eman ta zabal zazu

Universidad del País Vasco
Euskal Herriko Unibertsitatea

INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Departamento de Ciencia de la Computación e Inteligencia Artificial

Universidad del País Vasco (UPV/EHU)

Estrategias de visión por computador para la estimación de pose en el contexto de aplicaciones robóticas industriales: avances en el uso de modelos tanto clásicos como de Deep Learning en imágenes 2D y nubes de puntos

Ibon Merino Bermejo

Dirección

Basilio Sierra
Anthony Remazeilles

18 de julio de 2023

Agradecimientos

Con estas palabras voy a dar comienzo al final de esta larga etapa. Una etapa que ha tenido sus buenos y malos momentos, pero que ambos me han permitido desarrollarme en diferentes aspectos. Este recorrido que ha durado algo más de 4 años lo inicié siendo aún más ignorante de lo que soy ahora. No tenía ni idea de lo que me venía por delante.

Desde un inicio he tenido la suerte de encontrarme en el camino a personas maravillosas que me han ayudado a crecer como persona y como profesional. Primero quiero agradecer a Tecnalia por la oportunidad que me ha dado y a todos con los que he coincidido. Sobre todo quiero agradecer a la gente de mi grupo de robótica flexible. Gracias por todo lo que me habéis enseñado, por toda la ayuda que me habéis ofrecido y sobre todo por aguantar mis chistes malos.

Quiero agradecer a las 3 personas que han sido fundamentales en este recorrido: a mis directores de tesis, Basi y Anthony, por su paciencia y por su ayuda; y a Jon, por su dedicación y todo el apoyo que me ha dado durante todo este tiempo. Gracias a todos por hacer que este camino haya sido más llevadero.

Como las raíces son importantes y más aún si no te despegas de ellas, quiero agradecer al colegio donde estudié, Salesianos Deusto, y a toda la obra de los Salesianos que me han formado como persona. Han sido muchos años en ese colegio y posteriormente como monitor. Gracias por todo lo que me habéis enseñado y por todo lo que me habéis dado. Y gracias a todos los monitores y chavales que habéis compartido camino conmigo porque las experiencias adquiridas en esos años son invaluables.

También quiero agradecer a todos mis amigos que me han apoyado siempre. Por un lado a mis amigos de la universidad en especial a Judity e Itziar. Por otro lado a mi querida "Villa Lola" que me ha permitido distraerme, reírme y disfrutar en los momentos de más tensión. A mi "Zulo" que siempre han estado ahí presentes. A mis queridas "Txandaleras" que tienen un lugar especial en mi corazón. A mi pareja Gonzalo que me ha dado ánimos y fuerzas en los momentos más duros. Y a todos los demás amigos que han estado ahí siempre.

Como los postres, lo más dulce es el final. Quiero agradecer especialmente a mi

familia: a mi aita, que muy bien me ha enseñado que “más vale maña que fuerza” y me ha inspirado a ser resolutivo y perspicaz; a mi ama, que siempre ha estado ahí con su amor incondicional, tanto en lo bueno como en lo malo y de la que he aprendido que la vida son dos días y hay que sacarle el jugo a cada momento; a mi hermano, el pequeño de la casa, al que considero uno de mis mejores amigos; a mi aitite, que me ha demostrado que siempre hay que luchar y seguir hacia adelante; a mi amama, que es como mi segunda madre y todo lo he aprendido de ella; y al resto de mi familia por estar siempre ahí.

Tras todo esto, no me queda otra cosa que seguir en este camino de la eterna ignorancia que espero que me siga nutriendo de conocimientos, de experiencias y de personas maravillosas como lo ha hecho hasta ahora.

Resumen

La visión por computador es una tecnología habilitadora que permite a los robots y sistemas autónomos percibir su entorno. Dentro del contexto de la industria 4.0 y 5.0, la visión por computador es esencial para la automatización de procesos industriales. Entre las técnicas de visión por computador, la detección de objetos y la estimación de la pose 6D son dos de las más importantes para la automatización de procesos industriales. Para dar respuesta a estos retos, existen dos enfoques principales: los métodos clásicos y los métodos de aprendizaje profundo. Los métodos clásicos son robustos y precisos, pero requieren de una gran cantidad de conocimiento experto para su desarrollo. Por otro lado, los métodos de aprendizaje profundo son fáciles de desarrollar, pero requieren de una gran cantidad de datos para su entrenamiento.

En la presente memoria de tesis se presenta una revisión de la literatura sobre técnicas de visión por computador para la detección de objetos y la estimación de la pose 6D. Además se ha dado respuesta a los siguientes retos: (1) estimación de pose mediante técnicas de visión clásicas, (2) transferencia de aprendizaje de modelos 2D a 3D, (3) la utilización de datos sintéticos para entrenar modelos de aprendizaje profundo y (4) la combinación de técnicas clásicas y de aprendizaje profundo. Para ello, se han realizado contribuciones en revistas de alto impacto que dan respuesta a los anteriores retos.

Laburpena

Ordenagailu bidezko ikusmena robotei eta sistema autonomoei beren ingurunea hautemateko aukera ematen dien teknologia gaitzailea da. 4.0 eta 5.0 industriaren testuinguruan, ordenagailuen bidezko ikusmena funtsezkoa da prozesu industrialak automatizatzeko. Ordenagailuen bidezko ikusmen-tekniken artean, objektuen detekzioa eta 6D posearen estimazioa dira prozesu industrialen automatizaziorako garrantzitsuenetarikoak. Erronka horiei erantzuteko, bi ikuspegi nagusi daude: metodo klasikoak eta ikasketa sakoneko metodoak. Metodo klasikoak sendoak eta zehatzak dira, baina ezagutza aditu ugari behar dute garatzeko. Bestalde, ikasketa sakoneko metodoak erraz garatzen dira, baina datu asko behar dira entrenatzeko.

Tesi-memoria honetan, objektuak detektatzeko eta 6D posea estimatzeko ordenagailuen bidezko ikusmen-teknikei buruzko literaturaren berrikuspena aurkezten da. Gainera, honako erronka hauei erantzuna eman zaie: (1) ikuspegi-teknika klasikoen bidez posea estimatzea, (2) 2D ereduen ikaskuntza 3Dra transferitzea, (3) datu sintetikoak erabiltzea ikasketa sakoneko ereduak entrenatzeko, eta (4) teknika klasikoak eta ikasketa sakonekoak konbinatzea. Horretarako, eragin handiko aldizkarietan ekarpenak egin dira, aurreko erronkei erantzuteko.

Índice de contenidos

Índice de figuras

Índice de tablas

Parte I

Ámbito de investigación

CAPÍTULO 1

Introducción y motivación

1.1. Introducción

En los últimos años la industria ha experimentado muchos cambios a un ritmo vertiginoso. Ha pasado una década desde que se acuñó el término Industria 4.0 [1] y ya se contempla la siguiente revolución industrial, la Industria 5.0. Si la cuarta revolución industrial se centra en la automatización y el intercambio de datos mediante IoT, sistemas ciber-físicos o computación en la nube, la Industria 5.0 se centra en la sostenibilidad, resiliencia y en el bienestar humano mientras mantiene los criterios que marca la Industria 4.0. La Comisión Europea, quien bautizó esta nueva era, expone en su informe [2]:

> En lugar de preguntarnos qué podemos hacer con la nueva tecnología, nos preguntamos qué puede hacer la tecnología por nosotros.
>
> En lugar de pedir al trabajador de la industria que adapte sus habilidades a las necesidades de una tecnología que evoluciona rápidamente, queremos utilizar la tecnología para adaptar el proceso de producción a las necesidades del trabajador.

La visión por computador es un campo importante dentro de este paradigma. Es importante que los robots sepan identificar su entorno. Por ejemplo, de nada sirve que un robot sepa coger objetos si no sabe donde está el objeto. De la misma forma, un robot no puede ejecutar movimientos de forma amigable para el operario si no sabe donde está para poder evitarlo o actuar de forma más sutil cuando se aproxime a él. Esto es importante para el bienestar del operario y ayuda a adaptar la robótica al operario.

En la Figura 1.1 se muestra un brazo robótico colaborativo Doosan con una cámara 3D de luz estructurada Zivid. Este es un caso de uso típico de visión por computador en la industria. La visión por computador permite realizar tareas como localización de objetos [3, 4, 5], detección de objetos [6, 7, 8], tracking de objetos [9, 10, 11], detección de humanos [12, 13, 14], reconstrucción de objetos [15, 16, 17], segmentación semántica [18, 19], entre muchas otras, las cuales encajan perfectamente en este paradigma de Industria 4.0 e Industria 5.0.

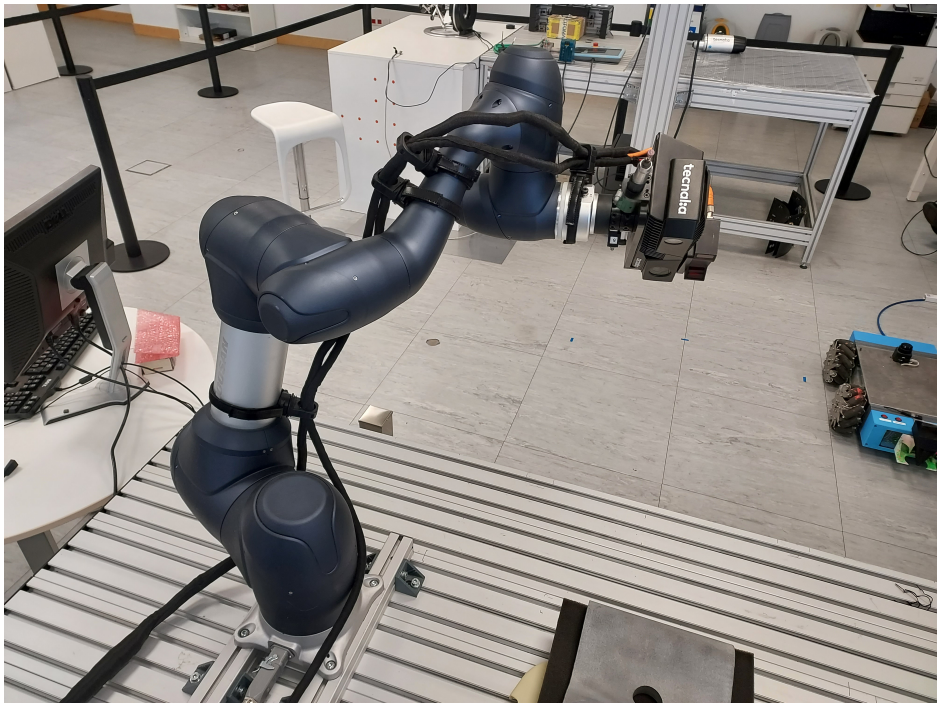


Figura 1.1: Brazo robótico colaborativo Doosan con una cámara 3D de luz proyectada Zivid.

