de los objetos en todo momento, siendo un robot de servicio autónomo capaz de identificar y reconocer cosas de su entorno, con el fin de manipularlas y realizar tareas.

Para que un robot sea capaz de reconocer, necesita modelos previamente guardados en su memoria con los cuales comparar para discernir que es qué, para generar estos modelos, el proceso general es el de obtener puntos clave, que contengan tanto las coordenadas de la posición y orientación de los objetos, como un descriptor correspondiente que defina las características visuales del punto en cuestión.

El problema también yace en el ser capaces de identificar estos puntos desde distintos ángulos de visión, por lo que necesitamos de alguna clase de algoritmo que nos empareje los puntos que hemos ido obteniendo a lo largo de diferentes instantes de tiempo. En un entorno complejo no controlado, el robot debe ser capaz de reconocer cualquier objeto (al menos en un caso ideal) sino es que la mayor cantidad de objetos, sin importar en qué lugar se encuentre, y también necesita un modelo preciso del tamaño y forma de cada cosa para ser capaz de agarrarlas, moverlas o utilizarlas.

MOPED nos hace el favor de solucionarnos estos problemas con ayuda de algoritmos de extracción de características invariantes con la escala para almacenarlas como puntos clave, llámese SIFT o SURF. Y de algoritmos de emparejamiento de esas mismas características como FLANN o "*Brute Force"*. En cuanto al problema de qué tan preciso es el reconocimiento o la estimación de pose, así como qué tan robusto es el modelado contra el ruido externo, resulta que tenemos un 98% de éxito en reconocimiento con múltiples imágenes o vistas de distintas cámaras, mientras que tenemos un 91% de éxito en reconocimiento con una única imagen o vista, esto de acuerdo a experimentos controlados con más de 2mil objetos llevados a cabo entre MOPED y HERB. [20]

Lo que nos plantea lo siguiente: ¿Cuáles son las limitaciones actuales y problemas que presenta MOPED en el modelado de objetos?
En el modelado actual de MOPED los problemas que encontramos van relacionados al rendimiento de reconocimiento, el cual está ultimadamente atado a encontrar suficientes características locales en un objeto dado. Si un objeto no cuenta con la textura suficiente, se localiza demasiado lejos como para obtener características nítidas y confiables, o tiene cualidades visuales como transparencia o reflejo de

la luz, los pasos obligados de extracción y emparejamiento de características puede que no encuentren suficientes correspondencias para llevar a cabo cualquier tipo de proceso de reconocimiento.

De acuerdo con el artículo oficial de MOPED, se necesita de un mínimo de 8 a 10 correspondencias necesarias para reconocer un objeto exitosamente y estimar su posición. Un problema actual del reconocimiento en MOPED es el de la necesidad de potenciar la habilidad en una escena o entorno de crear características o texturas que sean de ayuda para los objetos que carecen de ésta. [7]

Una problemática adicional en MOPED que surge en el reconocimiento de objetos basado en modelos, es la etapa de construcción del modelo en sí, puesto que a pesar de que es en su mayor parte automático, sigue necesitando de cierta supervisión humana, y es necesario escalar correctamente los objetos para lograr una estimación de la posición de los mismos desde una misma perspectiva.

Con el tiempo esto se ha ido mejorando gracias a la implementación de obtención de información vía láser, sensores infrarrojos o cámaras RGB-D (que detectan color y profundidad), dando así una mejor estimación de la posición de las características.

De igual forma se recomienda encarecidamente hacer múltiples pruebas a la hora de generar un nuevo modelo para la base de datos del reconocedor, ya que en el caso donde se modele usando una sola cámara y un método de promedio de la posición, sin utilizar la información de la profundidad real, esta se obtendría mediante una estimación dada la escala del modelo, y en estos casos 1mm de error en el tamaño de una lata de soda por ejemplo, se convertiría en un error en la estimación de la profundidad de hasta 3cm a una distancia de 1m. Lo que puede ocasionar un problema con el manipulador.

1.3 – Objetivos.

Como objetivos para el presente trabajo, se debe desarrollar un modelador que sirva para el reconocimiento de objetos MOPED, el sistema debe ser rápido, eficiente, práctico y se debe buscar que obtenga los mejores resultados posibles en cualquier escenario, o en su defecto, en el mayor número de escenarios posibles. Los modelos que se obtengan deben ser robustos y con una alta incidencia de reconocimiento, que funcione y sea compatible con la tecnología actual de Grupo Golem desarrollada en el IIMAS de la UNAM, se debe usar la mejor alternativa tanto en algoritmos de obtención de características, llámese SURF, SIFT, ORB, entre otros. Al igual que en algoritmos de emparejamiento como son Brute Force y FLANN por mencionar ejemplos.

Se debe de utilizar la última tecnología en estimación de posición, la cual debe ser dinámica y compatible con el resto del trabajo realizado por las otras sub divisiones de Grupo Golem, es decir, el modelador debe de encajar bien con los sistemas de reconocimiento que ya están incorporados en Golem y también debe ser práctico y fácil de utilizar en conjunto con los sistemas de manipulación, coordinación y planeación.

La información obtenida con el modelador debe ser confiable, y tener alta flexibilidad y disponibilidad para su manejo. Como otro de los objetivos del presente trabajo el modelador final deberá ser capaz de obtener los modelos en cuestión de forma rápida y con un bajo estado latente de máquina, que procese de manera eficaz sin malgastar los recursos de memoria del CPU. El modelador debe ser

modular para que resulte práctica su incorporación a un robot de servicio autónomo que realice múltiples tareas durante tiempos cortos o prolongados.

1.4 – Justificación.

El modelador de objetos 3D para el robot Golem III es importante no sólo para sus operaciones de visión y reconocimiento, de inteligencia y trato con el usuario, sino que es una parte crucial que impacta indirectamente el resto de los módulos de Golem, desde el hecho de que si el robot es incapaz de ver y reconocer los objetos, éste no puede manipularlos correctamente, sincronizar y planear sus tareas para ser capaz de realizar operaciones manuales y mecánicas, de igual forma, aunado al reconocimiento, también están los procesos de navegación y atención a determinados patrones o símbolos que podrían activar respuestas en el robot.

Así mismo, siendo que el reconocimiento es tan importante para la robótica de servicio, más aún para un robot del tipo mayordomo o mesero como en el caso de Golem III, el modelador cobra aún mayor importancia, ya que es debido a éste último que el reconocedor es capaz de sopesar características y posicionamientos encontrados en el medio con aquellos guardados en su base de información que le fueron facilitados por el modelador.

1.5 – *Metodología*.

La metodología que se siguió en el presente trabajo consistió en hacer uso de sensores de posición y profundidad, así como cámaras RGB (información del color en la imagen) para la obtención de la datos útiles del medio, en conjunto con algoritmos de librerías de visión por computadora como SIFT y FLANN para obtener características descriptivas de los puntos observados, así como emparejarlos en distintos instantes de tiempo (desde distintas perspectivas de vista).

De manera que se filtren todos aquellos puntos que no se repitan a lo largo de un conjunto de distintos ángulos de mira, esto otorgará la robustez necesaria al modelador para minimizar la incidencia de error en el reconocimiento.

En el caso del sensor se utilizó el Kinect SDK V2.0 de Microsoft Windows. Debido a su alta portabilidad y dinamismo. El módulo de Kinect Fusion fué particularmente útil en la obtención de puntos 2D en cada imagen a color que son traducidos a puntos 3D en el espacio de profundidad. Se utilizaron las funciones del SDK de Kinect para transformar cada punto clave al espacio de la cámara y después al de coordenadas globales con una referencia común.

Todas las distintas capturas fueron procesadas con SIFT para la obtención de puntos clave en la escena y con FLANN para asegurar que sólo se tomen en cuenta aquellos puntos más robustos.

Finalmente el sistema entrega un arreglo de los puntos filtrados que cuenten con un mínimo de repeticiones a lo largo de distintas vistas, cada punto cuenta con la información de coordenadas en el espacio XYZ tridimensional así como su vector descriptor que indica sus características visuales, todos los puntos con su origen de sistema de referencia ubicado en el centro del objeto, este conjunto final de puntos es el que será la entrada del reconocedor de objetos.

1.6 – Organización del presente trabajo.

En los posteriores capítulos de ésta tesis se tratarán los antecedentes, modelado con Kinect Fusion y propuesta del sistema de modelado, experimentos y conclusiones.

Empezando por el capítulo dos, los antecedentes, se buscará explicar el reconocimiento de instancias de objetos y su diferencia con el reconocimiento de categorías, se ahondará en las estrategias basadas en la detección de características globales y locales, ventajas y desventajas de cada una y la estimación de posición de los objetos basándonos en cada una. Se tratará a detalle el reconocimiento de objetos con MOPED, el modelado de objetos 3D como se hace actualmente en MOPED y otras estrategias de modelado.

En este capítulo igualmente se plantea explicar los distintos algoritmos de detección de características en los objetos, ventajas y desventajas de cada uno y su eficiencia en relación a tiempo de ejecución, entre otras especificaciones. De igual manera se tratarán los distintos algoritmos de emparejamiento o *"matching"* entre distintos instantes de tiempo y ángulos de mira de las capturas a los diferentes objetos, ventajas y desventajas de los métodos, así como su tiempo de respuesta en ejecución, rendimiento y porcentaje de acierto.

En el capítulo tres, modelado con Kinect Fusion y el desarrollo de la propuesta planteada, se hablará acerca de las ventajas y desventajas de modelar con el sensor Kinect, sus características, distintos tipos de funcionalidades, sensores y cuáles de éstas nos atañen a nosotros. Se explicará a detalle el proceso de modelado con Kinect Fusion, en todas sus etapas desde la obtención de la información de color y profundidad y su procesamiento al espacio de la cámara del sensor, la conversión de mapeo de profundidad obteniendo información de vértices y vectores normales en cada punto de la superficie escaneada, el seguimiento de la posición de la cámara; Estimación de pose con seis grados de libertad, la integración al volumen de reconstrucción y la obtención de vistas al volumen de reconstrucción mediante "*raycasting"* [4] al objeto de reconstrucción para generar una nube de puntos que nos sirva para producir imágenes de salida de tipo superficie, normales y a color.

[4] *Raycasting*: Es una técnica para solucionar problemas de computación gráfica y geométrica, consiste en "disparar" un rayo desde la perspectiva del observador y obtener información de distancia y posición del objeto gráfico analizado mediante intersecciones rayo-superficie.

Factores que involucran el rastreo o *"tracking"* [5], limitaciones del rastreo, el volumen de reconstrucción y algunos ejemplos básicos de su funcionamiento.

Los sistemas de referencia en el espacio local, global, del volumen y sus matrices de transformación de un espacio a otro, conversiones de los cuadros en el espacio de profundidad al de color o al de la cámara y viceversa.

En el último subcapítulo se verá la propuesta de modelador utilizando Kinect Fusion en conjunción con algoritmos de SIFT y FLANN para obtener una mejor eficiencia y robustez del sistema. También se utilizará Qt para producir una interfaz gráfica que haga más cómodo y dinámico el proceso de obtención de los modelos.

En el cuarto capítulo se puso en tela de juicio el rendimiento y operatividad del modelador mediante distintos experimentos, enfocados en variar la cantidad de capturas, de emparejamientos o *"matchings"*, la cantidad de puntos obtenidos en cada modelo, para distintos tipos de objetos y cuál es su incidencia de error o acierto en el reconocedor de Golem, finalmente se hizo un conglomerado de gráficas y presentación de resultados que nos dan un panorama general de los alcances y limitaciones del sistema propuesto, también de las condiciones específicas, si las hay, para que se puedan obtener mejores y más robustos modelos, las condiciones para obtener modelos de la manera más rápida y dinámica. Así como la conjunción de ambas; Rendimiento y rapidez para lograr una eficiencia óptima en todos los casos.

En el capítulo quinto y final del presente trabajo presentaré las conclusiones que obtuve del análisis de resultados del capítulo cuatro así como mis observaciones personales para que este trabajo pueda mejorarse y en qué situaciones convendrían o no hacer dichas mejoras, de igual manera describiré los puntos débiles y fuertes del sistema y bajo qué circunstancias pueden optimizarse y eliminarse respectivamente, describiré el impacto que este modelador tiene sobre Golem y en las áreas de visión, inteligencia artificial y la robótica de servicios.

[5] *Tracking*: Hace referencia al rastreo de la posición de la cámara con respecto a las coordenadas globales de la escena, el tracking se hace de manera iterativa para lograr una mejor convergencia de los puntos y como se interrelacionan unos con otros.

CAPÍTULO 2.
2 – Antecedentes.

2.1 – Reconocimiento de instancias y categorías de objetos.

2.1.1 – Conceptualización y Definiciones.

En el estudio del reconocimiento de objetos para diversas aplicaciones en robótica de servicios, un robot debe ser capaz de percibir su entorno en aglomeraciones no estructuradas de un conjunto no definido de objetos, con distintas variaciones de formas superficiales, texturas, tamaños, distancias y posiciones.

En el reconocimiento efectivo de objetos en interiores, se requiere de la habilidad para percibir tanto las categorías como las instancias en los mismos. En los sistemas actuales en reconocimiento de objetos, sigue sin ser predominante la existencia de aquellos que sean capaces de combinar ambos; Instancias específicas altamente texturizadas, y objetos genéricos sin textura en una escena desordenada. Un sistema que, implementándose con éxito, establece las raíces para una manipulación e interacción confiable y robusta en ambientes domésticos.

De estos dos tipos, el primero, el reconocimiento de las categorías, se basa en todas aquellas propiedades de forma y características globales de los objetos, el que tengan forma de platos, de ollas, de tarros, de botellas, entre otras por nombrar algunos ejemplos. Se basa en obtener descriptores de forma y contornos de los objetos, así como extracción de características en la textura muy generales. Mientras que el reconocimiento de características locales, también llamado de instancias de objetos, se enfoca en obtener marcas y señas discriminativas locales en la textura superficial de los mismos. Segmentar el objeto en parches para obtener textura por regiones. Tales como identificar una marca o un logo corporativo en la etiqueta de algunos productos, nombres comunes, combinaciones de colores específicos, entre otras.

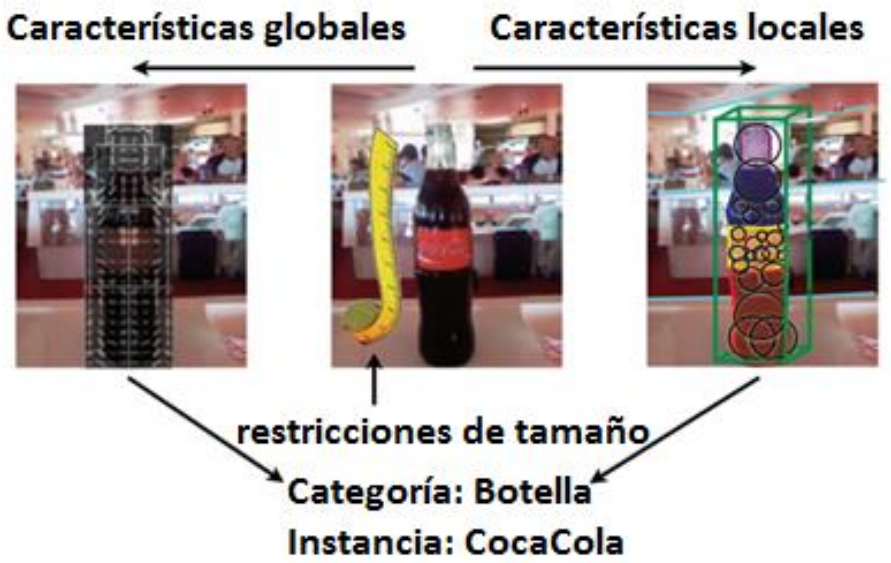

Fig. 2.1 – Reconocimiento de una botella de refresco utilizando características globales como contorno y forma (izquierda), y utilizando características locales como son parches por regiones en el objeto para obtener texturas y marcas discriminativas (derecha). En este caso la categoría es la botella y la instancia es la Coca-Cola. Traducida de: [6]

De igual manera, podemos definir a las características globales y locales por la manera en que se hacen sus descriptores para una comparación posterior, un descriptor no es sino un vector de determinados bin que actúa como un contenedor de números que describen iluminación, color, intensidad, textura, entre otras características inherentes a una imagen o a una región de la imagen, que puede ir desde un solo pixel hasta un conjunto de estos delineando una forma o contorno específico.

Los descriptores globales definirán la imagen en su totalidad con un único vector, generalmente de gran longitud, mientras que los descriptores locales, actuarán como un conjunto de vectores descriptores, cada uno anexo a una región o localidad particular de la imagen.

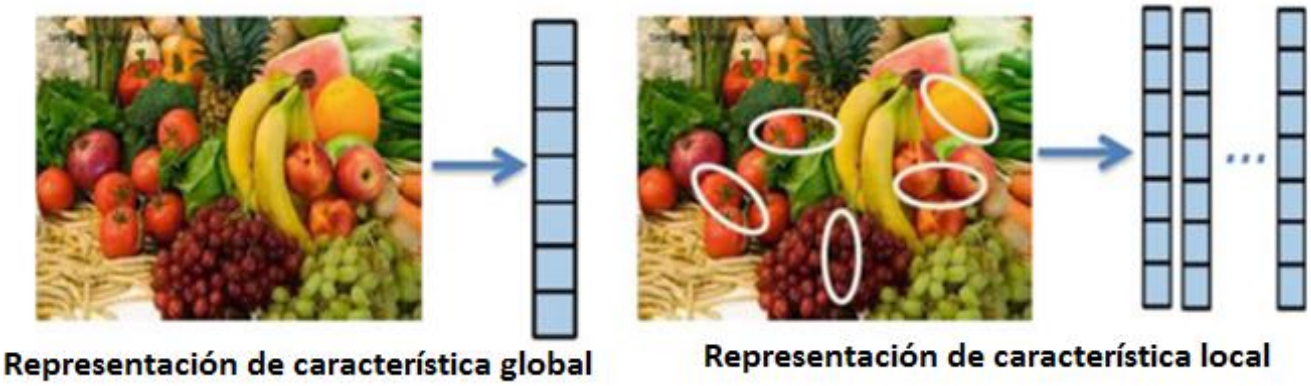

Fig. 2.2 – Representación mediante descriptores para características globales y locales. Traducida de: [30]

2.1.2 – Ventajas, desventajas, ejemplos y características.

Entre las principales estrategias de reconocimiento en características globales tenemos el HOG (Histograma de Gradientes Orientados), Co-HOG, matrices de forma y momentos invariantes como Hu y Zerinke. En cuanto a estrategias para reconocimiento de características locales algunos ejemplos son: SIFT, SURF, ORB, LBP, BRISK, MSER y FREAK.

Para crear algo de perspectiva, un ejemplo representativo de descriptor de características globales, el histograma de gradientes orientados (HOG), consiste en obtener una serie de vectores gradientes en las esquinas y bordes de los objetos para obtener las magnitudes de sus valores de inclinación y el sentido de los mismos. Lo cual ayuda a determinar y clasificar formas de los objetos en regiones planas y curvas. De igual manera que SIFT, los bordes y las esquinas siempre brindarán mucha información en relación a características tanto locales como globales.

Hablando de SIFT, como buen ejemplificador de los algoritmos para obtener características locales, nos ayuda a identificar características valga la redundancia, que son invariantes con la rotación y la escala, proveen de un emparejamiento robusto a lo largo de un rango afín de distorsión, cambios en la perspectiva 3D, adición de ruido y cambios de iluminación. Consiste en cuatro pasos principales que son: (1)Identificación de todos aquellos candidatos a puntos clave que son potencialmente invariantes con la escala y la orientación; (2)Una vez establecidos los candidatos se obtiene escala y ubicación de cada uno; (3)Se les asigna una o más orientaciones basadas en la ubicación de direcciones gradientes en la imagen local, esto último ayuda a que todas las futuras transformaciones sean invariables y tengan cierta estabilidad, y finalmente; (4)Se le asigna un descriptor a cada punto clave basándonos en los gradientes de la imagen cuantificados en la región alrededor del punto. Se discutirá más a fondo el algoritmo de SIFT en capítulos posteriores.

La aproximación más común en el reconocimiento de objetos basado en características locales es dividir el proceso en 2 partes; Extracción de las mismas y matching o emparejamiento con las de un modelo previamente guardado para confirmar la identificación del objeto. Si la plantilla utilizada como modelo no sirve para lograr un match directo, podría ser útil la implementación de plantillas en espacios característicos, aquellas que detallan el objeto a ser identificado en distintas condiciones como perspectivas variantes, iluminación, contrastes de color o "posiciones" aceptables para hacer el "*matching"*. Por ejemplo si en el "*matching"* se buscaba una cara, entonces el espacio característico consistirá de una base de datos de caras en distintas posiciones y condiciones de iluminación, etc.

En la detección y localización de objetos la aproximación más común es emplear algoritmos de escaneo por ventana, que consiste en pasar una ventana a lo largo de la imagen obtenida y tratar cada panel individual como si contuviera al objetivo o al entorno. Algunas variantes de esta estrategia es pasar una ventana clasificadora, que detecten partes específicas del objeto y después volver a unirlas para ensamblar un conjunto de ventanas que representan al objeto en su totalidad. Otra forma es simplemente extraer características de puntos clave en la imagen y luego establecer un perímetro alrededor de cada punto y clasificar estos perímetros en lugar de tratar con todas las posibles sub-ventanas. Sin embargo, como ya mencionamos, esto conlleva que si no hay suficientes características locales en la imagen, la información es insuficiente, pero puede respaldarse mediante echar un vistazo a partes de la imagen que se encuentren fuera de los parches asignados, llámense sub-ventanas o perímetros alrededor de puntos clave. Esto sería, ver la imagen como un todo, características globales.

Fig. 2.3 - En la imagen se puede apreciar el caso donde diferentes características
locales pueden ser interpretadas como la misma a pesar de estar en contextos distintos,
en estos casos, es conveniente mirar la imagen como un todo y obtener también características globales. Tomado de: [2]

Entre algunas de las ventajas y desventajas de ambos tipos de reconocimiento, encontramos que en las características locales, al ser sumamente dependientes de las regiones específicas a las que se miran, si la escala es demasiado pequeña, la perspectiva de mira hace que el objeto sea ofusco o ambiguo, y si al contrario, se aumenta demasiado la escala del objeto tal que las esquinas dejan de aparecer como esquinas, una zona que antes podría ser rica en marcas discriminativas ahora se convierte en una región

homogénea con parcial o nula aparición de distinciones. Sin embargo esta situación puede verse significativamente mejorada si se implementa un reconocimiento de características globales en conjunción con las locales. De este modo, en aquellos casos especiales donde se denota ambigüedad en la obtención de características en determinadas regiones, el objeto seguirá siendo identificable al aplicar un algoritmo como el de HOG donde adicionalmente se puede saber la forma y deformaciones de la zona en cuestión, esto aunado a la pose del objeto. Aunque el usar un algoritmo así en conjunto puede ocasionar problemas de rendimiento e insuficiencia de memoria.

Sin embargo las características locales presentan la ventaja de poder identificar y reconocer objetos y otorgan la posibilidad de construir aplicaciones a un nivel alto y robusto, mientras que por el otro lado del espectro, las características globales brindan sólo la posibilidad de detectar y clasificar el objeto, es decir; Decidir si el objeto en cuestión existe o no en la escena y clasificarlo de una manera primitiva de acuerdo a su figura y tamaño. Lo que nos funciona para múltiples aplicaciones de bajo nivel.

En cuanto a la estimación de la posición de un objeto mediante características globales, se obtiene mediante una diferenciación clara del objeto-objetivo del resto de su entorno, fácilmente identificable por su contorno, forma y tamaño, características de su modelo 3D como gradientes en bordes y esquinas, mientras que en el caso de las características locales, la posición de cada punto clave, sub-ventana y/o región de interés, se analiza en conjunto con una estimación de las coordenadas en 3D de cada uno de los puntos a partir de su imagen en 2D y sensores de profundidad. Muchas veces cada punto en el espacio correspondiendo a 1 pixel de la imagen.

Debido a que no es tan obvio el obtener la ubicación del objeto o su diferenciación del resto de la escena sólo a partir de sus descriptores locales (ya que el mismo conjunto de características locales puede ser encontrado fuera del objeto de interés y en otras locaciones de la misma imagen observada), para el modelado de los objetos, se deben establecer ciertos filtros, ya sea en la misma escena (limitando lo que se ve en la misma a sólo el objeto de interés) o mediante un algoritmo de reconocimiento conjunto al de modelado que sea capaz de filtrar la información no deseada, ya en el reconocimiento bien se puede sólo depender del modelo en si para establecer las pautas del objeto que se busca identificar.

En el caso de contar con múltiples nubes de puntos obtenidas de distintas perspectivas o imágenes, pueden alinearse para converger en un único modelo de referencia mediante correspondencias entre las características, las coordenadas de la posición del objeto pueden mejorarse notablemente mediante este tipo de técnicas, algunas estrategias populares para lograr esto son ICP (Iterative Closest Point) y el algoritmo de Levenberg-Marquardt.

2.2 – Métodos de reconocimiento de características locales y globales.

2.2.1 – Historia.

A lo largo del tiempo han surgido distintas aproximaciones a la solución del problema de reconocimiento de objetos, empezando desde 1965 con el método de Roberts, denominado *"Blocks world".* En éste, la misión era sumamente simplificada reduciéndose a trabajar sólo con poliedros del mismo color, estáticos y con un fondo uniforme. Al ser tantas las limitaciones impuestas, se hicieron avances considerables en el área. El sistema usaba una serie de detectores de bordes y que buscaba emparejar líneas para formular hipótesis de posición que se confirmaban al resolver para proyecciones en la cámara.

Los trabajos de Binford de 1971, sirvieron para extender la investigación de Roberts agregando el reconocimiento de superficies curvas llamadas cilindros generalizados.

En 1985, Lowe, fue uno de los primeros en alcanzar un sistema que conllevara entrenamiento y prueba de reconocimiento, ya utilizando modelos propiamente dichos que serían primitivos agrupamientos de líneas rectas y bordes, cuidando el detalle en la proximidad de los puntos finales de las mismas, la distancia, paralelismo y co-linealidad que las mismas presentaran. Las cuales posteriormente serían comparadas a la geometría del objeto para hallar correspondencias.

Una vez que se empezaron a implementar e intentar acelerar los procesos de emparejamiento y correspondencia entre características, surgieron métodos para este paso específico como interpretación de árboles, RANSAC y agrupamiento de posición. [13]

2.2.2 – Métodos basados en características globales.

Como algunos de los principales expositores de características globales están aquellos basados en momentos invariantes como Hu y Zerinke. Además de las matrices de forma y HOG.

Métodos que consumen una menor cantidad de memoria, son fáciles de compilar en procesos computacionales, son más rápidos y compactos. Sin embargo, no son invariantes a transformaciones importantes y son sensibles a las agrupaciones y la oclusión. En resumen, las características locales son más eficientes que las globales, pero en determinadas aplicaciones la distinción de categorías o detección utilizando características globales nos puede servir para la detección de copias ilegales, que muchas veces sólo han sufrido pequeñas modificaciones como recortes de partes específicas, cambios de escala o han sufrido cierta compresión. También al hacer clasificaciones de bajo nivel, que sólo requieran diferencias sencillas como forma, color o contorno, sin malgastar los recursos de procesamiento y el estado latente.

Generalmente los métodos de características globales usan diferencias en la forma como detección de bordes, histogramas de gradientes, y momentos geométricos invariantes (no ortogonales: Hu, y ortogonales: Zerinke), que en esencia son productos punto o proyecciones de una imagen sobre una base polinomial, para obtener características del objeto o la escena en base a la geometría.

2.2.3 – Métodos basados en características locales.

Ya que la mayoría de métodos basados en características locales suelen dividirse en aquellos que se encargan de detectar y extraer las características del medio en sí, tales como: Detector de esquinas Harris-Stephens y Shi-Tomasi, SIFT, SURF, ORB, BRIEF, LBP, BRISK, MSER y FREAK. Y aquellos que se encargan de hacer un matching o emparejamiento de los puntos extraídos ya sea con plantillas o con otro modelo para verificar la identificación del objeto; FLANN, Brute Force, Template Matching, distancia de Hamming, etc.

En este capítulo se revisan aquellos utilizados en el presente trabajo: SIFT para brindarnos una detección de características robustas e invariables con la escala, más una descripción de cada punto clave encontrado, que me servirá para el "*matching"*, y FLANN que bien no fué utilizado para hacer el reconocimiento entre la imagen y el modelo, (puesto que para esto se utiliza MOPED, el cual se verá en capítulos posteriores), pero si me sirvió mucho para hacer "*matching"* entre capturas tomadas al hacer

el modelo, lo cual nos garantizará modelos más robustos que sólo tomarán en cuenta aquellas características más confiables y visibles desde distintas perspectivas.

2.2.3.1 – Detección y extracción de descriptores: SIFT.

Dentro de los múltiples métodos para extraer características, el que se utilizó en el presente trabajo fué SIFT: Scale Invariant Feature Transform, o transformación de características invariantes con la escala por sus siglas en inglés. Las características extraídas mediante SIFT poseen suficientes propiedades que las hacen apropiadas para la correspondencia entre las mismas a lo largo de múltiples imágenes de un objeto o escena. Se puede extraer una alta cantidad de características SIFT que cubren la imagen de forma densa, sin perder en el sentido de que estas sean lo suficientemente robustas para ser identificables en distintas condiciones, lo que las hace perfectas para nuestro modelador de objetos, ya que proveen una base sólida para el reconocimiento de objetos.

El costo de extracción es minimizado aplicando un filtro de cascada para que sólo aquellas características con un costo más elevado de procesamiento y operaciones se obtengan sólo si pasan un filtro inicial. En una resolución de 500x500 pixeles, se tendrán alrededor de 2000 características estables, aunque estos valores variarán dependiendo del contenido mismo de la imagen, de si se desea que se consideren características aisladas, entre otros parámetros.

Al hacer correspondencias entre características SIFT se toma una distancia euclidiana entre los descriptores de cada punto. Generalmente en escenas muy saturadas, los puntos del fondo puede que no tengan ninguna correspondencia si se hace un matching contra una imagen previamente guardada en base de datos.

¿Por qué es importante la invariabilidad con la escala en la detección de características?

En la detección de puntos clave de interés, regiones con grandes cambios de color o textura son muy importantes. Uno de estos son las esquinas, puesto que es donde hay puntos de inflexión en color, forma, textura entre muchos otros parámetros. Al igual que sucede en bordes, texto dentro de la imagen o irregularidades como manchas o marcas. Puesto que las esquinas se mantienen como esquinas al cambiar la rotación, el ángulo de mira de una escena, o si la imagen se subdivide en ventanas individuales, pero no se mantienen como esquinas al cambiar la escala.

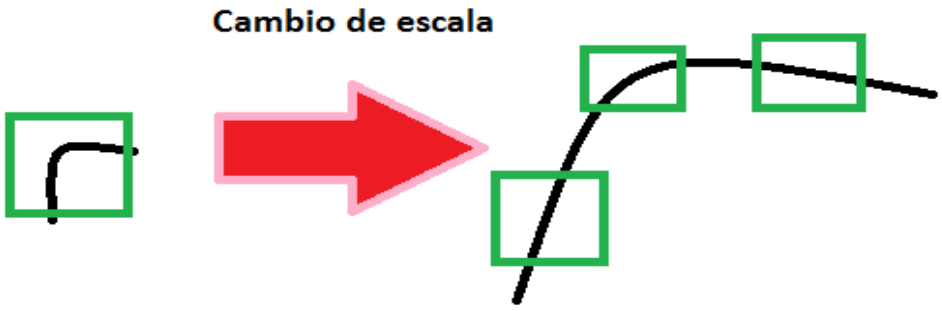

Fig. 2.4 - Demostración de variación en propiedades visuales de una esquina con la escala.

El algoritmo de SIFT consta de cuatro etapas principales:

1. Detección de extrema espacio-escalar.

Se utiliza el Laplaciano de Gaussiano (LoG) para obtener valores σ a lo largo de la imagen, LoG actúa como un detector de manchas de varios tamaños debido al cambio en σ, en otras palabras, σ funciona como un parámetro de la escala, de modo que se obtienen conjuntos (x, y, σ) por cada punto de interés encontrado por LoG en la imagen. Lo que significa que hay un punto clave potencial en la ubicación (x, y) a la escala de σ. Sin embargo debido a que LoG es muy costoso en procesamiento, SIFT utiliza una aproximación usando diferencia de Gaussianos (DoG), esta se obtiene como la diferencia entre desenfoques gaussianos de potenciales puntos con σ s distintas. El proceso se hace por octavas (diferentes rangos en el valor de σ), y ya que se encuentran estas diferencias de Gaussianas se hace un comparativo donde se filtra el valor extremo local, por ejemplo: Se toma un pixel que será comparado con sus 8 vecinos aledaños, sus 9 vecinos de la escala superior y sus 9 vecinos de la escala inferior, si es un extrema local, (es decir un máximo o un mínimo, el cual es importante porque sobresale de la media y esto lo hace más fácilmente identificable), se considera un potencial punto clave. Se menciona que los valores óptimos para realizar este cálculo son:

Número de octavas: 4.
Número de niveles de escala: 5.
Valor inicial para σ: 1.6.
El valor de k, donde cada diferencia Gaussiana será dada por σ y kσ: $\sqrt{2}$.

Véase [8] y [18].

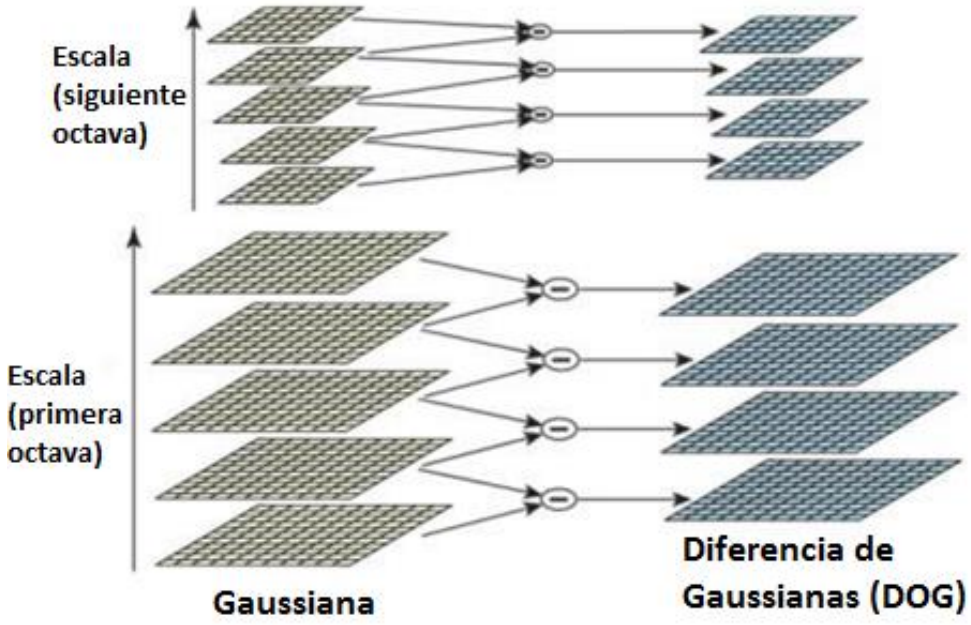

Fig. 2.5 - Pirámide Gaussiana donde se obtienen las DoG. Traducida de: [8]

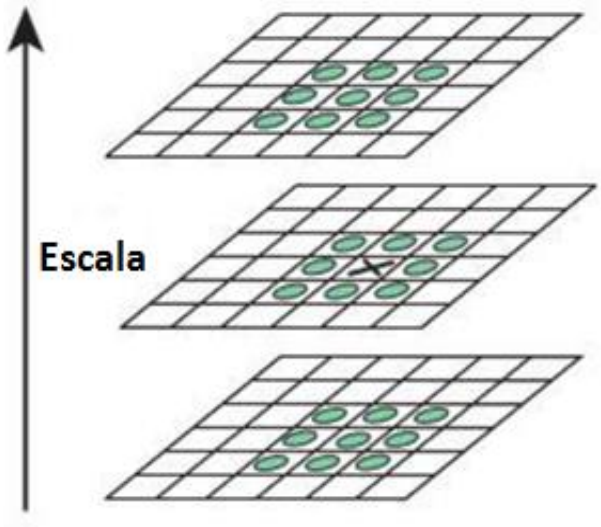

Fig. 2.6 - Comparativo entre un pixel con sus vecinos de la misma escala así como la inmediata inferior e inmediata superior para obtener el extrema local. Traducida de: [8]

2. Localización de puntos clave.

Una vez que los potenciales puntos clave fueron encontrados se le da una mayor precisión con métodos de expansión espacio-escalar con series de Taylor, se comparan valores de intensidad y si el mismo es menor a 0.03, se desprecia el punto. De igual forma se remueven los bordes ya que tienen alta respuesta en las diferencias Gaussianas.

3. Asignación de la orientación.

Se le asignan orientaciones a cada punto clave para obtener invariaciones con la rotación de la imagen. Lo hace con un gradiente tomado de los puntos circundando el vecindario del punto de interés. La orientación contribuye a que se logren correspondencias más estables.

4. Descriptor del punto clave.

Finalmente se le asigna un vector descriptor de 128 bin a cada punto clave, el cual se forma a partir de 16 sub-bloques tomados del vecindario alrededor de dicho punto, donde cada sub-bloque contiene un histograma de orientación de 8 bin, y subsecuentemente a estos 128 valores bin se les da una representación tipo vector por practicidad. En adición a esto se toman múltiples medidas de cambios de rotación e iluminación entre otros para alcanzar mayor robustez.

El algoritmo de SIFT se encuentra disponible en las librerías de código abierto de visión por computadora de OpenCV que se usará para el desarrollo del modelador. [8]

2.2.3.2 – Emparejamiento o "*matching*" de puntos: FLANN.

Para hacer una correspondencia entre las características encontradas se utilizan múltiples métodos, entre los cuales, los más utilizados suelen ser los de aproximación por vecinos cercanos o ANN,

(*Approximate nearest neighbor*). Cuando se habla del problema de la búsqueda del vecino más cercano en visión por computadora, se habla de encontrar un método que sirva como pre-proceso a un conjunto de puntos: "P" tal que para encontrar el vecino más cercano de un punto en un espacio métrico "M", llámese a dicho punto: "q", y a su vecino más cercano: "NN(q, P) $\in$ P" con respecto a una distancia métrica "d: M x M $\rightarrow \mathbb{R}$", se pueda realizar la operación NN(q, P) de forma eficiente.