El trabajo presentado tiene como motivación la preparación para esta Industria 5.0 (ver Sección 1.2), ya que se ha realizado en el centro de investigación orientado a las necesidades de la industria TECNALIA (ver Sección 1.3.1) en colaboración con la UPV-EHU (ver Sección 1.3.2) debido al trabajo de colaboración entre estos dos y con el autor. El trabajo realizado ha sido aplicado en varios proyectos (ver Sección 1.3.3) los cuales intentan desarrollar tecnologías habilitantes dentro de la Industria 5.0. Finalmente, se han definido unas hipótesis de investigación (ver Sección 1.4) las cuales se han intentado responder con las contribuciones realizadas (ver Capítulo 3).

1.2. Motivación

En las últimas décadas el sector industrial ha experimentado un cambio radical en su forma de producción. La automatización de procesos ha sido una de las claves para que la industria pueda competir en un mercado globalizado. Para ello, se han desarrollado nuevas tecnologías que permiten automatizar procesos de forma más eficiente y segura. Originalmente esta programación era predefinida, es decir, cada robot realizaba un movimiento o acción concreta repetidamente. Gracias a los avances en inteligencia artificial y sensórica, se ha dotado a los robots de mayor autonomía. Esto es, permitimos al robot actuar en concordancia con su entorno. Por ejemplo, si un robot tiene que realizar una tarea en un espacio de trabajo, es capaz de detectar los objetos que hay en ese espacio y planificar su trayectoria para realizar la tarea. Para realizar esto son necesarias varias tecnologías como una capacidad de adaptación para gestionar las órdenes dinámicas del robot o una visión artificial para detectar qué hay en nuestro espacio de trabajo.

En el contexto de este proyecto de tesis, dentro de la visión por ordenador se ha centrado en la detección de objetos y la estimación de su pose 6D (translación y rotación). La Figura 1.2 muestra un ejemplo de este problema.

Figura 1.2: Estimación de la pose de los objetos del dataset Linemod proyectada sobre una imagen de test.

Dentro de la detección de objetos y estimación de pose 6D, se pueden diferenciar dos tipos de técnicas: las técnicas clásicas y las de aprendizaje profundo. Llamamos técnicas o métodos clásicos a aquellas técnicas que predominaban antes del auge de los métodos de aprendizaje profundo. Estos métodos no requieren de una gran cantidad de datos para entrenar y han sido utilizados durante décadas de forma satisfactoria. Aun así, en esta última década los métodos de aprendizaje profundo han demostrado obtener unos mejores resultados para muchos casos y su uso se ha extendido. Es por ello que el uso de ambos tipos de técnicas todavía tiene cabida y se pueden utilizar en conjunto para obtener mejores resultados. Por eso la principal motivación de este proyecto de tesis ha sido estudiar las técnicas existentes de ambos tipos y extraer el máximo potencial de ambas combinándolas.

1.3. Contexto

1.3.1. TECNALIA

TECNALIA es un centro privado de investigación aplicada con varias sedes en todo el planeta. Es el centro de investigación aplicada y desarrollo tecnológico más grande de España, un referente en Europa y miembro del Basque Research and Technology Alliance (BRTA). TECNALIA tiene dos premisas principales: transformar investigación tecnológica en prosperidad y ser agentes de transformación de las empresas y de la sociedad para su adaptación a los retos de un futuro en continua evolución. TECNALIA dispone de 5 unidades operativas: *Transición energética, climática y urbana*; *Industria y movilidad*; *Salud*; *Digital*; y, *Lab Services*. Dentro de cada unidad operativa hay áreas de negocio. El autor pertenece al área de negocio *Medios de Producción y Robótica* de la unidad operativa *Industria y movilidad*. Más concretamente, a la plataforma de *Robótica para flexibilidad industrial*. Esta plataforma está especializada en el desarrollo de tecnologías robóticas inteligentes, autónomas, adaptativas y fáciles de programar.

Gracias a la plataforma de *Robótica para flexibilidad industrial* el autor ha podido colaborar en proyectos reales (Sección 1.3.3) para añadir habilidades de visión a los robots. La Figura 1.3 muestra el stand de TECNALIA en la Bienal de Máquina Herramienta (BIEMH) de 2022. En este stand se muestran las tecnologías habilitadoras de TECNALIA para la Industria 5.0. El autor participó en la creación de las herramientas de visión por computador para la detección de las piezas de los demostradores.

1.3.2. UPV/EHU

El autor ha estado ligado a la Universidad del País Vasco / Euskal Herriko Unibertsitatea (UPV/EHU) desde que comenzó sus estudios de Grado. Durante los últimos años ha colaborado con el grupo de Robótica y Sistemas Autónomos (RSAIT) de la UPV/EHU en varios proyectos de investigación.

Figura 1.3: Stand de TECNALIA de la BIEMH 2022

1.3.3. Proyectos

Durante el transcurso del doctorado del autor, éste ha participado activamente en proyectos de TECNALIA, tanto directamente ligados con su proyecto de tesis como otros que no lo están. Estos proyectos, que son reflejo de las necesidades tecnológicas que se demandan en entornos industriales, han dado sentido a esta proyecto de tesis doctoral. Los proyectos que han permitido desarrollar este proyecto de tesis, en su mayoría de financiación pública competitiva, han sido financiados por la Comisión Europea (Programa Horizon 2020) y el Gobierno Vasco (Programa Elkartek). La Figura 1.4 muestra el orden cronológico de los proyectos y el origen de la subvención.

Los proyectos ligados a este proyecto de tesis son los siguientes:

- **Elkarbot** (Figura 1.5): El proyecto Elkarbot con nº de proyecto KK-2020/00092 pertenece al programa de ayudas a la investigación colaborativa Elkartek. El principal objetivo de este proyecto es el de dotar al País Vasco de herramientas para su robotización. Para realizar esto se tienen en cuenta tres premisas a lograr: mitigar el riesgo percibido al desarrollo de tecnología de robótica flexible propia por los integradores; habilitar el uso de la robótica no solo en los sectores de automoción y aeronáutica; y coordinar mejor las importantes capacidades tecnológicas de la RVCTI (Red Vasca de Ciencia, Tecnología e Innovación) en torno a la robótica, en estrecha relación con el BDIH (Basque Digital Innovation Hub) y dentro del

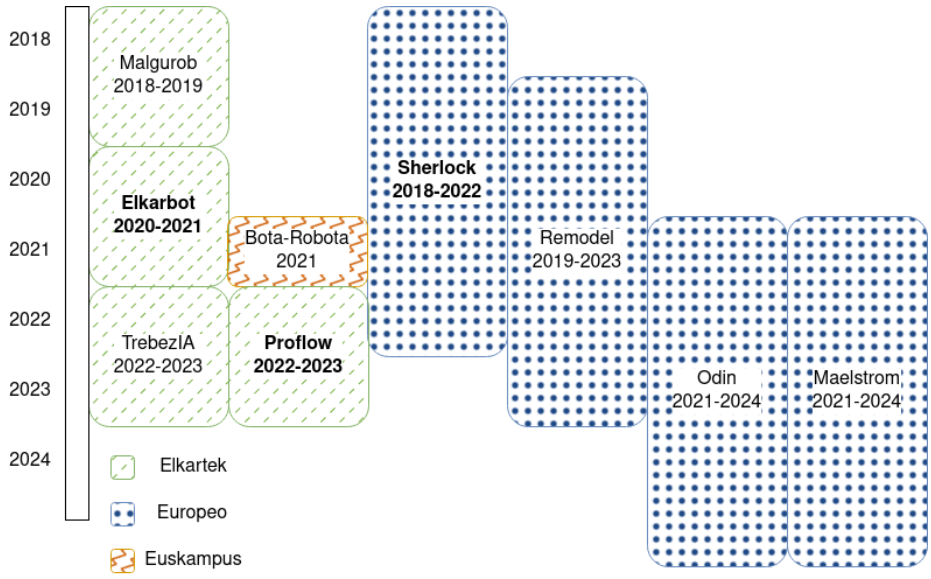


Figura 1.4: Orden cronológico de los proyectos y origen de la subvención. En negrita están aquellos proyectos ligados al proyecto de tesis.

contexto de BRTA (Basque Research and Technology Alliance). Este proyecto ha tenido una duración de dos años empezando en 2020. La principal aportación del autor en este proyecto ha sido diseñar algoritmos de clasificación de piezas industriales usando arquitecturas de aprendizaje profundo 3D preentrenadas con modelos 2D. Esto ha permitido identificar piezas de forma efectiva usando cámaras 3D. Como resultado de este proyecto se ha publicado el siguiente artículo:

Merino, I., Azpiazu J., Remazeilles A., Sierra B., 2021. **3D Convolutional Neural Networks Initialized from Pretrained 2D Convolutional Neural Networks for Classification of Industrial Parts**. Sensors 21, 1078. MDPI. https://doi.org/10.3390/s21041078

- **Sherlock - Seamless and safe human centred robotic applications for novel collaborative workplaces** (Figura 1.6): Es un proyecto europeo bajo el programa Horizon 2020 cuyo indicador del acuerdo de subvención (Grant Agreement) es 820689. Sherlock tiene como objetivo introducir las últimas tecnologías en seguridad robótica en entornos de producción, habilitándolas con mecatrónica inteligente y cognición basada en IA, creando, de esta forma, estaciones HRC (Human Robot Collaboration) eficientes que están diseñadas para ser seguras y garantizar la aceptación y bienestar de los operarios. Este proyecto tiene una du-

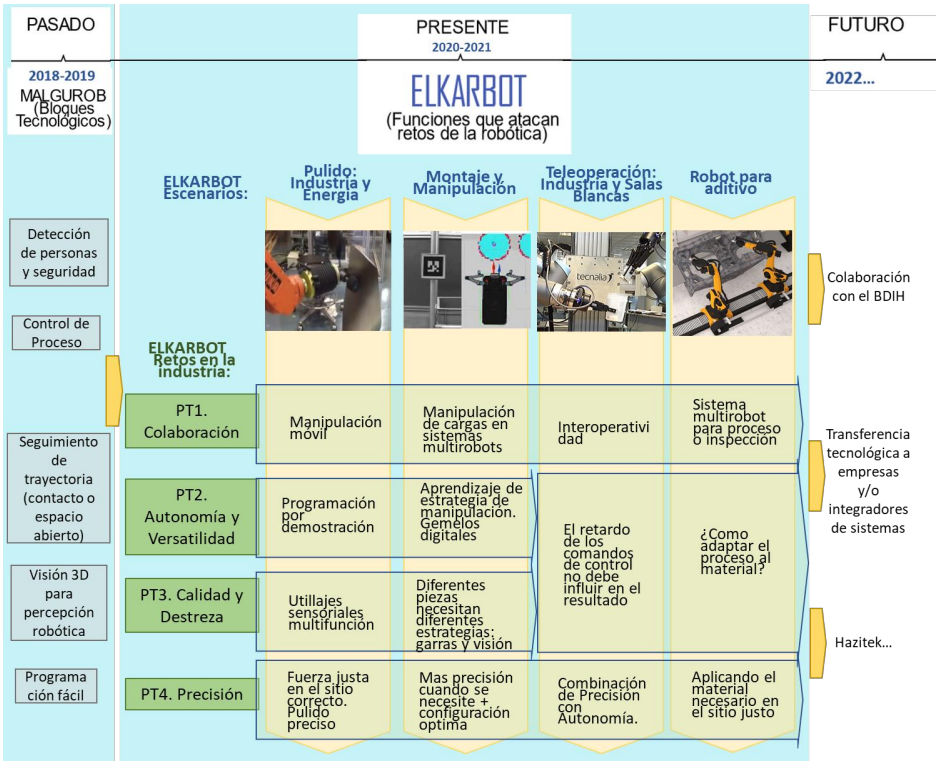


Figura 1.5: Paquetes de trabajo del proyecto Elkarbot para responder a los retos de la robótica industrial del futuro.

ración de cuatro años entre 2018 y 2022. En este proyecto el autor ha contribuido diseñando técnicas de reconocimiento de piezas industriales usando descriptores locales y globales. Inicialmente estas técnicas han permitido clasificar piezas industriales y una vez identificadas obtener la pose 6D de la pieza para poder cogerla. Durante este proyecto se han publicado los siguientes artículos:

Merino, I., Azpiazu, J., Remazeilles, A., Sierra, B., 2019. **2d Image Features Detector and Descriptor Selection Expert System**. Computer Science & Information Technology (CS & IT), AIRCC Publishing Corporation. https://doi.org/10.5121/csit.2019.91206

Merino, I., Azpiazu, J., Remazeilles, A., Sierra, B., 2019. **2D Features-based detector and descriptor selection system for hierarchical recognition of industrial parts**. International Journal of Artificial Intelligence & Applications (IJAIA), AIRCC Publishing Corporation. https://doi.org/10.5121/ijaia.2019.10601

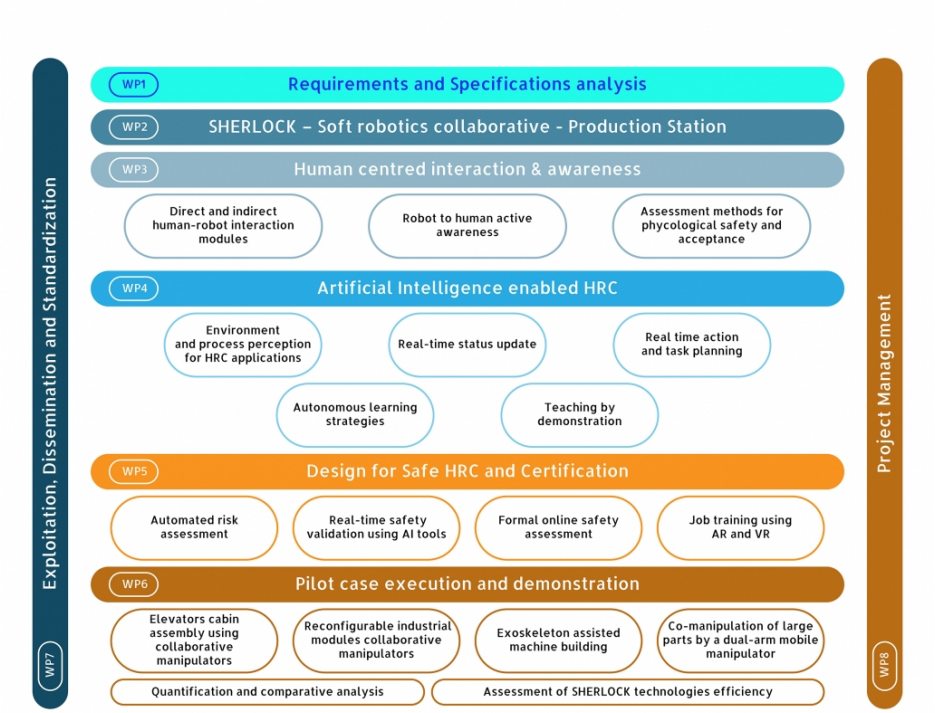


Figura 1.6: Paquetes de trabajo del proyecto Sherlock con sus respectivas tareas.

Merino, I., Azpiazu, J., Remazeilles, A., Sierra, B., 2020. **Histogram-Based Descriptor Subset Selection for Visual Recognition of Industrial Parts**. Applied science 10, 3701. MDPI. https://doi.org/10.3390/app10113701

J.L. Outón, I. Merino, I. Villaverde, A. Ibarguren, H. Herrero, P. Daelman, B. Sierra, 2021. A Real Application of an Autonomous Industrial Mobile Manipulator within Industrial Context. Electronics 10, 1276. MDPI. https://doi.org/10.3390/electronics10111276

- **PROFLOW** (Figura 1.7): PROFLOW (Producción Fluida para la Industria Inteligente) es un proyecto perteneciente al programa Elkartek con nº de proyecto KK-2022/00024. En la producción fluida, no-lineal o matricial basada en células de fabricación flexibles y configurables, las estaciones de trabajo no están físicamente conectadas, como ocurre con las cintas transportadoras en las líneas convencionales de producción, sino que se trata de islas interconectadas mediante robots

Figura 1.7: Descripción gráfica del proyecto PROFROW

móviles. La adopción de un paradigma de producción fluida o matricial como la propuesta en PROFLOW favorece que las empresas manufactureras vascas mejoren su capacidad de respuesta y adecuación a un contexto marcado por un elevado y creciente nivel de complejidad en la demanda de productos (i.e., muchas variantes de producto a fabricar en series cada vez más cortas) permitiéndoles mejorar su competitividad en un mercado globalizado. Este proyecto tiene una duración de dos años comprendidos entre 2022-2023. En este proyecto se ha diseñado un algoritmo de estimación de pose 6D fusionando las predicciones de modelos de aprendizaje profundo. Los resultados de la investigación se han publicado en:

Merino, I., Azpiazu, J., Remazeilles, A., Sierra, B, 2023. **Ensemble of 6 dof pose estimation from state-of-the-art deep methods**. Neurocomputing, Volume 541. Elsevier. https://doi.org/10.1016/j.neucom.2023.126270

El autor ha participado también en proyectos de investigación que han contribuido a la carrera investigadora del autor, pero que no están ligados directamente con la tesis. Estos proyectos son los siguientes:

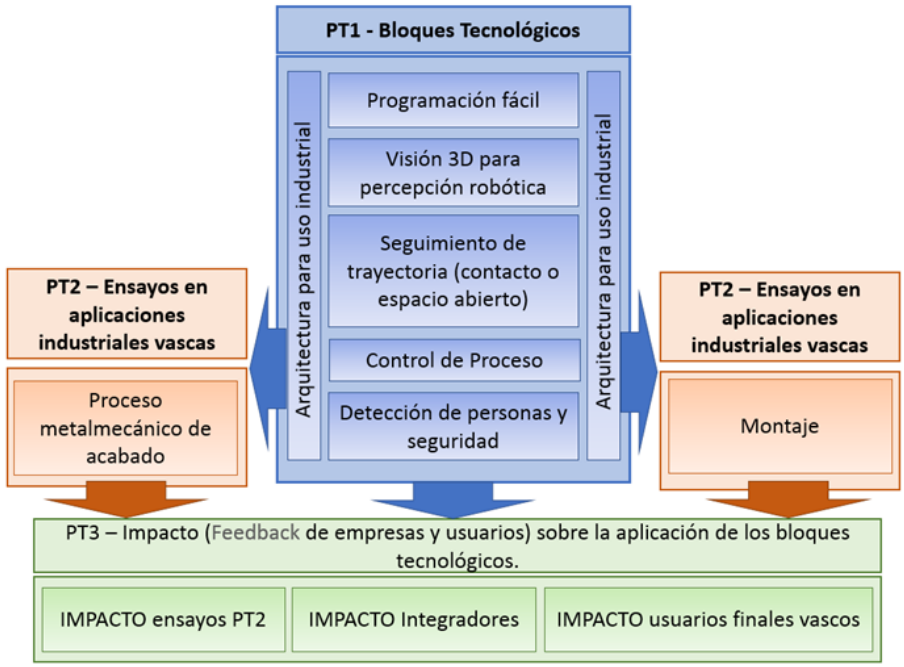


Figura 1.8: Partes del proyecto Malgurob y la conexión entre éstas.

- **Malgurob** (Figura 1.8): Malgurob es un proyeto dentro del programa Elkartek que tiene como objetivo desarrollar bloques tecnológicos que preparen a las empresas vascas para la siguiente ola de innovación robótica. Esta ola rompe con los esquemas actuales de desarrollo de equipamiento para centrarse en software y capacidades tanto de entendimiento como de decisión de los robots. La robótica flexible que es como se llama a este paradigma donde los robots son más flexibles y autónomos, es la tendencia que se está marcando dentro del contexto industrial. Para disminuir el gap existente actualmente en las empresas vascas en cuanto a robótica flexible se refiere, estos bloques tecnológicos servirán para convertir a éstas en líderes en el sector. Este proyecto se ha realizado entre los años 2018 y 2019. Se ha contribuido diseñando algoritmos de visión por computador 3D para detectar objetos industriales. Los algoritmos de visión han permitido detectar piezas industriales y obtener su pose 6D, para después poder cogerlas con un brazo robótico.