FLANN actúa como una librería de selección rápida de algoritmos ANN para que en la amplia cantidad de casos posibles, pueda obtenerse la mejor opción con una optimización de parámetros de forma automática. Muchos de los algoritmos contenidos en FLANN son propuestos por sus autores y otros que ya existen en la literatura han sido mejorados para lograr un mejor rendimiento.

Entre los principales tipos de algoritmos de búsqueda del vecino más cercano se encuentran los arboles de particiones, técnicas de división o "hashing" y técnicas gráficas de vecindarios.

Los árboles de particiones son de los más conocidos algoritmos de NN (nearest neighbor = vecino más cercano), que a pesar de ser muy eficientes con espacios de pocas dimensiones, su rendimiento se ve acortado rápidamente para información con alto número de dimensiones. Como alternativa a estos se encuentran los k-d trees o árboles con una dimensión: k, Consisten en hacer divisiones con híper planos en cada nodo, haciendo particiones del espacio por cada híper plano, así, puntos a la izquierda del plano serán representados por un sub árbol izquierdo de ese nodo, mientras que los puntos a la derecha del plano serán representados por el sub árbol derecho. Sin embargo se encontró que usando colas de prioridad y haciendo una modificación en cómo se toma la distancia entre puntos y sus vecinos cercanos se puede acelerar la búsqueda, al igual que limitando el tiempo que duran las búsquedas, deteniéndola antes después de examinar un número fijo de nodos-hoja. (Los nodos más externos dentro de la estructura del árbol de particiones).

Los algoritmos de búsqueda "hashing", tienen una alta dependencia de sus funciones, las cuales se han ido mejorando con múltiples investigaciones basadas en técnicas de aprendizaje como: parameter sensitive hashing, spectral hashing, randomized LSH hashing from learned metrics, (Donde LSH representa: "locality sensitive hashing"), kernelized LSH, learnt binary embeddings, shift-invariant kernel hashing, semi-supervised hashing, optimized kernel hashing y complementary hashing.

Sin embargo los experimentos del mismo artículo de FLANN, [17], específicamente en el capítulo 4, describen como en la práctica, estos algoritmos generalmente son superados por estructuras de partición del espacio como *randomized k-d trees*.

Finalmente, las técnicas gráficas de vecino más cercano, trabajan construyendo una estructura gráfica en la cual los puntos son vértices conectados mediante bordes a sus vecinos más cercanos. Entonces se usan distintas estrategias como seleccionar algunos elementos dispersos en la gráfica como "semillas" desde las cuales se hace una exploración estilo "*best-first*" donde se van desglosando nodos, uno a la vez, escogiéndolos mediante alguna regla o función heurística, esto, para aproximar los puntos iniciales con sus vecinos. Estos métodos gráficos sufren de altos costos de procesamiento para la construcción de las estructuras k-NN gráficas. Pero algunos autores alivian el costo utilizando gráficos por aproximación de vecino más cercano. Sin embargo algunos de estos métodos todavía deben ser evaluados para considerar su futura incorporación en FLANN. [17] y [19]

2.2.4 – Otras estrategias y métodos de modelado y reconocimiento.

Más tarde en 1987, surgieron métodos que jugaban con el agrupamiento de posición, tomado por Stockman, Thompson y Mundy. Donde la idea se basaba primordialmente en discretizar el espacio de posibles transformaciones de objetos, buscar el valor máximo entre éstas y eligiendo al final la transformación más robusta como la posición del real del objeto. Un ejemplo de esto sería obtener pares de esquinas con correspondencias, elegir el par que tenga el valor máximo de estos y utilizar las transformaciones de rotación y traslación de ese último par para obtener la posición del objeto desde el modelo a la escena. Sin embargo el sistema requiere de una gran cantidad de memoria entre más complejas sean las transformaciones a efectuar. [14]

En cuanto a los modelos que buscan correspondencias entre características, definitivamente el más exitoso o robusto es RANSAC, planteado por Fischler y Bolles en 1981, el cual se basa en encajar información del modelo en entornos ruidosos, con significativa cantidad de puntos aislados fuera del área de interés, en otras palabras, el objeto a identificar. El procedimiento básico del método es: Escoger un conjunto de información para formular una hipótesis (un conjunto de parejas de puntos), se calcula el valor de la misma y se determina el número de puntos consistentes (puntos pertenecientes al objeto), si el valor de la hipótesis es lo suficientemente grande, el algoritmo regresa la misma ya que tiene la mayor estabilidad, en caso contrario se repite el proceso con una nueva hipótesis. Existen múltiples variaciones del método como: MLESAC, Lo-RANSAC y PROSAC, donde se hacen re-estimaciones para generar hipótesis más robustas o donde simplemente se selecciona aquella que posee más potencial en lugar de la más estable. [11]

En esta época se comenzaron a implementar métodos para la obtención del objeto sin la necesidad de examinar de forma explícita cada una de las imágenes del conjunto de datos obtenidos. Esto, por ejemplo, era conseguido por el método de Landan y Wolfson en 1988 donde utilizaban tercias de puntos para definir un cuadro afín e invariante, de donde obtenían puntos que eran discretizados en una “hash table” o tabla de datos dispersos, junto a un identificador del modelo y las mismas imágenes de consulta. La ventaja del método era que no era necesaria una búsqueda cúbica de todos los posibles tercetos de puntos, sin embargo el tamaño de la misma “hash table” y el número de cuadros afín utilizados podía llegar a convertirse en una desventaja.

Más tarde con el surgimiento de modelos de apariencia, dado que mostraron un gran porcentaje de éxito, comenzaron a concentrarse intereses en los mismos, y en los procedimientos para obtener características locales, uno de los precursores fue SLAM (Software Library for Appearance Modeling) en 1995 por Murase y Nayar, el cual ya usaba un conjunto de imágenes desde distintas poses representadas por un vector característico, al añadirse otra imagen nueva, el reconocimiento de la misma involucra buscar el vecino más cercano en el espacio característico.

Una vez que surgió el trabajo de Lowe en características locales utilizando un operador de diferencia de gaussianos para mantener la escala invariante (SIFT), se dio una explosión en el número de métodos que utilizaban apariencia dispersa, múltiples puntos de interés y métodos con descriptores. Se hizo aparente que la nueva mecánica para obtener emparejamientos exitosos consistía en detección y extracción de puntos de interés como Harris y DOG, descriptores como SIFT y un emparejamiento eficiente de estos descriptores entre distintas escenas, o entre modelo y escena.

Los subsecuentes trabajos se enfocaron en hacer más eficiente cada una de estas etapas individualmente. En el caso de detectores de regiones de interés; Hessian-Affine, Haris-Affine y MSER. Sobre los cuales es importante destacar que en la práctica, necesitan tanto dar un alto porcentaje de correspondencia entre características, es decir que sean robustas, así como generar una gran cantidad de características para poder hacer reconocimientos robustos. [11]

En cuanto a descriptores de regiones y puntos de interés, el trabajo de Mikolajczyk y Schmid comparó múltiples descriptores concluyendo que dentro de los que tienen alta dimensionalidad los más robustos eran SIFT y PCA-SIFT. [13]

2.3 – Modelado 3D.

2.3.1 – Definición y uso para el reconocimiento.

En el tema del reconocimiento de objetos 3D, el modelado es el proceso mediante el cual se genera un conjunto de información que nos servirá para "entrenar" o predisponer nuestro sistema de reconocimiento para que pueda hacer un "*matching"* entre las nuevas entradas que le lleguen, ya sean imágenes u otros modelos, con aquellos modelos ya almacenados en su base de datos, para buscar comparaciones y correspondencias que indicarán la identificación y/o clasificación del objeto en cuestión.

Entiéndase por identificación, al hecho de preguntar: "¿Eres el objeto que busco?", a lo cual el sistema deberá entregar una salida de si o no, siendo así un método muy minimalista. Mientras que la clasificación se entiende como un problema algo más complejo, al preguntar: "¿Tú qué clase de objeto eres?", cuando hablamos de categorías y características globales. O incluso más concretamente: "¿Tú qué objeto eres?", cuando hablamos de instancias de objetos y características locales. Esta diferencia de grados de complejidad se ilustra bien en las figuras 2.7 y 2.8.

En el caso de modelos 3D, estamos hablando de módulos de aprendizaje para un reconocimiento del tipo clasificatorio. Es decir; Una etapa de entrenamiento que le sirva a nuestro sistema para discernir entre un objeto y otro, traducido en varios modelos que dictaminen las características específicas de cada objeto a reconocer. Un modelo 3D basado en características locales, como el que se utilizó en el presente trabajo, utilizará algún tipo de descriptores más sus respectivas coordenadas en el espacio 3D respecto a un sistema de referencia centrado en el objeto, (el cual se utiliza en MOPED), y se construirá además, mediante el método de MOPED, que se deriva de las clásicas aproximaciones en el problema de "*structure from motion*" como veremos en el siguiente sub-capítulo.

Fig. 2.7 – Módulo de aprendizaje general, de identificación simple. Tiene una imagen de entrada y la salida es un simple "si" para designar la respuesta que debe tener el sistema con esa entrada específica.

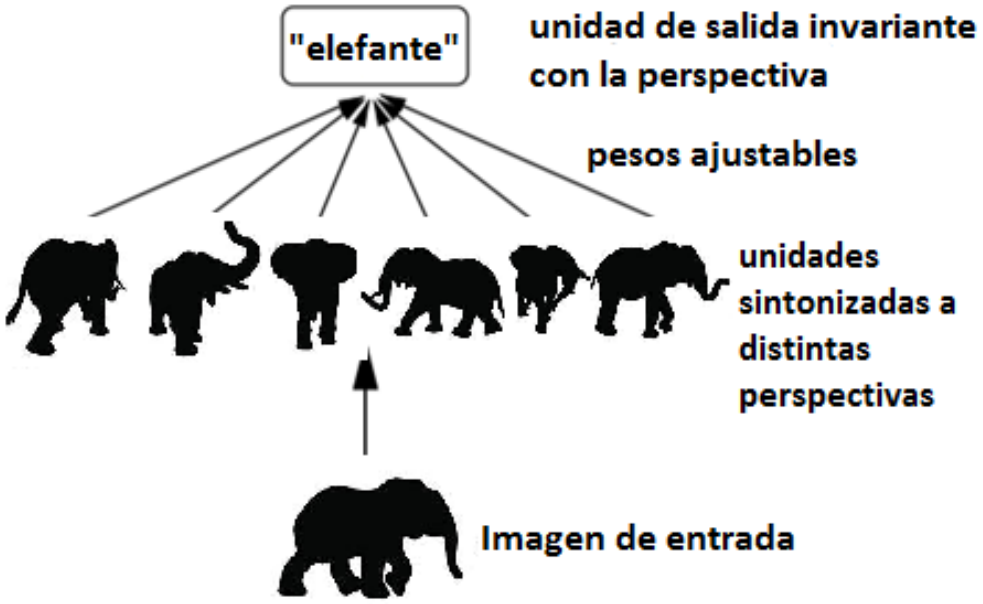

Fig. 2.8 – Módulo de aprendizaje específico, un clasificador. Entrenado para responder de manera invariante a la perspectiva de un determinado objeto. De manera que se obtenga la salida: "elefante" independientemente de la vista.

2.3.2 – Métodos basados en "*Structure from motion*".

Dentro de los métodos basados en "*structure from motion*" que se basan en reunir características que construyan la estructura del objeto mientras se mueve la perspectiva desde la cual se obtiene la información son: Triangulación, el problema de 2 cuadros "*structure from motion*", factorización y el *bundle adjustment*. Discutiremos sobre cada uno de ellos:

Triangulación.

Empezando desde lo más básico en "*structure from motion*", se tiene el problema de la triangulación que consiste en estimar la ubicación de un punto en una coordenada 3D cuando este es visto desde

múltiples perspectivas. El detalle de la triangulación es que se conocen las ubicaciones de cada una de las cámaras que observan el punto, lo que facilita un poco el cálculo. El problema principal se reduce a una estimación de posición de un objeto 3D a partir de un conjunto de proyecciones en puntos 2D, que no es sino un caso particular de la alineación basada en características.

El problema de la estimación de posición también se le conoce como calibración extrínseca, el recobrar posición dadas 3 correspondencias, que es el mínimo de información necesaria, conocido como el problema de perspectiva en 3 puntos (P3P) extensible a grandes cantidades de puntos que conjuntamente se les llama PnP o perspectiva desde "n" puntos de vista.

Una solución sencilla consiste en obtener el punto promedio o el más cercano a ambos puntos en el espacio 3D vistos desde todas las perspectivas, un ejemplo simple para 2 cámaras sería:

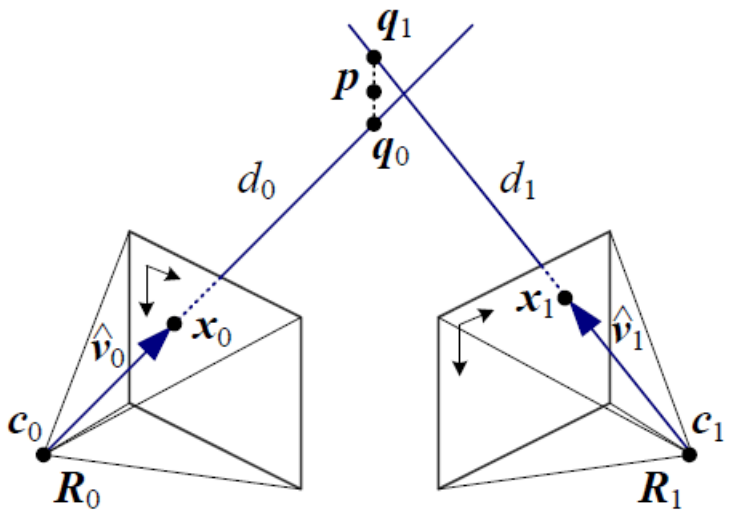

Fig. 2.9 – Triangulación de 2 perspectivas, se busca el punto "p" más cercano tanto a "q_0" como a "q_1". Tomado de: [28]

Donde "R" representa la rotación de la cámara, "c" el centro de la misma, "x" es el punto de la imagen o el pixel, "v" es la distancia del centro de la cámara al pixel y "d" la distancia desde el punto 3D hasta su proyección en la imagen en 2D.

Para encontrar el valor óptimo "p" se puede hacer una operación de mínimos cuadrados. Aunque una opción alternativa que puede dar mejores estimaciones es, si algunas de las cámaras están más cercanas al punto 3D que otras es la de minimizar el residuo en las ecuaciones de medición de la matriz de la cámara. [36]

Two frame structure from motion.

En el problema de estructura desde movimiento en 2 cuadros, se hace un primer aproximamiento a lo que es "*structure from motion*", donde ya no contamos con la información de la posición ya sea, de los puntos 3D o de las cámaras que los capturan, por lo que debemos de recuperar la estructura 3D del objeto y la posición de las correspondencias en las imágenes de manera simultánea.

Ahora se toma en consideración los valores de rotación y traslación para cada cámara, y se toma un plano que contiene los puntos de cada centro de cámara más el punto 3D de interés "p", al cual se le conoce como plano epipolar. Los epipolos son aquellos puntos en sus respectivos planos de imagen de sus respectivas cámaras que encaran el plano de la otra cámara, siendo así que ambos epipolos, y ambos centros de ambas cámaras están todos sobre la misma línea tridimensional.

Una de las principales aplicaciones de "*structure from motion*" de 2 cuadros o capturas es la interpolación de vistas, que nos sirve para generar una animación 3D de una perspectiva de una escena 3D hacia otra. Para lograr los mejores resultados se sugiere "facilitar" la entrada y salida de parámetros de la cámara usando un coseno elevado así como mover la cámara a lo largo de una trayectoria circular. También es posible re proyectar los puntos 3D ya obtenidos hacia vistas de perspectivas entre medio usando triangulación, siempre que los puntos en ambas cámaras tengan correspondencias conocidas.

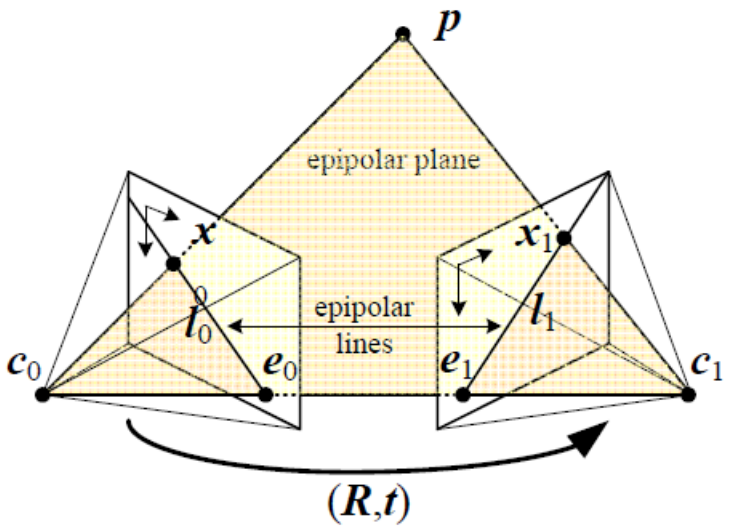

Fig. 2.10 – Structure from motion de 2 cuadros o cámaras. El plano epipolar conformado por los puntos coplanares: "p", "c_1" y "c_2". Los epipolos "e_0" y "e_1" conforman las líneas epipolares: l_0 y l_1 junto con los puntos de cada plano: "x_0" y "x_1" que observan al punto común en 3D "p". Si se considera la primera cámara en el origen, la segunda es definida por una rotación y traslación: "R" y "t" respectivamente. Tomado de: [28]

Factorización.

Se basa en obtener estructura y movimiento a partir de un seguimiento de características, como en el procesamiento de secuencias de video, se puede inferir la geometría completa de un objeto a partir de sus características (si son lo suficientemente discernibles), al ser puestas en movimiento.

En este método, se obtiene una matriz general de todos los puntos de cada captura para todas las capturas, y se descompone esta matriz mediante SVD (descomposición en valores singulares) para obtener 2 matrices fundamentales: "M" (motion = movimiento) y "S" (structure = estructura). De las cuales "S" contiene todos los puntos del objeto en el espacio 3D, y "M" contiene las matrices de movimiento que van ligadas a todos los puntos de determinada captura, para transformar del sistema de coordenadas del objeto 3D al sistema de coordenadas en 2D de la imagen específica.

Tal que: **X = MS.** Donde "X" es la matriz que contiene todos los puntos de todas las capturas desde el valor x_{11} hasta x_{MN}, siendo "M" el número total de capturas y "N" el número total de puntos por captura,

el cual debe ser el mismo para todas las capturas, es decir, todos los puntos necesitan tener una correspondencia en cada una de las perspectivas. Lo cual se convierte en una clara desventaja de este método, aunque se puede trabajar alrededor del problema utilizando subconjuntos densos de puntos y después utilizar estimaciones ya sea de cámara o puntos para llenar los puntos faltantes. El proceso de factorización estándar también puede extenderse a métodos más flexibles como ortografía escalada, movimiento rígido multi-cuerpo, actualizaciones secuenciales a la factorización, adición de líneas y planos, re-escalamiento de mediciones para incorporar incertidumbres de ubicaciones individuales, entre otras. [28]

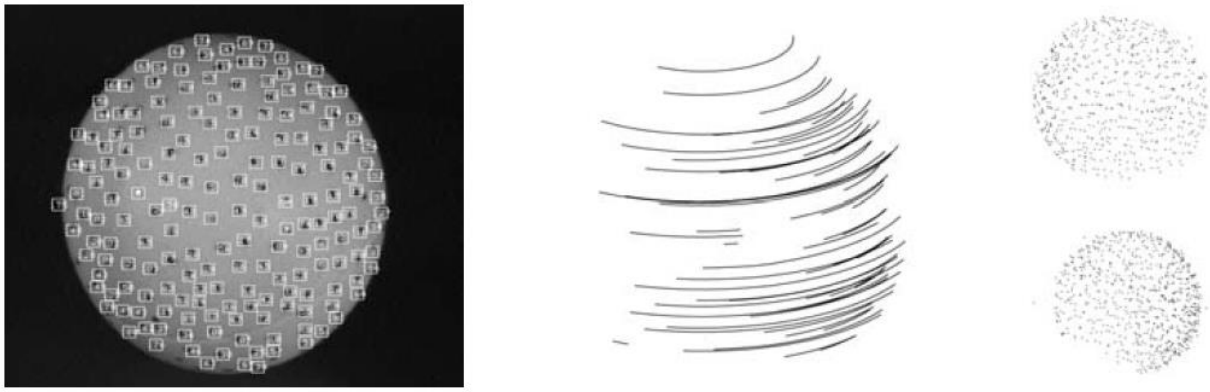

Fig. 2.11 – Reconstrucción 3D de la factorización de una pelota de ping pong giratoria. (a)Imagen de muestra con características seguidas marcadas. (b)Sub-muestra de una corriente de características en movimiento, formando una geometría esférica. (c)Dos vistas distintas del modelo 3D reconstruido. Tomado de: [28]

Bundle Adjustment.

Este método se basa en la minimización del error de re proyección entre las ubicaciones de la imagen de puntos observados y puntos predichos. En otras palabras, minimizar la distancia entre el punto 2D real proyectado a partir del punto 3D y el punto estimado dados los parámetros de la cámara iniciales. Se logra utilizando algoritmos no lineales de mínimos cuadrados, uno de los más efectivos siendo Levenberg-Marquardt ya que es sencillo de implementar y converge rápidamente desde un amplio rango de predicciones iniciales.

Las ecuaciones normales tienen una estructura de bloques dispersos dada la falta de interacción entre parámetros para diferentes puntos 3D y cámaras. Si los puntos 3D son conocidos o fijados, las ecuaciones de cada cámara se vuelven independientes unas de otras.

Para el caso de problemas de bundle adjustment donde los conjuntos de cámaras e imágenes se vuelven muy grandes, suelen aplicarse algoritmos de incremento, donde nuevas cámaras se van agregando conforme pasa el tiempo, lo cual suena lógico cuando muchas veces la información viene de una cámara de video o vehículos en movimiento. Conforme nueva información es obtenida, se puede implementar una actualización de las estimaciones de manera incremental, sin embargo esto sólo es útil para problemas de mínimos cuadrados lineales, mientras que para problemas no lineales como los de structure from motion, se pueden hacer múltiples pasadas por los datos. Pero como los puntos antiguos van desapareciendo de la vista conforme pasa el tiempo, se necesita mantener una ventana o filtro para mantener la visibilidad de cierta cantidad de cuadros conforme se siguen obteniendo más capturas, algoritmos que también se utilizan para tracking en tiempo real y para realidad aumentada.

Cabe añadir que a pesar que "*bundle adjustment"* y otras técnicas de mínimos cuadrados son métodos comunes de elección para solucionar problemas de "*structure from motion"* dada su robustez, sufren de problemas de inicialización, y no llegan a converger sin los parámetros de entrada adecuados por lo que se recomienda usar valores conservadores a la hora de elegir qué puntos y cuáles cámaras se utilizarán para inicializar el algoritmo. [28]

2.3.3 – Otros métodos.

Además de métodos que se basan en la obtención de características naturales extraídas de imágenes reales de objetos y escenarios para la creación de modelos en 3D, existen aquellos que se valen de modelos creados de forma sintética, partiendo de ideas básicas de manipulación de imágenes, alterando sintéticamente cambios de iluminación, escala, posición, rotación de los objetos, fondo e incluso el agregado de otros distractores para hacer sus conjuntos destinados al entrenamiento de modelos para el reconocimiento más robustos. Una base de datos que hizo uso de estas técnicas fue NORB. [31] Escalando a no sólo imágenes sino modelos completos renderizados en 3D para asemejar lo más posible las condiciones de la vida real, y subsecuentemente emparejados con imágenes que si fueron tomadas del mundo real, surgen métodos como el de "3D feature maps". [32] Que a pesar de obtener buenos resultados en el reconocimiento de objetos optimizando parámetros de renderización en los modelos, el método queda limitado a los valores de inicialización de estos mismos parámetros de renderización, haciéndolo no adecuado para muchas tareas de reconocimiento o detección. Además que la estimación de la posición es calculada de forma directa mediante una hipótesis generada para clases de objetos genéricos y post-procesada con estimación de pose 2D-3D quedando pendiente una mejora a un esquema de optimización iterativo como ICP o ICE. [33]

Existen otros métodos que se enfocan más en hacer una aproximación a los modelos biológicos de la corteza visual del ser humano y primates, le llaman reconocimiento de objetos por medio de características inspiradas en corteza visual, el método consiste en 2 principales tipos de procesamiento en su forma más sencilla, primero trata de aplicar filtros de Gabor a la imagen de entrada para incrementar la selectividad de objetos y hacer un "*template matching*", para después hacer un "*max pooling*", donde se conjuntan las entradas a través de una operación de máximos de posición y escala, obteniendo así una gradual invariancia respecto a la escala y la traslación. Siguiendo el comportamiento de cortezas visuales donde durante los primeros pocos cientos de mili-segundos de procesamiento visual, se sigue una especie de jerarquía consecutiva donde los campos receptivos de neuronas tienden a hacerse más grandes junto con el conjunto de estímulos que provocan la respuesta de una neurona, también llamado estímulo óptimo.

Sin embargo es curioso como a pesar de sólo realizar 2 procesamientos importantes, a diferencia de algoritmos que usan computaciones más complejas como estimaciones de distribuciones de probabilidad o la selección de componentes faciales para su uso con SVM (máquinas de vectores de soporte), o el simple hecho de no utilizar información geométrica como algunos de los mejores sistemas existentes en visión por computadora, basados en modelos de forma y apariencia, y detección de la posición relativa de cada componente junto a sus valores asociados de detección. A pesar de esto, el sistema de corteza visual sólo por basarse en inspiraciones biológicas es capaz de competir con estos algoritmos, quizás en parte por su construcción de cambio gradual y tolerancia escalar que imita muy bien el procesamiento visual biológico que se ha estado refinando por la evolución a lo largo de la historia. [34]

2.4 – MOPED.

2.4.1 – Reconocimiento de objetos con MOPED.

Siguiendo la estructura de los métodos previamente vistos basados en "*structure from motion*", ahora nos enfocaremos en MOPED, que trata de la misma estrategia de multi-perspectiva de un objeto, tomando su información desde todos los ángulos posibles, pero combinando con algoritmos robustos de obtención y emparejamiento de características así como estimación de pose y uso de agrupaciones por regiones que contengan descriptores afines, haciendo el cálculo de manera iterativa con ICE para recombinar la pose y unir agrupaciones de otras agrupaciones de características hasta obtener una convergencia adecuada.

El reconocimiento en MOPED se basa en el respaldo de una base de datos de modelos, que nos ayuden a identificar cada objeto dentro de una escena y estimar la posición de cada uno dentro de la misma.

Entradas de MOPED.

Como argumento de entrada tendremos un set de "M" imágenes al que denominaremos: "I". Tal que:

$$I = \{I_1, \ldots, I_m, \ldots, I_M\}, \quad I_m = \{K_m, T_m, g_m\}. \qquad (1)$$

En el caso más general, cada imagen es capturada usando una cámara calibrada de forma distinta, por lo que, cada imagen I_m es definida por una matriz de dimensiones 3x3 de parámetros intrínsecos de la cámara denominada K_m, otra matriz de dimensiones 4x4 de parámetros extrínsecos de la cámara denominada T_m con respecto a un cuadro de referencia global y una matriz con los valores de cada pixel proyectado en la escena denominada g_m.

MOPED es indiferente de la cantidad de imágenes introducidas, así que es igualmente válido tanto un conjunto de M imágenes de múltiples cámaras extrínsecamente calibradas como el caso simplificado de una sola imagen tomada de una sola cámara que a su vez es el mismo cuadro de referencia global. Es decir; (M = 1) y ($T_1 = I$, donde I es una matriz identidad de 4x4) respectivamente.

La otra entrada de MOPED son los modelos, cada modelo a ser reconocido, necesita pasar primero por una etapa previa de entrenamiento, en la cual un modelo de nube de puntos del objeto es creado y guardado dentro de la base de datos. Primero un conjunto de imágenes del objeto en distintas posiciones y ángulos es tomado, de las cuales se extraen descriptores locales de las características naturales usando SIFT, SURF o algún otro algoritmo que sea capaz de obtener puntos clave de interés de manera robusta. Después usando una transformación de unión y correspondencia entre los puntos de distintas perspectivas se obtiene un modelo en 3D de nube de puntos del objeto. Cada punto 3D es vinculado a un descriptor que es producido de un agrupamiento de descriptores correspondidos en las distintas perspectivas. Finalmente se hace una alineación y escalamiento apropiados para cada modelo para que las dimensiones sean coherentes a las del objeto en el mundo real, junto a un sistema de referencia que por simplicidad es definido en el centro del objeto.

Se puede representar tal que "O" es el conjunto de modelos de objetos. Cada modelo es definido por su ID o identificación "o" así como un conjunto de características C_0. Tal que:

$$O = \{o, C_0\}, \quad C_0 = \{C_{1;o}, \ldots, C_{i;o}, \ldots, C_{N;o}\}. \qquad (2)$$

Cada característica está representada por una coordenada en el espacio de tres dimensiones: $P = [X, Y, Z]^T$ en el sistema de coordenadas centrado en el objeto, y un descriptor "D" cuyo tamaño

dependerá del tipo de algoritmo utilizado, por ejemplo k = 128 si se utiliza SIFT, y k = 64 si se utiliza SURF. Donde "k" es el número de bin o espacios contenedores del vector descriptor. Lo cual se puede expresar como:

$$C_{i;o} = \{P_{i;o}, D_{i;o}\}, \quad P_{i;o} \in \mathbb{R}^3, \ D_{i;o} \in \mathbb{R}^k. \qquad (3)$$

A la unión de todas las características de todos los objetos en "O" se le denomina como: $\mathbf{C} = U_{o \in O} C_0$.

Véase [7].

Salidas de MOPED.

La salida de MOPED será un conjunto de hipótesis "H". Cada hipótesis de reconocimiento de un objeto $H_h = \{o, T_h\}$ está representada por el ID o identificación del objeto "o" y una matriz de 4x4 "T_h" que corresponde a la posición del objeto respecto al marco de referencia de la imagen.

Etapas de reconocimiento y componentes de MOPED.

En la siguiente sección se explica el cuadro de referencia MOPED en su forma más básica, para los requerimientos básicos de aplicaciones en robótica. Se asume la configuración más común de reconocimiento de objetos donde se cuenta sólo con un grupo pequeño de imágenes, generalmente no más de diez, y un cálculo de ICE de dos estimaciones de agrupaciones más una unión final de agrupaciones para evitar detecciones múltiples que pudiesen no haber convergido. Así es como MOPED se asegura de tener un buen equilibrio entre alta precisión en el reconocimiento y mantener un bajo estado latente en el sistema. Es importante señalar que si se trabaja con un conjunto amplio y simultaneo de imágenes se necesitaría un mayor número de iteraciones.

1. Extracción de características.

La primera fase consiste en una extracción de características de cada imagen de forma independiente, tal que si se tiene un conjunto de imágenes "I" con características locales C_n. Cada imagen $I_n \in I$ será procesada tal que:

$$I_n = \{K_n, T_n, C_n\}, \quad C_n = \{\text{Extracción\_de\_característica}(g_n). \qquad (4)$$

De tal forma que cada característica "$C_{i;n}$", tomada de una imagen "n" es definida por su coordenada en pixeles $p_{i;n} = [x, y]^T$ y su correspondiente vector descriptor: "$d_{i;n}$" tal que:

$$C_n = \{C_{1;n}, \ldots, C_{i;n}, \ldots, C_{in;n}\}, \quad C_{i;n} = \{p_{i;n}, d_{i;n}\}. \qquad (5)$$

Y se define a la unión de todas las características locales extraídas de todas las imágenes "n" como: $f = U_{n=1}^{N} C_n$.

2. Correspondencia de características.

Se buscan correspondencias uno a uno entre características obtenidas del conjunto de imágenes de entrada y del modelo del objeto guardado en la base de datos. Si "M" es un match o correspondencia entre una característica de la imagen: $C_{i;n}$ y una característica del modelo: $C_{j;o}$, tal que:

$M^o_{i;n}$ = {($C_{i;n}$, $C_{j;o}$), si: $C_{i;n} \leftrightarrow C_{j;o}$ Y 0, si: no hay correspondencia}. (6)

Al conjunto de correspondencias dadas entre un objeto "o" y una imagen "n" está representada como: $\boldsymbol{M}^o_n$ = $U_{\forall i} M^o_{i;n}$. Y los conjuntos de correspondencias $\mathbf{M}^o$ y $\mathbf{M}_n$ definidos de forma equivalente son: $\mathbf{M}^o = U_{\forall i;n} M^o_{i;n}$ y $\mathbf{M}_n = U_{\forall i;o} M^o_{i;n}$.

3. Agrupamiento de características.

Las características emparejadas a un objeto particular son agrupadas en el espacio de la imagen (x, y), independiente para cada imagen. Dado que espacialmente las características cercanas son más probables a pertenecer a la misma instancia de objeto, se agrupa el conjunto de coordenadas de características p $\in \boldsymbol{M}^o_n$, produciendo así un conjunto de otros conjuntos de características espacialmente cercanas.

Cada conjunto K_k definido por una identidad de objeto "o", un índice de imagen "n" y un subconjunto de correspondencias al objeto en la imagen I_n, tal que:

K_k = {o, n, $\mathbf{M}_k \subset \boldsymbol{M}^o_n$ }. (7)

Al conjunto de todos los agrupamientos le llamamos: "K".

4. Estimación I. Generación de la hipótesis.

Cada conjunto es analizado de forma independiente en cada imagen en la búsqueda de objetos que tengan correspondencias. Se utiliza RANSAC y Levenberg-Marquardt (LM) para encontrar múltiples hipótesis de objetos con una buena estimación de la posición de cada uno de estos. Cada hipótesis se puede escribir como:

h = {o, k, T_h, $\mathbf{M}_h \subset \mathbf{M}_k$}. (8)

donde "o" es la identidad del objeto, k es el índice de la agrupación de correspondencias entre instancias locales de esa hipótesis, T_h es la matriz de transformación 4x4 que define la posición del objeto de la hipótesis respecto al marco de referencia global y $\mathbf{M}_h$ es el subconjunto de correspondencias que son consistentes con la hipótesis h.

5. Agrupación de conjuntos.

Ya que el mismo objeto puede estar presente en múltiples agrupaciones de correspondencias e imágenes, las distintas posiciones son proyectadas desde cada imagen a un sistema de referencia global, y aquellos conjuntos que tienen características comunes y compatibles conforman otros conjuntos más grandes, estos nuevos conjuntos se definen como:

K_k = {o, $\mathbf{M}_k \subset \mathbf{M}^o$}. (9)

6. Estimación II. Refinación de la pose.

En este paso ya la mayoría de puntos aislados o que no aportan información útil han sido removidos, y cada conjunto contendrá correspondencias a únicamente una instancia de un objeto, a lo largo de múltiples imágenes. Se vuelve a utilizar RANSAC y LM en conjunto para obtener posiciones que sean más consistentes y en la que cada hipótesis multi-perspectiva se puede definir como:

$$H = \{o, T_H, \mathbf{M}_H \subset \mathbf{M}_k\}. \qquad (10)$$

7. Recombinación de las posiciones.

Finalmente se unen las instancias de objetos que tienen posiciones similares, esto para evitar que se generen falsos positivos de detecciones de múltiples objetos. Donde un conjunto de hipótesis **H**, tal que: $H_h = \{o, T_H\}$, es la salida final de MOPED. [7]

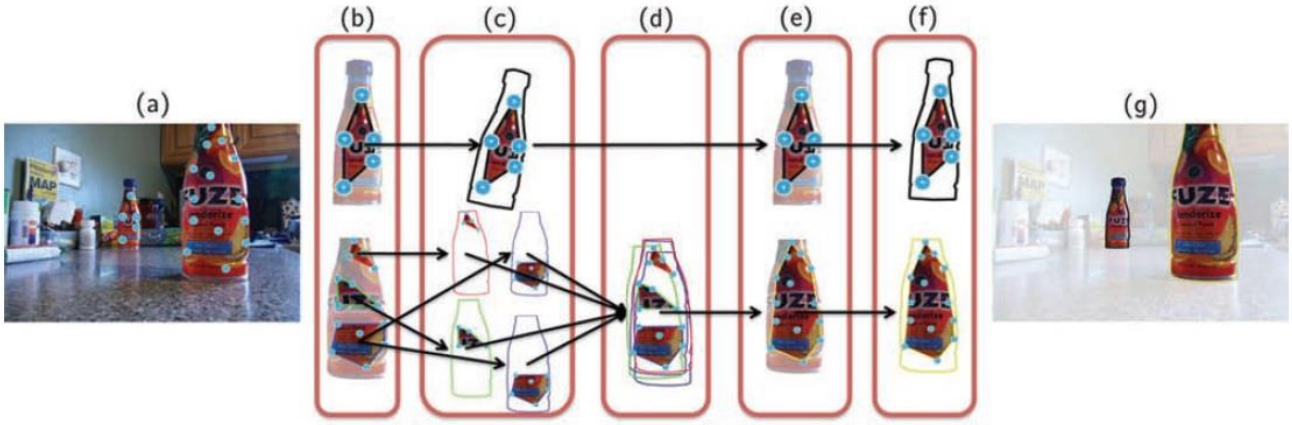

Fig. 2.12 - Ilustración de 2 iteraciones de ICE. Los contornos de colores representan estimaciones de posición. (a)Extracción de características y correspondencias. (b)Agrupación de características. (c) y (d) Generación de hipótesis. (e)Agrupación de conjuntos. (f)Refinación de la pose. (g)Resultado Final. Tomado de: [7]

Fig. 2.13 – Reconocimiento de objetos con escenas del mundo real. (Arriba)Escena de alta complejidad donde MOPED reconoce 27 objetos, algunos parcialmente cubiertos. Usando una base de datos con 91 modelos y una resolución de imagen de 1600x1200. (Abajo)Escena de complejidad intermedia donde MOPED procesa la imagen de 640x360 en 87ms y encuentra todos los objetos, (exceptuando la lata verde de sopa ya que no está en la base de datos). Tomado de: [7]

2.4.2 – Modelado de objetos 3D con MOPED.