- **Bota-Robota** (Figura 1.9): Bota-Rotoba es un proyecto dentro del Programa Euskampus Resiliencia Covid-19 cuyo objetivo es desarrollar un robot para la desin-

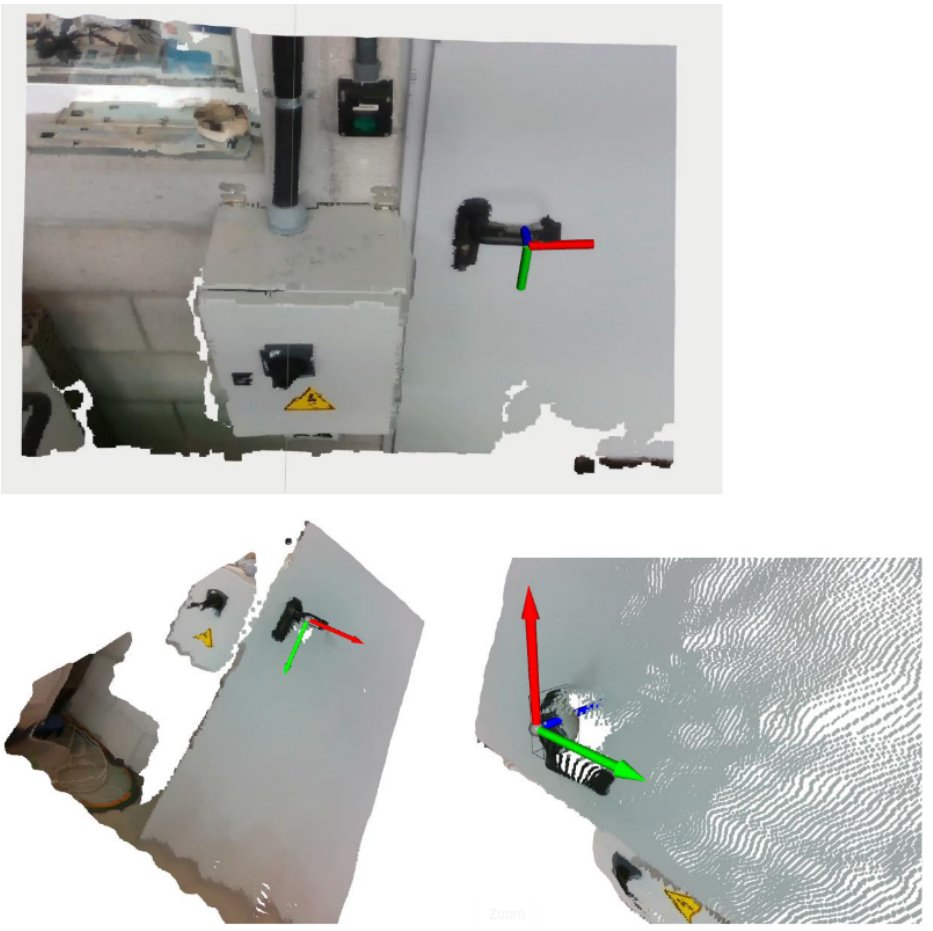


Figura 1.9: Detección de una manilla de una puerta del proyecto Bota-Robota

fección Covid-19 capaz de abrir puertas para la navegación autónoma dentro de edificios. Este proyecto se ha desarrollado durante 2021. La participación en este proyecto ha sido de experimentación con aprendizaje por refuerzo para permitir la apertura de puertas mediante un sistema robótico. Las pruebas definidas utilizan un algoritmo de aprendizaje por demostración para aprender a abrir puertas sin la necesidad de realizar un gran número de demostraciones. Con 10 demostraciones humanas del guiado mediante un joystick, el robot es capaz de replicar las demostraciones realizadas.

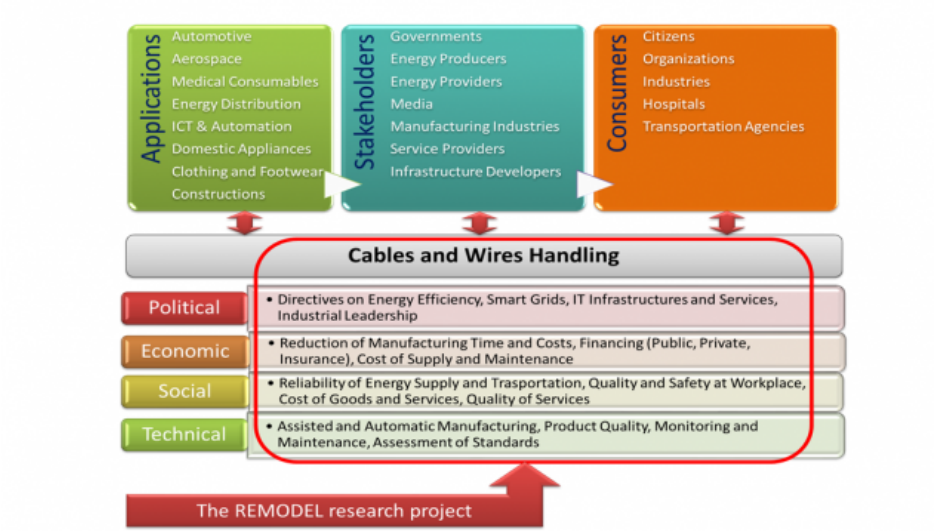


Figura 1.10: Resumen del proyecto Remodel en los distintos ámbitos de acción.

- **Remodel - Robotic tEchnologies for the Manipulation of cOmplex DeformablE Linear objects** (Figura 1.10): El proyecto Europeo Remodel, con indicador del acuerdo de subvención 870133, tiene como objetivo capacitar un nuevo entorno de producción que posibilita la manufacturación de productos compuestos por múltiples cables o alambres por medio de robots de dos brazos e incluso integrarlo completamente en la cadena de diseño de productos. El proyecto Remodel tiene una duración de 4 años empezando en 2019. El autor ha contribuido en este proyecto diseñando un detector de conectores basado en técnicas de aprendizaje profundo. Estos conectores son piezas pequeñas de difícil detección.

- **Odin - Open-Digital-Industrial and Networking pilot lines using modular components for scalable production** (Figura 1.11): El proyecto Odin pertenece al programa Europeo Horizon 2020 con el indicador del acuerdo de subvención 101017141. El objetivo principal de este proyecto reside en desarrollar tecnologías que permitan no sólo demostrar que los nuevos sistemas de producción basados en robots son técnicamente factibles, sino que además son eficientes y sostenibles para su inmediata introducción en planta. Entre esas tecnologías están: robots colaborativos y áreas de trabajo colaborativas entre robot-humano, robots autónomos y planificación de tareas basadas en inteligencia artificial, robots móviles y herramientas reconfigurables, gemelos digitales y comisiones virtuales, e integración de robots orientados al servicio y arquitecturas de comunicación. Este proyecto está comprendido entre los años 2021 y2024 Dentro de este proyecto se

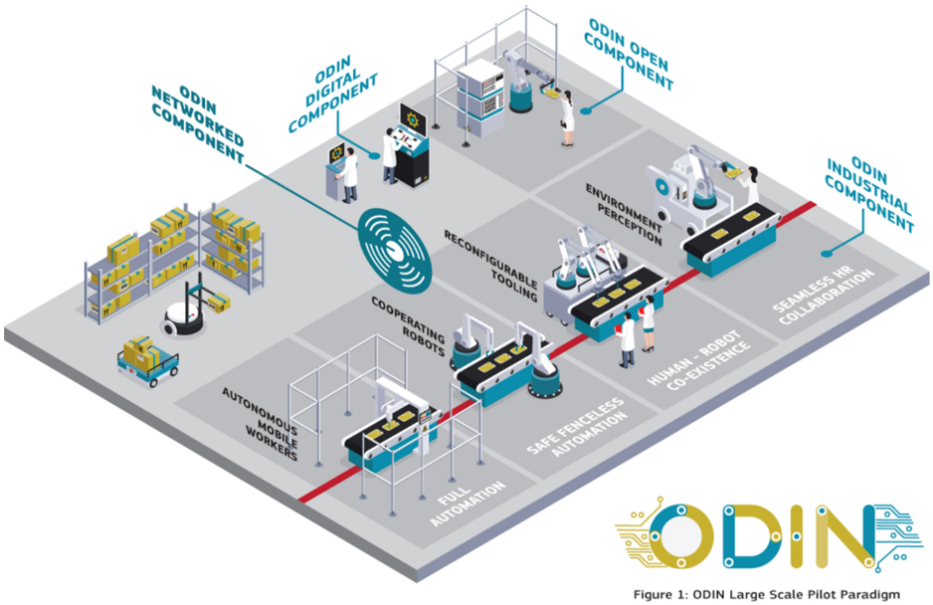


Figura 1.11: Componentes que forman parte del proyecto Odin

han diseñado algoritmos de CAD matching para la localización de piezas grandes.

- **Maelstrom - Smart technology for MArinE Litter SusTainable RemOval and Management** (Figura 1.12): Maelstrom es un proyecto financiado por la comisión europea bajo el programa Horizon 2020 con indicador del acuerdo de subvención 101000832. El proyecto Maelstrom busca generar tecnologías que permitan la extracción de basura marina en los diferentes ecosistemas de las costas Europeas. El proyecto tiene cuatro pilares: definir una propuesta de confianza y científicamente sólida para el entendimiento de la acumulación de basura marina y las condiciones de la vida marina en esas acumulaciones; evaluar la repercusión de la retirada de basura marina al momento de su extracción como en los posteriores dos años; idear y montar una tecnología escalable, replicable, inteligente y automatizada para la identificación, la extracción y clasificación de basura marina desde el fondo marítimo hasta la superficie con el compromiso de las comunidades locales; y proveer una mezcla de soluciones químicas y mecánicas verdes que permitan reciclar todo el conjunto de material orgánico y plástico recogidos durante la fase de extracción y evaluar adecuadamente su calidad para un nuevo ciclo de producción y mercantilización. Maelstrom comenzó en 2021 y durará hasta finales de 2024. En este proyecto se ha diseñado una célula de separación de basura

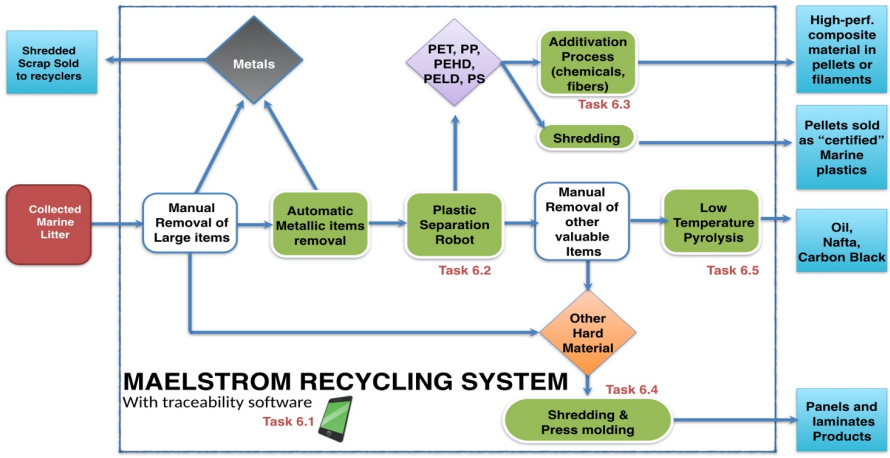


Figura 1.12: Pipeline del sistema de gestión de residuos marinos del proyecto Maelstrom y las salidas que se obtienen de cada proceso.

marina utilizando métodos de aprendizaje profundo en imágenes hiperespectrales. La separación de la basura se ha realizado mediante un robot paralelo Adept de Omron.

- **TrebezIA** (Figura 1.13): TrebezIA es un proyecto perteneciente al programa Elkartek con nº de proyecto KK-2021/00033. Este proyecto parte de dos anteriores Malgurob y Elkarbot los cuales se centraron en desarrollar bloques funcionales de bajo nivel para resolver tareas muy especificas como la segmentación de piezas, optimización de trayectorias para evitar obstáculos, definición de puntos de agarre o mejora de la precisión de robots. Aun así, siguen existiendo necesidades en las fases de pre-producción, producción y post-producción. Este proyecto tiene una duración de dos años comprendidos entre 2022-2023. Este proyecto ha servido como inicio del algoritmo de estimación de pose 6D fusionando las predicciones de modelos de aprendizaje profundo diseñado en el proyecto PROFLOW.

La participación en estos proyectos ha permitido al autor desarrollar su carrera investigadora y ha servido como base para el desarrollo de este proyecto de tesis. En ellos se han identificado necesidades tecnológicas que han servido como motivación para el desarrollo de este proyecto de tesis: la mejora de precisión de detección, la difícil integración de todos los sistemas de visión en entornos industriales y robóticos, la necesidad de datos sintéticos para entrenar algoritmos de aprendizaje profundo, entre otras.

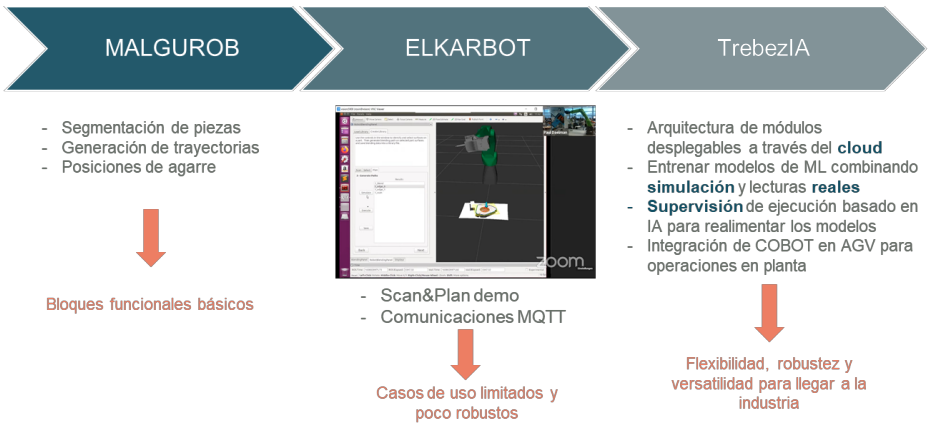


Figura 1.13: Logros obtenidos en los proyectos Malgurob y Elkarbot, los cuales preceden al proyecto TrebezIA, y los objetivos de este proyecto

1.4. Hipótesis de la investigación

Como se ha mencionado en la anterior sección, se quiere estudiar el comportamiento de técnicas clásicas y de aprendizaje profundo en la detección de objetos y la estimación de pose 6D. Además, se quiere estudiar cómo se puede mejorar el rendimiento de los métodos de aprendizaje profundo mediante el uso de técnicas clásicas. Por ello, se plantean las siguientes hipótesis:

- **H1**: La fusión de técnicas clásicas permite mejorar los resultados obtenidos de manera individual.
- **H2**: Transferencia de aprendizaje de algoritmos de aprendizaje profundo 2D a su versión 3D mejora la precisión obtenida.
- **H3**: Entrenar algoritmos con datasets sintéticos permite obtener mejores resultados que entrenarlos con sólo datos reales limitados.
- **H4**: La combinación de técnicas clásicas y de aprendizaje profundo aumenta la precisión y la generalización.

Las contribuciones realizadas dan respuesta a estas hipótesis. En el Capítulo 3 se exponen las contribuciones realizadas.

1.5. Estructura de la memoria

Esta memoria está dividida en dos partes. En la Parte I se expone la investigación realizada. Ésta a su vez está dividida en Capítulos. En total hay 4 Capítulos: Introducción y motivación, Técnicas de visión para la detección de objetos, Contribuciones y Conclusiones.

El Capítulo 1 es el capítulo actual y en él se expone la motivación y las hipótesis de la investigación. En el Capítulo 2 se hace un estado del arte sobre las diferentes técnicas de visión por computador, primero se introduce y se presentan los tipos de tareas que existen en visión por computador (2.1), después se continúa con técnicas clásicas 2D (2.2), técnicas clásicas 3D (2.3), técnicas de aprendizaje profundo (2.4), y por último, sobre datasets y benchmarks (2.5). En el Capítulo 3 se exponen las contribuciones que se han realizado para dar respuesta a las hipótesis presentadas en la Sección 1.4. Finalmente, en el Capítulo 4 se exponen las conclusiones que se han obtenido durante el proyecto de tesis.

Por último, en la Parte II están como anexos los artículos que presentan las contribuciones expuestas en el Capítulo 3.

CAPÍTULO 2

Técnicas de visión por computador para la detección de objetos.

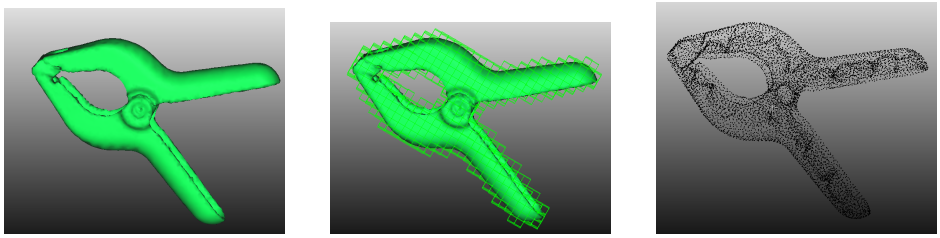
2.1. Introducción

Uno de los problemas clave tratados en este proyecto de tesis es la estimación de pose 6D de objetos observados por un sensor de visión. Esta consiste en estimar la posición y orientación de un objeto en el espacio. Esta pose está compuesta por 6 dimensiones, que se pueden expresar de distintas maneras. Una de ellas y la más común es: x, y, z, θ, ϕ, ψ (los tres primeros para la posición y los 3 últimos para la rotación). Estas dimensiones se pueden representar mediante una matriz de rotación y una matriz de traslación o mediante una matriz de transformación homogénea. Hay que tener en cuenta que esta pose 6D es relativa a un sistema de coordenadas concreto. Como norma general, se suele utilizar el sistema de coordenadas de la cámara, ya que es el sistema de coordenadas que se utiliza para obtener las imágenes. Para entender mejor la estimación de pose 6D, es importante tener en cuenta los siguientes conceptos.

Esa estimación de pose 6D se realiza a partir de la información visual adquirida por la cámara. Estos datos adquiridos por la cámara se pueden representar de diferentes maneras, formatos o tipologías. Por un lado tenemos los datos 2D, que pueden ser imágenes RGB, imágenes térmicas, imágenes de profundidad, etc; y por otro lado tenemos los datos 3D, que pueden ser nubes de puntos, voxels, etc. Entre los datos 2D las

imágenes RGB son las más comunes. Estas son imágenes 2D que contienen información de color. Cada pixel de una imagen RGB contiene información de color en los canales Rojo (R: Red), Verde (G: Green) y Azul (B: Blue). Cualquier imagen 2D puede estar compuesta de uno o varios canales. En el caso de las RGB son 3 canales (R, G y B), pero también existen imágenes 2D en escala de grises que solo tienen un canal, como por ejemplo las imágenes térmicas o las de profundidad. Las imágenes de profundidad son imágenes 2D que contienen información de la profundidad. Cada pixel de una imagen de profundidad contiene información de la distancia a la que se encuentra el objeto que se está observando. Por eso, aunque son imágenes 2D, contienen información 3D e incluso se pueden transformar en nubes de puntos.

El uso de datos 3D es menos común que el uso de datos 2D, debido a que los datos 3D son más costosos de obtener y procesar. Aun así, existen sensores que pueden obtener datos 3D de forma relativamente sencilla, como por ejemplo los sensores de profundidad estilo Kinect. Las nubes de puntos son estructuras de datos 3D que contienen un conjunto de puntos en el espacio, cada uno con su coordenada x, y y z. Además, cada punto también puede disponer de información de color y de su normal. El vóxel es la contraparte 3D del píxel. Es la unidad cúbica que compone una matriz tridimensional. Al igual que los píxeles, no tienen información de la posición en el espacio, sino que viene dada por su posición en la estructura de datos. Muchas veces los vóxeles se utilizan para representar una nube de puntos, ya que es una forma de reducir la cantidad de datos. A mayor tamaño de vóxel mayor cantidad de puntos abarcará y será más fácil procesarla, pero menor será la resolución y, por consiguiente, la precisión que se obtendrá. La Figura 2.1 muestra un ejemplo de un objeto del dataset YCB-Video en formato malla, voxel y nube de puntos.



(a) Malla (Mesh). **(b)** Voxel. **(c)** Nube de puntos.

Figura 2.1: Pinza del dataset YCB-Video.

Además, también es importante definir el resto de tareas típicas de la visión por computador. La clasificación es una de las tareas más básicas, pero a su vez más utilizadas, de la visión por computador. Consiste en dar una clase a una imagen. La segmentación semántica es una extensión de la clasificación. Consiste en etiquetar la imagen a nivel de

píxel o punto, de esta forma se divide la imagen en diferentes regiones, cada una de ellas perteneciente a una clase diferente. La segmentación de instancia consiste en localizar en la imagen todas las instancias diferentes. La segmentación panóptica es una combinación de las dos anteriores, la cual consiste en dividir la imagen en diferentes regiones, cada una de ellas perteneciente a una instancia diferente, y además a cada instancia se le asigna una clase. La detección 2D consiste en encontrar objetos en imágenes. Estas detecciones se dan en forma de *Bounding Box 2D* (caja delimitadora). La Figura 2.2 [20] muestra un ejemplo de detección 2D, segmentación semántica, segmentación de instancia y segmentación panóptica para ver sus diferencias. Otra tarea importante es el tracking o seguimiento de objetos. Consiste en seguir un objeto durante una secuencia (en un vídeo por ejemplo).