Durante la etapa de entrenamiento de un Sistema de reconocimiento, es decir; el crear los modelos de los objetos, requiere de objetos con una geometría particular como superficies planas o bordes seguidos de líneas rectas, modelos en CAD, o depender del uso de marcas o señalizaciones, mientras que en MOPED, el modelado se hace en base a las características locales naturales de cada objeto, con SIFT o SURF por ejemplo, y haciendo una unión de estas características con correspondencias entre distintas perspectivas se consigue un modelo de nube de puntos de cada objeto después claro, de hacer una alineación y escalamiento a las dimensiones del mundo real desde la imagen y el espacio de la cámara que se utiliza.

Para crear un modelo en 3D de un objeto, primero se necesita obtener un conjunto de imágenes del mismo en distintas posiciones, de tal manera que se cubra la mayor cantidad posible de características en todas las vistas, se obtienen correspondencias de cada captura con sus semejantes aledañas para que los puntos que lleguen al modelo final sean los más robustos, es decir sean observables desde la mayor cantidad de ángulos de mira. Cada una de estas correspondencias tendrá una asignada característica única, la cual puede ser cualquiera de las obtenidas de las múltiples capturas, un promedio de las mismas, o calcularse mediante agrupaciones con ICE.

Generalmente las correspondencias en múltiples perspectivas pueden verse como proyecciones en 2D sobre una imagen de los puntos 3D del objeto, tal que la relación formal entre un punto 3D: $\mathbf{P}_i = [X_i, Y_i, Z_i\ 1]^T$ y su contraparte en 2D para una imagen dada "j": $\mathbf{p}_{ji} = [x_{ji}, y_{ji}, 1]^T$, puede expresarse como:

$$\mathbf{p}_{ji} \equiv K\, T_j\, \mathbf{P}_i \qquad (11)$$

$$\text{Donde: } T_j = \begin{bmatrix} R_j & t_j \\ 0 & 1 \end{bmatrix} \qquad (12)$$

Siendo "K" una matriz de calibración de la cámara con parámetros intrínsecos de dimensión 3x3, T_j es la matriz de transformación de la cámara del espacio 3D al 2D de la imagen, compuesta por R_j y t_j que son la rotación y traslación de la cámara respecto a un cuadro de referencia global respectivamente.

Al minimizar la suma de errores de re proyección, (véase Eq. 14). Puede obtenerse el mejor conjunto de parámetros para la transformación adecuada. Dados: $\mathbf{P} = [P_1 \ldots P_N]$, $\mathbf{R} = [R_1 \ldots R_M]$, $\mathbf{t} = [t_1 \ldots t_M]$ donde "N" es el número de puntos 3D del objeto y "M" el número de posiciones de la cámara.

Entonces el mejor conjunto de parámetros (**P*, R*, t***) está dado por:

$$\text{score}(\mathbf{P}, R_j, t_j) = \sum_{i=1}^{N} [\mathbf{p}_{ji} - \text{proj}(\mathbf{P}_i, R_j, t_j)]^2 \qquad (13)$$

$$(\mathbf{P^*, R^*, t^*}) = \underset{\mathbf{P}, \mathbf{R}, \mathbf{t}}{\arg min} \sum_{j=1}^{M} \text{score}(\mathbf{P}, R_j, t_j) \qquad (14)$$

Donde "proj( )" representa la función de perspectiva no lineal dada por:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \text{proj}(\mathbf{P}_i, R_j, t_j) \qquad (15)$$

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} x/w \\ y/w \end{pmatrix} \quad \begin{pmatrix} x \\ y \\ w \end{pmatrix} = K \left[R_j \mid t_j \right] \mathbf{P}_i \qquad (16)$$

El objetivo de las funciones es obtener el error de re proyección para cada punto del objeto en el espacio 3D, hacer una sumatoria del cuadrado de todos los errores para cada una de las capturas, y finalmente obtener los valores mínimos y óptimos de rotación, traslación y conjunto de puntos que saldrá de una determinada captura "j".

El óptimo conjunto de parámetros puede ser obtenido con un método de minimización no lineal genérico como Levenberg-Marquardt. Una forma sencilla de facilitar el proceso es una inicialización de todos los puntos en el plano XY y la posición de todas las cámaras en la misma posición arbitraria, de frente a ese plano XY, debería ser suficiente para obtener resultados precisos después de la convergencia. El modelo de salida consistirá en un conjunto "**P**" de "N" puntos con coordenadas 3D y sus correspondientes vectores descriptores SIFT, SURF o el que se utilice, más otro conjunto de "M" posiciones de cámara, cada una con sus respectivas matrices "R" y "t" de rotación y traslación respectivamente a un sistema de referencia global.

A pesar de que este modelo final ya sería utilizable, se puede hacer una alineación más a las coordenadas del mundo real: "R_a" y "t_a", así como un escalamiento "s" a las dimensiones reales del objeto, por cada punto 3D: $\mathbf{P}_i$, tal que:

$$\mathbf{P'}_i = \begin{bmatrix} sR_a & t_a \\ 0 & 1 \end{bmatrix} \mathbf{P}_i \qquad (17)$$

Donde $\mathbf{P'}_i$ son los puntos finales del modelo. [7]

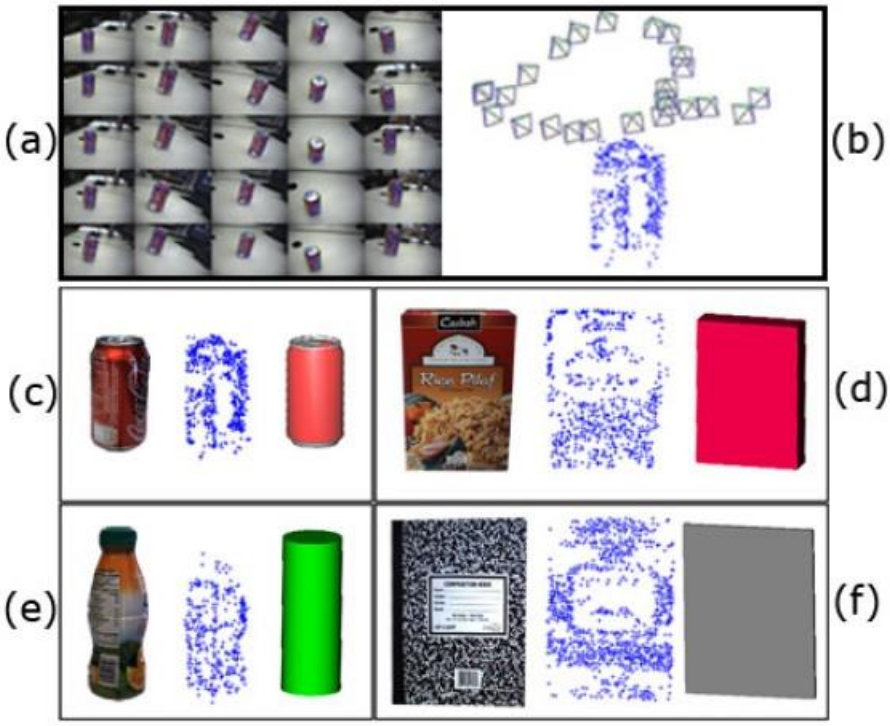

Fig. 2.14 - Aprendizaje de modelos de nube de puntos en 3D. (a-b)Imágenes de entrenamiento para una lata de soda y modelado 3D a través de distintas posiciones de cámara. (c-f)Modelos aprendidos para una lata de soda, caja de arroz, botella de jugo y cuaderno. Cada imagen contiene el objeto, su modelo 3D como nube de puntos y una representación virtual en el espacio de trabajo de un robot de servicio. Tomado de: [7]

CAPÍTULO 3.

3 - Modelado de objetos 3D con Kinect Fusion.

3.1 – El sensor Kinect.

Un sensor Kinect, marca registrada de Microsoft (llamado comúnmente Kinect), es un dispositivo físico que contiene cámaras, un arreglo de micrófonos y un acelerómetro así como una gama de software capaz de procesar color, profundidad, luz infrarroja, información de esqueleto y cara, además de sonido y detección de movimiento.

Fig. 3.1 - Imagen de un sensor Kinect V2.0 para Windows. Tomado de: [21]

Si nos ponemos a hablar de las especificaciones de cada uno de los componentes del Kinect tendremos:

- Una cámara RGB que guarda información en 3 canales con una resolución en su primera versión de 1280x960 pixeles, después incrementada a 1920x1080 pixeles.

- Un arreglo de múltiples micrófonos, cuatro en total, que permiten encontrar la dirección de las ondas sonoras, el origen de las mismas y grabar el audio.

- Un acelerómetro de 3 ejes configurado para un rango de 2 veces la aceleración de la gravedad, nos sirve para determinar la orientación e inclinación actual del Kinect.

- Finalmente un emisor receptor de luz infrarroja, que nos sirve para cuantificar la profundidad respecto a la placa frontal del Kinect, los rayos infrarrojos son reflejados en los objetos, y al regresar al Kinect transforman la información a una distancia en metros.

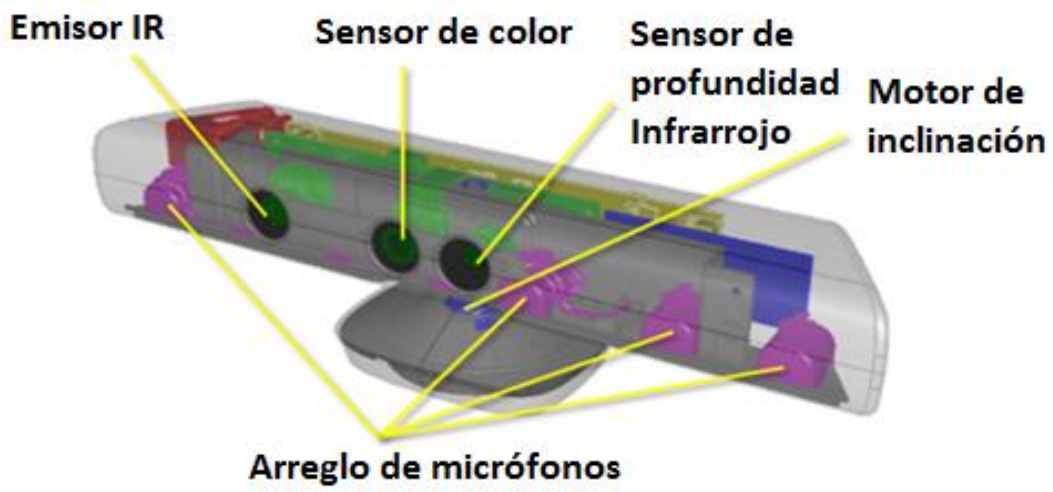

Fig. 3.2 - Esquema de los componentes de un sensor Kinect V1. Traducida de: [22]

El campo de visión del Kinect es de unos 43° en el eje vertical por unos 57° en el eje horizontal. Con un rango de inclinación vertical de ±27°, cuenta con una corriente tanto en los datos de color como de profundidad de unos 30fps (frames per second = cuadros por segundo). El formato del audio es de 16kHz a una modulación por impulsos codificados de 24bits con convertidor analógico-digital (ADC), que incluye una señal de procesamiento que cuenta con cancelación de eco acústico y supresión de ruido. El acelerómetro 2G/4G/8G está configurado para el rango 2G con un límite superior de precisión de 1°.

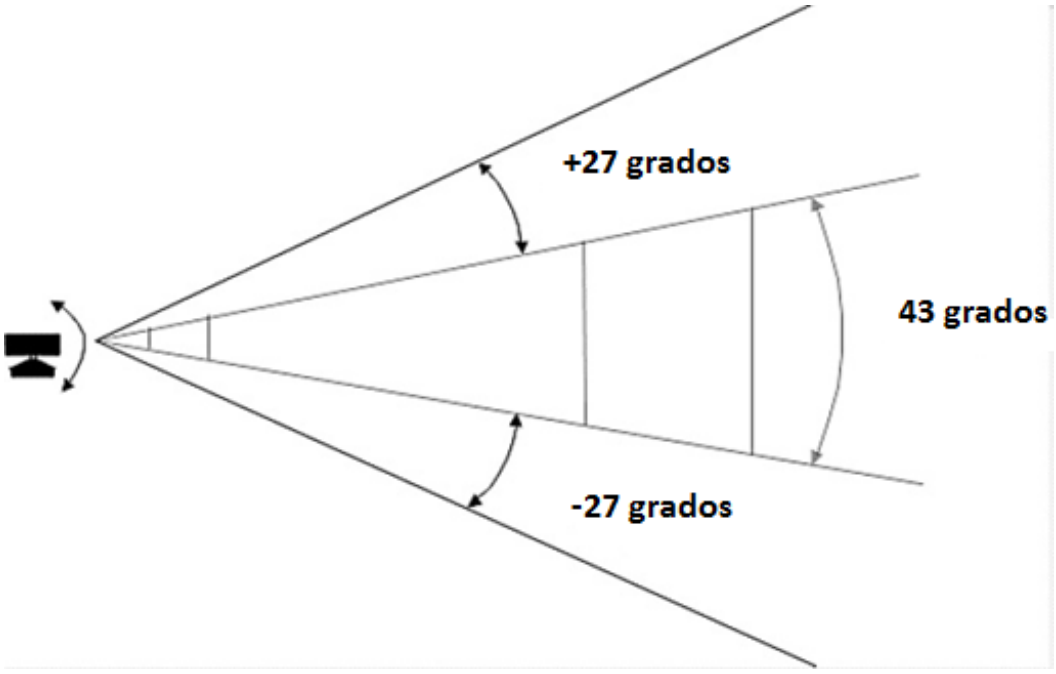

Fig. 3.3 - Rango del espacio de interacción del sensor visual de Kinect. Traducida de: [23]

Un Kinect obtiene información de color, profundidad y datos de esqueleto un cuadro a la vez. Por lo que hay distintos espacios coordenados por cada tipo de torrente de información, así como transformaciones entre los distintos espacios.

Espacio de color.

Cada cuadro capturado por el Kinect obtiene una imagen a color de todo el campo de visión del sensor de color en ese instante. Un cuadro está compuesto por pixeles, donde el número de los mismos dependerá del tamaño del cuadro que es especificado por la resolución del sensor, siendo en la versión 2.0 de Kinect SDK de unos 1920x1080 pixeles de resolución para la imagen a color y de unos 512x424 pixeles de resolución para la imagen de profundidad. En la imagen a color, cada pixel en una ubicación particular (x, y) contiene valores para los canales Rojo, Verde y Azul que conforman el color de la escena. Siendo un valor RGB de (0, 0, 0) el color negro y un valor de (255, 255, 255) el color blanco.

Espacio de profundidad.

El sensor de profundidad captura por cada cuadro una imagen en escala de grises de todo el campo de visión del sensor de profundidad, (es importante aclarar que el sensor de profundidad está ubicado ligeramente aledaño al sensor de color, por lo que para utilizar la información conjunta de ambos sensores es necesario realizar una alineación de los campos de visión ya que están viendo ligeramente distintas perspectivas). Cada pixel en la imagen de profundidad contiene una distancia cartesiana expresada en milímetros, desde el plano de la cámara hasta el objeto más cercano en esa coordenada particular (x, y), las coordenadas de un cuadro de profundidad no representan unidades físicas en la habitación o escena, sino la ubicación de un pixel en la imagen de profundidad.

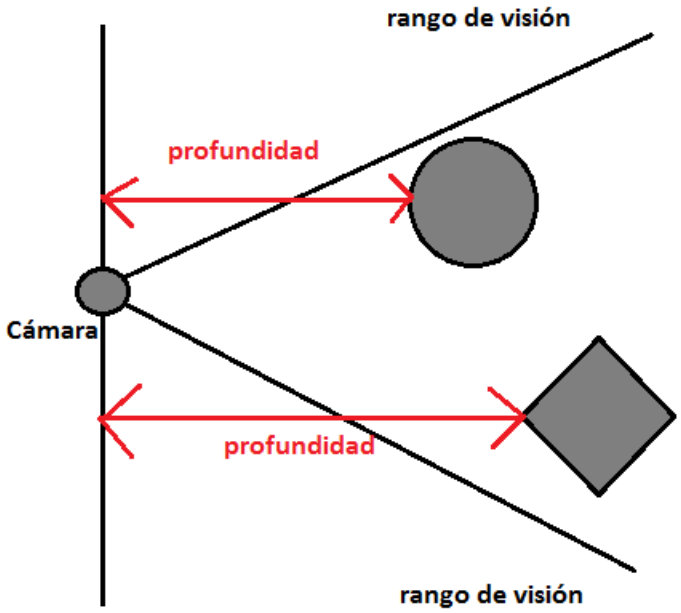

Fig. 3.4 - Medición de los valores de profundidad de Kinect.

Cuando el sensor Kinect toma valores de profundidad muy cercanos o muy lejanos no pueden ser medidos con seguridad, y los marca como “muy cerca”, “muy lejos” o “desconocido” si ningún objeto o superficie fue detectado en ese pixel particular. El sensor tiene 2 rangos de medición, el default que está disponible tanto para Kinect para Windows como para Kinect para Xbox360, y el rango cercano que sólo está disponible para el sensor Kinect para Windows, que será el utilizado en este trabajo.

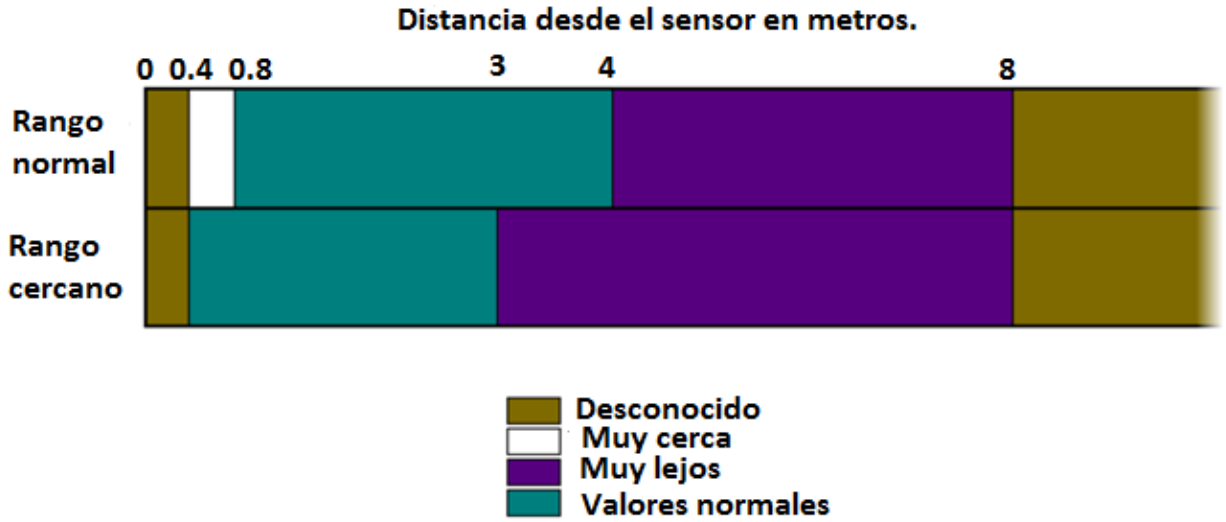

Fig. 3.5 - Rangos Default o normal y cercano de medición en el sensor de profundidad de Kinect.

Espacio esqueleto.

Cada cuadro, la imagen de profundidad es procesada por Kinect para obtener información esqueleto. La cual contiene coordenadas de posición en 3D para esqueletos humanos de hasta dos personas en el campo de visión del sensor de profundidad. La posición del esqueleto y de cada una de sus articulaciones se guardan como coordenadas en el espacio (x, y, z) expresadas en metros a diferencia del espacio de profundidad. Toda información de tracking en Kinect opera en el espacio 3D con el mismo sistema de referencia, esto incluye la información de esqueleto y el espacio de la cámara.

Espacio de la cámara.

El sistema de referencia del espacio de la cámara, servirá para operar la información en 3D de Kinect es un sistema de mano derecha, similar a como se definiría un espacio de cámara en computación gráfica.

Fig. 3.6 - Sistema de coordenadas del espacio de la cámara y por extensión de esqueleto del Kinect. Tomado de: [25]

El origen del sistema de referencia 3D de Kinect se encuentra en el centro del sensor Infrarrojo.
X crece positivamente hacia la izquierda del sensor.
Y crece positivamente hacia arriba y depende directamente de la inclinación del sensor.

Z crece positivamente hacia la dirección que el sensor está apuntando.
1 unidad = 1 metro.

Véase [21].

3.2 – Kinect Fusion.

Kinect Fusion es un sistema que nos permite escanear objetos y ambientes en 3D utilizando el sensor Kinect, permitiendo la creación de modelos de los mismos. Ya que es muy versátil es posible la interacción con la escena mientras se sigue escaneando el entorno y al mismo tiempo se manipula una escena virtual del mismo generada en tiempo real.

Fig. 3.7 - De izquierda a derecha: Información de entrada de Kinect Fusion tomada directamente del sensor Kinect aún con ruido e incompleta, salidas de Kinect Fusion generadas únicamente con la información del sensor de profundidad como mapeados en tiempo real a color y sombreado en escala de grises respectivamente. Tomado de: [26]

Si se corre el sistema en un ritmo no interactivo se consiguen mayores volúmenes de reconstrucción. A partir de los cuales puede generarse una nube de puntos o una red 3D de superficie.

Kinect Fusion integra múltiples perspectivas de una escena mientras se mueve el sensor alrededor de la misma, de tal manera que va integrando esta información de profundidad a un volumen de reconstrucción en voxeles que se va volviendo más robusto entre más perspectivas se agreguen a la escena sin demasiado ruido o movimientos bruscos mientras se hace el modelo. La posición de la cámara es seguida en su ubicación y orientación lo que permite que se haga un promedio de las perspectivas creando una fusión.

Etapas en la reconstrucción del volumen.

- La primera etapa es la conversión de la información del cuadro de profundidad a profundidad de punto flotante en metros. Seguida de una conversión opcional a una nube de puntos en un espacio 3D en el sistema de coordenadas de la cámara y normales a la superficie en cada uno de estos puntos que contienen la orientación de la misma superficie y nos sirve para la función de Kinect Fusion "*AlignPointClouds*" que proporciona una moderada alineación entre perspectivas.

- La segunda fase consiste en calcular la posición de la cámara respecto a un sistema de referencia global y común a todas las ubicaciones del sensor. Y continúa siguiendo esta posición mientras pasa por cada cuadro usando un algoritmo de alineación iterativo. Ya sea con una función de *"AlignPointClouds"* que se encarga de alinear ya sea nubes de puntos del mismo volumen de reconstrucción con otras nuevas capturadas por el sensor de profundidad o de forma autónoma para alinear 2 perspectivas de cámara de la misma escena. Y la segunda opción: *"AlignDepthToReconstruction"*, que provee un seguimiento de la cámara más preciso cuando se trabaja con un volumen de reconstrucción, sin embargo es susceptible a cambios en el entorno como objetos en movimiento.
- La tercera etapa se encarga de la fusión o integración de la información de profundidad desde la posición del sensor actual hasta una única representación volumétrica del espacio de la cámara. Se hace continuamente cuadro por cuadro, haciendo un promedio constante para reducir ruido y manejar de forma dinámica cambios en la escena. Los espacios vacíos pueden irse llenando si por ejemplo se mueve el sensor a la parte de atrás de un objeto donde antes no se contaba con esa información por tener sólo perspectivas frontales o laterales.
- Se puede hacer un *"raycasting"* sobre el volumen de reconstrucción desde la posición donde se encuentre el sensor y generar una nube de puntos que puede sombrearse para renderizar una imagen visible del volumen de reconstrucción en 3D.

El tamaño del volumen de reconstrucción generalmente va desde los $8m^3$ y con una resolución de 1-2mm por voxel, entiéndase por voxel un pixel en el espacio de 3 dimensiones.

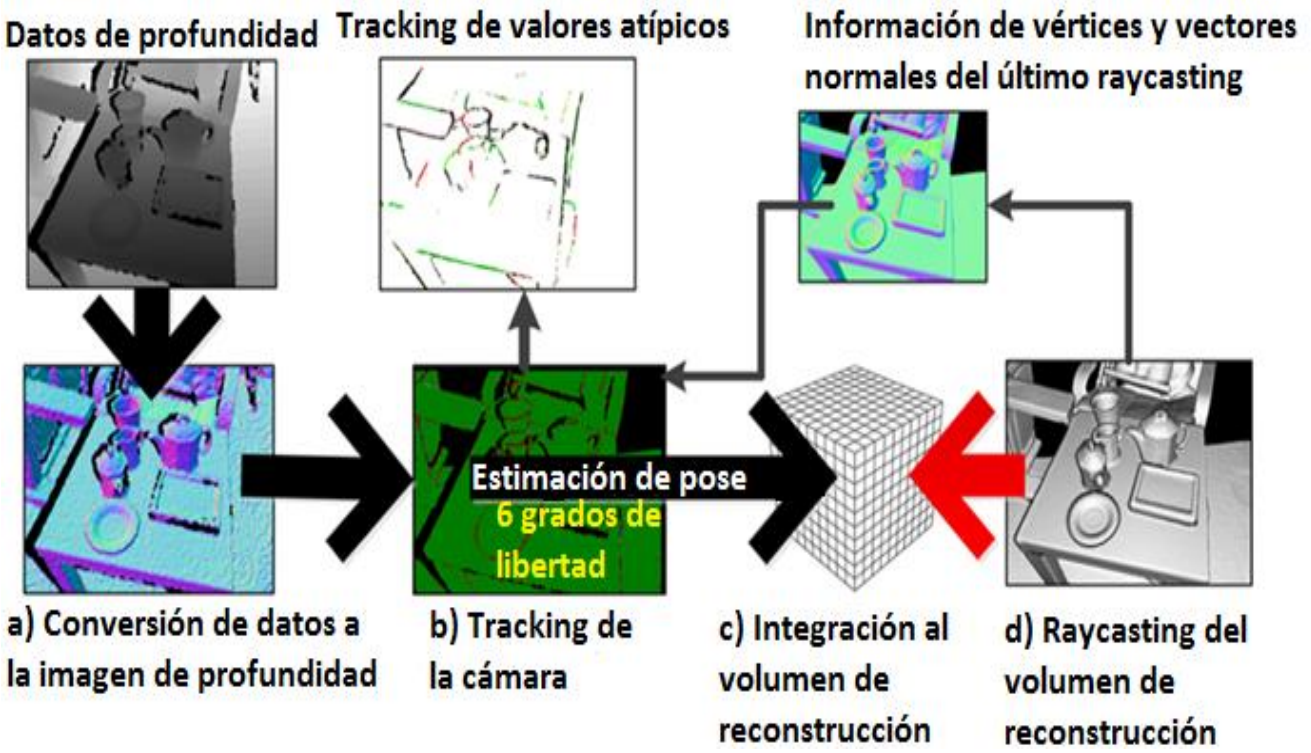

Fig. 3.8 - Esquema de las etapas de reconstrucción de Kinect Fusion.
(a)Conversión de la información de profundidad a punto flotante en metros y suavizado. (b)Seguimiento de la cámara en perspectivas distintas a 6 grados de libertad, cuidando movimiento e inclinación del sensor. (c)Integración de las múltiples perspectivas en el volumen de reconstrucción. (d)*Raycasting* para obtener una nube de puntos e imagen renderizada del volumen para su visualización. Traducida de: [27]

El sistema actual de Kinect Fusion a pesar de su buen funcionamiento en interiores en espacios de hasta $7m^3$, siendo capaz de mapear y reconstruir, puede extenderse al grado de ser capaz de generar volúmenes de reconstrucción del tamaño de un edificio entero, lo cual por el momento representa una tarea complicada, ya que en condiciones extenuantes y distancias largas, Kinect Fusion ya no es capaz de mantener la alineación entre perspectivas y el seguimiento de la posición del sensor respecto al volumen. Además claro, que requeriría mucho consumo de memoria. Son problemas que podrían solucionarse usando Kinect Fusion dentro de un cuadro de referencia de sub-mapeo manejando problemas con memoria y procesamiento. Sin embargo, considerándose problemas clásicos de SLAM igual se puede atacar con buenos resultados para representaciones dispersas, pero que requerirían más trabajo en modelado denso.

Entiéndase por SLAM el problema de la localización y mapeado simultáneos, que tiene como soluciones estrategias de discretizar el espacio utilizando sub-particiones del entorno de trabajo de robots móviles para tener un mayor control y predicción en cada sub-conjunto. [26] y [27]

3.3 – Sistema propuesto para modelar los objetos.

3.3.1 – Conjunción de MOPED y Kinect Fusion para el modelado.

En la propuesta del modelador de objetos 3D que se desarrolló para el robot Golem III del IIMAS de la UNAM, se utilizaron los conceptos de MOPED para realizar el modelado en conjunción con una poderosa herramienta que nos facilita mucho el trabajo además de hacer el proceso rápido y dinámico, el sensor Kinect.

La estrategia que se siguió, muy parecida a la empleada en MOPED, consistió en que se tomaran múltiples capturas a lo largo de una trayectoria circular alrededor del objeto de interés, en cada captura se tomaron los datos de coordenadas 3D de los puntos observados, esto mediante una transformación interna que se hizo con ayuda de las funciones de Kinect Fusion, es decir, a partir de la información del cuadro de profundidad, se transformaron los valores al espacio de la cámara, y posteriormente mediante la matriz de transformación de la cámara al mundo, (obtenida del volumen de reconstrucción de Kinect Fusion), se obtienen los valores de los puntos 3D del objeto respecto al mismo sistema de referencia, esto entre captura y captura con cierto margen de error de alineación entre los puntos. El cual se intenta minimizar más tarde mediante un "*matching*" realizado en un rango de capturas con FLANN.

Puesto que la matriz de transformación de la cámara al mundo del volumen de reconstrucción estará actualizándose constantemente, se debe tener especial cuidado al manejar el sensor, de manera que no se hagan movimientos bruscos o demasiado rápidos en el proceso de capturar, ya que esto podría ocasionar que el sensor se pierda y los puntos de la siguiente captura sean demasiado alejados del promedio de la ubicación de los otros puntos previamente censados.

Ya que buena parte del trabajo de alineación y puesta de coordenadas respecto a la misma referencia lo hace Kinect Fusion por nosotros, nos facilita la adición de algoritmos de mínimos cuadrados como Levenberg-Marquardt para minimizar errores de re proyección, sin embargo una adición del mismo en cierta medida podría ayudar a mejorar la convergencia, al igual que algoritmos iterativos como ICP.

Sin embargo dentro de este proceso no se considerarán todos los puntos del objeto, sino sólo aquellos puntos clave que presenten características robustas extraídas mediante algoritmos de detección de

características locales naturales, para así identificar exactamente de qué objeto se trata y no sólo como un clasificador en el caso que se usarán características globales como el contorno o la forma del objeto.

Para esto se utilizaron librerías de visión por computadora que cuenten con estos algoritmos, en este trabajo se utilizó la librería de código abierto OpenCV que contiene algoritmos de detección como SIFT, SURF, ORB entre otros, más otros para realizar correspondencias como Brute Force o FLANN.

Se extraen las características de la imagen (mediante SIFT) de color tomada con la cámara RGB del Kinect, y posteriormente con ayuda de la función de "*CoordinateMapper*" de Kinect Fusion se hace una superposición de los puntos extraídos de la imagen de color a la de profundidad mediante vectores de puntos en el espacio de la cámara "*CameraSpacePoints*" que es una estructura dentro de Kinect Fusion. Esto se hace porque como ya mencionamos con anterioridad, la cámara de color y el sensor infrarrojo para la profundidad están muy cercanos el uno del otro, pero no exactamente en la misma posición, por lo que sus ángulos de visión son ligeramente distintos y necesitan alinearse.

Una vez que los puntos sean mapeados desde los cuadros de color y profundidad al espacio de la cámara y subsecuentemente al espacio de coordenadas globales. Se realiza un filtrado de aquellos puntos que se obtuvieran como "ceros" ya sea porque el sensor Kinect tuvo errores para determinar su posición, porque estaban muy cerca o muy lejos del sensor en sí (figura 3.5), o porque son puntos aislados demasiado lejos del objeto como para tener sentido.

El guardado de cada punto se hace en una estructura que contendrá su vector de coordenadas 3D, su vector descriptor de 128 bin, un vector de valores RGB que contendrán el color del pixel en sí (esto ayuda como entrada al reconocedor para tener una mejor estimación), y un número identificador para que sepamos en todo momento de a qué número de captura pertenecen los puntos.

3.3.2 – Conjunción de SIFT y FLANN para la descripción de características.

Posteriormente de que se hayan guardado los puntos de todas las capturas en una estructura, se hace un barrido final a lo largo de los mismos con FLANN, la idea es hacer un "*matching"* en cada una de las capturas tomadas, a lo largo de un rango de 5 capturas, es decir; Al llegar a la captura "n", se hará un "*matching*" desde la captura: "n-2" hasta la captura: "n+2" para el número "M" de capturas que se hayan tomado durante el modelado.

El objetivo de este "*matching*" es filtrar sólo aquellas características con mayor incidencia de detección de un buen número de perspectivas de vista del objeto, por lo menos tres. Una vez hecho el "*matching*" a lo largo de las 5 capturas para una captura "n", FLANN nos arrojará una lista de los puntos clave que encontraron correspondencia entre dos capturas, de modo que el "*matching*" debe hacerse de forma incremental, primero la captura "n-2" con la "n-1", a ese resultado hacerle un "*matching*" con la captura "n", a ese resultado hacerle un "*matching*" con la captura "n+1" y finalmente a ese resultado hacerle un "*matching"* con la captura "n+2".

Posteriormente, debe hacerse un proceso de "rastreo" donde se ubicarán exactamente de donde proviene cada punto del match y cuántas veces se repitió cada uno, al final sólo se considerarán para el modelo final los puntos que tuvieron al menos 3 correspondencias y máximo 5 (porque como el rango es de 5 capturas es el máximo número de veces que pueden repetirse).

El rango puede modificarse, al igual que el número de correspondencias por punto, esto se tomará en cuenta para los experimentos que se harán del modelador en conjunto con el reconocedor de objetos de Golem.

Ya que se terminaron de rastrear los puntos a sus respectivas capturas, en aquellos que tuvieron cierto número de correspondencias en varias perspectivas se hará un promedio de sus coordenadas, sus descriptores y sus valores de color, que se guardarán en el modelo final. Siempre cuidando que aquellos puntos que ya fueron considerados, sean marcados y evitados posteriormente para evitar información redundante. (Si hay un punto que se repite 5 veces, al hacer un barrido de los que se repitieron 4 veces, es posible encontrarnos con estos puntos de nuevo).

Se hace un promedio de las coordenadas que se le resta a cada una para así localizar el sistema de referencia en el centro del objeto, y finalmente se guardan en un archivo con formato *.xml* que servirá como entrada al reconocedor de objetos.

Fig. 3.9 – Matching entre dos capturas realizado con FLANN para características SIFT en OpenCV.
El filtro aquí utilizado a pesar de funcional, falta ser refinado a una tolerancia menor ya que muchas correspondencias son erróneas debido a que se detectan "match" donde no los hay, pero igual se consideran ya que los descriptores son parecidos.
(El matching se realizó exclusivamente para este trabajo pero las capturas se obtuvieron online [37]).

3.3.3 – Interfaz gráfica.

Para facilitar el manejo del modelador de objetos 3D, se utilizan las funciones dentro de una interfaz gráfica por practicidad, la interfaz se realizó en Qt 5.7 con compatibilidad para Microsoft Visual Studio 2015 en 64 bits. La interfaz incluye dos objetos de visualización principales y cuatro botones, para asegurar que el proceso de modelado sea simple y certero, la mayor parte del trabajo ya lo hace el algoritmo que se desarrolló para este trabajo.

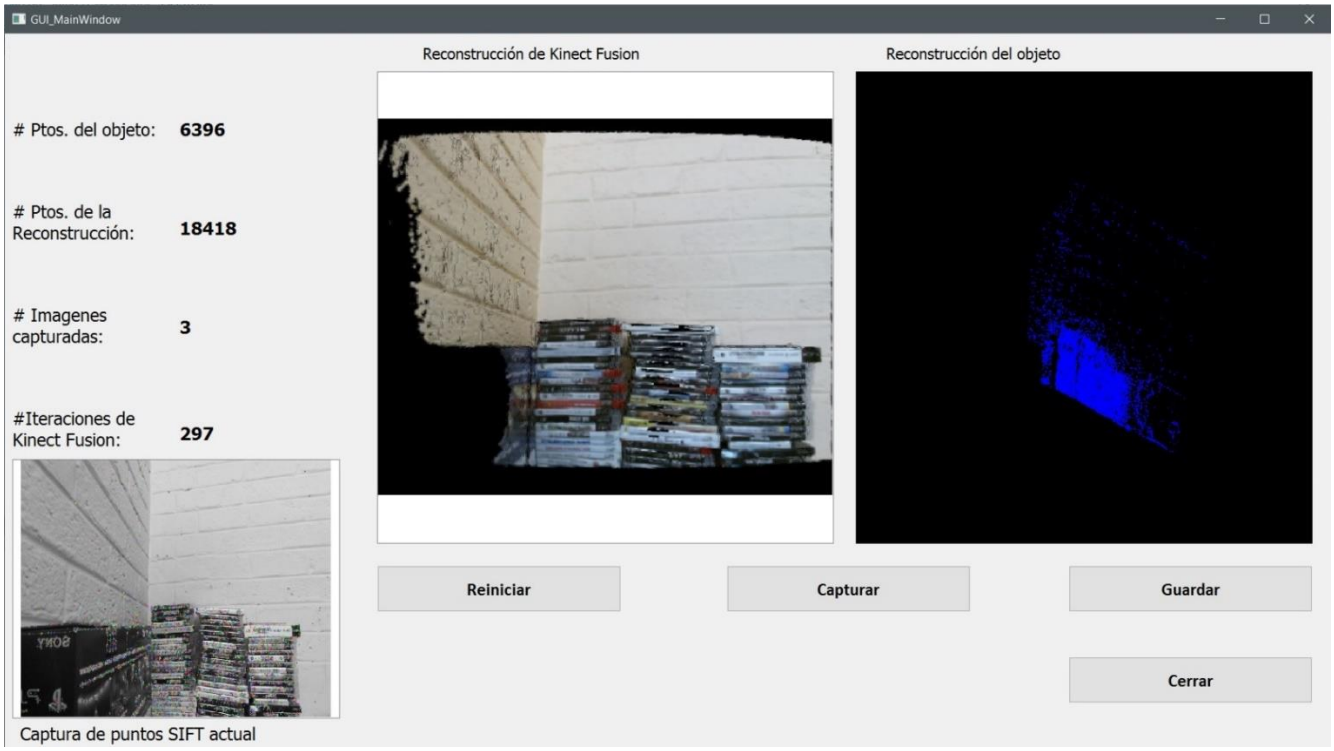

Fig. 3.10 – Interfaz gráfica del modelador de objetos 3D para el robot Golem III.
Se puede visualizar la reconstrucción con Kinect Fusion en tiempo real de una pila de discos así como la reconstrucción de nube de puntos 3D del objeto. Nótese que hay puntos aislados de la pared que entran al modelo, por lo que se requiere un filtro para sólo enfocar el objeto.

A continuación se describirán los componentes de la interfaz y sus respectivos funcionamientos:

Visualizador de la reconstrucción de Kinect Fusion.

En esta pantalla se puede visualizar el renderizado actual obtenido del volumen de reconstrucción generado con las imágenes de color y profundidad, nos ayuda a calcular distancias adecuadas a las cuales tomar las capturas (si el sensor está muy cerca del objeto el haz infrarrojo puede no detectar correctamente las distancias y esto se ve reflejado en el visualizador como manchas negras), así como la correcta actualización de la matriz de transformación de la cámara al mundo que puede ser incorrecta si las nuevas vistas del volumen comienzan a traslaparse con las anteriores.

Visualizador de la reconstrucción del objeto con OpenGL.

Es un objeto con una clase de OpenGL (librería de código abierto de graficación) heredada que actúa como un lienzo que va dibujando los puntos del modelo preliminar del objeto conforme se van tomando las capturas, se puede manipular con el mouse y rotarse en todas direcciones para dar un mejor sentido de lo que sucede con la reconstrucción. Uno puede darse cuenta si hay puntos aislados que no

pertenecen al objeto que sin embargo fueron capturados si aparecen en el visualizador, el mismo objeto que controla al visualizador le pregunta cada medio segundo al modelo preliminar cuál es su información para redibujar la escena constantemente en tiempo real.

Botones de las funciones principales.

Dentro de los cuatro botones de la aplicación, el botón cerrar cierra la aplicación y detiene todos los procesos de reconstrucción, el de reiniciar se encarga de hacer un reseteo al objeto de reconstrucción para volver a obtener sus valores de nube de puntos y renderizado que permita su visualización en pantalla, así como reiniciar la matriz de posición de la cámara a la última posición obtenida. El botón de capturar se encarga de obtener una captura de todo lo que está viendo el sensor en ese instante y guarda los puntos en un arreglo preliminar y finalmente el botón guardar se encarga de hacer el matching con FLANN del cual se habló en el subcapítulo pasado, guarda el modelo final y cierra la aplicación.

Elementos adicionales.

Otros de los elementos de la interfaz son contadores que nos permiten saber el número de puntos del objeto detectados en la captura actual, el número total de puntos de la reconstrucción, la cantidad de imágenes capturadas y el número de iteraciones que lleva Kinect Fusion desde que se inicia el programa, (se hace una iteración donde se obtienen nuevas imágenes de color y profundidad y se agregan al volumen de reconstrucción cada 300ms).

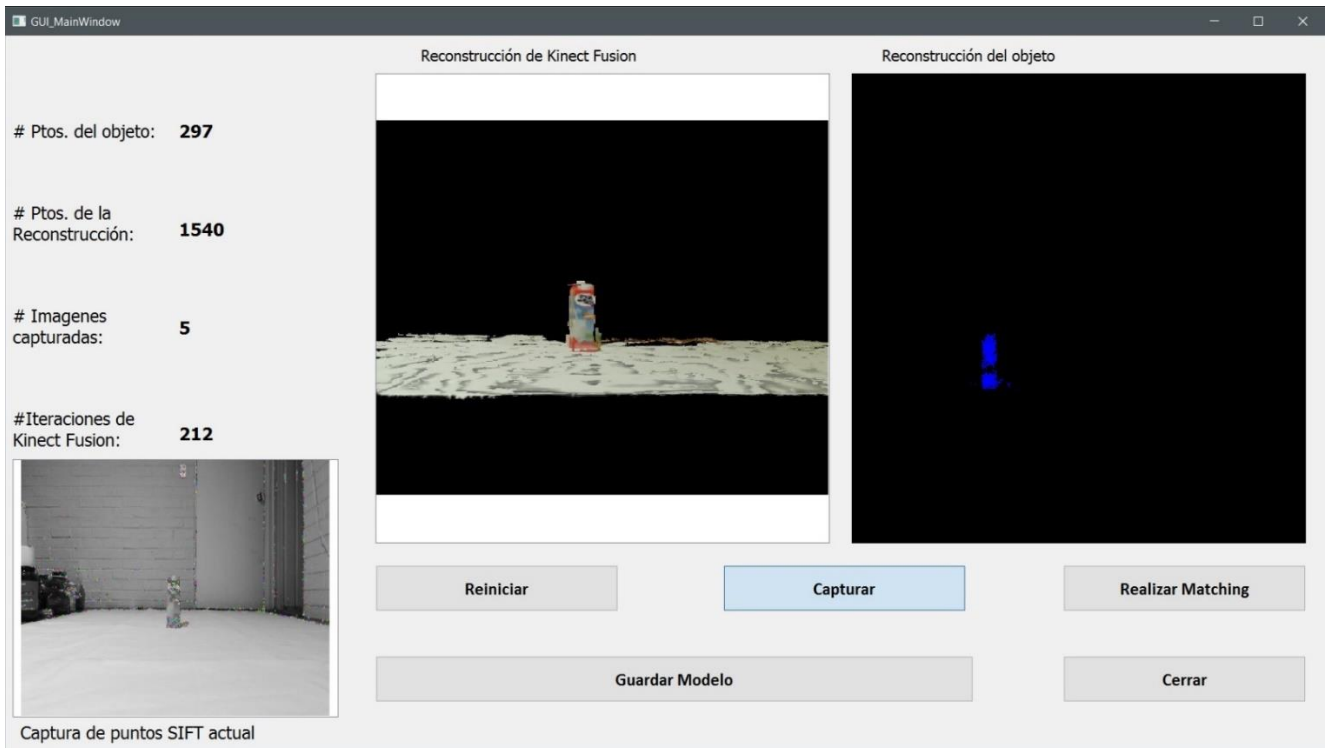

Fig. 3.11 – Versión preliminar de la interfaz gráfica del modelador. A pesar de ser una versión ya funcional en la medida de generar modelos, esta cuenta con un botón extra innecesario: "Realizar matching" ya que durante el proceso se ideó dividir el proceso del matching en partes para ver cómo se comportaban los modelos. Se guardaban los matchings de determinado número de capturas y al final se conjuntaban en el modelo final.

CAPÍTULO 4.

4 – Experimentos.

4.1 – Metodología.

Para la metodología de pruebas y experimentos me basé en la obtención de por lo menos diez modelos de diez objetos distintos respectivamente, que sirvieron para probar las características del modelador así como su eficiencia en el reconocimiento con al menos cien escenas distintas, que consistieron en cien fotografías a color en ambientes y entornos distintos donde varían aspectos como iluminación, oclusión (muchos objetos en una escena), cubrir parcialmente el objeto de interés, distancia al objeto de interés y vistas en diferentes perspectivas (frontal, lateral, isométrico, atrás).

Específicamente lo hice de la siguiente manera:

Utilicé un conjunto de diez objetos comunes o de fácil acceso para el banco de pruebas:

1. Caja de leche Forti.
2. Empaque de avena Quaker.
3. Empaque de jugo Bonafina.
4. Lata de verduras.
5. Empaque de jugo Boing.
6. Envase de Shampoo Caprice.
7. Caja de pasta dental.
8. Caja de salsa de tomate.
9. Caja de té de flores.
10. Caja de tinte para el cabello.

Fig. 4.1 – Conjunto de diez objetos principales para probar el modelador de objetos 3D junto con el reconocedor MOPED de Golem.

Originalmente se utilizaron cuatro objetos adicionales en el banco de pruebas; Un envase de crema Alpura, un envase de Parmesano, un jugo de la cosecha pura y un antiadherente para cocinar, sin embargo, estos objetos fueron descartados después de una prueba inicial de reconocimiento dado que no contaban con textura suficiente para poderles extraer las características naturales SIFT necesarias. Subsecuentemente utilicé estos mismos objetos pero para hacer pruebas de reconocimiento en conjunto con otros objetos conocidos y desconocidos dentro de la escena.

Fig. 4.2 – Conjunto original de objetos planteados para el banco de pruebas, al final se excluyeron cuatro objetos debido a falta de textura en los mismos y se intercambiaron por otros más viables.

La metodología que se siguió fué que se utilizaran diez imágenes por cada objeto, si son diez objetos, esto nos dará un total de 100 imágenes para nuestro banco de pruebas, más otras diez imágenes adicionales que sirven para probar objetos en conjunto y en casos particulares que podrían resultar interesantes.

Se prueban en concreto tres aspectos principales en el reconocimiento con el modelador de objetos; Distancia, iluminación y perspectiva o punto de mira del objeto.

De las 10 imágenes que usé por cada objeto;

- Tres imágenes fueron enfocadas a probar la distancia (cerca de la cámara, a media distancia y lejos de la cámara).
- Tres imágenes trataron el aspecto de la iluminación en interiores (iluminación clara o luz de día, iluminación media o moderada en un horario vespertino e iluminación oscura en un horario nocturno.
- Cuatro imágenes fueron en relación a la perspectiva desde cómo se mira el objeto (vista frontal, lateral, por la parte de atrás y finalmente una vista isométrica).

De este modo me aseguré de cubrir aspectos del modelado y reconocimiento en relación a la escala invariante (distancia a la cámara), características naturales extraídas a distintos grados de iluminación y también distintas perspectivas cubiertas por el modelo 3D de nube de puntos del objeto.

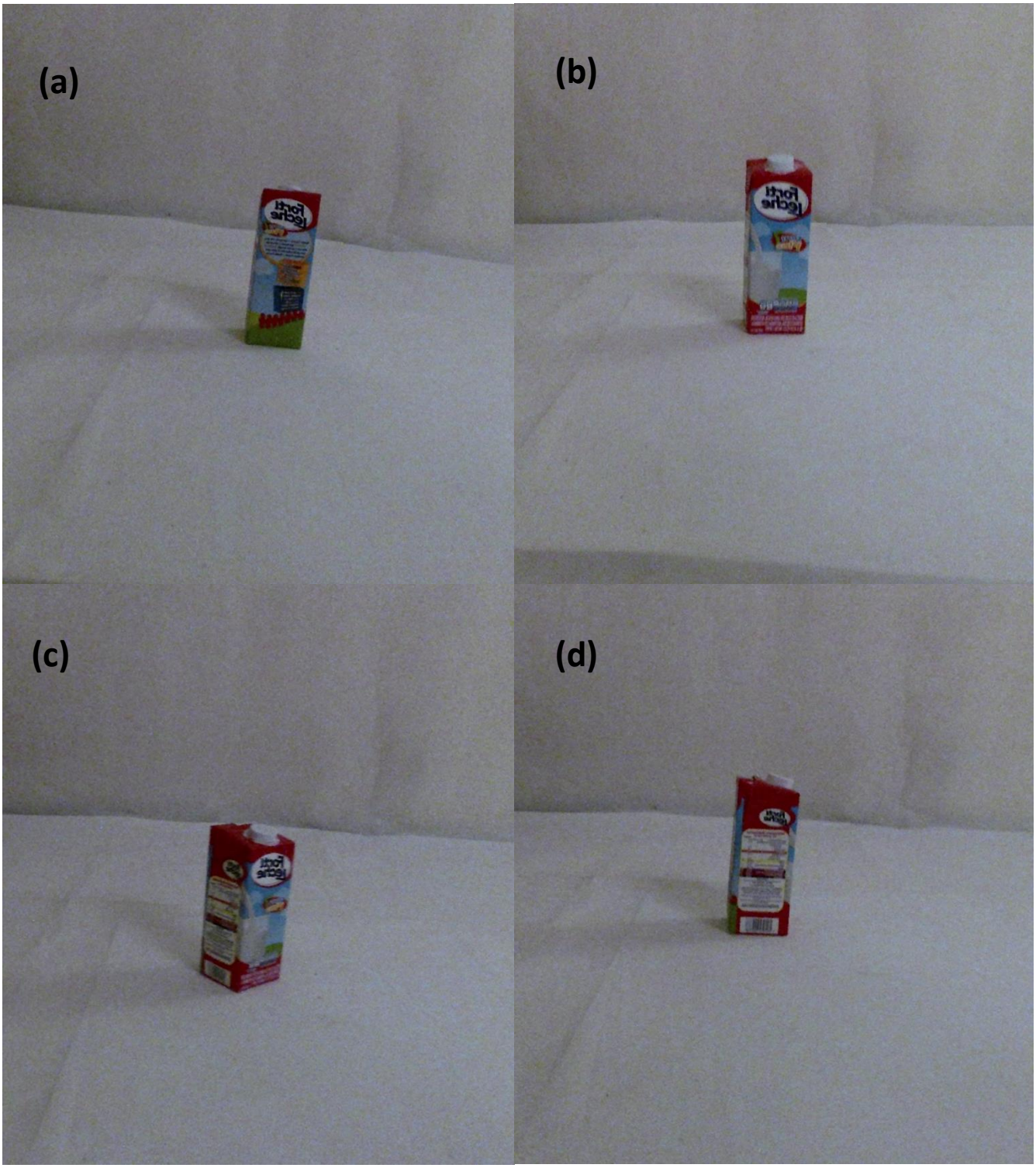

Fig. 4.3 – Imágenes de prueba para el reconocimiento de una leche Forti variando la perspectiva. (a)Vista de atrás. (b)Vista frontal. (c)Vista isométrica. (d)Vista lateral.

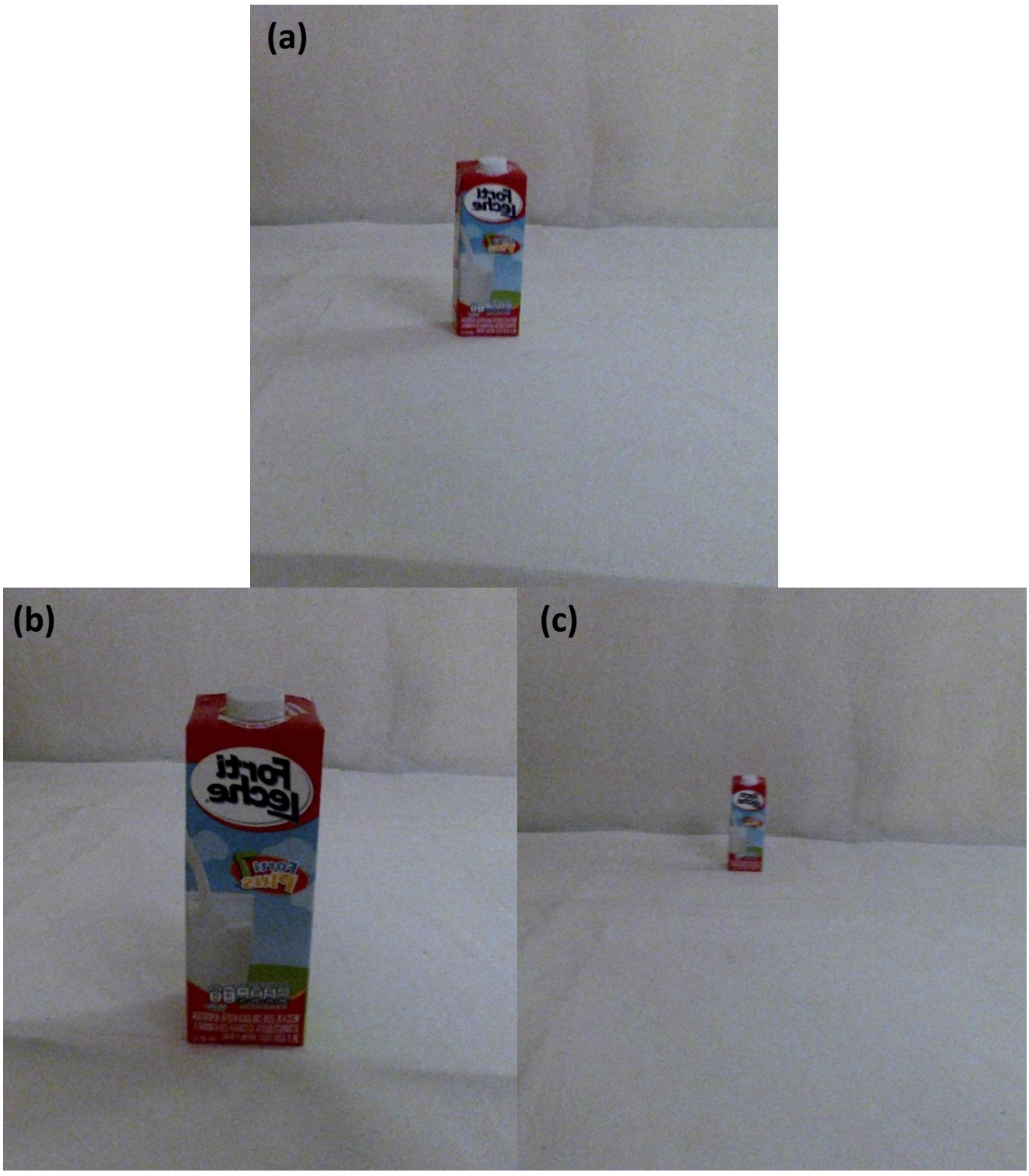

Fig. 4.4 – Imágenes de prueba para el reconocimiento. Distintas distancias hacia la cámara de una Leche Forti. (a)Distancia media, (50cm). (b)Distancia cercana a la cámara, (20cm). (c)Distancia lejos de la cámara, (80cm).

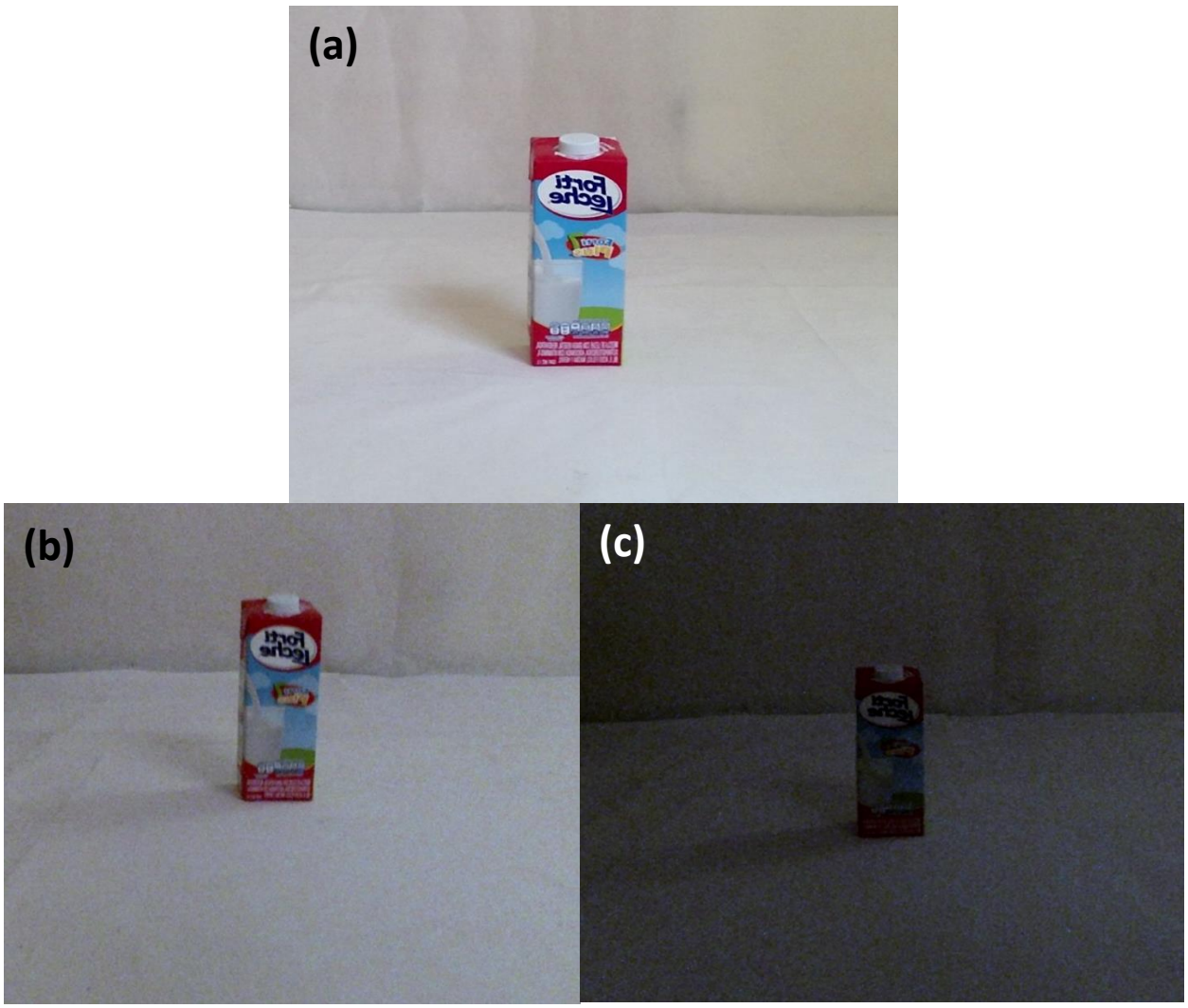

Fig. 4.5 – Imágenes de prueba para el reconocimiento. Distintas condiciones de iluminación en el objeto (Leche). (a)Iluminación clara, luz de día. (b)Iluminación media, horario vespertino. (c)Iluminación oscura, horario semi-nocturno.

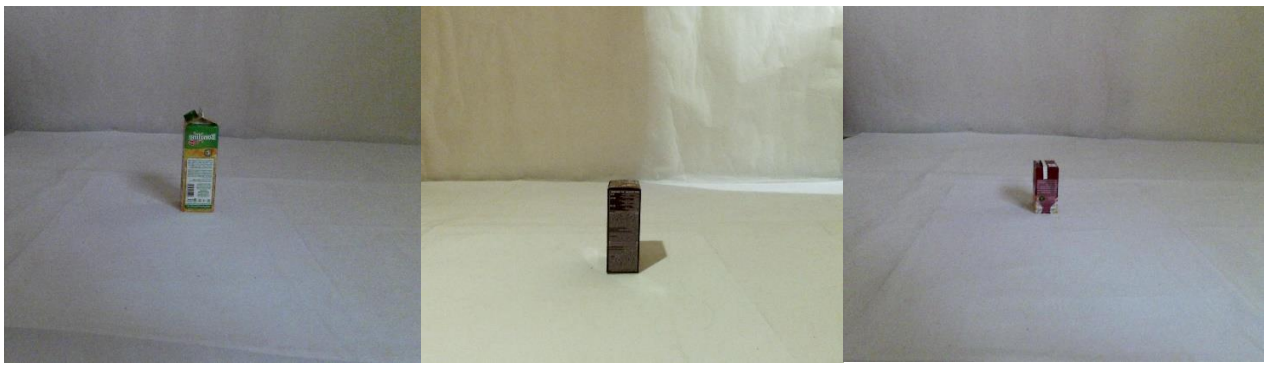

Fig. 4.6 – Perspectiva lateral para distintos objetos. (De izquierda a derecha; Jugo Bonafina, tinte para el cabello y una caja de salsa de tomate).

De igual manera, como ya mencionamos se usarán algunas imágenes de escenas con objetos en conjunto y reconocimiento de distintos modelos en estas escenas.

Fig. 4.7 – Conjunto de algunas imágenes que muestran agrupaciones de distintos objetos conocidos y no conocidos para probar la eficiencia de los modelos en circunstancias de oclusión (parcialmente cubiertos por otros objetos), variaciones en perspectiva, distancia y reconocer objetos discerniéndolos de otros en el mismo entorno.

4.2 – Resultados.

Los resultados obtenidos de los experimentos me dieron información importante sobre el funcionamiento del modelador de objetos 3D para el robot Golem III, desde en qué circunstancias los modelos son más favorables, el procedimiento ideal para obtener los mismos, y algunas recomendaciones para evitar posibles errores en el reconocimiento, así mismo me doy cuenta de cuáles son las limitaciones del sistema propuesto y como podría mejorarse u optimizarse para su integración en Golem.

Para medir los resultados me basé en una recopilación y análisis de datos que me arrojó el reconocedor de objetos 3D basado en MOPED que está implementado actualmente en Golem, me fijé en la cantidad de correspondencias encontradas entre la escena y el modelo generado del objeto, la cantidad de "clusters" o agrupaciones de correspondencias entre características de la escena y el modelo que son espacialmente cercanas entre sí, también medí la cantidad de objetos detectados por MOPED en la primera estimación de ICE, donde la cantidad de iteraciones en RANSAC se mantiene alta en contraste con la cantidad de iteraciones de Levenberg-Marquardt que se mantiene baja, esto para descubrir múltiples hipótesis de objetos aunque con una estimación de posición inicial algo burda, después medí la cantidad de objetos reconocidos por MOPED en la segunda estimación de ICE, que al contrario que la primera, las iteraciones en RANSAC son bajas y en LM son altas para conseguir buenas estimaciones de posición para cada agrupación de agrupaciones de correspondencias que sea consistente con la geometría multi-perspectiva del objeto. Lo ideal en esta última medición fué obtener una única estimación de posición, puesto que sólo realicé pruebas para una escena y un modelo 3D de nube de puntos por objeto en cada vez. Y finalmente medí el grado de reconocimiento de los objetos dentro de la escena de acuerdo con las etapas que realiza MOPED, es decir; Qué tan lejos en las etapas de reconocimiento de MOPED llegó ese modelo para esa escena en particular, las agrupé de la siguiente manera:

- **Etapa o grado 1** = Extracción de características naturales SIFT de la escena.
- **Etapa o grado 2** = Obtención de correspondencias entre la escena y el modelo del objeto guardado.
- **Etapa o grado 3** = Agrupación de correspondencias entre características espacialmente cercanas entre sí en conjuntos de las mismas.
- **Etapa o grado 4** = Detección de por lo menos un objeto en la primera estimación de ICE en MOPED.
- **Etapa o grado 5** = Obtención de conjuntos de otros conjuntos de correspondencias que potencialmente pueden pertenecer al mismo objeto único.
- **Etapa o grado 6** = Reconocimiento de uno o más objetos en la segunda estimación de ICE en MOPED con su correspondiente estimación de la posición, independientemente de si esta está correctamente orientada o no.
- **Etapa o grado 7** = Filtrado y recombinación de la posición del objeto para que esté correctamente centrada en el mismo.

Con estas cinco mediciones principales por cada conjunto escena-modelo realicé un análisis de resultados para cuantificar la eficiencia del modelador de objetos 3D a lo largo de las 110 imágenes o escenas y los 10 objetos con sus respectivos modelos en nube de puntos.

Los datos los obtuve mediante la interfaz del reconocedor de Golem en cada caso.

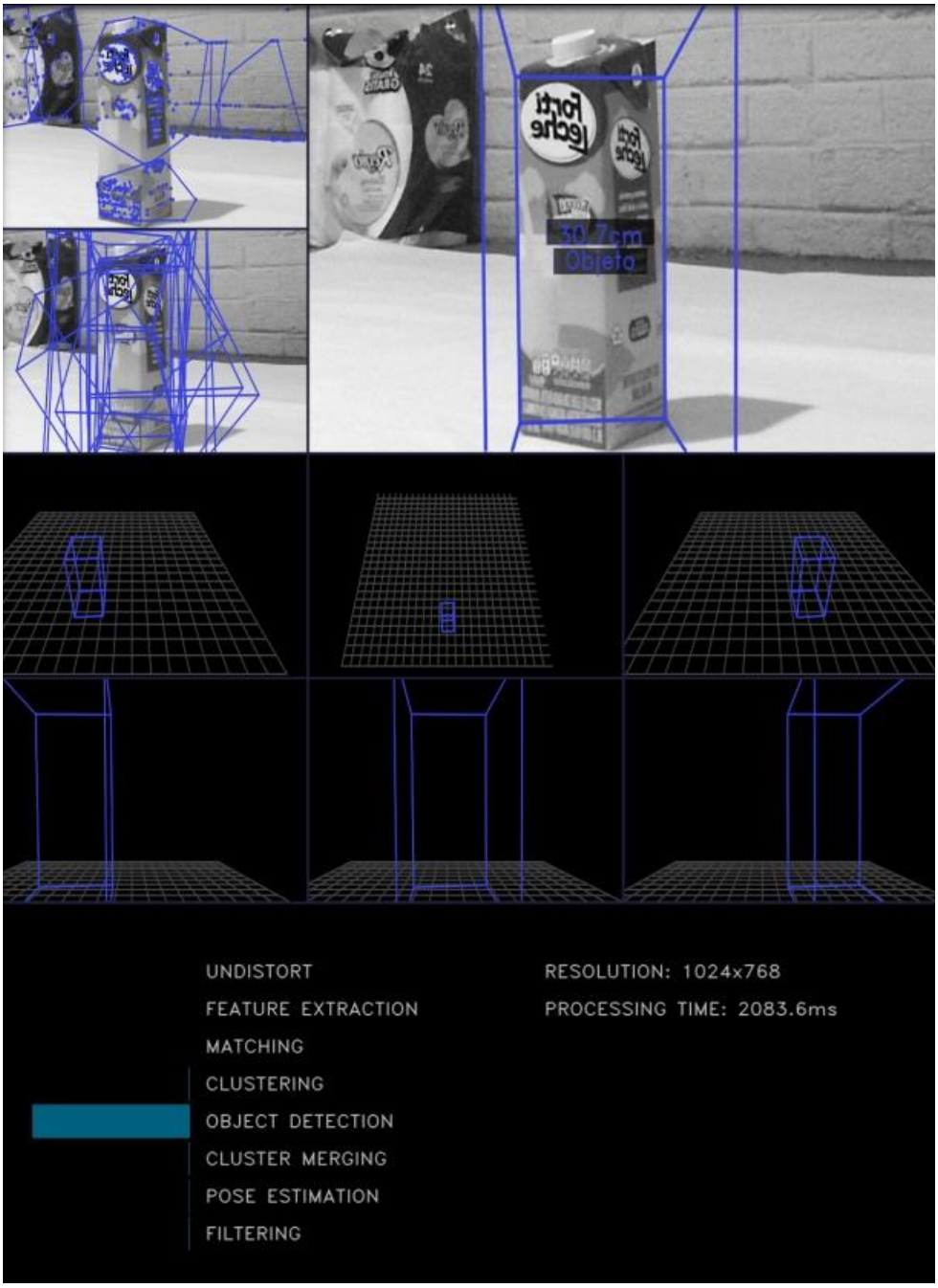

Fig. 4.8 – Reconocimiento de un cartón de leche utilizando el reconocedor MOPED de Golem.

En el cuadro superior izquierdo se aprecia la etapa de obtención de correspondencias y agrupaciones de las mismas en la escena, en el cuadro inmediatamente debajo de este se observa la primera estimación de ICE y detección de objetos a partir de esas agrupaciones de correspondencias, sin embargo la estimación de posición no es muy buena, en el cuadro a la derecha de este último (un poco más grande) tenemos la estimación de posición del objeto y reconocimiento del mismo si se encontró, el equivalente a la segunda estimación de ICE en MOPED. En los cuadros inferiores vemos la localización del objeto en relación a la escena observable y datos estadísticos como el tiempo de procesamiento, resolución de la imagen y una relación de las etapas de MOPED ya mencionadas, (en este caso como se dieron múltiples detecciones del objeto en la primera estimación con relación a la segunda que fue sólo una, nos muestra una barra azul grande en “Object Detection”, si hubieran más detecciones de objetos

en la segunda estimación, la barra azul de “Pose Estimation” sería un poco más grande). Aquí el grado de reconocimiento o etapa alcanzada en MOPED sería 6: Reconocimiento del objeto y estimación de la posición en su segunda estimación de ICE, por lo que podría considerarse “buena”, sin embargo está incompleta puesto que no llega a obtener una recombinación adecuada de la posición, en la etapa 7 de MOPED, que al ser la final se consideraría un reconocimiento exitoso.

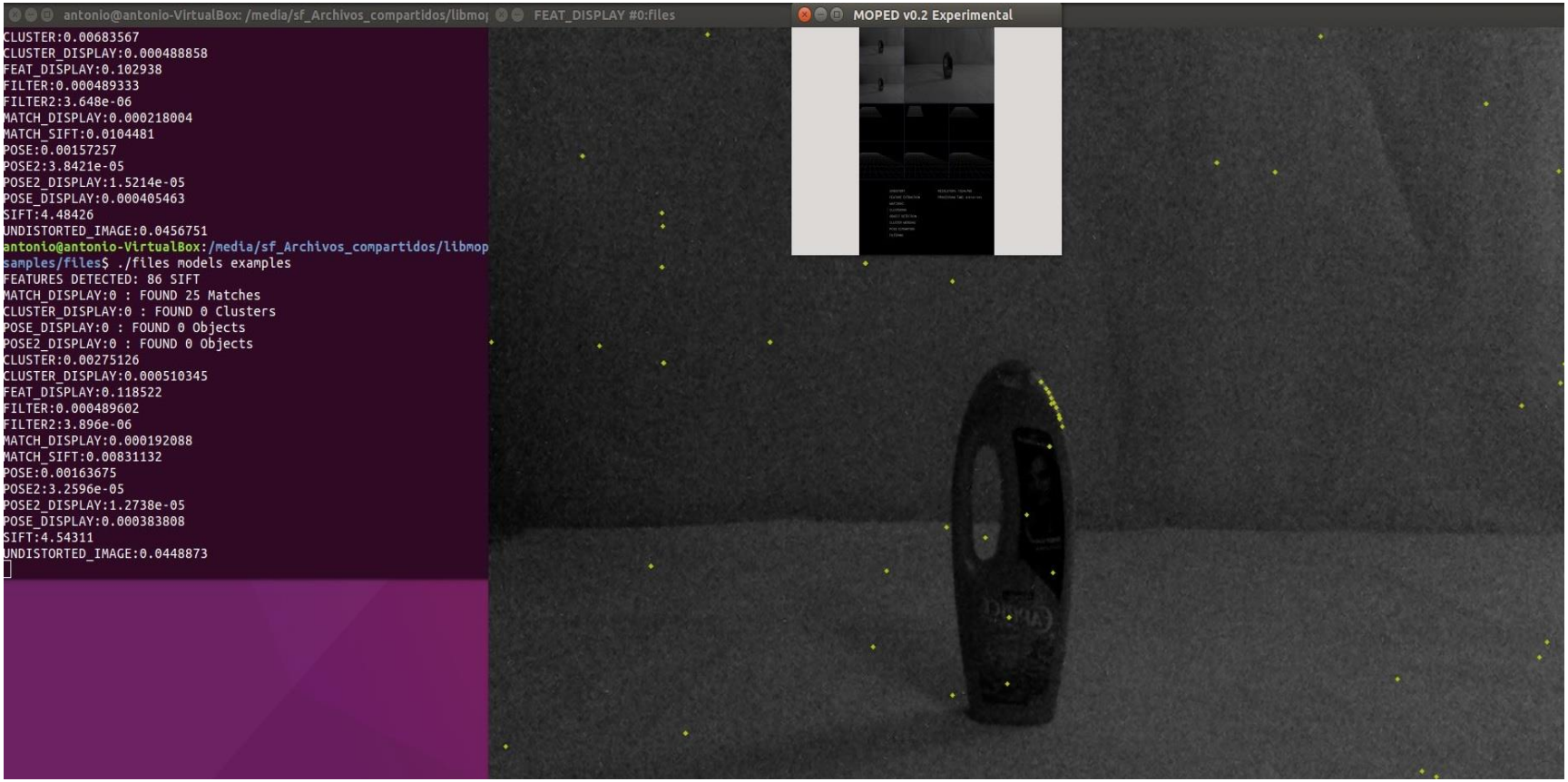

Fig. 4.9 – Interfaz del reconocedor de objetos de Golem en Linux.

La pantalla muestra la extracción de características SIFT de la escena, en este caso de la perspectiva frontal de un shampoo con condiciones de iluminación oscura o semi-nocturna, tanto en la pantalla que muestra el reconocimiento MOPED (aquí se muestra más pequeña), como en la aplicación de consola, podemos notar que no se encontraron correspondencias suficientes entre la escena y el modelo, (apenas unas 25), por lo cual no se obtuvieron agrupaciones de características, ni tampoco se pudieron generar hipótesis de objetos a partir de las mismas; “Found 0 clusters”, “Pose_display: Found 0 Objects” y “Pose2_display: Found 0 Objects”. Por lo que se considera un reconocimiento fallido que sólo alcanzó el grado o etapa 2 de MOPED: Obtención de correspondencias.