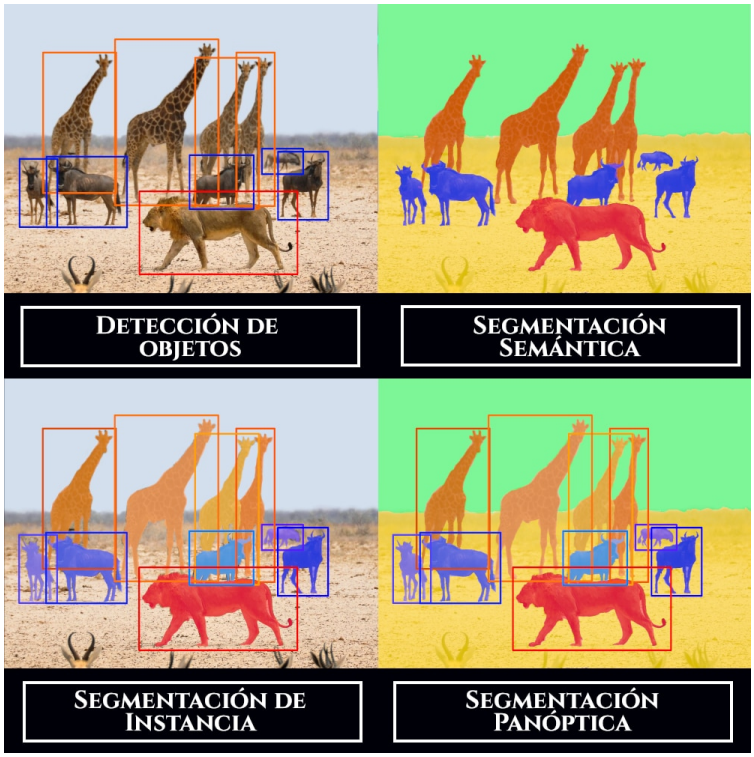


Figura 2.2: Comparativa de una detección 2D, segmentación semántica, segmentación de instancia y segmentación panóptica a la misma imagen. (Origen de la imagen: lamaquinaoraculo.com)

Muchas de las tareas de visión por computador que se han mencionado anteriormente se pueden realizar con datos 2D o 3D. Además, la salida de una tarea puede ser

de diferentes tipologías de datos. Cuando hacemos una detección 2D obtenemos un Bounding Box 2D y cuando hacemos una detección 3D obtenemos un Bounding Box 3D. Aun así, existen métodos que permiten hacer una correspondencia entre datos 2D y 3D, como por ejemplo PnP [21] (*Perspective-n-Point*). Este método permite obtener la pose 6D de un objeto a partir de una detección 2D y un modelo 3D del objeto.

La Figura 2.3 muestra el conjunto de las tareas que se pueden realizar con visión por computador.

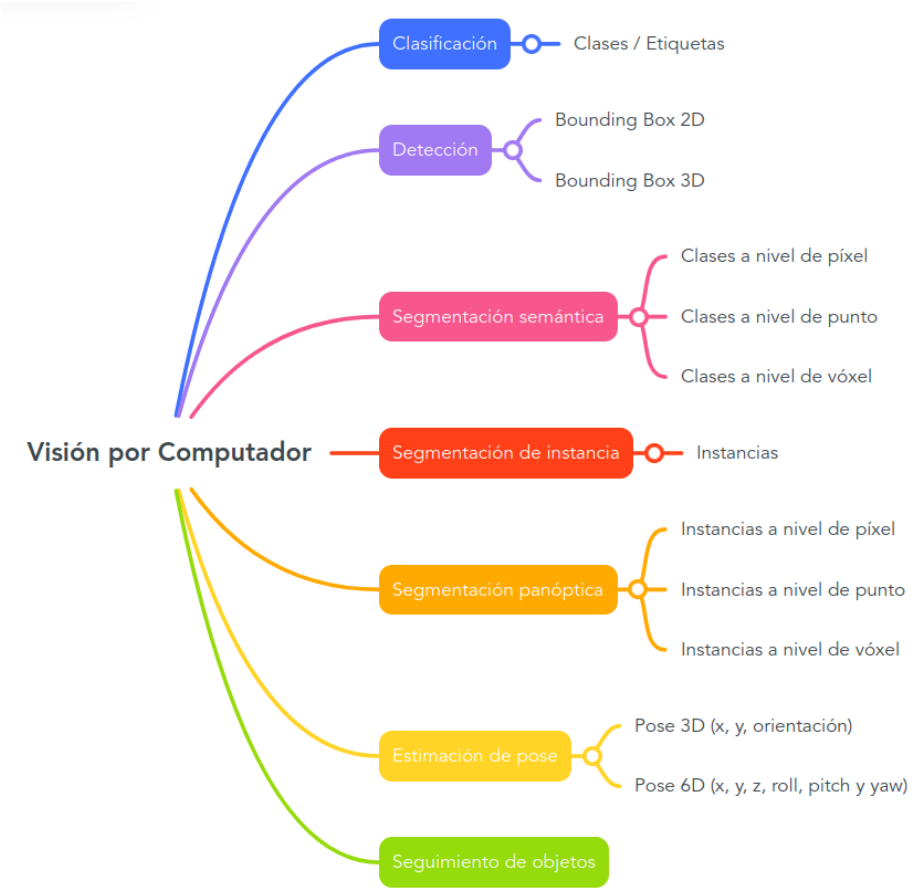


Figura 2.3: Tareas más comunes de visión por computador con el output que generan.

En este capítulo, se van a presentar técnicas clásicas 2D (Sección 2.2), que se pueden

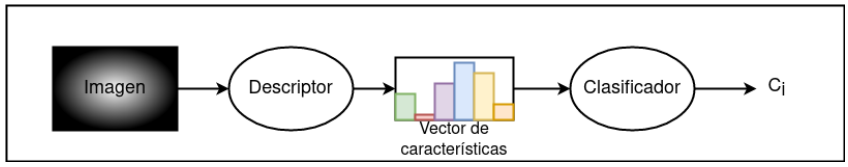
dividir entre descriptores globales (Subsección 2.2.1), descriptores locales (Subsección 2.2.2), y la busqueda de correspondencias y clasificación (Subsección 2.2.3). Posteriormente, se van a presentar técnicas 3D (Sección 2.3) para la estimación de pose 6D: métodos locales basados en parches (Subsección 2.3.1), métodos locales basados en nubes de puntos (Subsección 2.3.2) y métodos basados en plantillas (Subsección 2.3.3). Además, se van a introducir las bases del aprendizaje profundo (Sección 2.4) y las arquitecturas y técnicas más utilizadas como son las redes neuronales convolucionales (Subsección 2.4.1), las redes neuronales recurrentes (Subsección 2.4.2), los autoencoders (Subsección 2.4.3), los modelos generativos adversariales (Subsección 2.4.4), Transformers (Subsección 2.4.5), modelos de difusión (Subsección 2.4.6) y aprendizaje transferido (Subsección 2.4.7). Por último, se va a presentar el estado del arte de los datasets más utilizados para la estimación de pose 6D (Sección 2.5), como el benchmark BOP (Subsección 2.5.1) y herramientas para la generación de datasets sintéticos (Subsección 2.5.2).

2.2. Técnicas clásicas 2D

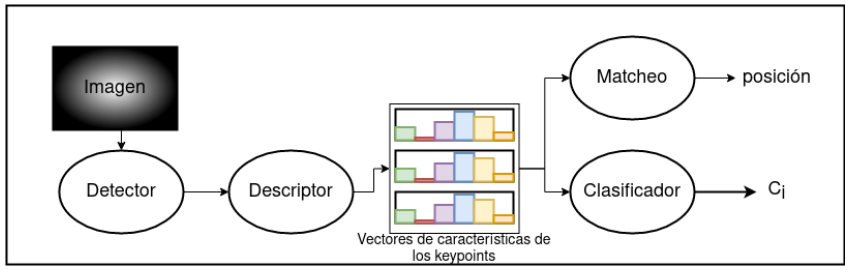
Dentro de la visión por ordenador más tradicional o visión clásica, el pipeline de detección de objetos ha seguido un patrón determinado, el cual ha consistido en buscar características concretas o importantes en las imágenes, describir o codificar esas características y buscar correspondencias entre esas características codificadas. La Figura 2.4 muestra los *pipelines* o flujos que siguen la gran mayoría de algoritmos de visión clásica. El primer flujo, la descripción global, está recogido en 2.4a. En este contexto de descripción global, se busca extraer características de la imagen en su totalidad formando un vector de características de tamaño concreto que depende del tipo de descriptor. Este vector de características se usa para clasificar la imagen. El segundo flujo, la descripción local, se resume en 2.4b. En este contexto primero se detectan características concretas las cuales se llaman *keypoints* o puntos de interés que pueden ser esquinas, bordes, *blobs* o parches de la imagen concretos. Una vez detectados estos puntos de interés, se describen cada uno de ellos mediante un descriptor local. De esta forma obtenemos un vector de características por cada punto de interés. Estos vectores de características se pueden utilizar para clasificar la imagen en su totalidad, similar a lo que se hace con los descriptores globales, o *matchear* (emparejar) esos vectores con los de modelos conocidos para lograr obtener la localización de esos objetos conocidos (estimación de pose). La Figura 2.5 muestra la relación entre los diferentes métodos clásicos locales y globales del estado del arte. Las flechas indican que un método se basa en otro, que es una mejora respecto al anterior o que es una combinación de varios métodos.

2.2.1. Descriptores globales

Los descriptores globales nos permiten describir una imagen en su totalidad y formar un vector de características único para toda la imagen.



(a) Descriptores globales



(b) Descriptores locales

Figura 2.4: Flujos de visión por ordenador 2D clásico

Entre ellos podemos encontrarnos HOG (*Histograms of Oriented Gradients*) [22] que es un histograma de gradientes orientados normalizado localmente, HSOG (*Histograms of the Second-Order Gradients*) [23] que calcula los gradientes de segundo orden para capturar la curvatura relacionada con propiedades geométricas, GIST [24] que utiliza una representación de la estructura espacial dominante de una escena que integra un conjunto de dimensiones perceptuales que llaman *Spatial Envelope*, y MPEG-7 [25, 26] que está basado en el estándar con el mismo nombre e incluye diferentes descriptores basados en histogramas como CLD (*Color Layout Descriptor*), EHD (*Edge Histogram Descriptor*) y SCD (*Scalable Color Descriptor*).

Existe un gran número de descriptores locales que se pueden utilizar como descriptores globales calculando su histograma. A este grupo de algoritmos se les llama histogramas de patrones equivalentes [27]. Entre ellos podemos encontrar:

- LBP (*Local Binary Pattern*) [28] es el más conocido de ellos el cual es robusto frente a variaciones de luz.
- LBP da pie a una gran variedad de diferentes descriptores locales que describen texturas y tienen un nivel de caracterización bajo para calcular su histograma.
- STU (*Simplified Texture Unit*) [29] consigue reducir el rango de posibles valores sin una pérdida significativa en el poder de caracterización.