Volviendo a las diez imágenes iniciales planteadas en la metodología para medir el reconocimiento de cada objeto individualmente, (sin escenas de conjuntos de objetos), tenemos que variaríamos la perspectiva, distancia e iluminación, los resultados obtenidos por MOPED para un empaque de jugo se muestran a continuación:

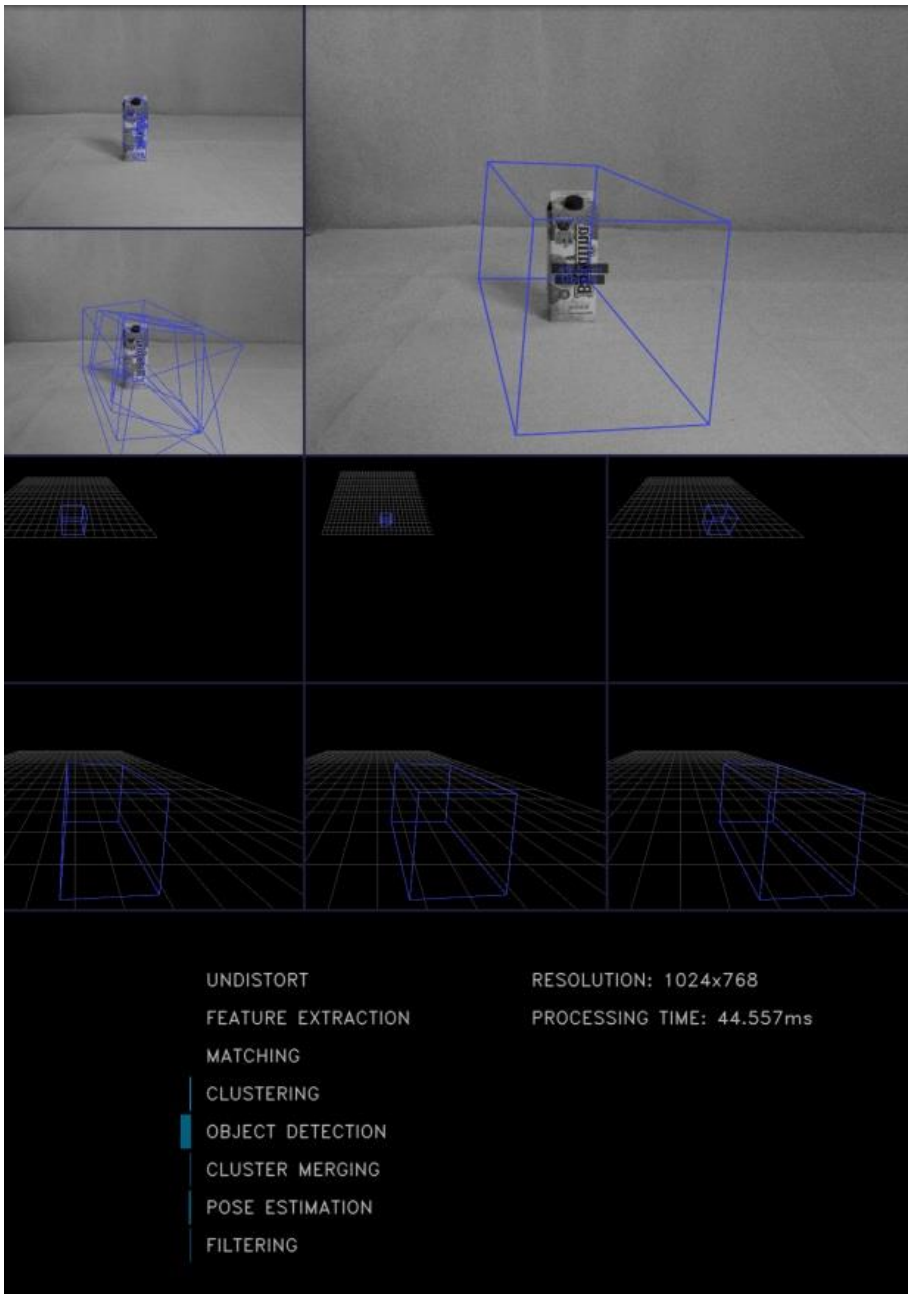

Fig. 4.10 – Reconocimiento de un empaque de jugo en condiciones estándar, (distancia e iluminación media con una perspectiva frontal). A pesar que el reconocimiento del objeto es efectivo, y hay una buena estimación de la pose aproximada, la orientación de la posición no está correctamente centrada en el mismo y abarca un rango demasiado amplio. Esto puede generarse cuando el modelo tiene puntos aislados que se guardan al resto de "puntos buenos" durante el proceso de modelado, o también puede deberse a un modelo no completamente cerrado o con ciertos errores de alineación, de lo cual hablaré más adelante.

El procedimiento para modelar los objetos que utilicé consiste en darle vueltas al objeto tomando capturas de distintas perspectivas, para filtrar puntos del entorno, utilicé una lona blanca como solución sencilla, sin embargo es posible que por pelusas, cabellos o cualquier otra pequeña imperfección detectada por el sensor puedan darse ciertos errores en la generación de los modelos a forma de ruido, y esto mismo genere errores en el reconocimiento.

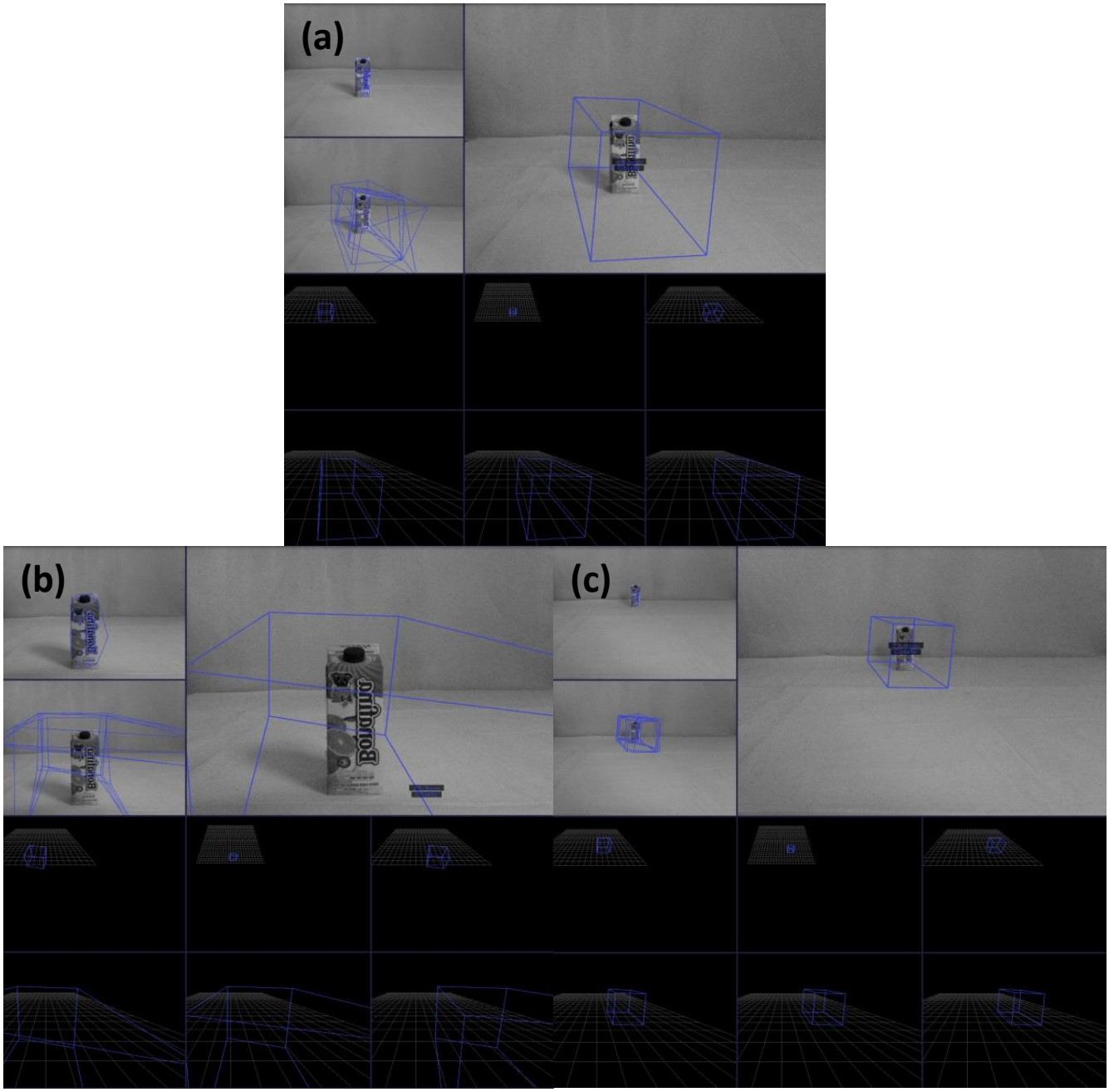

Fig. 4.11 – Reconocimiento de un empaque de jugo a distintas distancias de la cámara. Nótese que en la distancia corta, hay menos rango reconocido del lado derecho, quizás en parte porque tomé menos puntos para el modelo de este lado. (a)Distancia media a aproximadamente 50cm. (b)Distancia corta a unos 20cm. (c)Distancia larga a 80cm.

En general para este objeto se obtuvieron buenos resultados en el reconocimiento a diferentes distancias, con una estimación, aunque imperfecta, de la posición, cabe señalar que aunque el rango sea amplio y quizás esto no facilite la manipulación del robot con el objeto, la localización sigue siendo robusta, por lo que quizás con un pequeño filtro en la supuesta directriz de manipulación podría arreglarse este inconveniente. (Como que se enfoque sólo en el centro del área de reconocimiento detectada).

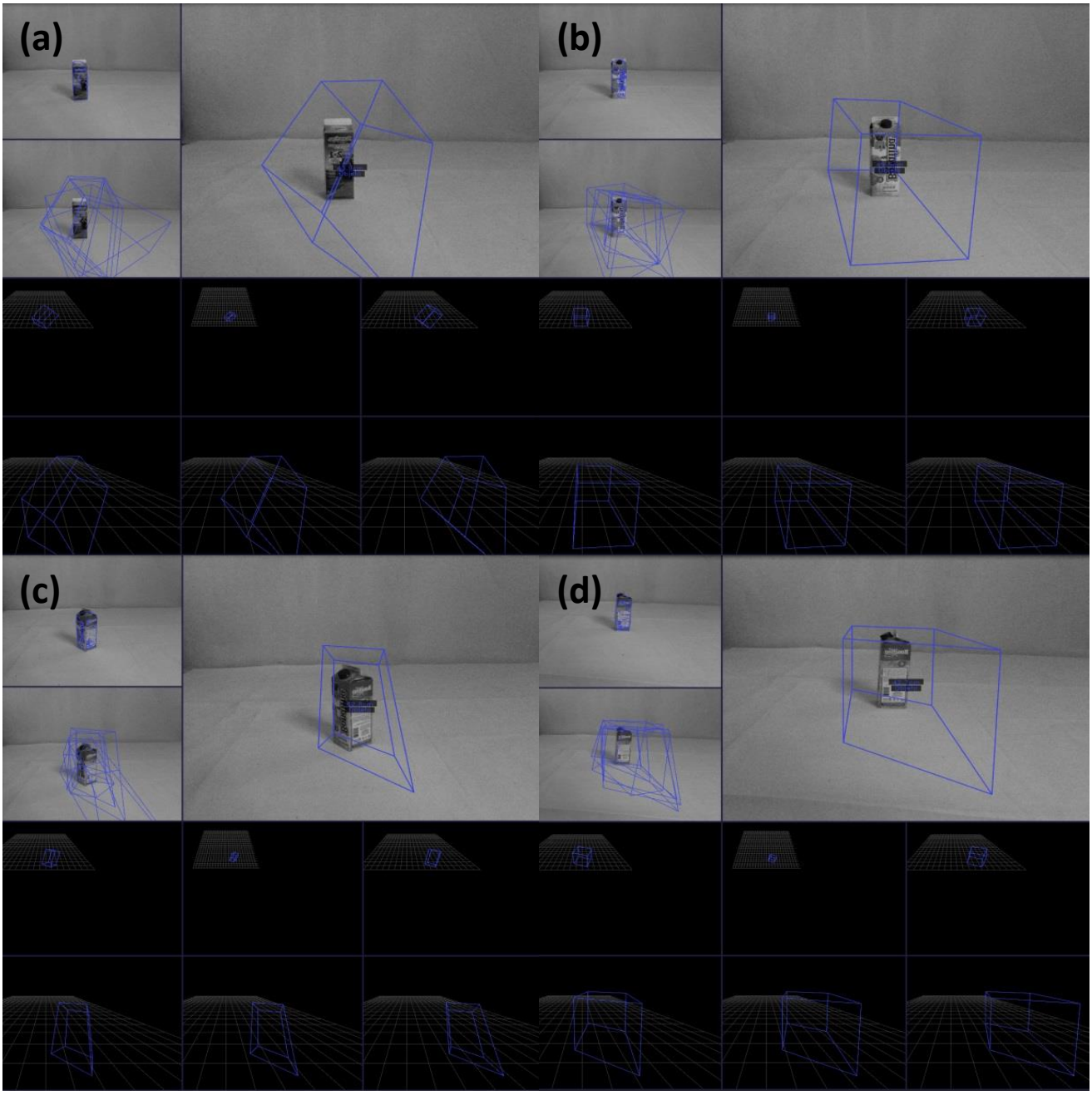

Fig. 4.12 – Reconocimiento de un empaque de jugo a distintas perspectivas con el mismo modelo. (a)Perspectiva de atrás. (b)Perspectiva frontal. (c)Perspectiva isométrica. (d)Perspectiva lateral. Sin embargo cabe señalar que aquí es más notorio el segundo problema del modelador de objetos, la correcta alineación de puntos para conseguir una estimación de la posición estable junto a su orientación.

Para la alineación de puntos en el modelo, inicialmente se confió completamente en la etapa de integración y procesamiento de cada nuevo cuadro de mira de la cámara al volumen de reconstrucción de Kinect Fusion, al obtener los puntos en el espacio de la cámara a partir de las imágenes de color y de profundidad alineadas, se hacía una multiplicación por la matriz actual de transformación del espacio de la cámara al espacio de coordenadas globales, sin embargo muchas veces, la alineación de puntos era pésima e incluso los puntos entre una perspectiva y otra eran separados por distancias grandes de espacio. Una alternativa que le dio una solución parcial a este problema fué la de implementar ecuaciones de movimiento rígido de mínimos cuadrados utilizando SVD (descomposición en valores

singulares) en la etapa de captura del modelador, para que entre una y otra captura, fuera realizando una alineación parcial de la captura actual con la guardada previamente, a pesar que esto solucionó enormemente el problema de la distancia entre puntos de una captura y otra, (poniéndolos todos a la misma distancia gracias a la matriz de transformación de traslación), parece que la rotación entre puntos no fué del todo efectiva, por lo que los modelos siguen siendo en ocasiones inestables en su reconocimiento, específicamente en la etapa de recombinación de la pose realizada por MOPED, ya que no es capaz de realizar una estimación de la orientación precisa.

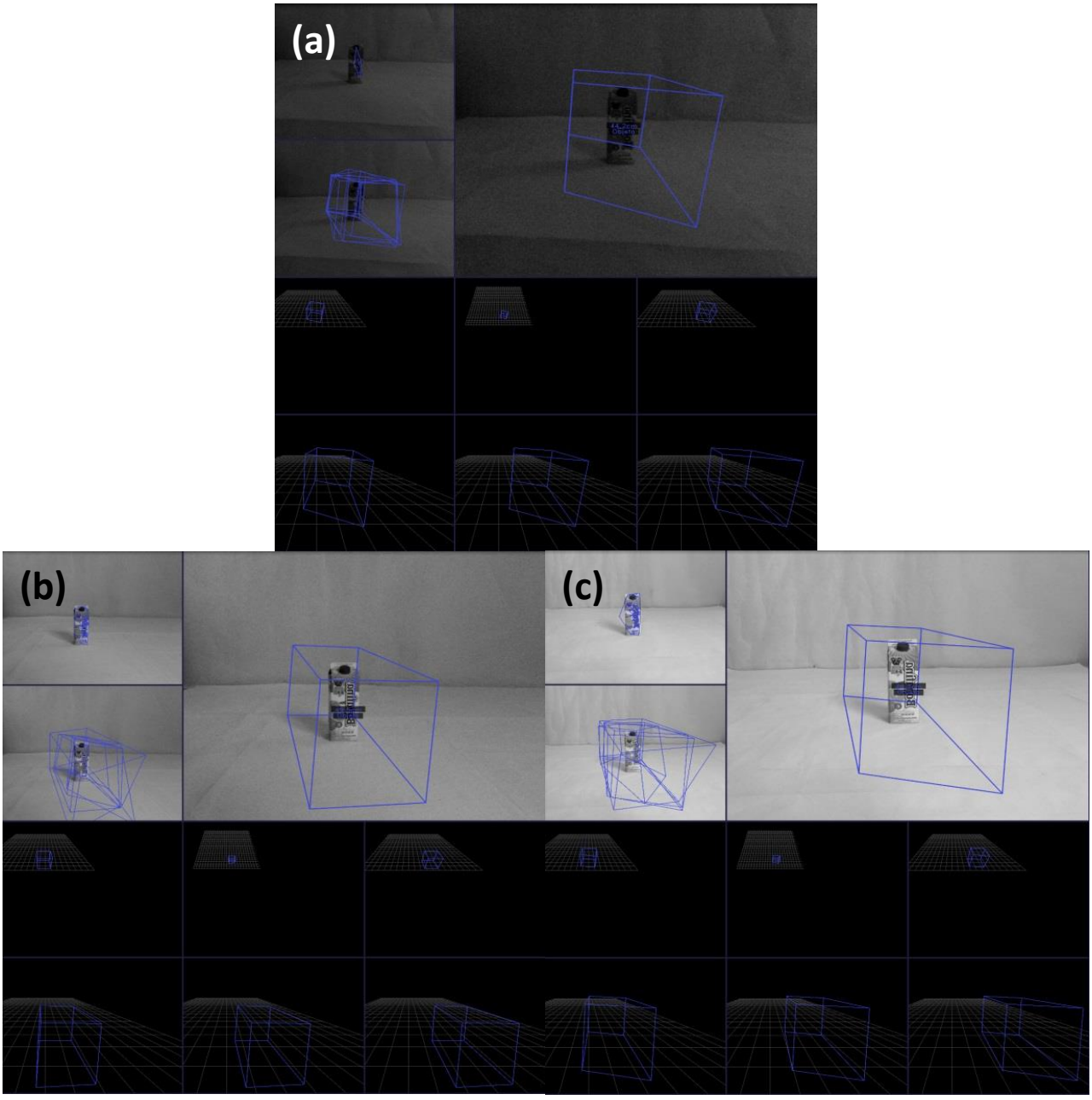

Fig. 4.13 – Reconocimiento de un empaque de jugo a distintas condiciones de iluminación. (a)Iluminación oscura o semi-nocturna. (b)Iluminación media o vespertina. (c)Iluminación clara o con luz de día.

A pesar que la iluminación oscura puede crear dificultades para notar las características naturales del objeto con la misma intensidad, en este caso el modelo fué lo suficientemente robusto para ser capaz de leerlas, sin embargo esto no siempre ocurre.

A continuación se presentan los resultados en conjunto para todo el banco de pruebas de 100 imágenes y 10 objetos o modelos obtenidos, en cada una de las 5 categorías mencionadas previamente, (cantidad de correspondencias, agrupaciones, objetos en la primera y segunda estimación de MOPED y su grado o etapa de reconocimiento alcanzada).

Cantidad de correspondencias encontradas en Objetos (Lotes 1 y 2)

Capturas	Avena	Jugo Boing	Jugo Bonafina	Lata de verduras	Leche	Pasta dental	Salsa de tomate	Shampoo Caprice	Té de flores	Tinte del cabello
Distancia 1 - Cerca	245.00	484.00	408.00	179.00	231.00	419.00	350.00	133.00	244.00	237.00
Distancia 2 - Media	250.00	319.00	236.00	94.00	139.00	265.00	151.00	82.00	145.00	148.00
Distancia 3 - Lejos	72.00	112.00	47.00	30.00	39.00	98.00	66.00	34.00	63.00	76.00
Iluminación 1 - Claro	260.00	338.00	315.00	161.00	232.00	274.00	256.00	277.00	142.00	273.00
Iluminación 2 - Medio	110.00	319.00	236.00	100.00	139.00	177.00	143.00	78.00	150.00	166.00
Iluminación 3 - Oscuro	34.00	29.00	44.00	14.00	44.00	75.00	21.00	25.00	77.00	39.00
Perspectiva 1 - Frontal	257.00	319.00	236.00	89.00	142.00	256.00	151.00	78.00	138.00	287.00
Perspectiva 2 - Lateral	151.00	126.00	176.00	50.00	121.00	102.00	61.00	18.00	142.00	134.00
Perspectiva 3 - Atrás	274.00	125.00	108.00	10.00	155.00	180.00	105.00	82.00	116.00	206.00
Perspectiva 4 - Isométrico	243.00	162.00	236.00	104.00	87.00	246.00	236.00	87.00	162.00	222.00

Tabla 1 – Cantidad de correspondencias encontradas por imagen y modelo 3D del objeto.

Agrupaciones de Correspondencias MOPED de los Objetos (Lotes 1 y 2)

Capturas	Avena	Jugo Boing	Jugo Bonafina	Lata de verduras	Leche	Pasta dental	Salsa de tomate	Shampoo Caprice	Té de flores	Tinte del cabello
Distancia 1 - Cerca	2.00	14.00	1.00	1.00	2.00	5.00	1.00	3.00	3.00	2.00
Distancia 2 - Media	1.00	1.00	1.00	1.00	1.00	2.00	1.00	1.00	1.00	1.00
Distancia 3 - Lejos	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Iluminación 1 - Claro	1.00	1.00	1.00	1.00	1.00	3.00	1.00	1.00	1.00	1.00
Iluminación 2 - Medio	2.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	2.00	1.00
Iluminación 3 - Oscuro	0.00	1.00	1.00	0.00	1.00	6.00	1.00	0.00	6.00	2.00
Perspectiva 1 - Frontal	1.00	1.00	1.00	1.00	1.00	3.00	1.00	1.00	1.00	1.00
Perspectiva 2 - Lateral	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00	1.00	1.00
Perspectiva 3 - Atrás	1.00	1.00	1.00	1.00	1.00	3.00	1.00	1.00	1.00	1.00
Perspectiva 4 - Isométrico	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Tabla 2 – Cantidad de agrupaciones generadas a partir de las correspondencias en la escena y el modelo.

Cantidad de Objetos detectados en Generación de Hipótesis MOPED (Estimación 1) (Lotes 1 y 2)

Capturas	Avena	Jugo Boing	Jugo Bonafina	Lata de verduras	Leche	Pasta dental	Salsa de tomate	Shampoo Caprice	Té de flores	Tinte del cabello
Distancia 1 - Cerca	3.00	17.00	4.00	4.00	1.00	3.00	4.00	4.00	1.00	0.00
Distancia 2 - Media	4.00	4.00	4.00	4.00	4.00	3.00	4.00	4.00	3.00	1.00
Distancia 3 - Lejos	4.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00	2.00
Iluminación 1 - Claro	4.00	4.00	4.00	4.00	4.00	3.00	4.00	4.00	4.00	4.00
Iluminación 2 - Medio	4.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00
Iluminación 3 - Oscuro	0.00	4.00	4.00	0.00	4.00	4.00	0.00	0.00	0.00	0.00
Perspectiva 1 - Frontal	4.00	4.00	4.00	4.00	4.00	6.00	4.00	4.00	4.00	4.00
Perspectiva 2 - Lateral	4.00	4.00	4.00	4.00	4.00	4.00	4.00	0.00	2.00	4.00
Perspectiva 3 - Atrás	0.00	4.00	4.00	0.00	4.00	0.00	4.00	2.00	4.00	4.00
Perspectiva 4 - Isométrico	4.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00

Tabla 3 – Cantidad de objetos detectados en la primera estimación de ICE de MOPED.

Cantidad de Objetos detectados en el refinamiento de la posición (Estimación 2) (Lotes 1 y 2)

Capturas	Avena	Jugo Boing	Jugo Bonafina	Lata de verduras	Leche	Pasta dental	Salsa de tomate	Shampoo Caprice	Té de flores	Tinte del cabello
Distancia 1 - Cerca	2.00	1.00	1.00	1.00	1.00	1.00	2.00	1.00	1.00	0.00
Distancia 2 - Media	2.00	1.00	1.00	1.00	1.00	2.00	3.00	1.00	1.00	1.00
Distancia 3 - Lejos	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Iluminación 1 - Claro	1.00	1.00	1.00	1.00	1.00	1.00	2.00	1.00	1.00	2.00
Iluminación 2 - Medio	1.00	1.00	1.00	1.00	1.00	1.00	3.00	1.00	1.00	1.00
Iluminación 3 - Oscuro	0.00	0.00	1.00	0.00	1.00	1.00	0.00	0.00	0.00	0.00
Perspectiva 1 - Frontal	2.00	1.00	1.00	1.00	1.00	2.00	1.00	1.00	1.00	1.00
Perspectiva 2 - Lateral	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00	1.00	2.00
Perspectiva 3 - Atrás	0.00	1.00	1.00	0.00	2.00	0.00	1.00	1.00	1.00	3.00
Perspectiva 4 - Isométrico	1.00	1.00	1.00	1.00	1.00	2.00	2.00	1.00	2.00	1.00

Tabla 4 – Cantidad de objetos reconocidos en la segunda estimación de ICE de MOPED, enfocada a la aproximación de la posición del objeto.
En este caso como sólo se utilizó un modelo y una escena en cada caso, el caso ideal es que reconozca un objeto, siendo la cantidad de dos o más, errores en la estimación y el peor caso posible siendo cero, o que no haya sido capaz de detectar al objeto.

Grado de Reconocimiento alcanzado por los Objetos en MOPED (Lotes 1 y 2)

Capturas	Avena	Jugo Boing	Jugo Bonafina	Lata de verduras	Leche	Pasta dental	Salsa de tomate	Shampoo Caprice	Té de flores	Tinte del cabello
Distancia 1 - Cerca	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	3.00
Distancia 2 - Media	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00
Distancia 3 - Lejos	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00
Iluminación 1 - Claro	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00
Iluminación 2 - Medio	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00
Iluminación 3 - Oscuro	2.00	4.00	6.00	2.00	6.00	6.00	3.00	2.00	3.00	3.00
Perspectiva 1 - Frontal	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00
Perspectiva 2 - Lateral	6.00	6.00	6.00	6.00	6.00	6.00	6.00	2.00	6.00	6.00
Perspectiva 3 - Atrás	3.00	6.00	6.00	3.00	6.00	3.00	6.00	6.00	6.00	6.00
Perspectiva 4 - Isométrico	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00

Tabla 5 – Grado o etapa de reconocimiento en MOPED alcanzado por el modelo en la escena correspondiente, siendo 7 el mejor caso posible donde finalizó todo el proceso de reconocimiento en MOPED, y 1 o 0 el peor caso posible donde o bien sólo fué capaz de extraer características de la escena o ni siquiera eso fué capaz de obtener.

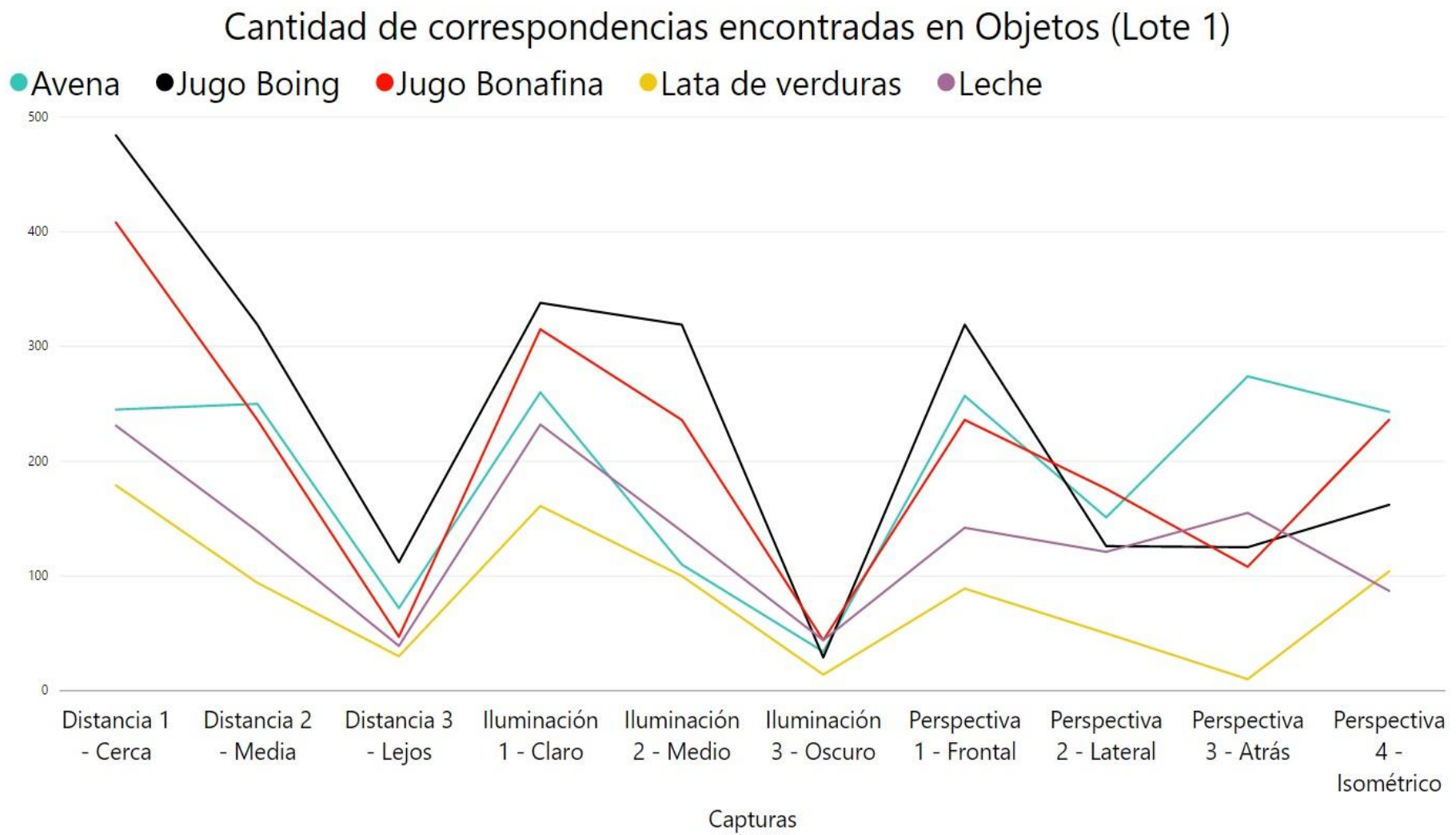

Fig. 4.14 – Comparación de cantidad de correspondencias obtenidas para el primer lote de objetos. Ya que las escenas utilizadas fueron fondos blancos homogéneos donde sólo existía el objeto, nos da una buena estimación de cuantas correspondencias del objeto en escena fueron encontradas respecto las del modelo 3D.
Tanto en esta como la siguiente gráfica podemos notar como el número de correspondencias encontradas decrementa conforme el objeto se aleja de la cámara y de igual manera conforme la iluminación se vuelve más oscura, el número de correspondencias encontradas disminuye, también, interesantemente para nuestros modelos, las perspectivas isométrica y frontal tienen mayor número de correspondencias que la lateral o la trasera del objeto. Esto último quizás a la forma en que se obtuvieron los modelos, empezando siempre desde una perspectiva frontal o isométrica, tendiendo a obtener más características desde estas vistas en el modelo.

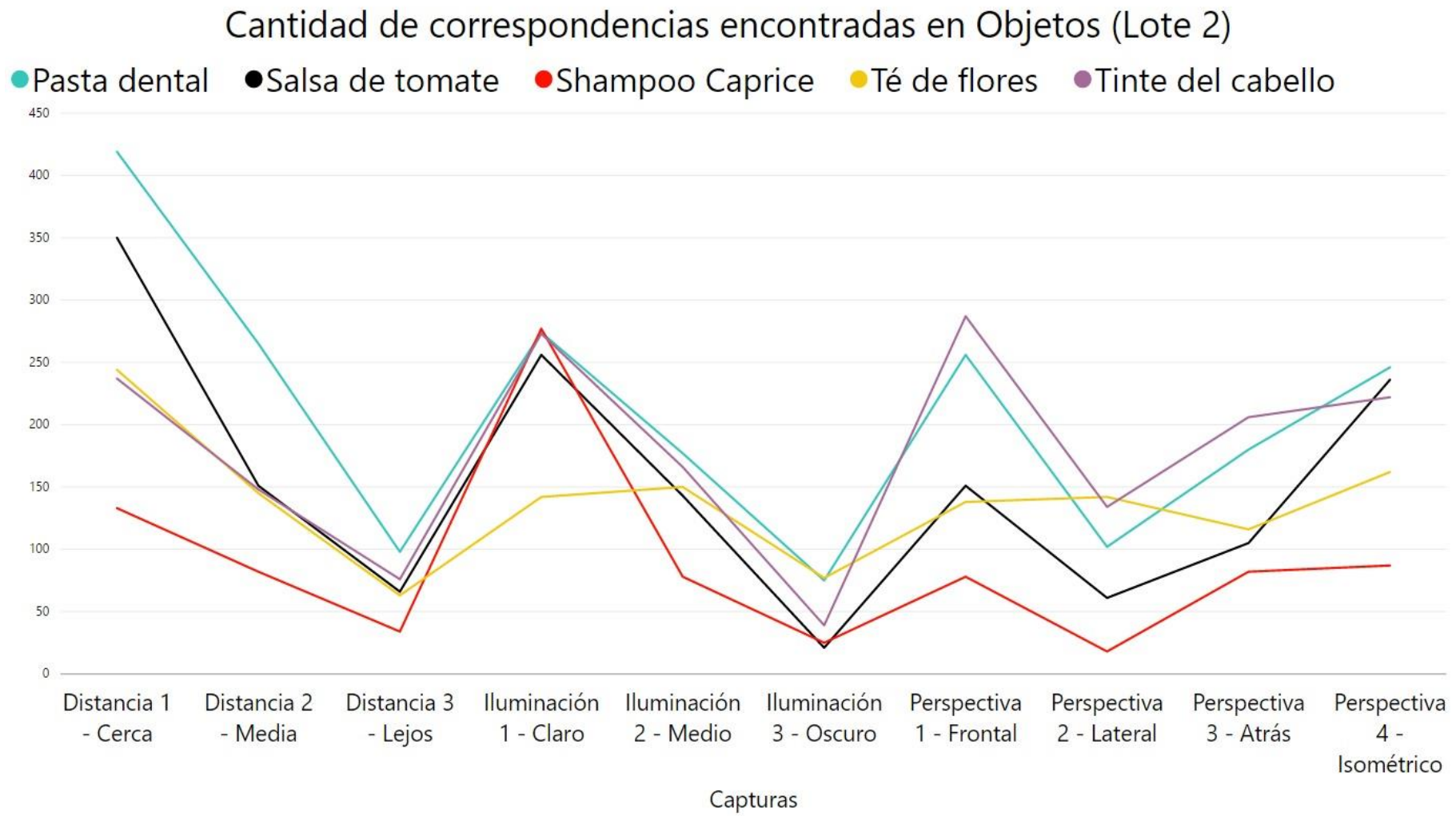

Fig. 4.15 – Comparación de cantidad de correspondencias obtenidas para el segundo lote de objetos. En este lote la cantidad más baja de correspondencias detectadas en los casos de perspectivas, fue la vista lateral.

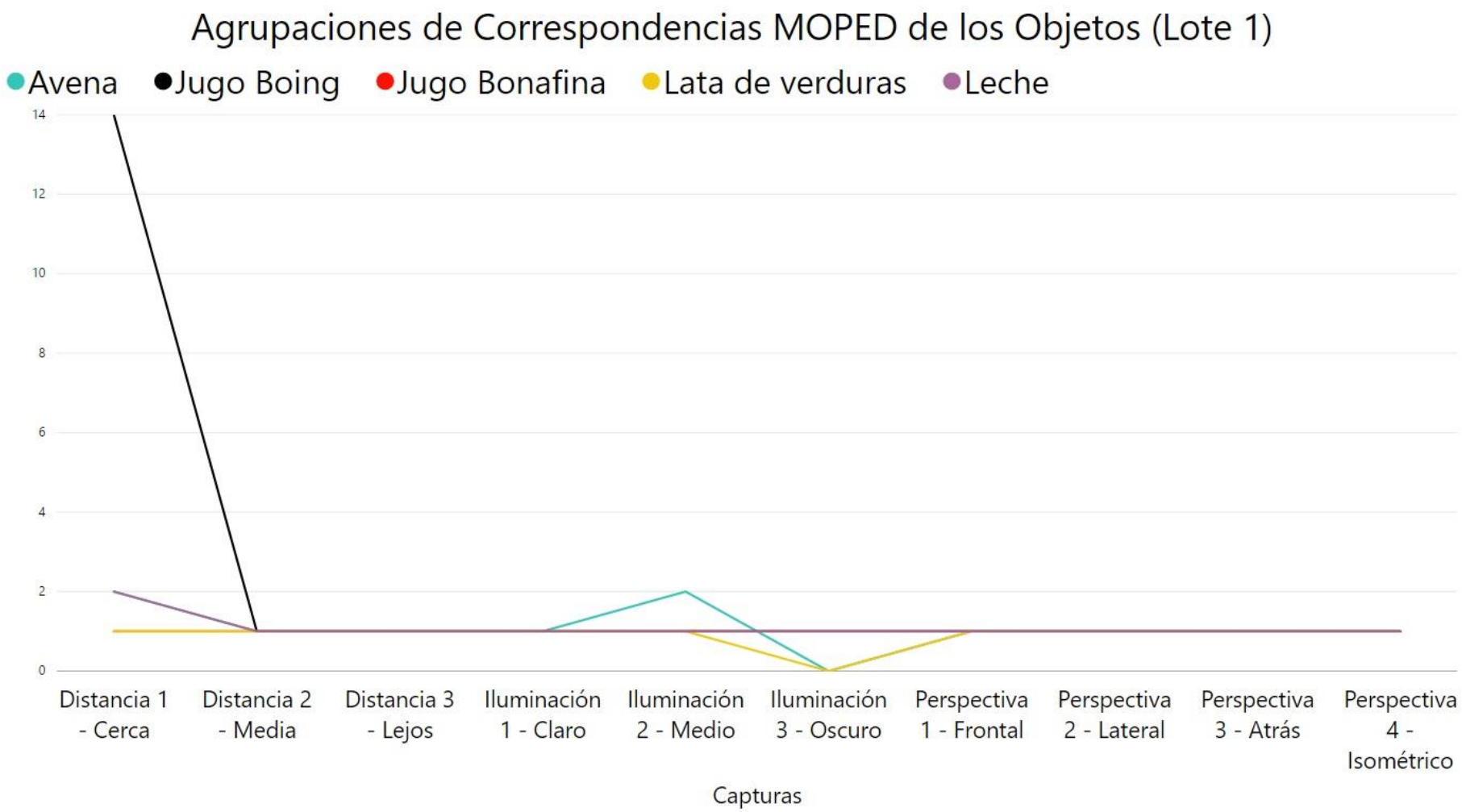

Fig. 4.16 – Comparación entre objetos del lote 1 y sus agrupaciones de correspondencias obtenidas por captura.

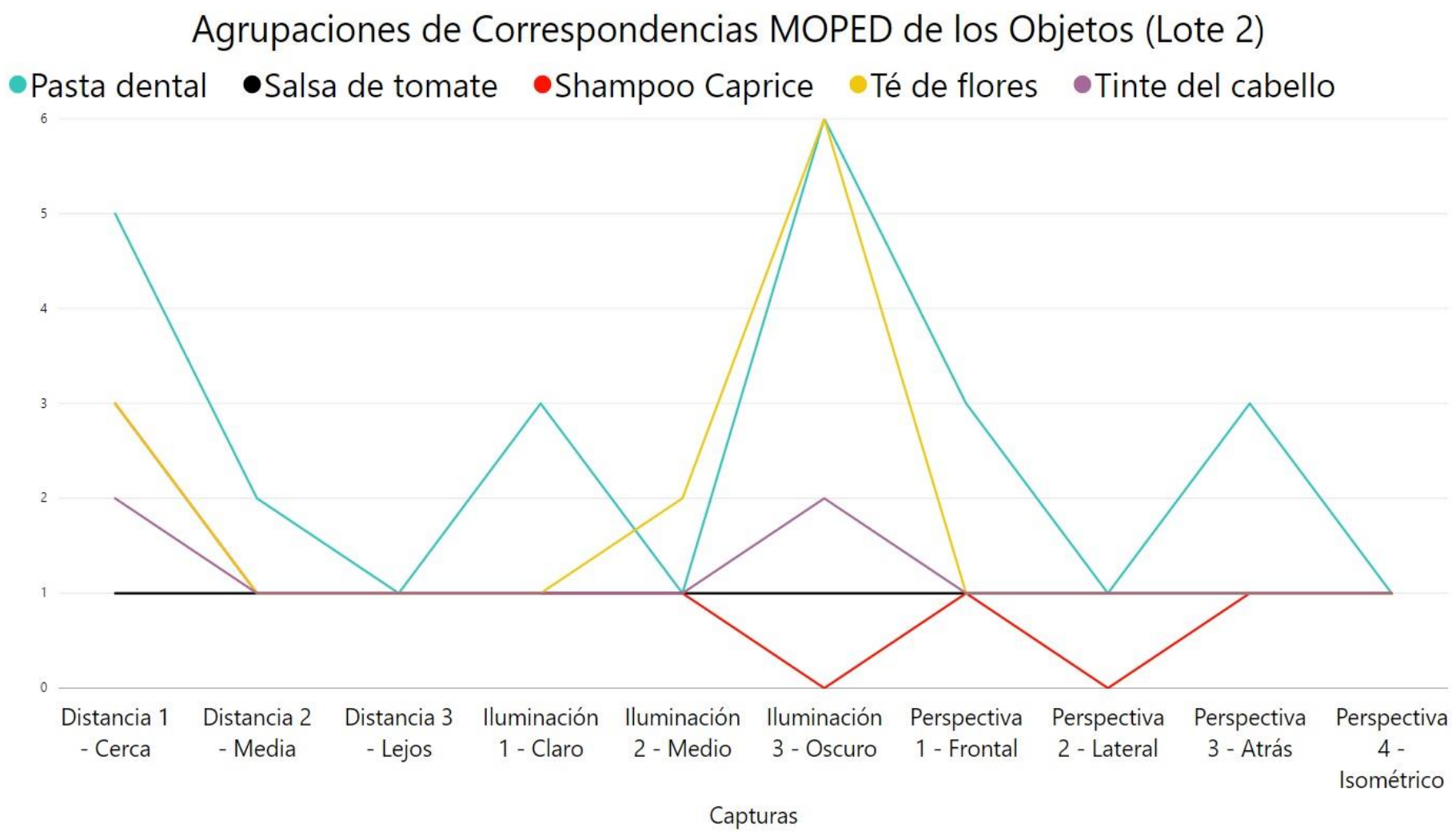

Fig. 4.17 - Comparación entre objetos del lote 2 y sus agrupaciones de correspondencias obtenidas por captura.

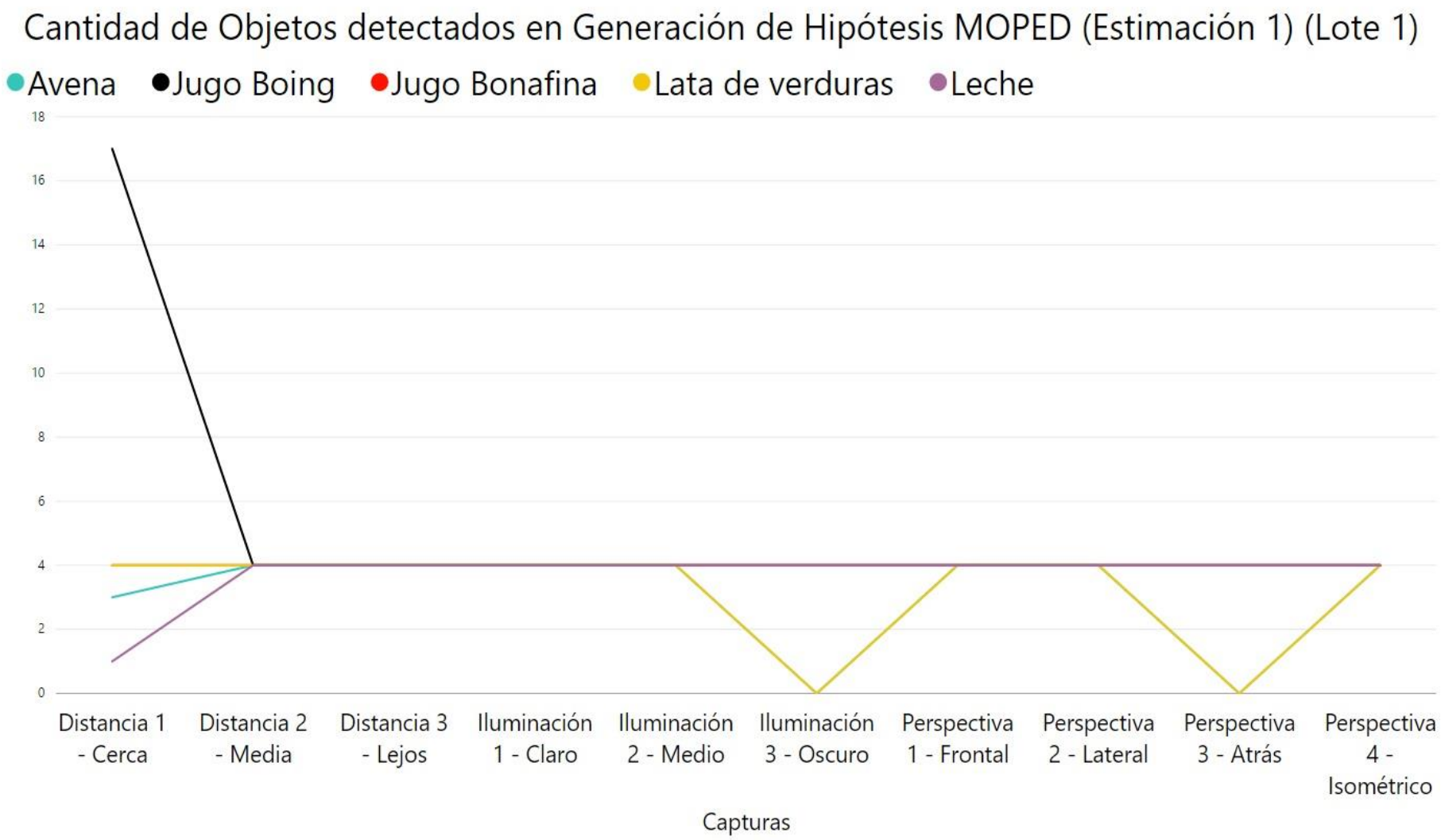

Fig. 4.18 – Cantidad de objetos detectados en la primera estimación ICE en MOPED para el lote 1 de objetos.

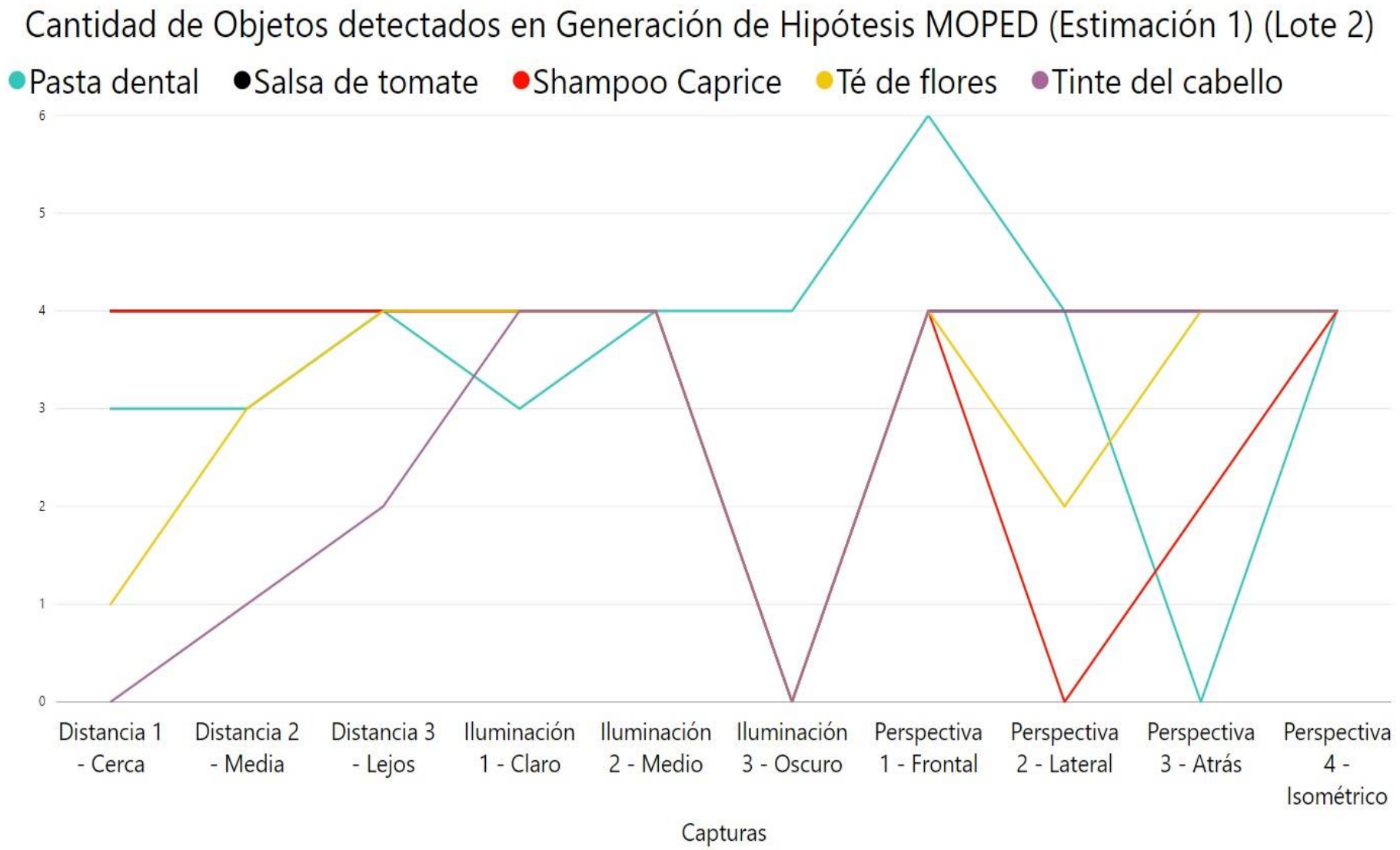

Fig. 4.19 - Cantidad de objetos detectados en la primera estimación ICE en MOPED para el lote 2 de objetos.

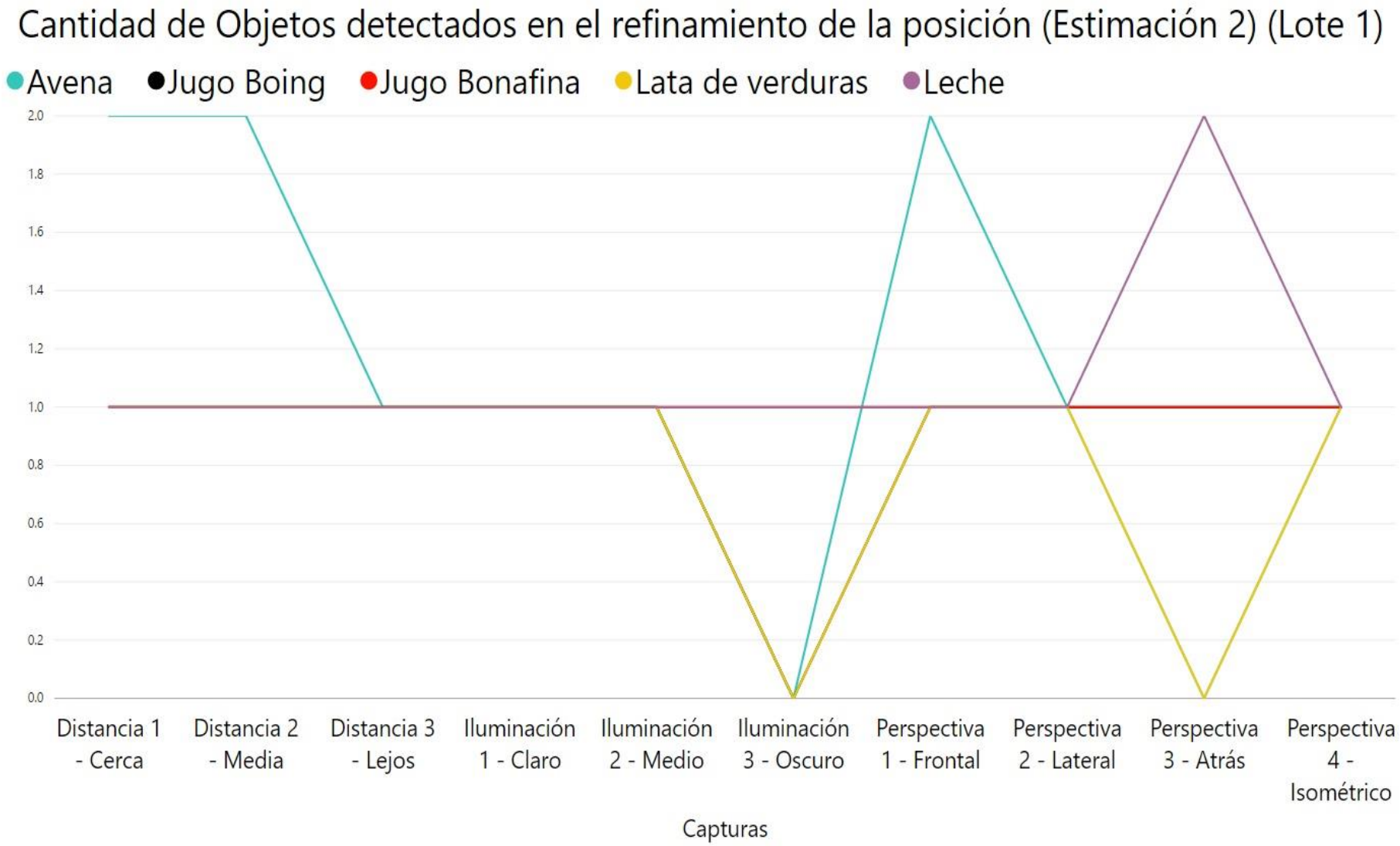

Fig. 4.20 – Cantidad de objetos reconocidos por MOPED en la segunda estimación de ICE para el primer lote de objetos.

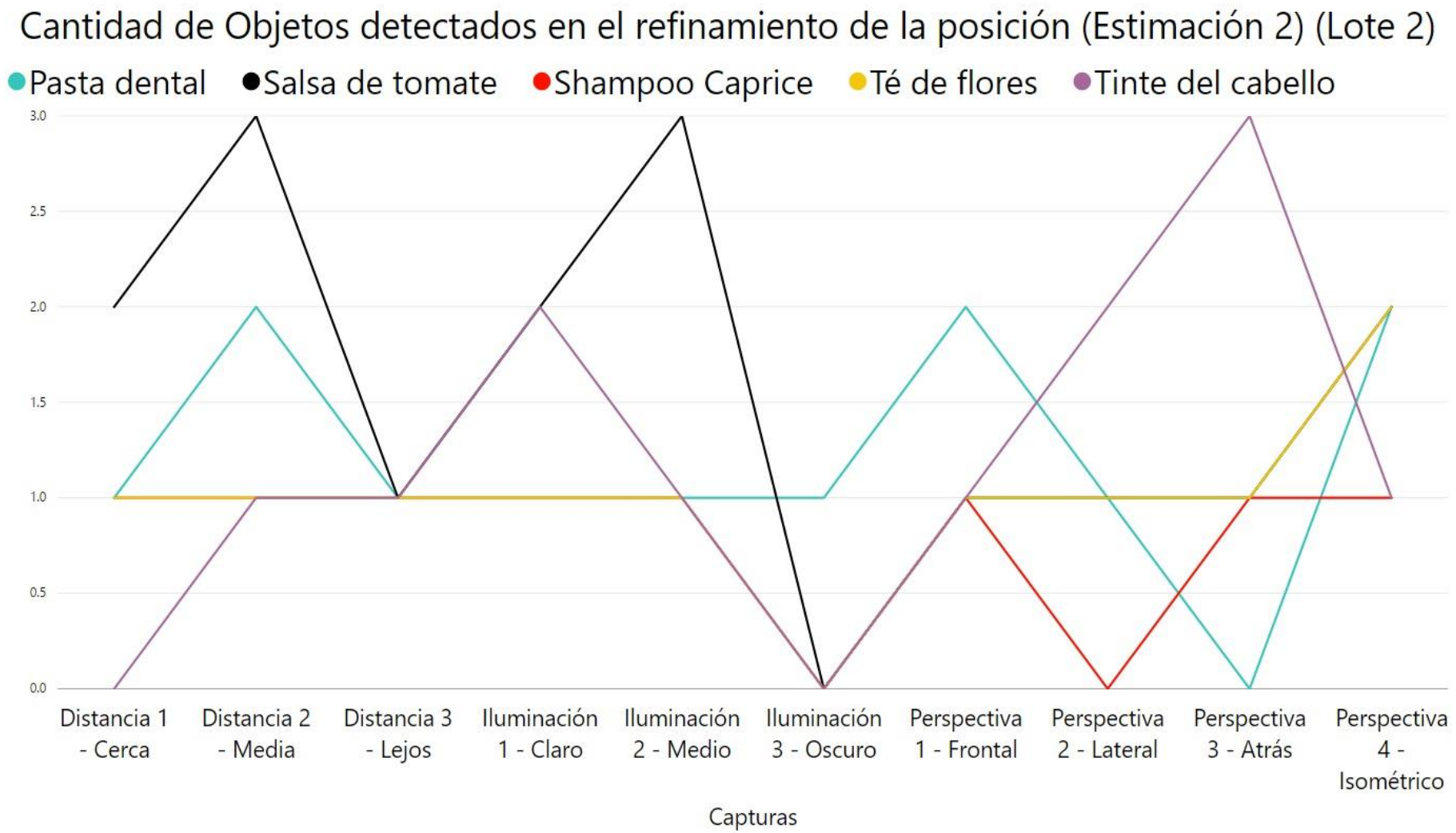

Fig. 4.21 - Cantidad de objetos reconocidos por MOPED en la segunda estimación de ICE para el segundo lote de objetos.

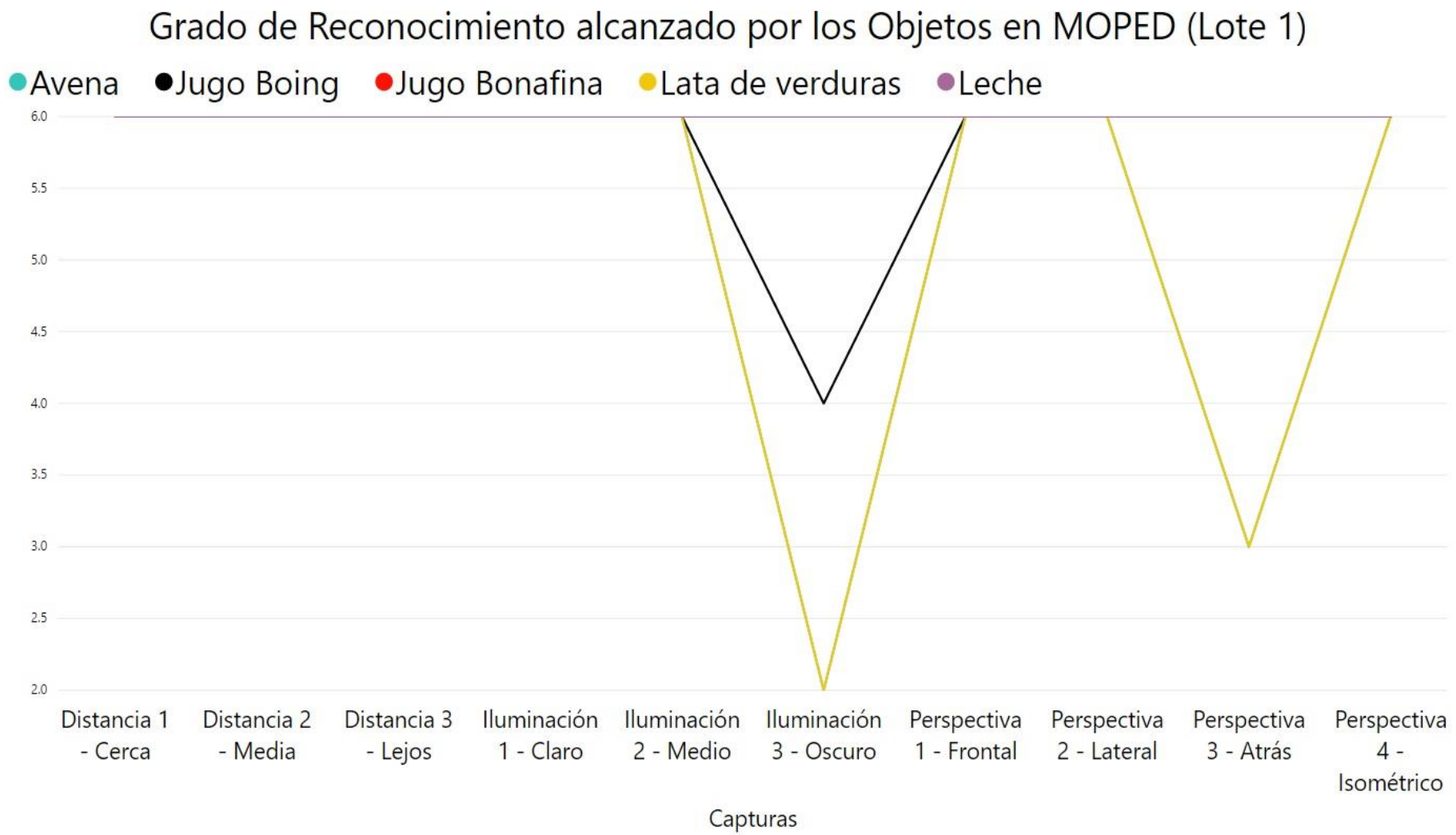

Fig. 4.22 – Grado de reconocimiento o etapa alcanzada en MOPED por el modelo del objeto en cada captura para el lote 1 de objetos.

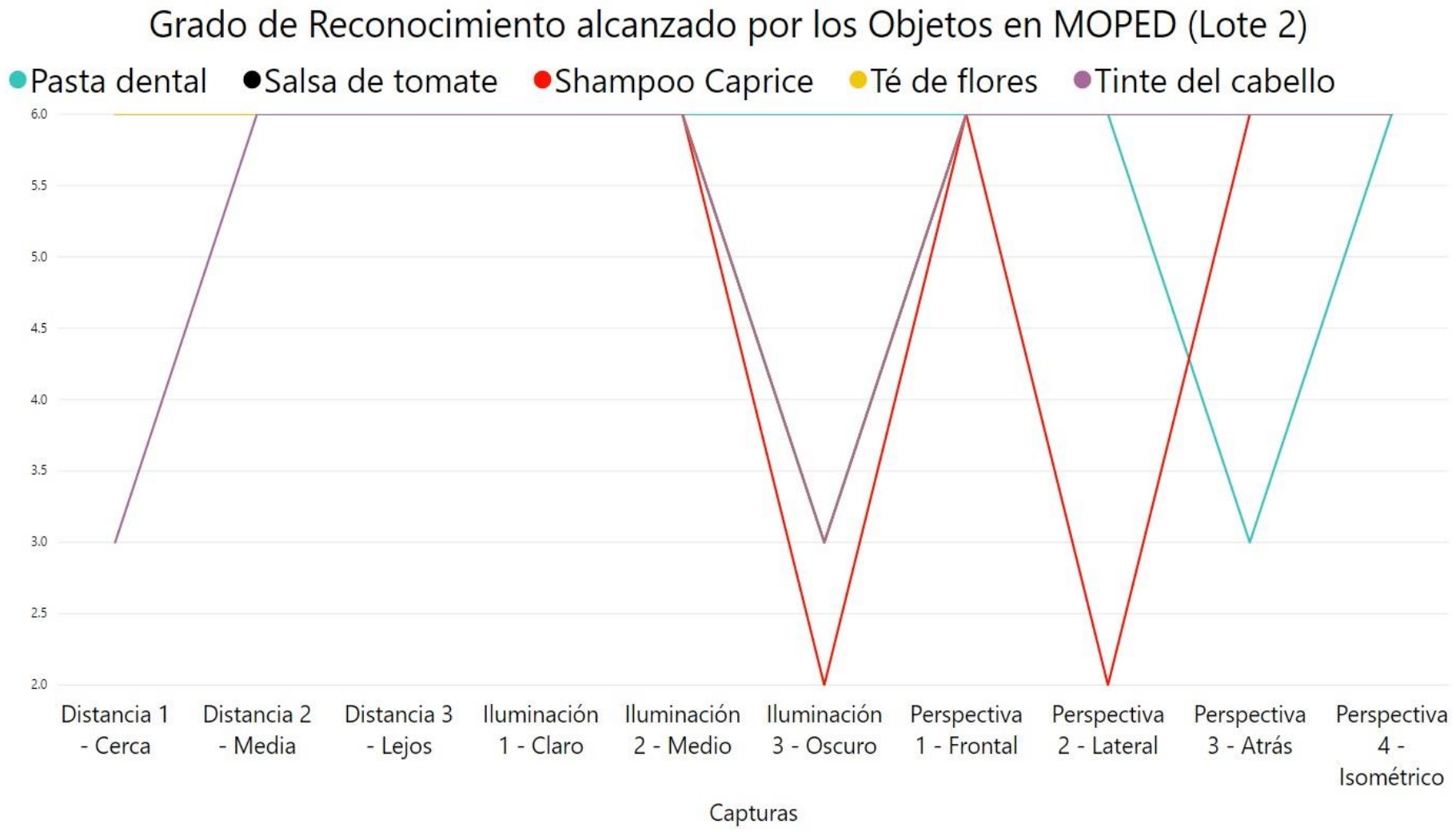

Fig. 4.23 - Grado de reconocimiento o etapa alcanzada en MOPED por el modelo del objeto en cada captura para el lote 2 de objetos.

Ahora en cuanto al reconocimiento en escenas donde se tienen múltiples objetos, tanto conocidos como desconocidos, se hizo un análisis en un total de 30 casos distintos, construidos a partir de al menos unas doce o trece capturas tomadas de escenas con múltiples objetos, variando el modelo utilizado en cada caso, este análisis inicial se hizo de forma aleatoria, y subsecuentemente realicé una prueba con nueve capturas específicas para dos objetos principales que habían respondido bien en pruebas anteriores; el empaque de jugo y la leche, los resultados son presentados a continuación:

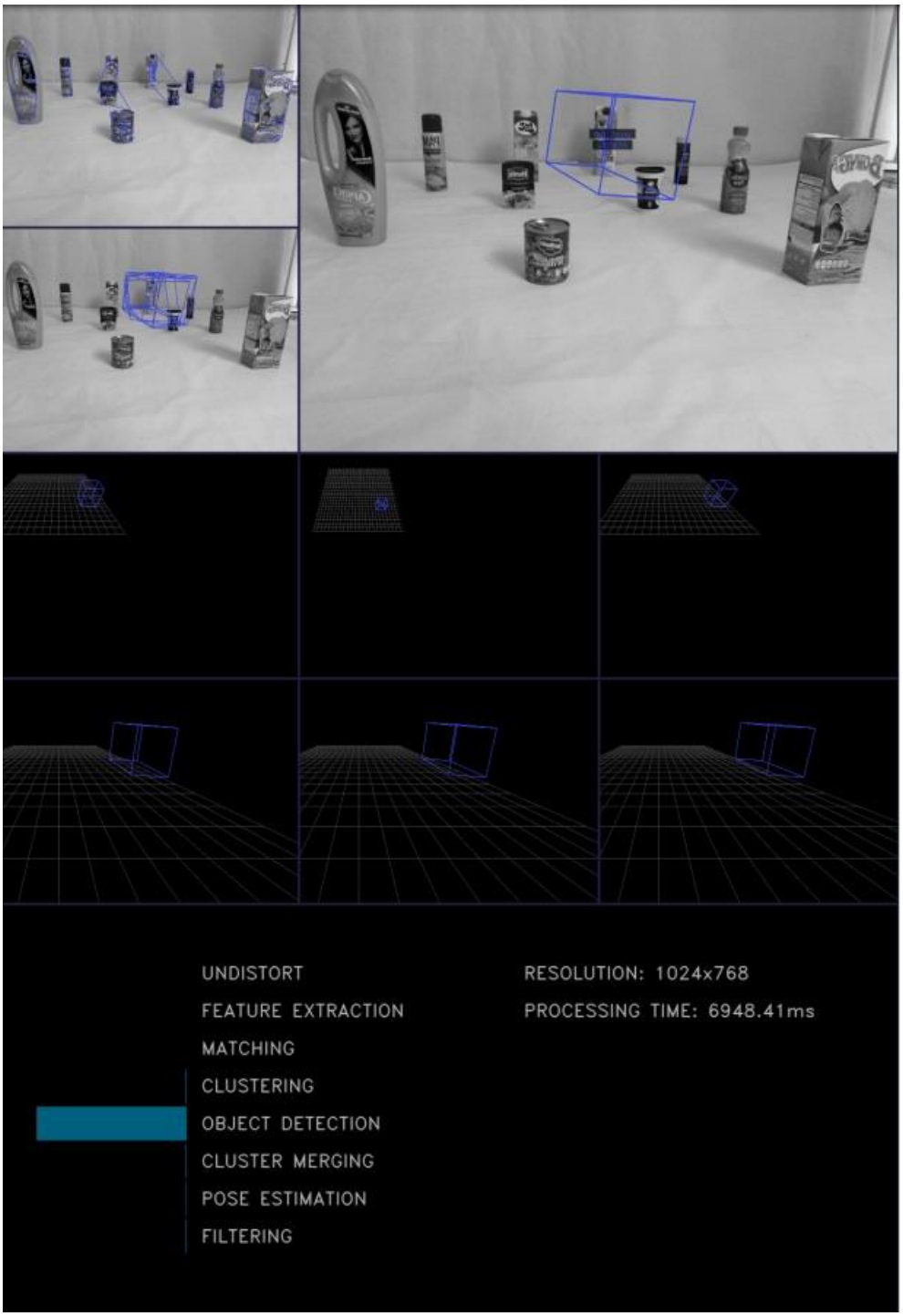

Fig. 4.24 – Reconocimiento MOPED de un empaque de jugo en una escena con múltiples objetos, de un modelo obtenido con el modelador de objetos 3D para el robot Golem III.

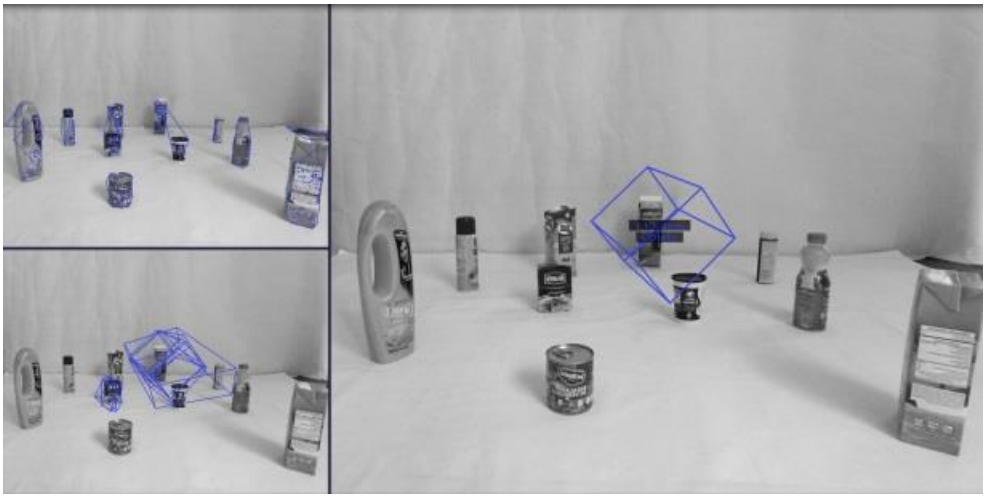

Fig. 4.25 - Reconocimiento de la escena anterior con una rotación aleatoria de todos los objetos, ahora con el empaque de jugo en su cara trasera y a una distancia lejana de la cámara.

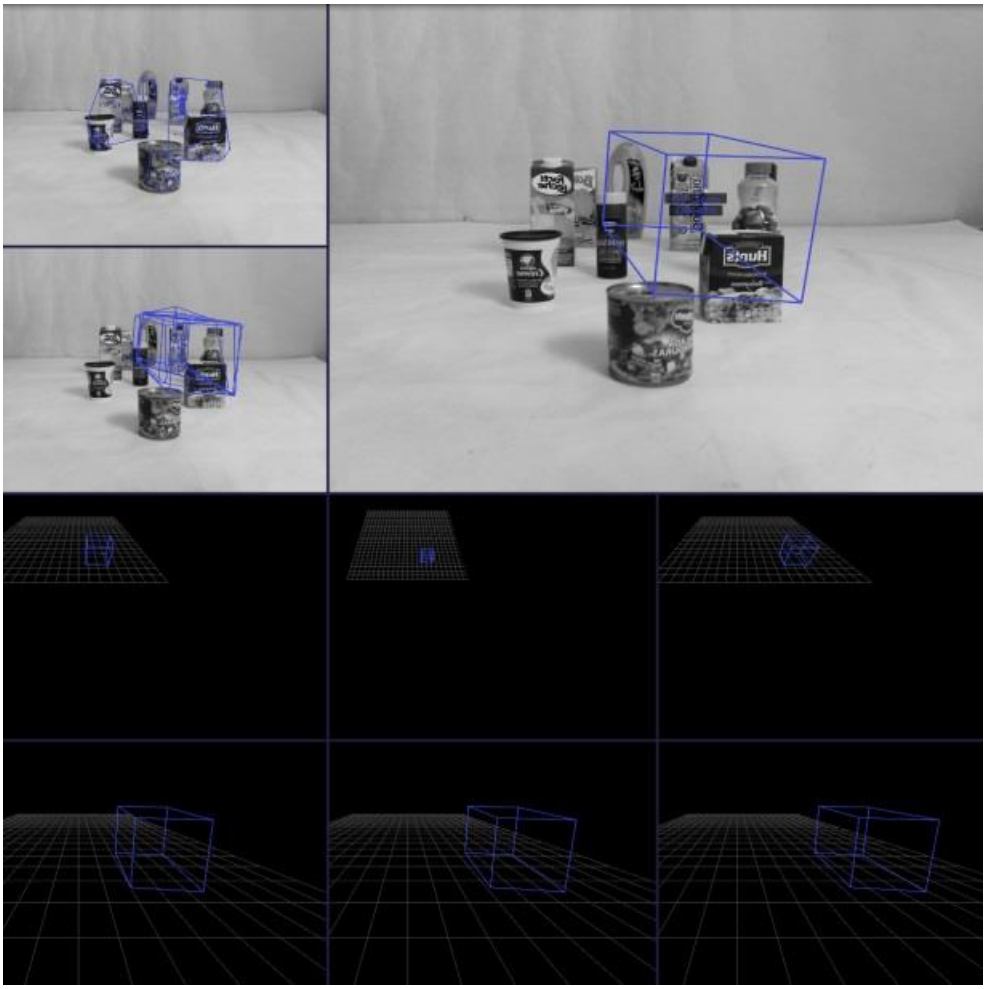

Fig. 4.26 – Reconocimiento del mismo objeto ahora en otro conjunto de objetos, arreglados a manera de hilera, unos frente a otros.