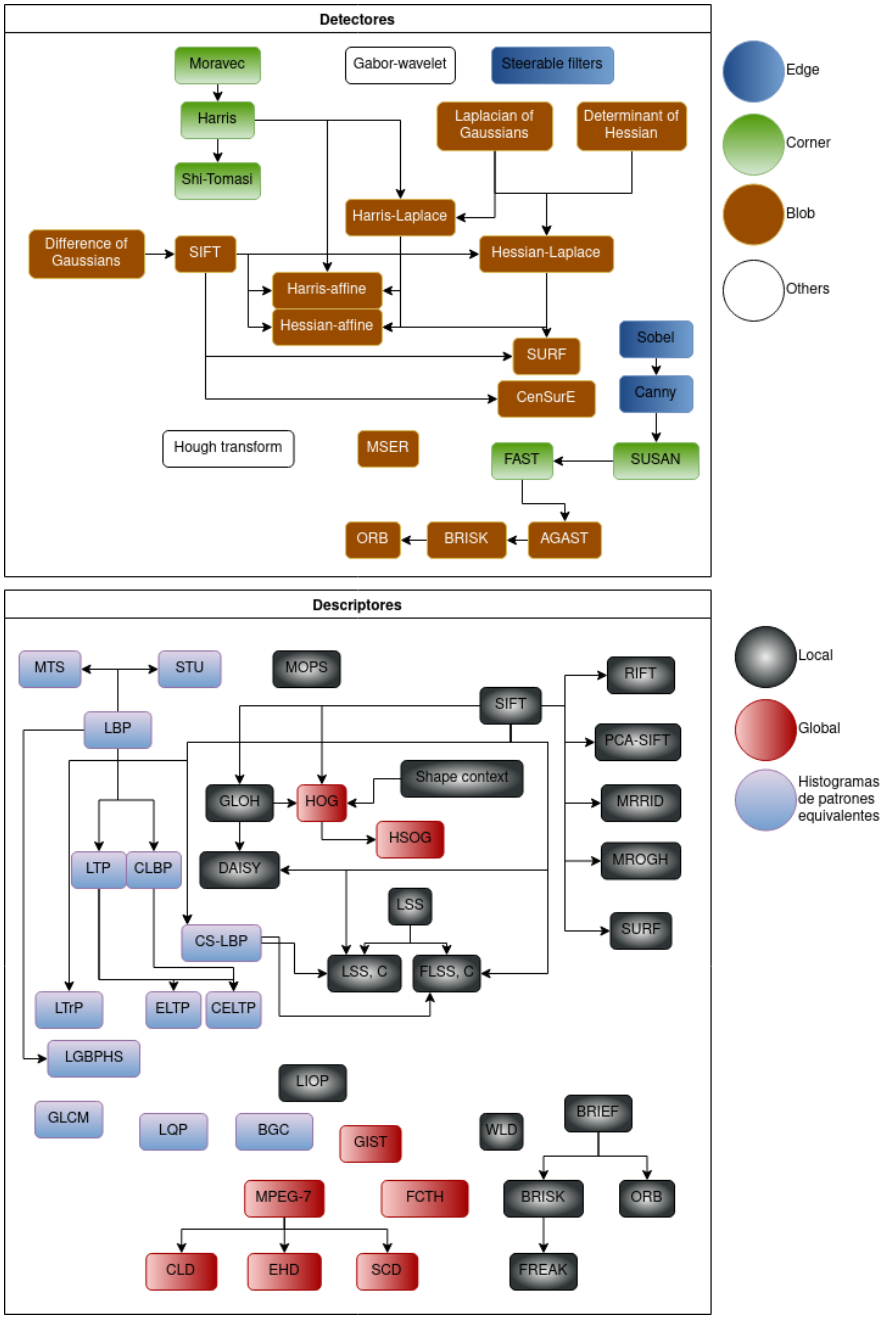


Figura 2.5: Relación entre los diferentes métodos clásicos locales y globales del estado del arte

- MTS (*Modified Texture Spectrum*) [30] es una versión simplificada del LBP utilizando un subset de los píxeles.

- CLBP (*Completed Local Binary Pattern*) [31] combina tres diferentes versiones del LBP modificado para mejorar la invarianza a la rotación.

- LTP (*Local Ternary Pattern*) [32] es una generalización del LBP más discriminativa y menos sensible al ruido en regiones uniformes pero deja de ser estrictamente invariante a transformaciones a nivel de grises.

- ELTP (*Enhanced Local Ternary Pattern*) [33] ataca el problema que tiene LTP utilizando una estrategia adaptativa para seleccionar el umbral o *threshold*.

- CELTP (*Completed Enhanced Local Ternary Pattern*) [33] utiliza la misma estrategia que el CLBP usando ELTP.

- LTrP (*Local Tetra Pattern*) [34] extrae mayor información utilizando derivadas de mayor orden.

- BGC (*Binary Gradient Contours*) [35] presenta 3 formas diferentes de calcular el contorno del gradiente binario de forma parecida a LBP.

- Añadiendo filtros Gabor a LBP obtenemos el descriptor LGBPHS (*Local Gabor Binary Pattern Histogram Sequence*) [36].

- LQP (*Local Quantized Pattern*) [37] es una generalización del LBP que utiliza un vector de cuantificación y una tabla de búsqueda para utilizar más pixeles para la caracterización del patrón local y un nivel de caracterización mayor sin sacrificar simplicidad y eficiencia computacional,

- GLCM (*Gray Level Coocurrences Matrices*) [38] está basado en describir las dependencias espaciales en tonos grises. Es un descriptor de texturas muy simple pero muy utilizado en muchas aplicaciones.

- WLD (*Weber Local Descriptor*) [39] es un descriptor local muy simple pero la vez muy potente y robusto basado en la ley de Weber [40].

Como está definido en el pipeline de los descriptores globales (Figura 2.4a), una vez obtenido el vector de características global (como descriptor global o histograma de patrones equivalentes), se puede utilizar para clasificar la imagen en su totalidad. Para ello, se utilizan algoritmos de aprendizaje automático o de matcheo (ver Sección 2.2.3.1).

2.2.2. Descriptores locales

Para hacer una correspondencia (*matching*) de características, es necesario primero encontrar puntos o regiones en la imagen que tengan cierta significancia, es decir, que sean características del objeto. Para ello, se utilizan detectores de puntos de interés (*keypoint detectors*); estos pueden ser detectores de bordes, de esquinas, de regiones (*blobs*) o de otro tipo.

2.2.2.1. Detectores

Un detector de puntos de interés localiza puntos que tengan ciertas características para que sean reconocibles, como por ejemplo, bordes o esquinas. La Figura 2.6 muestra un ejemplo de detección de puntos de interés en una imagen de una pieza industrial.

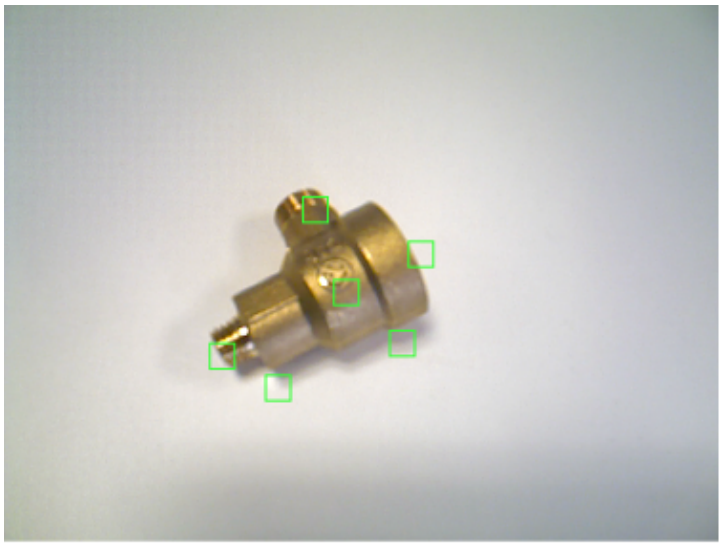


Figura 2.6: Detección de puntos de interés en una pieza industrial.

Entre los detectores de bordes más conocidos están los Steerable filters [41], Sobel [42] y Canny [43, 44, 45]. Los filtros orientables o *Steerable filters* permiten sintetizar filtros de orientaciones arbitrarias a partir de filtros básicos. El detector de bordes Sobel se basa en un operador isotrópico para obtener el gradiente en un vecindario de 3x3. El detector Canny, en cambio, detecta bordes buscando máximos en el gradiente de imágenes con un suavizado gaussiano.

Los detectores de esquinas son algo más complejos, ya que buscan las intersecciones de dos bordes. Los más conocidos son Moravec [46], Harris y Stephens [47], Shi-Tomasi [48], SUSAN [49] y FAST [50]. Moravec [46] busca similitudes en los pixeles cercanos mediante la suma de diferencias cuadradas. Harris y Stephens [47] analizan el detector de esquinas de Moravec para mejorarlo aplicando la autocorrelación. Shi-Tomasi [48] añade otra mejora más a los anteriores que es sustituir la función de puntuación $R = \lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)^2$ por $R = min(\lambda_1, \lambda_2)$, donde λ_1 y λ_2 son los valores propios de la matriz del ruido del movimiento. SUSAN [49] usa máscaras circulares para dar una respuesta isotrópica. FAST [50] incorpora una prueba de alta velocidad para excluir a un gran número de no esquinas. FAST da inicio a una familia de algoritmos de detección de puntos de interés llamada AST (*Accelerated Segmented Test*), la cual es una modificación del detector SUSAN que se presenta en FAST. Uno de los métodos perteneciente a la familia AST es AGAST (*Adaptative and Generic Accelerated Segmented Test*) [51]. Éste cambia la forma en la que se construye el árbol de decisión para el test AST para obtener mejores resultados. Otro método de esta familia es BRISK (*Binary Robust Invariant Scalable Keypoints*) [52], el cual añade invarianza a la escala y espacio. Siguiendo con esta familia de algoritmos, ORB [53] añade invarianza rotacional, por lo que incluyendo los aportes de BRISK, ORB es capaz de detectar puntos de interés en múltiples escalas y rotaciones.

Los blobs, en cambio, son los más elaborados y los más recientes, aunque muchos de ellos son y tienen su base en algoritmos del siglo pasado. Una de esas bases es Difference of Gaussians (DoF) [54, 55] el cual es un filtro de paso de banda espacial que atenúa frecuencias en la imagen de escala de grises que están lejos de la banda céntrica. DoF es la base para uno de los algoritmos de detección, descripción y matcheo de características más usado del estado del arte: SIFT [56]. Este algoritmo, que está compuesto por un detector y un descriptor, es invariante a la translación, invariante al escalado, invariante a la rotación, parcialmente invariante a los cambios de iluminación y robusto ante pequeñas distorsiones geométricas locales. Otro algoritmo base para las siguientes generaciones de algoritmos es Laplacian of Gaussians [57]. Este algoritmo aplica un kernel que aproxima el aplicar un filtro gaussiano de suavizado con el Laplaciano $L(x, y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$, es decir aproximar la Ecuación 2.1.

$$LoG = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2 + y^2}{2\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.1}$$

Otra de las bases de la detección de blobs es el determinante del Hessiano [57]. Este algoritmo lo que busca son extremos en espacio-escala del determinante del Hessiano (DoH, Determinant of Hessian), el cual obtiene algo mejores selecciones de escala que el operador Laplaciano. El algoritmo SURF [58] está basado en este operador, pero utiliza imágenes integrales para calcularlo de forma mucho más eficiente. SURF está también inspirado en SIFT, siendo SURF varias veces más rápido que SIFT.