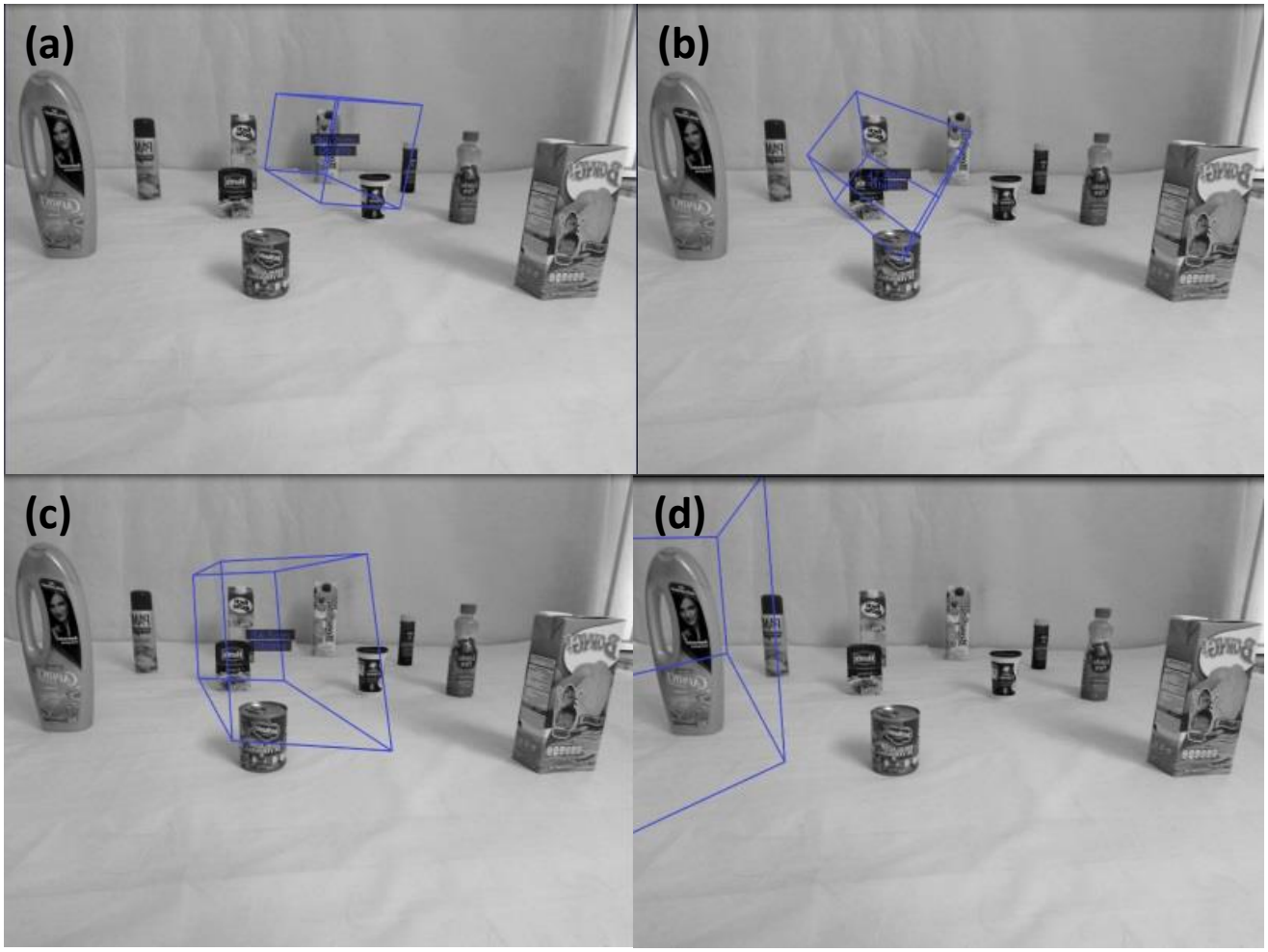

Fig. 4.27 – Reconocimiento de distintos objetos en una escena de múltiples objetos ordenados de forma aleatoria, la estimación de la posición no es perfecta debido a la naturaleza de la alineación en los modelos y puntos aislados debido al ruido. (a)Reconocimiento del empaque de jugo. (b)Reconocimiento del cartón de leche Forti. (c)Reconocimiento de la caja de salsa de tomate. (d)Reconocimiento de la botella de shampoo.

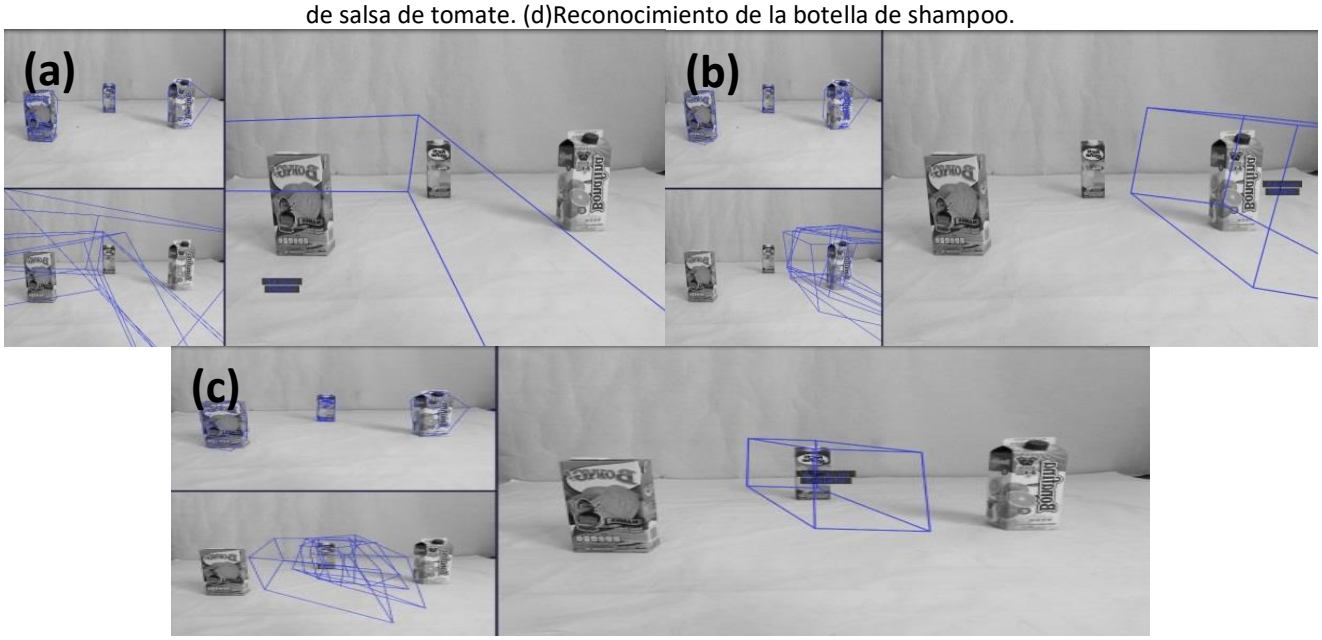

Fig. 4.28 – Reconocimiento de una triada de objetos. (a)Empaque de jugo Boing. (b)Empaque de jugo Bonafina. (c)Cartón de leche.

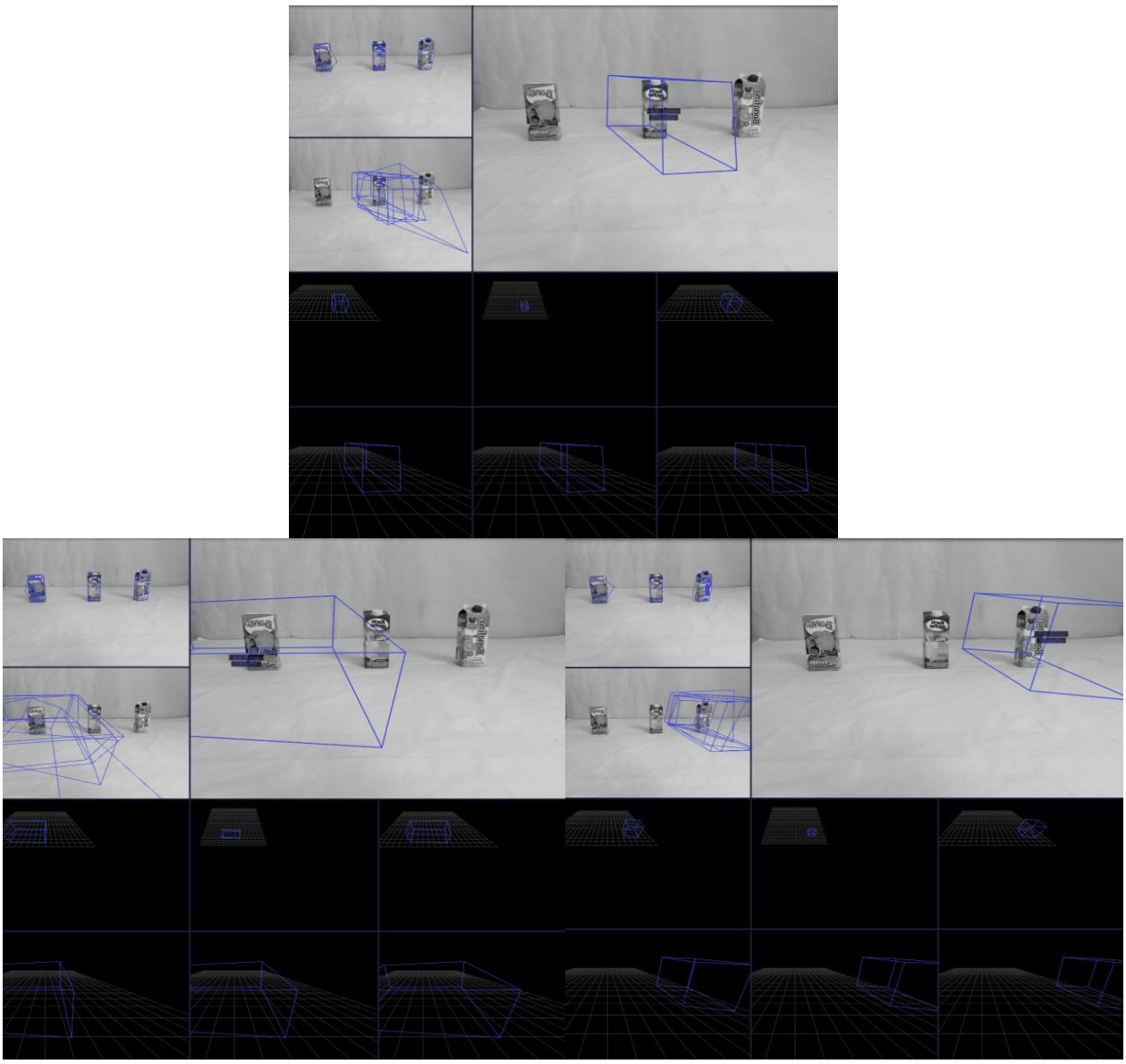

Fig. 4.29 – Reconocimiento de la misma triada de objetos anterior, esta vez a la misma distancia de la cámara.

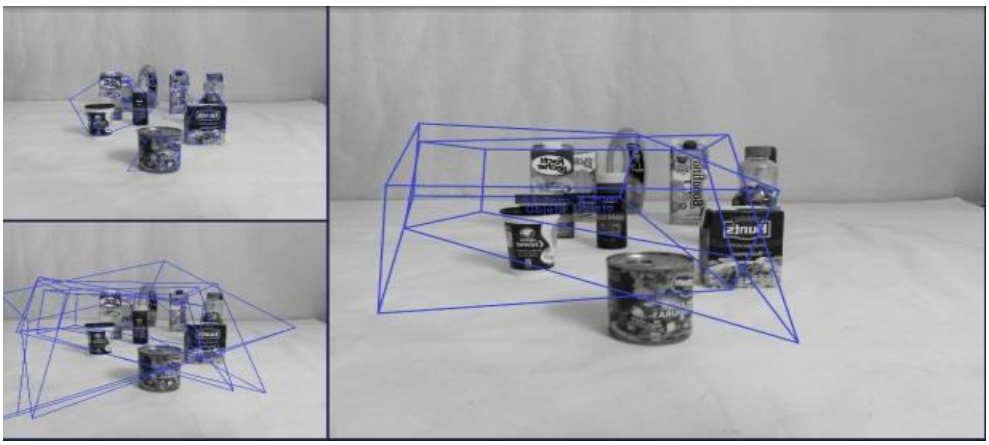

Fig. 4.30 – Reconocimiento de un empaque de jugo Boing, con una estimación de posición muy mala, que cubre otros objetos y además presenta una detección múltiple del objeto, sin embargo, cabe señalar que al menos fue capaz de detectar el objeto a una distancia considerable además de estar parcialmente cubierto por los demás objetos. (Hasta atrás, a un lado de la botella de shampoo y justo más allá de donde se encuentra la leche).

Ahora se presentarán los resultados para todos los conjuntos de objetos, comenzando con el arreglo general de 30 casos aleatorios y después con el experimento controlado de nueve capturas para dos modelos.

Número de captura de conjunto	Cantidad de correspondencias encontradas
01	469.00
02	523.00
03	530.00
04	495.00
05	477.00
06	403.00
07	388.00
08	375.00
09	535.00
10	562.00
11	435.00
12	404.00
13	380.00
14	460.00
15	497.00
16	1,247.00
17	1,269.00
18	1,244.00
19	1,337.00
20	1,508.00
21	1,384.00
22	560.00
23	611.00
24	494.00
25	517.00
26	387.00
27	483.00
28	398.00
29	530.00
30	526.00

Número de captura de conjunto	Cantidad de agrupaciones de correspondencias
01	7.00
02	6.00
03	7.00
04	9.00
05	8.00
06	3.00
07	3.00
08	3.00
09	3.00
10	3.00
11	3.00
12	14.00
13	8.00
14	4.00
15	9.00
16	27.00
17	21.00
18	17.00
19	28.00
20	37.00
21	30.00
22	11.00
23	12.00
24	11.00
25	11.00
26	13.00
27	10.00
28	11.00
29	11.00
30	13.00

Tablas 6 y 7 – Muestran los datos de las pruebas aleatorias con conjuntos de objetos para correspondencias y agrupaciones de las mismas en cada caso.

Número de captura de conjunto	Cantidad de objetos detectados en la primera estimación
01	4.00
02	4.00
03	4.00
04	4.00
05	4.00
06	4.00
07	4.00
08	4.00
09	4.00
10	4.00
11	4.00
12	6.00
13	4.00
14	4.00
15	4.00
16	8.00
17	4.00
18	6.00
19	8.00
20	16.00
21	19.00
22	4.00
23	4.00
24	4.00
25	4.00
26	2.00
27	4.00
28	4.00
29	4.00
30	4.00

Número de captura de conjunto	Cantidad de objetos reconocidos en la segunda estimación
01	3.00
02	1.00
03	1.00
04	2.00
05	1.00
06	1.00
07	1.00
08	1.00
09	1.00
10	1.00
11	1.00
12	1.00
13	2.00
14	1.00
15	2.00
16	2.00
17	1.00
18	4.00
19	2.00
20	2.00
21	2.00
22	1.00
23	3.00
24	1.00
25	1.00
26	1.00
27	1.00
28	1.00
29	1.00
30	1.00

Tablas 8 y 9 – Muestran los datos de las pruebas aleatorias con conjuntos objetos para la cantidad de objetos detectados en la primera y segunda estimación de ICE en MOPED.

Número de captura de conjunto	Grado de reconocimiento de acuerdo a la etapa alcanzada en MOPED
01	6.00
02	6.00
03	6.00
04	6.00
05	6.00
06	6.00
07	6.00
08	6.00
09	6.00
10	6.00
11	6.00
12	6.00
13	6.00
14	6.00
15	6.00
16	6.00
17	6.00
18	6.00
19	6.00
20	6.00
21	6.00
22	6.00
23	6.00
24	6.00
25	6.00
26	6.00
27	6.00
28	6.00
29	6.00
30	6.00

Tabla 10 – Demuestra la última etapa alcanzada en el reconocedor MOPED por cada conjunto escena-modelo.

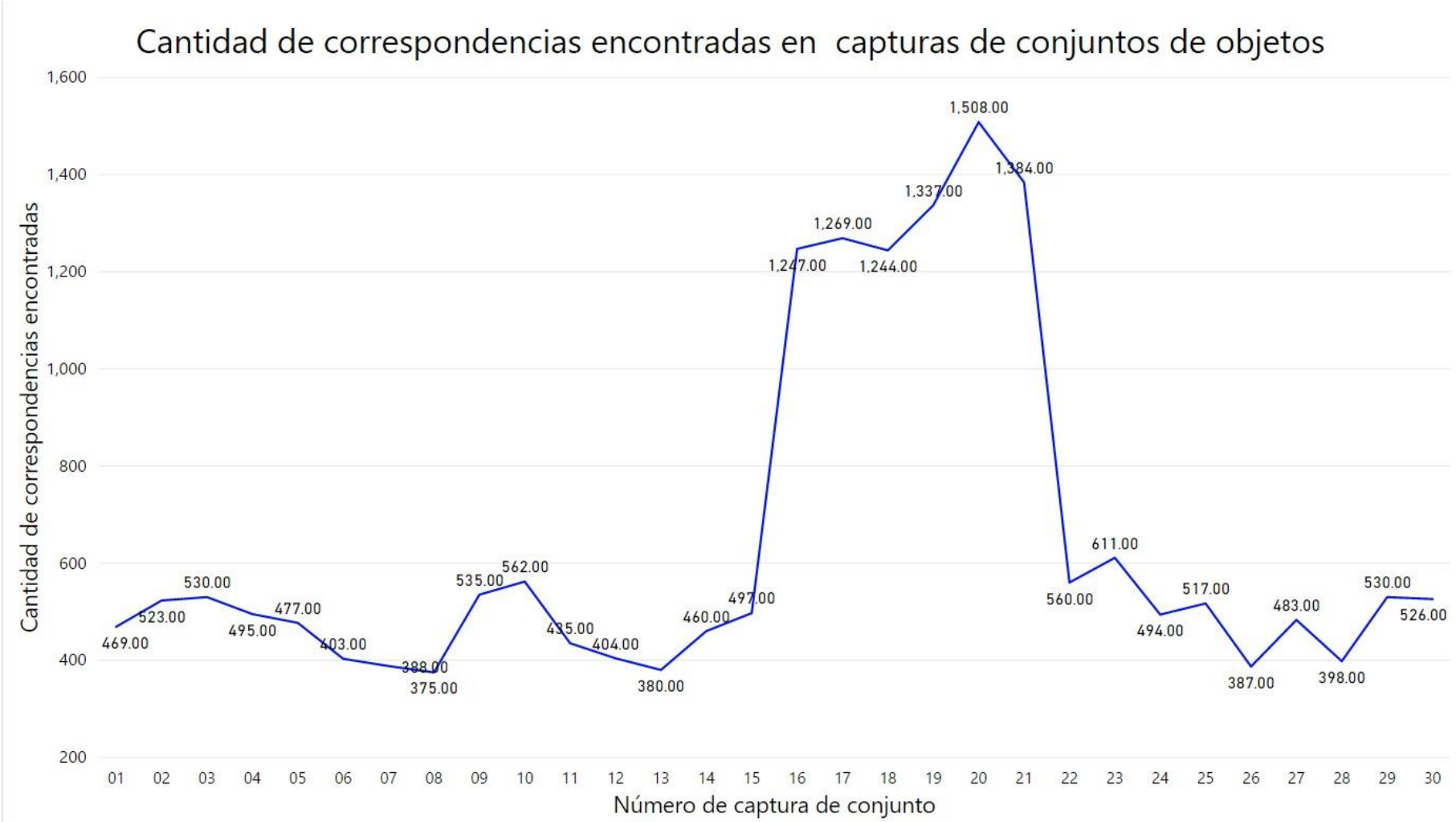

Fig. 4.31 – Cantidad de correspondencias encontradas en las escenas de conjuntos de objetos.

Fig. 4.32 – Cantidad de agrupaciones de correspondencias encontradas en las capturas de conjuntos de objetos.

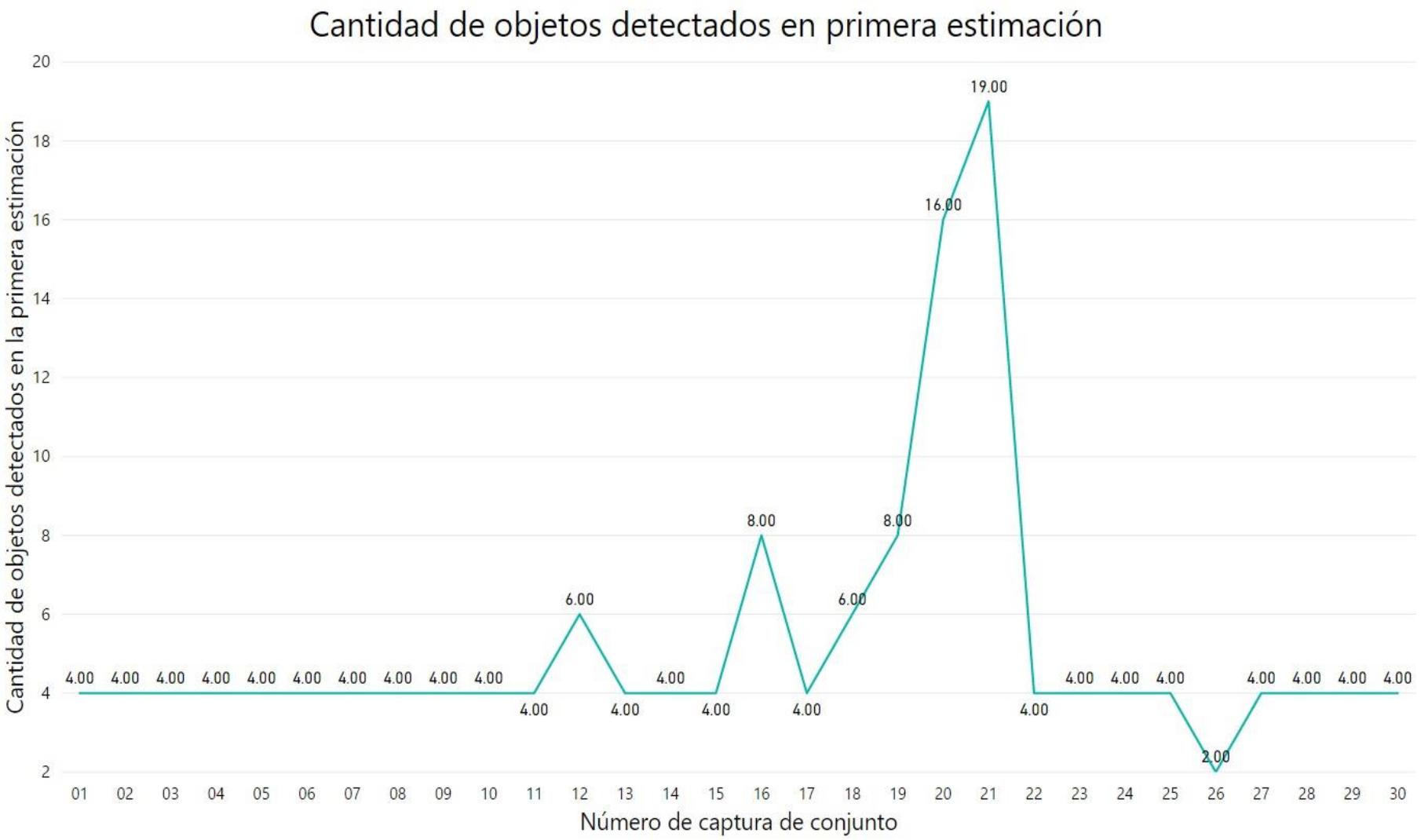

Fig. 4.33 – Cantidad de objetos detectados en la primera estimación de MOPED en las capturas de conjuntos de objetos.

Fig. 4.34 – Cantidad de objetos reconocidos en la segunda estimación de MOPED en las capturas de conjuntos de objetos.

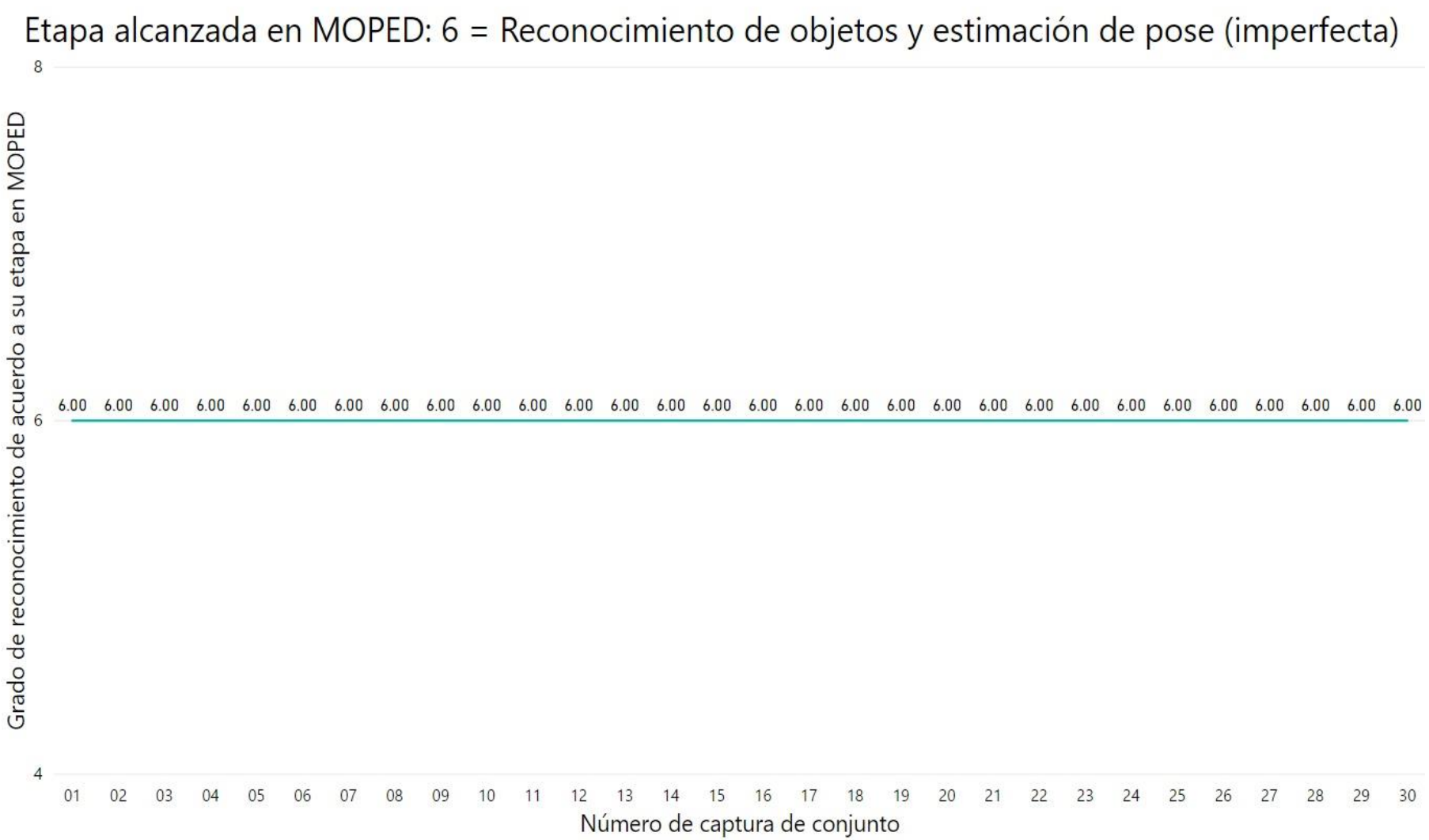

Fig. 4.35 – Etapa alcanzada en MOPED por los modelos en las capturas de conjuntos de objetos.

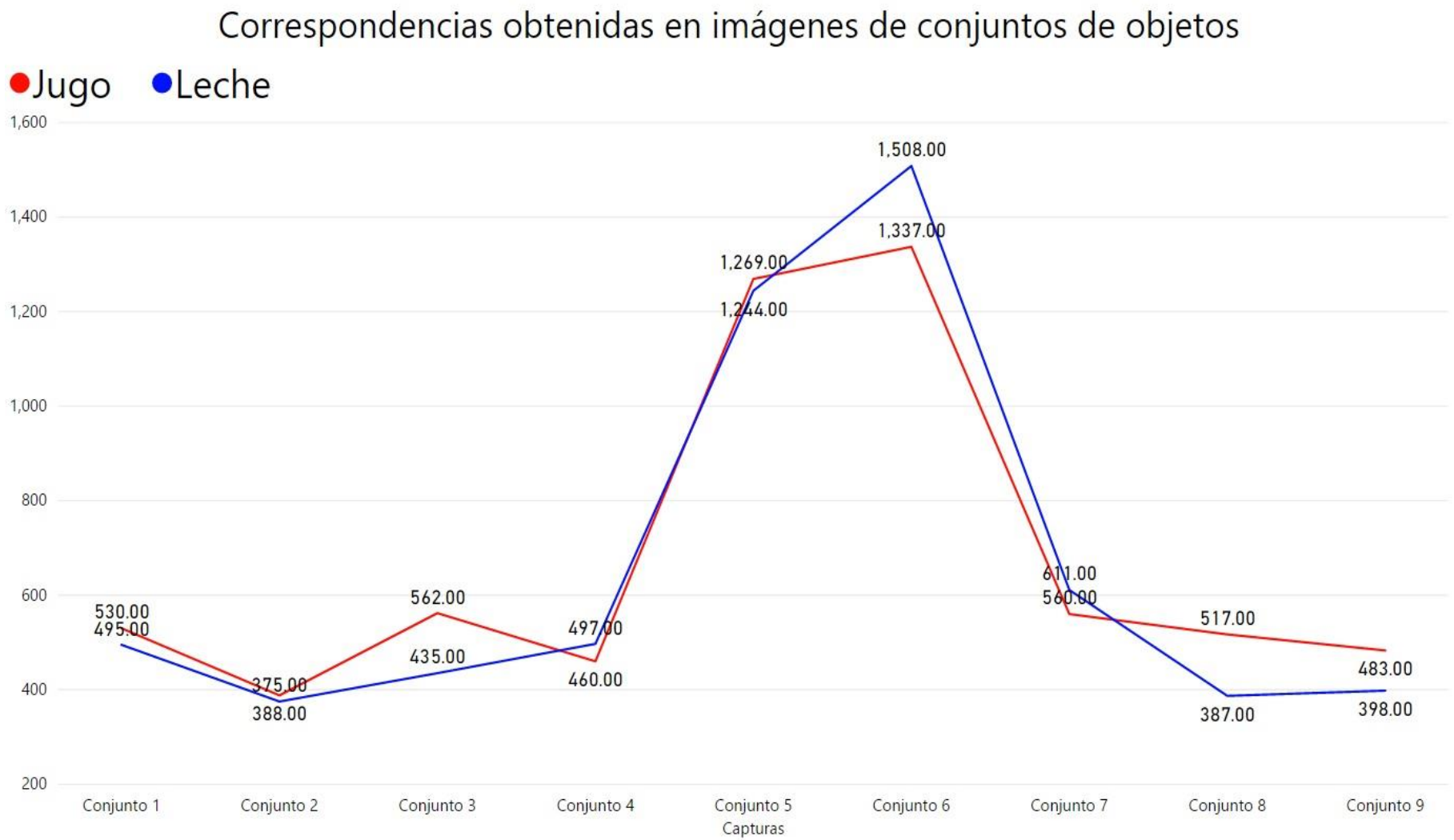

Fig. 4.36 – Correspondencias encontradas en el experimento de dos modelos con nueve capturas de conjuntos de objetos.

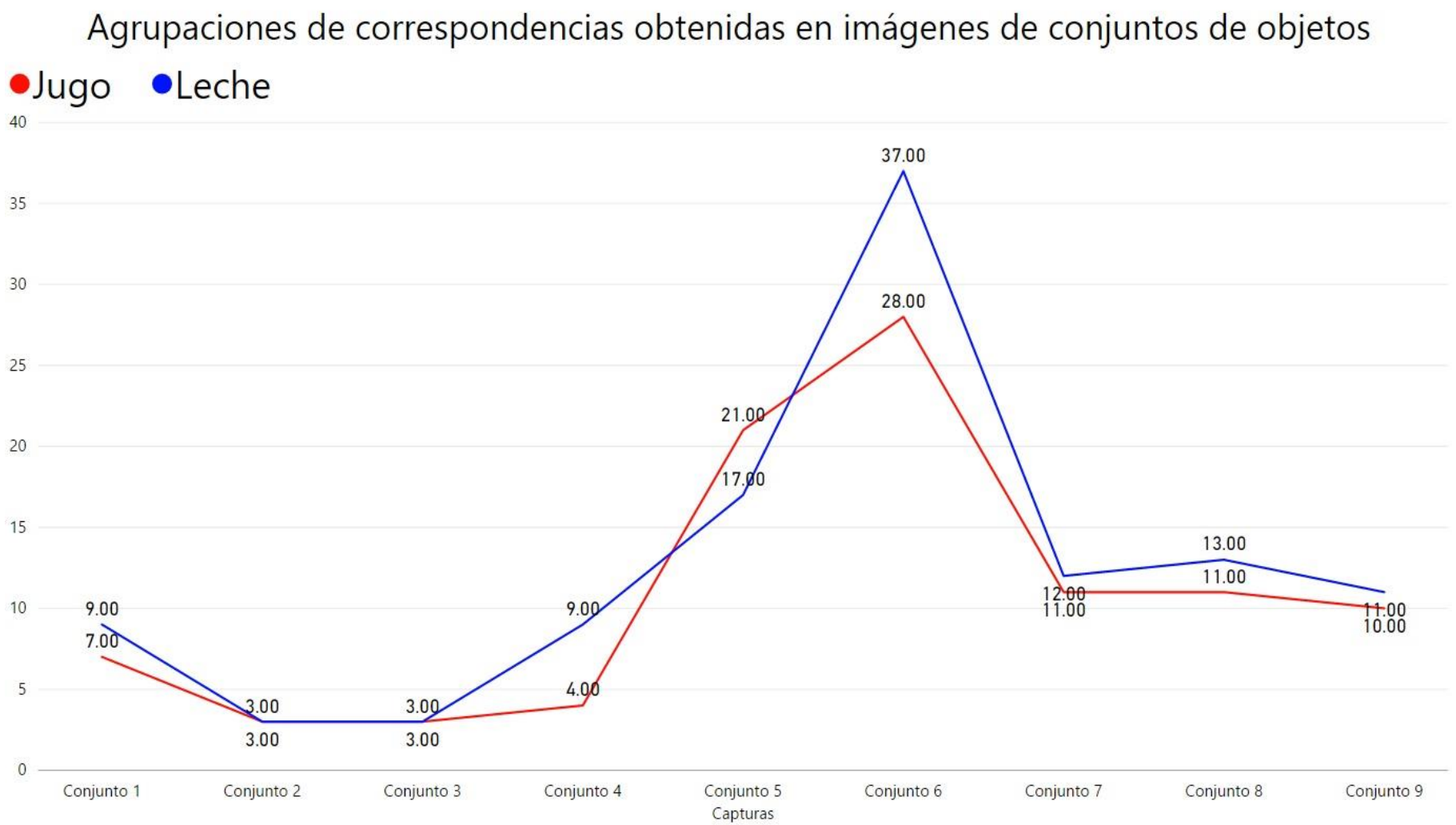

Fig. 4.37 – Agrupaciones de correspondencias encontradas en el experimento de dos modelos con nueve capturas de conjuntos de objetos.

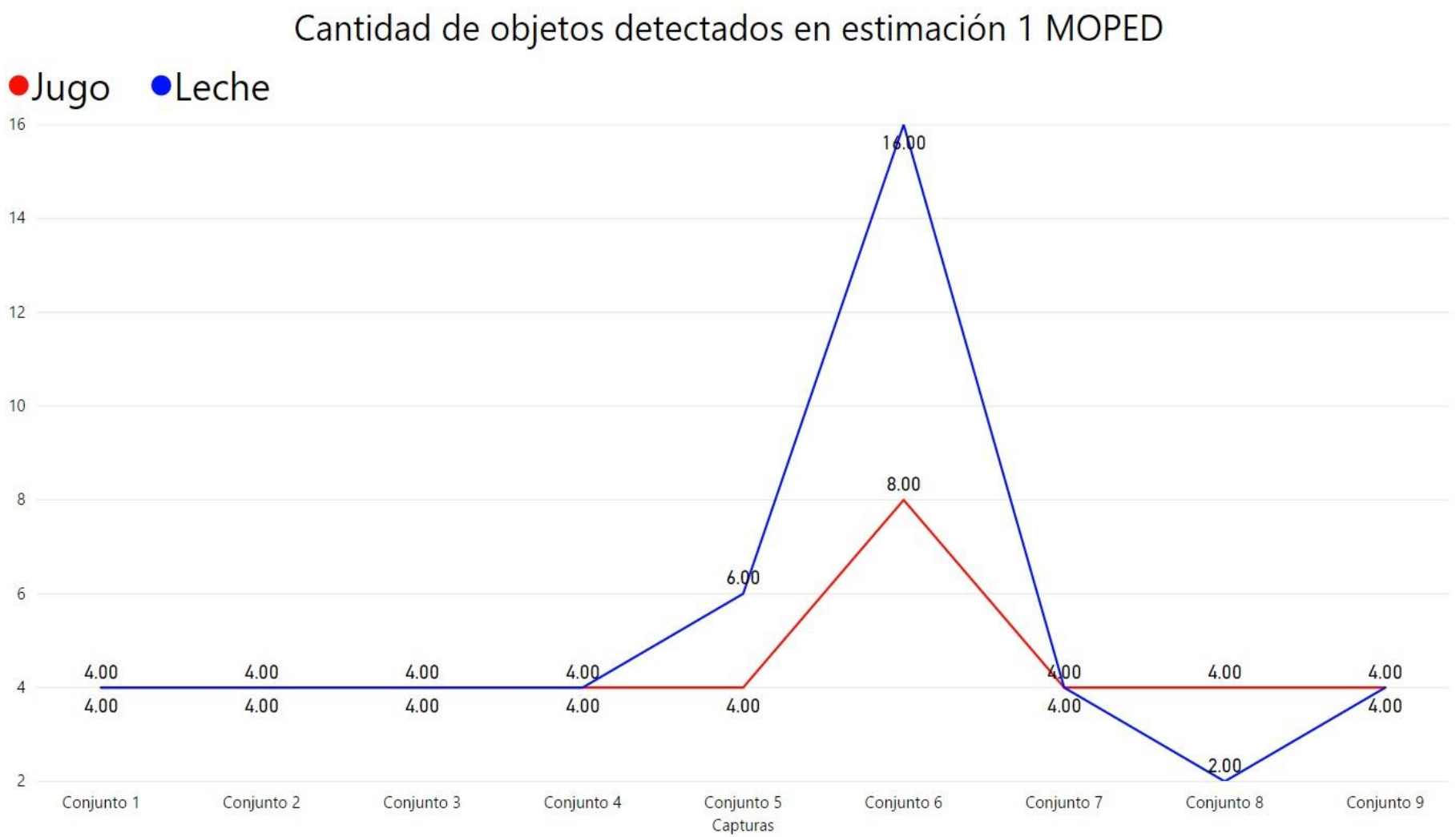

Fig. 4.38 – Cantidad de objetos detectados para la primera estimación ICE en MOPED en el experimento de dos modelos con nueve capturas de conjuntos de objetos.

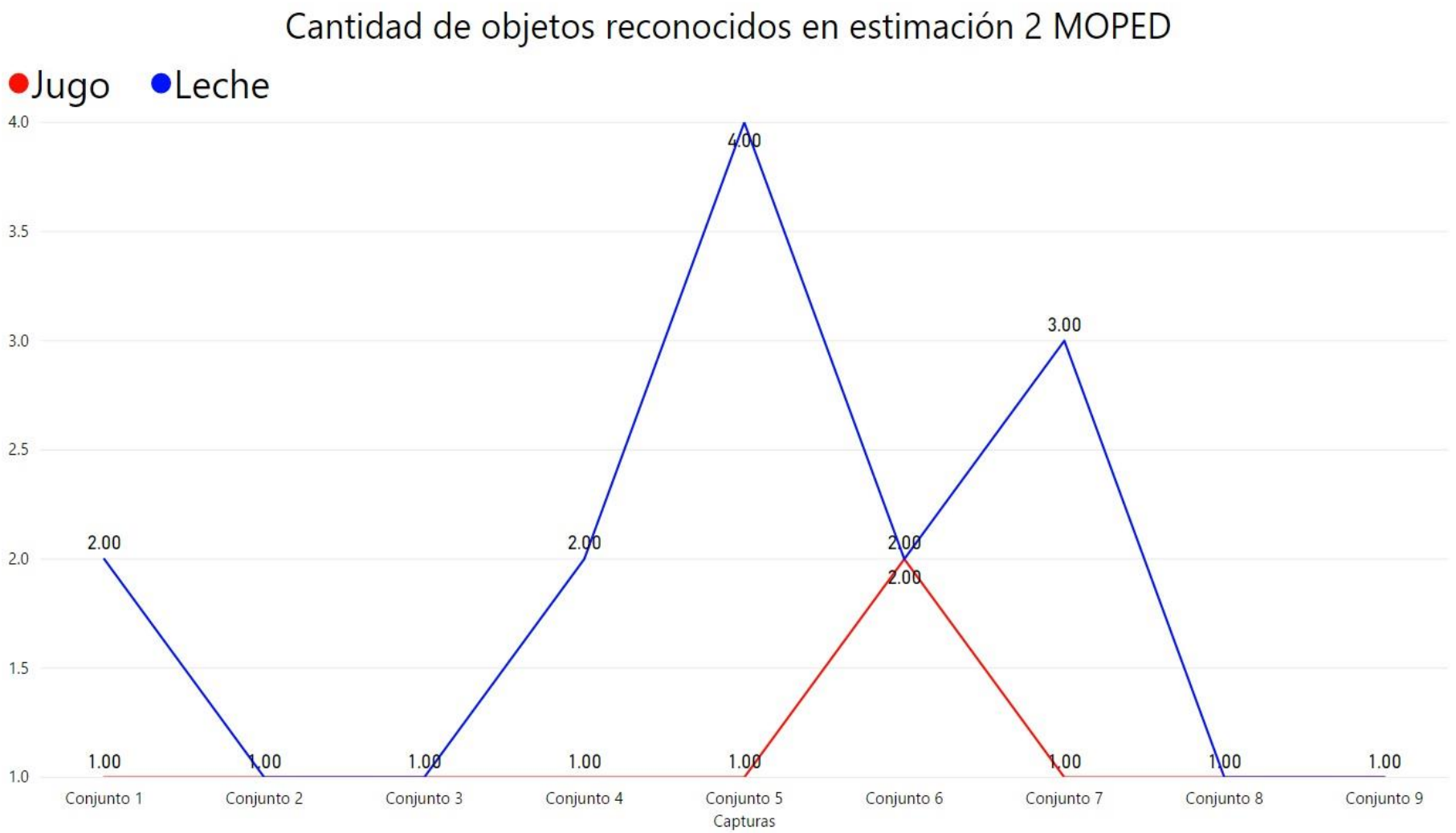

Fig. 4.39 – Cantidad de objetos reconocidos en la segunda estimación ICE de MOPED en el experimento de dos modelos con nueve capturas de conjuntos de objetos.

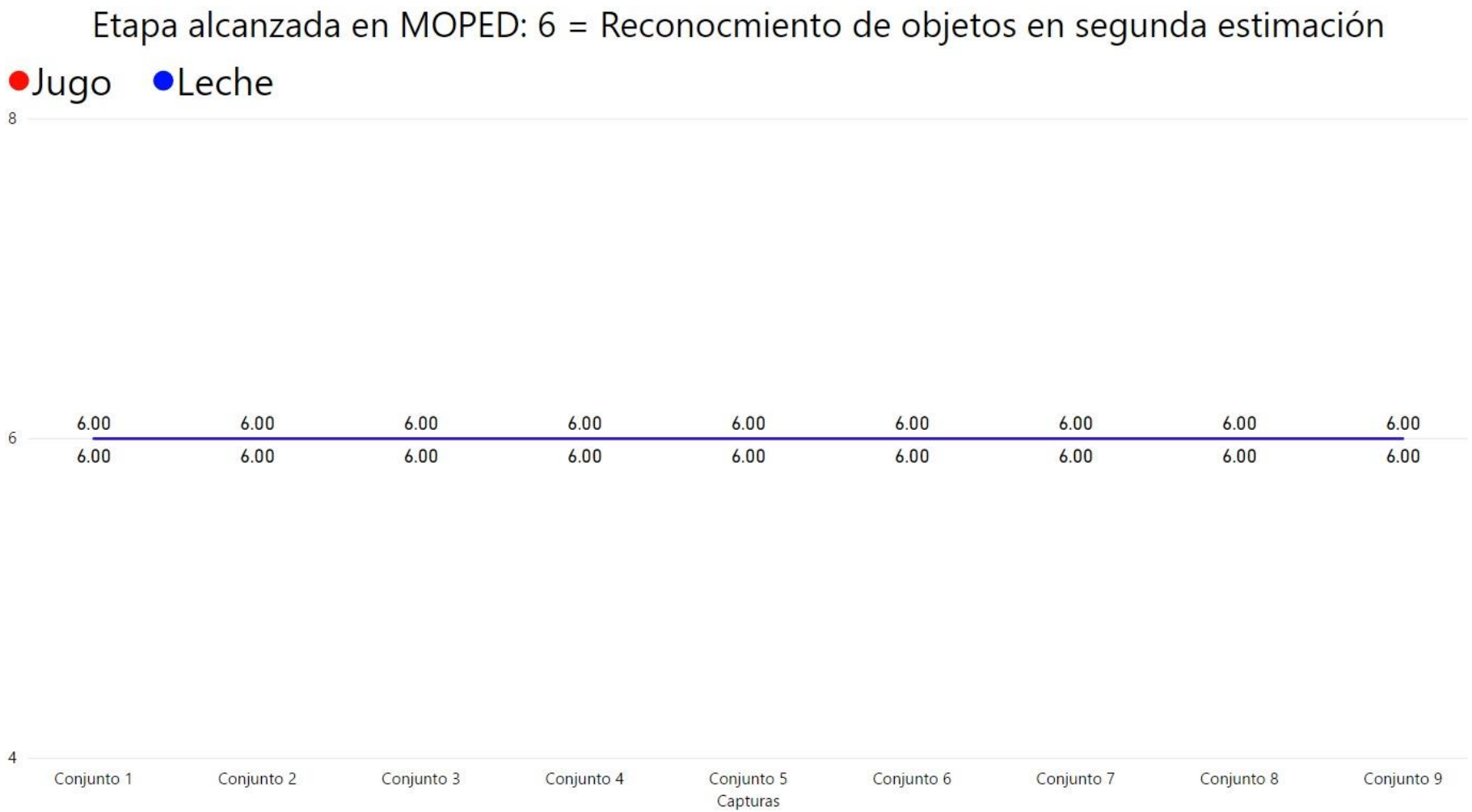

Fig. 4.40 – Etapa alcanzada en MOPED por el modelo en el reconocedor de Golem para el experimento de dos modelos con nueve capturas de conjuntos de objetos.

4.3 –Análisis de resultados.

De todo el conjunto de resultados, hay una serie de aseveraciones que se pueden hacer, primero, en aspectos generales, la cantidad de correspondencias obtenidas por el reconocedor entre el modelo y la escena va directamente ligada a qué tan cerca o lejos está el objeto de la cámara y a su vez a la cantidad de iluminación dentro de la habitación en ese momento, siendo las condiciones más favorables en luz de día o con iluminación clara y a una distancia prudente pero cercana a la cámara. También podemos inferir por la manera en que se realizaron los modelos, y los resultados obtenidos en el reconocimiento en distintas perspectivas, que la cara o lado por el que se empiece a realizar un modelo y la cantidad de puntos tomados en cada lado u orientación del objeto irá fuertemente ligado a qué predilección tenga ese mismo objeto a ser reconocido, si se comenzó el modelo por una perspectiva frontal y se tomaron muchas capturas iniciales en la cara frontal del objeto, es más probable que el reconocedor sea capaz de percibir el objeto con mayor facilidad si la cámara está encarando su lado frontal que cualquier otro.

Para las estimaciones de ICE en MOPED, encontramos que en su mayor parte el promedio de detecciones iniciales fue de 4 objetos por escena, las cuales en su mayoría fueron reducidas a una sola estimación de pose y reconocimiento del objeto con su segunda estimación, lo cual nos dice que al menos en ese sentido el modelador está haciendo un buen trabajo para proporcionar modelos robustos al reconocedor para lograr esto, sin embargo hubo varios casos aislados dentro de los experimentos que nos muestran casos que, aunque pocos, no logra reconocer siempre todas las perspectivas en ciertos objetos, esto, en la mayoría de casos porque ciertas perspectivas de algunos objetos no proporcionan suficientes características naturales y textura para obtener un número suficiente de correspondencias, (como en el lado lateral del shampoo que es casi completamente uniforme, o en la parte trasera de la lata de verduras que casi no cuenta con textura muy definida).

También se dieron algunos casos con identificaciones repetidas de un mismo objeto, de igual forma, dado que el modelo en ocasiones resulta inestable por tener traslapes de puntos unos con otros, ya que la alineación entre capturas no resulta suficiente para cubrir este margen de error. Sin embargo, las estadísticas generales son positivas ya que en el 88% de los casos, el reconocedor MOPED llegó a una convergencia donde obtuvo reconocimiento del objeto. (Como se aprecia en la tabla 5).

De igual forma en el 100% de los casos analizados con conjuntos de objetos, se llegó a una etapa preliminar de estimación de la posición y reconocimiento del objeto, sin embargo sólo en el 66% de los casos se llegó a una estimación de posición de un solo objeto sin errores por identificaciones repetidas. (Como se puede apreciar en los gráficos de las figuras 4.34 y 4.35).

Interesantemente en la mayoría de los casos donde no hubo una convergencia a una única identificación del objeto en la segunda iteración de ICE en MOPED, resultó ligado directamente con una identificación que excedía los 4 objetos promedio detectados en el resto de los casos durante la primera iteración de ICE en MOPED. (Como se observa en las gráficas de las figuras 4.33 y 4.34 o en las figuras 4.18, 4.19, 4.20 y 4.21).

Se observa que en general hubo más errores de identificaciones repetidas en las capturas de conjuntos de objetos que en las pruebas individuales, esto tiene lógica, ya que al añadir mayor complejidad a la escena es más probable que el reconocimiento se vuelva un poco más inestable, sin embargo a lo largo de todos los experimentos realizados observamos que MOPED llega en el mayor número de las veces a una etapa 6 de reconocimiento y estimación de la posición, siendo esta situación sólo mejorable por la

etapa de la recombinación de la posición y filtrado, pero por el momento los modelos no son tan robustos como desearía, debido a errores en puntos aislados que puedan colarse al modelo y una alineación de los puntos entre captura y captura poco robusta.

Estos problemas pueden solucionarse agregando una etapa de filtrado y de alineación de forma iterativa con cada captura tomada, de igual forma es altamente probable que el problema esté en el rendimiento de la computadora, ya que al no tener yo demasiado poder de procesamiento, sólo he podido correr el loop general de Kinect Fusion en un lapso en intervalos de 300 milisegundos, quizás aminorando este lapso a uno inmediato, que esté obteniendo una estimación, alineación e integración del volumen de reconstrucción observado a mayor velocidad, y con un mayor poder de procesamiento, podríamos obtener mejores matrices de transformación de la posición que nos den una alineación más precisa en los puntos del modelo.

Los resultados finales del experimento se concentran en la siguiente tabla resumen con su respectivo gráfico:

Capturas	*Cantidad de correspondencias encontradas. (Promedio).*	*Cantidad de agrupaciones de correspondencias MOPED. (Promedio).*	*Cantidad de objetos detectados en la primera estimación. (Promedio).*	*Cantidad de objetos reconocidos en la segunda estimación. (Promedio).*	*Grado de reconocimiento alcanzado en MOPED. (Promedio).*
Distancia 1 – Cerca	***293***	***3.4***	***4.1***	***1.1***	***5.7***
Distancia 2 – Media	***182.9***	***1.1***	***3.5***	***1.4***	***6***
Distancia 3 – Lejos	***63.7***	***1***	***3.8***	***1***	***6***
Iluminación 1 – Claro	***252.8***	***1.2***	***3.9***	***1.2***	***6***
Iluminación 2 – Medio	***161.8***	***1.2***	***4***	***1.2***	***6***
Iluminación 3 – Oscuro	***40.2***	***1.8***	***1.6***	***0.3***	***3.7***
Perspectiva 1 – Frontal	***195.3***	***1.2***	***4.2***	***1.2***	***6***
Perspectiva 2 – Lateral	***108.1***	***0.9***	***3.4***	***1***	***5.6***
Perspectiva 3 – Atrás	***136.1***	***1.2***	***2.6***	***1***	***5.1***
Perspectiva 4 - Isométrico	***178.5***	***1***	***4***	***1.3***	***6***
Con múltiples objetos en la escena	***647.6***	***11.76***	***5.23***	***1.46***	***6***

Tabla 11 – Tabla de resultados promedios finales.

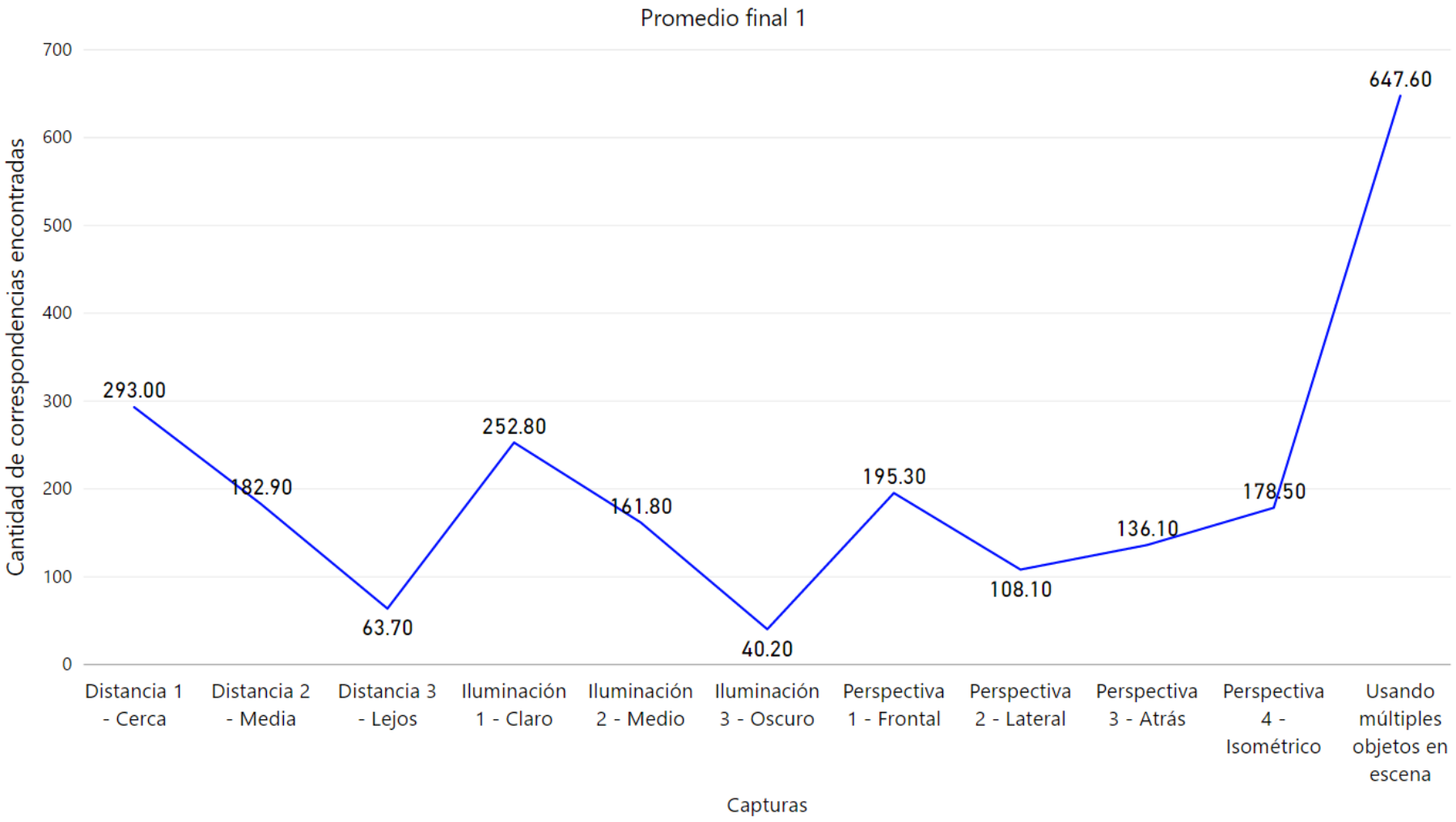

Fig. 4.41 – Cantidad de correspondencias encontradas en promedio para todos los objetos.

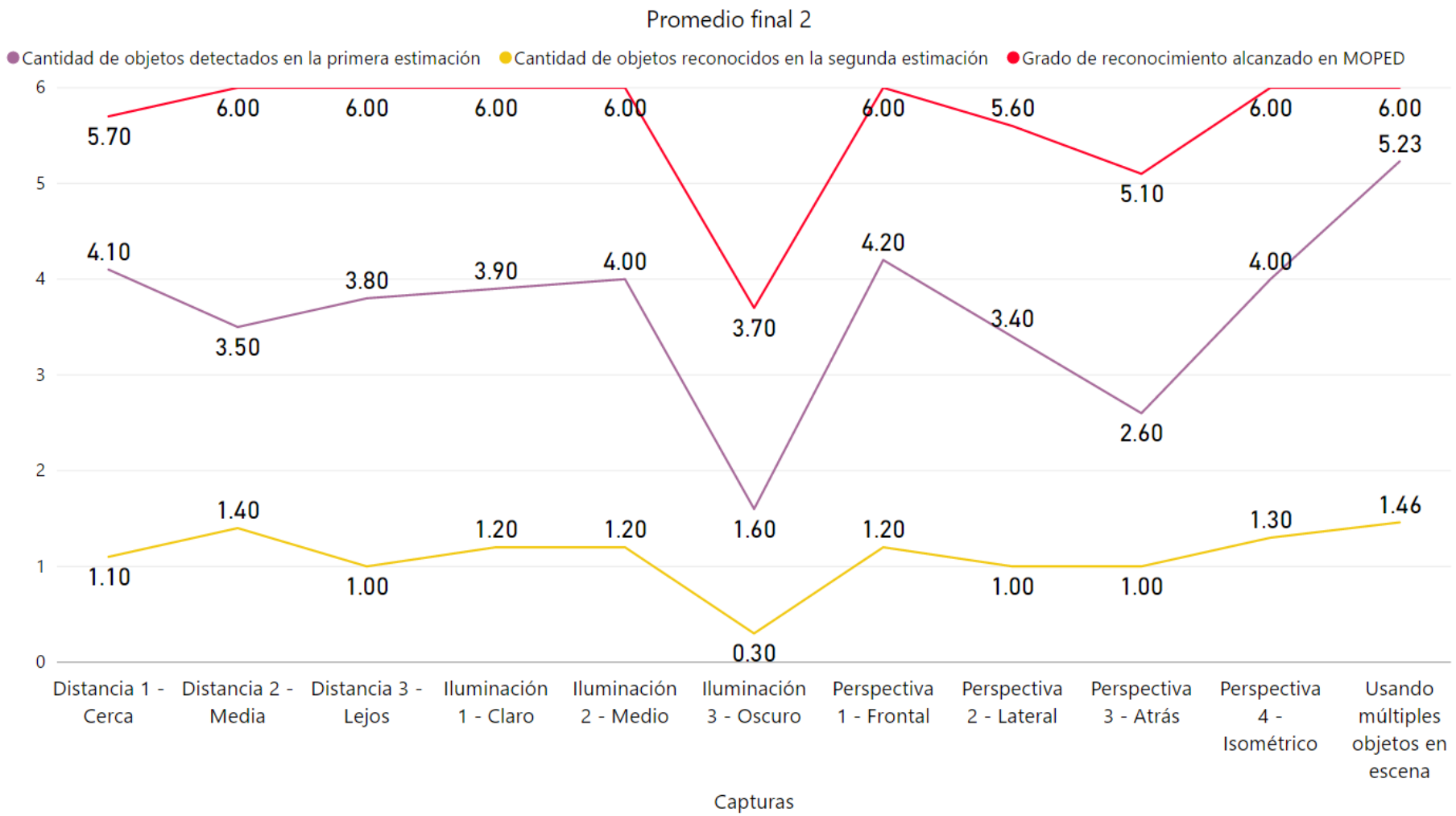

Fig. 4.42 – Cantidad de objetos detectados y reconocidos para la primera y segunda estimación, así como el grado de reconocimiento alcanzado en MOPED en promedio para todos los objetos.

Objetos a reconocer	Modelos									
	Leche	Jugo Bonafina	Té de flores	Shampoo	Jugo Boing	Avena	Lata de verduras	Salsa de tomate	Pasta dental	Tinte para el cabello
Leche	80	0	0	0	10	0	0	0	0	0
Jugo Bonafina	0	100	0	10	0	0	0	0	0	0
Té de flores	0	0	80	0	0	0	0	0	0	0
Shampoo	0	0	0	70	10	0	0	10	0	0
Jugo Boing	10	0	0	0	70	0	0	10	0	0
Avena	0	0	0	0	0	80	0	0	0	0
Lata de verduras	0	0	0	0	0	0	70	0	0	0
Salsa de tomate	0	0	0	0	0	0	10	60	0	0
Pasta dental	0	0	0	0	0	0	0	0	50	0
Tinte para el cabello	0	0	0	0	0	0	0	0	0	60
No reconoció ningún objeto	10	0	20	20	10	20	20	20	50	40
TOTAL	100	100	100	100	100	100	100	100	100	100

Tabla 12 – Tabla de confusión para el modelador de objetos 3D propuesto.

CAPÍTULO 5

5 – Conclusiones.

Como primer punto para las conclusiones destaco que el objetivo del modelador de objetos 3D se cumplió satisfactoriamente, sin embargo, hay ciertos aspectos del proceso de reconocimiento que podrían mejorarse todavía, en cuestión de obtención de información tanto de color, como profundidad del objeto, lograr una alineación de puntos parcial entre capturas, extracción de características naturales y emparejamiento de las mismas para obtener aquellos puntos más robustos el desempeño es bueno y cumple con su propósito, además que se adapta bien con el reconocedor que ya existe en Golem, las restricciones para la obtención de modelos son mínimas, consistentes en contar con un espacio homogéneo visualmente donde colocar el objeto para tener el menor ruido posible durante el proceso de modelado.

En este trabajo se utilizó una simple lona de pellón blanca para filtrar las características indeseables que pudiese detectar el sensor, esto acompañado de usar el Kinect con un filtro de distancia en profundidad más un ángulo de cierta inclinación sobre el piso y el objeto, a pesar de tener cierta informalidad, probó ser un método práctico, rápido y sencillo para capturar el objeto de reconstrucción sin ruido externo.

Con ayuda de un proceso de obtención de matrices de rotación y vector de traslación entre una captura y otra utilizando la función de SVD de OpenCV (descomposición en valores singulares) de una matriz de covarianza obtenida con la multiplicación de las matrices de puntos en una captura y en otra. Basándonos en el método de Least-Squares Rigid Motion Using SVD. [38]

El visualizador de la interfaz gráfica provee de una manera práctica y eficiente de estimar la posición de las coordenadas entre capturas, de estimar si hay puntos de ruido si alguna captura se tomó incorrectamente y también podemos borrar la última captura para tener un mejor control sobre el proceso, lo cual también se verá reflejado en el visualizador, el cual puede rotarse en cualquier dirección para facilitar la detección de errores o anomalías en el modelo.

El usar Kinect nos brinda modalidad y adaptabilidad en entornos en interiores para robots de servicio, ya que podemos censar en todo momento y en un amplio rango la profundidad y la distancia a posibles obstáculos así como obtener en tiempo real una captura con información de color y profundidad que adjunto a que el robot posea una base de datos con modelos de los objetos hace que pueda reconocer los mismos sólo introduciendo la información de Kinect directamente al reconocedor de MOPED.

En cuanto a la etapa de reconocimiento, esta presenta algunos problemas para detectar la estimación de la posición del objeto, puesto que si detecta matches así como agrupaciones entre el modelo guardado con el modelador y la imagen presentada como escena de prueba. Es capaz de determinar la posición en la primera iteración de ICE, pero a la hora de estimar la segunda posición, a pesar de que sí lo hace, esta suele venir acompañada de múltiples detecciones a pesar de sólo ser un objeto, por lo que la misma no está lo suficientemente refinada, esto es debido a que al obtener los puntos clave con SIFT, y subsecuentemente hacer el matching con FLANN, se hace un promedio de las coordenadas que encontraron correspondencias, los puntos quedan suspendidos sobre posiciones incorrectas a lo que se muestra en la escena, por lo tanto MOPED se confunde y no es capaz de refinar la pose más allá de la primera etapa. Un problema similar surgió sobre la elaboración del proyecto, cuando MOPED no era capaz ni siquiera de detectar ningún objeto en la escena, a pesar que si detectaba correspondencias

entre el modelo y la misma, esto debido a que Kinect al tener su sentido de orientación horizontal invertido, (el eje x incremental hacia la izquierda), se necesitaba de presentar imágenes invertidas para que pudiese estimar correctamente la posición y orden de los puntos para detectar el objeto exitosamente, de modo que el problema se traslada aquí en el sentido de una alineación entre las nubes de puntos entre una captura y otra, esta desalineación, que a pesar que se acerca mucho con la descomposición en valores singulares para la obtención de la matriz de rotación, es imperfecta y los puntos con correspondencias en ocasiones suelen salir muy alejados en el espacio, y al promediarlos acaban en posiciones completamente disonantes con la verdadera organización de características naturales inherentes al objeto.

Este problema puede solucionarse aplicando un subproceso de ICP o de alineación entre puntos cuando se obtienen las capturas, que, a pesar que Kinect Fusion ya tiene esta función implementada en el proceso de alineación e integración del volumen de reconstrucción, los puntos clave no se sacan de este, sino de las imágenes de color y profundidad respectivamente trasladadas al espacio de coordenadas de la cámara y finalmente al espacio de coordenadas globales, por lo que podrían verse como capturas "aisladas" que nunca se integran al volumen en sí, de forma que no se hace este subproceso de alineación propio de funciones como "*AlignDepthframeToReconstruction*", "*AlignPointClouds*" o "*ProcessFrame*", esta última que incluye tanto el proceso de alineación de "*AlignDepthframeToReconstruction*" para alinear la última imagen de profundidad obtenida al resto del volumen como la función de integración de ambos.

Alternativamente se puede utilizar la función de "*AlignPointClouds*" de manera independiente con 2 capturas de profundidad transformadas a *"frame"* de nube de puntos para alinearlas sin necesidad del volumen de reconstrucción, esto se intentó implementar en algún punto del trabajo, sin embargo el error de alineación no mejora significativamente respecto al visto con anterioridad utilizando la matriz de transformación de la posición de Kinect Fusion o con el método implementado con SVD, quizás un subproceso completo utilizando ICP o Levenberg-Marquardt o quizás algún conjunto de técnicas de alineación de puntos empleadas en la librería de código abierto de PCL pudiesen servir igualmente.

En cuanto a las condiciones ideales para obtener modelos, recomendaría que fuera a luz de día con una iluminación clara, a una distancia cercana-media entre el objeto y el sensor (alrededor de 30cm a 50cm), e intentar evitar en la medida de lo posible ambientes con iluminación demasiado oscura, (al grado que no sea posible discernir los colores del objeto en cuestión).

El procedimiento con el que se obtuvieron los mejores modelos fué el de empezar con el objeto en una posición isométrica e ir tomando capturas desde aquí hasta la parte trasera del mismo e intentar cerrar el círculo completo en un promedio de 8 a 12 capturas aproximadamente.

A pesar de una imperfecta recombinación y estimación de la posición y orientación del objeto, en términos de detección del objeto y obtención de correspondencias, la conjunción de SIFT y FLANN probó ser bastante buena con un promedio del 89% de los casos donde MOPED superó la etapa 4 donde ya detecta objetos en base al modelo y la escena. (Gráficos de las figuras 4.22 y 4.23, así como la tabla 5). Modelos como la leche o el empaque de jugo "Bonafina" prueban el enorme potencial que tiene el modelador de objetos 3D, ya que se comprobó exitoso en el reconocimiento a todas las distancias, todas las perspectivas del objeto y todas las condiciones de iluminación propuestas.

BIBLIOGRAFÍA.

1. Manuel Hernández, Carlos Calles, Juan Rodríguez. (2015). “Robótica: Análisis, modelado, control e implementación”. Ciudad Victoria, Tamaulipas, México. Omnia Publisher SL.

2. Murphy K., Torralba A., Eaton D., Freeman W. (2006). “Object Detection and Localization Using Local and Global Features”. In: Ponce J., Hebert M., Schmid C., Zisserman A. (eds) Toward Category-Level Object Recognition. Lecture Notes in Computer Science, vol. 4170. Springer, Berlin, Heidelberg.

3. Dalal Navneet, Triggs Bill. (s.f.). “Histogram of Oriented Gradients for Human Detection”. INRIA Rhone-Alps, ˆ655 avenue de l’Europe, Montbonnot 38334, France.

4. Pedro F. Felzenszwalb, Daniel P. Huttenlocher. (2005, enero). “Pictorial Structures for Object Recognition”. D.P. International Journal of Computer Vision, Volume 61, Issue 1, pp 55–79.

5. Ross Girshick. (2013, abril). “Deformable part models”. UC Berkeley. CS231B Stanford University, Guest Lecture.

6. Kate Saenko, Sergey Karayev, Yangqing Jia, Alex Shyr, Allison Janoch, Jonathan Long, Mario Fritz, Trevor Darrell. (2011, septiembre). “Practical 3-D Object Detection Using Category and Instance-level Appearance Models”. San Francisco, CA., USA. IEEE/RSJ International Conference on Intelligent Robots and Systems.

7. Alvaro Collet, Siddhartha Srinivasa. (2011, septiembre). “The MOPED framework: Object Recognition and pose estimation for manipulation”. The International Journal of Robotics Research. SAGE Publications.

8. David G. Lowe. (2004, enero). “Distinctive Image Features from Scale-Invariant Keypoints”. Vancouver, B.C., Canada. International Journal of Computer Vision. Volume 60, Issue 2, pp 91-110.

9. Marco Antonio Chavarria Fabila. (2009, junio). “Monocular Pose Estimation on Global and Local Features”. Tesis de Doctorado. Christian-Albrechts-Universität zu Kiel eingereicht hat.

10. Satoshi Fujimoto. (2015). “Satoshi Robato Fujimoto – Kinect Fusion V2”. Kumamoto, Japón. Disponible en <https://github.com/SatoshiRobatoFujimoto/KinectFusionV2/tree/master/KinectFusionV2> [en línea] [Consulta: 15 de diciembre de 2016].

11. James Philbin. (2010). “Scalable Object Retrieval in Very Large Image Collections”. Tesis de Doctorado. University of Oxford.

12. Grupo Golem. (2013). IIMAS, UNAM. Disponible en <http://golem.iimas.unam.mx/gallery.php?lang=en&sec=gallery> [en línea] [Consulta: 15 de abril de 2017].

13. K. Mikolajczyk and C. Schmid. (2003). “A performance evaluation of local descriptors”. IEEE Conference on Computer Vision and Pattern Recognition.

14. G. Stockman. (1987). “Object recognition and localization via pose clustering”. Computer Vision, Graphics and Image Processing, Volume 40. pp 361-387.

15. Alvaro Collet, Dmitry Berenson, Siddhartha S. Srinivasa, Dave Ferguson. (2009, Agosto). “Object recognition and full pose registration from single image for robotic manipulation”. ICRA’09 Proceedings of the 2009 IEEE international conference on Robotics and Automation.

16. Juan Luis Elorriaga. (2015, junio). “Robótica Industrial y Robótica de Servicio”. Asociación Española de Robótica y Automatización Tecnologías de la Producción. Disponible en <https://es.slideshare.net/innobasque/juan-luis-elorriaga-aer> [en línea] [Consulta: 07 de abril de 2017].

17. Marius Muja, David G. Lowe. (2014). “Scalable Nearest Neighbor Algorithms for High Dimensional Data”. IEEE Transactions on pattern analysis and machine intelligence. Vol. 36, Issue 11, pp 2227-2240.

18. OpenCV Development Team. (2011-2014). “OpenCV 3.0 Dev Documentation”. Disponible en <http://docs.opencv.org/3.0beta/doc/py_tutorials/py_feature2d/py_sift_intro/py_sift_intro.html#sift-intro> [en línea] [Consulta: 14 de abril de 2017].

19. T. B. Sebastian, B. B. Kimia. (2002). “Metric-based shape retrieval in large databases”. Proc. IEEE Conf. Comput. Vis. Pattern Recog., Vol. 3, pp 291–296.

20. Srinivasa S.S., Ferguson D., Helfrich C.J., Berenson D., Collet A., Diankov R. (2010). “HERB: A Home Exploring Robotic Butler”. Autonomous Robots. Vol. 28, pp 5–20.

21. Microsoft. (2014). “Kinect for Windows v2 sensor”. Disponible en <https://news.microsoft.com/kinect-for-windows-v2-sensor-2/#Q4v2YW2O3Tcl7eXS.97> [en línea] [Consulta: 22 de abril de 2017].

22. Microsoft. (2010). “Kinect for Windows Sensor Components and Specifications”. Disponible en <https://msdn.microsoft.com/en-us/library/jj131033.aspx > [en línea] [Consulta: 22 de abril de 2017].

23. Microsoft. (2010). “Kinect for Windows Sensor Interaction Space”. Disponible en <https://msdn.microsoft.com/en-us/library/hh973071.aspx> [en línea] [Consulta: 22 de abril de 2017].

24. Microsoft. (2010). “Kinect for Windows Sensor Coordinate Spaces”. Disponible en <https://msdn.microsoft.com/en-us/library/hh973078.aspx#Depth_Ranges> [en línea] [Consulta: 22 de abril de 2017].

25. Microsoft. (2010). “Kinect for Windows Sensor Coordinate Mapping”. Disponible en <https://msdn.microsoft.com/en-us/library/dn785530.aspx> [en línea] [Consulta: 22 de abril de 2017].

26. Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Andrew Fitzgibbon. (2011, octubre). “Kinect Fusion: Real-time dense surface mapping and tracking”. ISMAR ’11 Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality, pp 127-136.

27. Microsoft. (2014). “Kinect Fusion”. Disponible en < https://msdn.microsoft.com/en-us/library/dn188670.aspx> [en línea] [Consulta: 23 de abril de 2017].

28. Richard Szeliski. (2010, septiembre). “Computer Vision: Algorithms and Applications”. Springer.

29. Ville Kyrki. (2002, octubre). “Local and global feature extraction for invariant object recognition”. Tesis de Doctorado. Lapeenranta University of Technology.

30. Hassaballah M., Abdelmgeid Aly Amin, Alshazly Hammam A. (2016, febrero). “Image features detection, description and matching”. Volume 630 of the series Studies in Computational Intelligence. pp 11-45.

31. Y. LeCun, F. J. Huang, and L. Bottou. (2004). “Learning methods for generic object recognition with invariance to pose and lighting”. In Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 97–104.

32. J. Liebelt, C. Schmid, and K. Schertler. (2008). “Viewpoint-independent object class detection using 3d feature maps”. In Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 1–8.

33. Bernd Heisele, Gunhee Kim, Andrew J. Meyer. (2009, septiembre). “Object recognition with 3D Models”. In A. Cavallaro, S. Prince and D. Alexander, editors. Proceedings of the British Machine Conference. pp 29.1-29.11.

34. Serre T., Wolf L., Poggio T. (2005, junio). “Object recognition with features inspired by visual cortex”. Computer vision and pattern recognition (CVPR) IEEE Computer Society Conference. San Diego, CA, USA.

35. Maximilian Riesenhuber, Tomaso Poggio. (2000). “Models of object recognition”. Cambridge, Massachusetts, USA. Nature Neuroscience, 3. pp 1199-1204.

36. Sutherland, I. E. (1974). “Three-dimensional data input by tablet”. Proceedings of the IEEE, Vol. 62, Issue 4. pp 453–461.

37. Sungjoon Choi, Qian-Yi Zhou, Stephen Miller, Vladlen Koltun. (2016). “A large dataset of object scans”. Disponible en <http://redwood-data.org/3dscan/dataset.html?c=plant&i=8680> [en línea] [Consulta: 28 de abril de 2017].

38. Olga Sorkine-Hornung and Michael Rabinovich. (2017, enero). “Least Squares Rigid Motion using SVD”. ETH Zurich.