CenSurE [59] es otro detector escala-espacio que permite detección en tiempo real pensado principalmente para registro de imágenes (buscar correspondencias entre imágenes para fusionarlas), especialmente para odometría; teniendo en cuenta dos criterios principalmente, estabilidad de los descriptores con los cambios de perspectiva y precisión en la localización consistente para cambios de perspectiva.

Krystian Mikolajczyk y Cordelia Schmid definieron en 2002 otros 4 tipos de detectores [60, 61]: Harris-Laplace, Harris-affine, Hessian-Laplace y Hessian-affine. Harris-Laplace combina el detector de esquinas Harris con una representación Gaussiana de la escala-espacio. Harris-affine es un refinamiento del detector Harris-Laplce, el cual es invariante a la escala y funciona bien para regiones isotrópicas que se ven desde el mismo ángulo de visión, el cual añade invarianza a trasformaciones afines. Hessian-Laplace, en cambio, utiliza la matriz hessiana en vez del detector Harris para detectar los puntos de interés y, partiendo de estos puntos iniciales, Hessian-affine refina estas detecciones mediante un método iterativo (el mismo que en Harris-affine) para volverlo invariante a transformaciones afines.

En 2004, Matas et al. introducen un nuevo descriptor llamado *Maximally Stable Extremal Regions* (MSER) [62]. Este detector de regiones eficiente y rápido (cercano a la frecuencia de imágenes) permite detectar un conjunto de regiones extremas afínmente invariantes. Existen otros detectores que buscan otro tipo de características en la imagen, como por ejemplo los Gabor-wavelets [63] y las transformaciones de Hough [64].

2.2.2.2. Descriptores

Una vez detectados los puntos o regiones de interés, en la mayoría de los casos hay que describirlos para poder buscar correspondencias entre ellos. Esta descripción es un *encoding* de las características de esos puntos o regiones de interés, es decir, trasformar la información en la imagen a un vector de información el cual poder comparar o analizar. Existen muchos tipos de descriptores y la mayoría de ellos se utilizan con un detector concreto. Aun así, muchos de ellos son intercambiables y las combinaciones posibles son muchas.

Entre los más utilizados se encuentran SIFT [56], SURF [58], BRISK [52] y ORB [53], los cuales también tienen su propio detector y se han presentado en la Subsección 2.2.2.1. Brisk ensambla un descriptor de cadena de bits a partir de comparaciones de intensidad obtenidas del muestreo dedicado de cada vecindario de los puntos de interés. El descriptor de ORB, en cambio, está basado en BRIEF [65] añadiendo consciencia de la rotación de forma eficiente.

SIFT ha sido y sigue siendo uno de los métodos más utilizados para detectar, describir y emparejar puntos, y muchos autores han intentado mejorarlo o superarlo. SURF, por ejemplo, buscaba ser más rápido que SIFT obteniendo características parecidas a éste. Otro de ellos es PCA-SIFT [66] el cual sustituye los histogramas utilizados en SIFT por

aplicar un análisis de componentes principales a los parches de gradientes normalizados. RIFT [67] también está basado en SIFT, pero en vez de buscar las orientaciones dominantes del parche normalizado, se divide el parche circular normalizado en anillos concéntricos y se calcula un histograma de las orientaciones de gradiente para cada anillo. MRRID y MROGH [68] son dos descriptores que agrupan las características locales en función de sus órdenes de intensidad en varias regiones de apoyo. La diferencia entre estos dos descriptores es que el primero está basado en gradientes mientras que el segundo lo está en intensidades. GLOH [69] es una extensión de SIFT que se calcula el descriptor para una rejilla de localización log-polar con tres cuadrículas en la dirección radial y 8 en la dirección angular. Inspirado en SIFT y en este último, DAISY [70] se calcula más rápido que los algoritmos en los que se basa pero con ciertas ventajas sobre SURF. LIOP [71], en cambio, explota eficazmente la información local utilizando el orden de intensidad de todos los puntos vecinos muestreados. Este muestreo es invariante a la rotación para evitar estimaciones de rotaciones locales consistentes, y por ende, tener un mayor poder discriminativo.

Otros descritores como MOPS [72] no siguen un patrón parecido a SIFT. MOPS es un descriptor que consiste en los valores de intensidad normalizada de sesgo/ganancia de unos parches de 8×8 seguido de un algoritmo adaptativo de supresión no máxima (*non-maximal suppression*). Shape context [73] captura en cada punto de interés la distribución del resto de puntos respecto a él, obteniendo una caracterización discriminativa global. De esta forma, puntos correspondientes en dos formas parecidas tendrán parecidos descriptores. LSS [74] es un método que busca describir auto-similitudes internas locales para poder identificar objetos que no compartan propiedades de imagen comunes (color, textura o esquinas) pero que si tengan una forma geométrica similar. LSS,C y FLSS,C [75] calculan el descriptor LSS dentro de una rejilla de localización cartesiana y el último añade también un método de cálculo del LSS más rápido.

FREAK [76] es un descriptor basado en BRISK en el sampleo de patrones y en ORB para definir el conjunto optimo de pares de muestreo los cuales se comparan sus intensidades para definir el descriptor. Además añade un mecanismo para la orientación similar a lo que se realiza en BRISK.

2.2.3. Búsqueda de correspondencias y clasificación

Siguiendo el esquema del pipeline de los descriptores locales (Figura 2.4b), una vez obtenido el vector de características de cada punto de interés, se pueden utilizar para clasificar la imagen en su totalidad o para buscar correspondencias entre ellas. En el caso de los descriptores globales (Figura 2.4a), los vectores de características se utilizan para clasificar la imagen en su totalidad. Cuando hacemos una descripción local tenemos que buscar la correspondencia de esas características en otra imagen o en los modelos conocidos. Para ello, el método de búsqueda de correspondencias más habitual es buscar para cada vector de características el vector de características más parecido. Cuando

se realiza esta búsqueda de correspondencias, se pueden utilizar diferentes métodos para calcular la distancia entre los vectores de características, como por ejemplo la distancia euclídea (Ecuación 2.2, para dos vectores P y Q de dimensionalidad n) o la distancia de Hamming (Ecuación 2.3). Una vez tenemos la distancia entre los vectores de características de dos imágenes, se realiza una búsqueda de correspondencias por fuerza bruta, es decir, probando todas las posibilidades.

$$d_E(P,Q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2} \tag{2.2}$$

$$d_H(P,Q) = \sqrt{\sum_{i=1}^{n} h(p_i, q_i)} \tag{2.3}$$

$$h(p,q) = \begin{cases} 0 & \text{si } p = q \\ 1 & \text{si } p \neq q \end{cases} \tag{2.4}$$

Para evitar realizar la búsqueda de correspondencias por fuerza bruta, se pueden utilizar métodos que aproximen esa solución o ayuden a realizar la búsqueda de manera más eficiente. Uno de esos métodos es la bolsa de características o *Bag-of-Features* [77] también llamado *Bag-of-Visual-Words*. Es un método para representar los vectores de características obtenidos tras detectar y describir los puntos o regiones de interés. Estos vectores de características se agrupan (*Vector Quantization* o *clustering*) y se asocia cada grupo a un centroide. Estos centroides representan las características principales presentes en todo el dataset de entrenamiento. A esto se le conoce como *codewords* o *visual words* y al conjunto de ellas *codebook* o *visual vocabulary* La imagen de entrada o *query* de la cual se han obtenido sus características, se calcula un histograma de frecuencia del centroide más cercano (normalmente utilizando la distancia euclídea, pero puede ser otra distancia) a cada característica.

Otra forma de representar características es mediante el *Geometric hashing* [78]. Este método consiste en codificar los vectores de características generando por cada par de características una base y representando al resto de características en esa base. Las trasformaciones cuantificadas se guardan en una tabla hash como llaves y los índices de las características que forman las bases como valores.

Existen muchos otros métodos para representar los vectores de características de diferente manera como PCA [79, 80], LLE [81] o ICA [82], que consisten en reducir la dimensionalidad de los vectores de características para que el cálculo de las distancias sea más rápido.

Como en muchos otros ámbitos el uso de algoritmos o técnicas de aprendizaje automático permite identificar patrones para poder clasificar o buscar correspondencias de manera más eficiente.

2.2.3.1. Técnicas de aprendizaje automático

En función del tipo de dataset que tenemos y el objetivo que se quiere conseguir se utilizan un tipo de técnica u otras. Llamamos aprendizaje supervisado cuando tenemos datos etiquetados, es decir, tenemos ejemplos del comportamiento que se quiere replicar. Cuando esos datos están sin etiquetar, es decir no tenemos información de la variable dependiente, lo llamamos no supervisado. Existen también casos en los que el dataset está parcialmente etiquetado, por ejemplo tenemos una pequeña cantidad de datos etiquetados y muchos sin etiquetar; estos casos se les llama semi-supervisados. Existen muchos más tipos de métodos de aprendizaje automático como auto-supervisado, *one-class* o detección de anomalías, pero en esta investigación vamos a centrarnos en los supervisados, ya que son los que más repercusión tienen en la estimación de pose.

Entre las técnicas de aprendizaje automático supervisado más utilizadas en visión por computador se encuentran los basados en el vecino más cercano, los vectores de soporte y los árboles de decisión.

Las técnicas basadas en cercanía o vecino más cercado (k-NN o K- Nearest Neighbor) se pueden usar como clasificadores y son las más utilizadas para buscar correspondencias entre características. De hecho el método anteriormente explicado para la busqueda de correspondencia de características es un método basado en la busqueda del vecino más cercano. En el caso de clasificación de imágenes, si estamos utilizando descriptores globales se buscan los k vecinos más cercanos (siendo $k \geq 1$) dentro del dataset de entrenamiento y se clasifica la imagen por mayoría de votos. Esta mayoría puede ser ponderada en función de la cercanía u otro tipo de votación. En el caso de localización de objetos o busqueda de correspondencia de características, consiste en buscar por cada punto de interés al vecino más cercano entre las características del modelo para luego buscar la homografía y obtener la translación y rotación del objeto dentro de la imagen. Esa búsqueda de correspondencias se puede hacer a fuerza bruta (probando todas las posibilidades) o utilizar métodos de aproximación del vecino más cercano que agilicen la búsqueda de éste. La Figura 2.7 muestra un ejemplo del k-NN para $k = 3$ y $k = 5$ donde hay que clasificar el punto blanco en verde o en rojo. Para $k = 3$ el punto nuevo se clasificaría como verde, mientras que para $k = 5$ el punto se clasificaría como rojo. Por eso, la elección de la k es un parámetro a tener en cuenta.

Los vectores de soporte también conocidos como Support Vector Machines (SVM) [83] son unos modelos de aprendizaje automático no probabilistas, binarios y lineales que buscan maximizar el margen entre dos clases definiendo uno o varios hiperplanos. Es uno de los métodos más robustos de predicción basado en marcos de aprendizaje

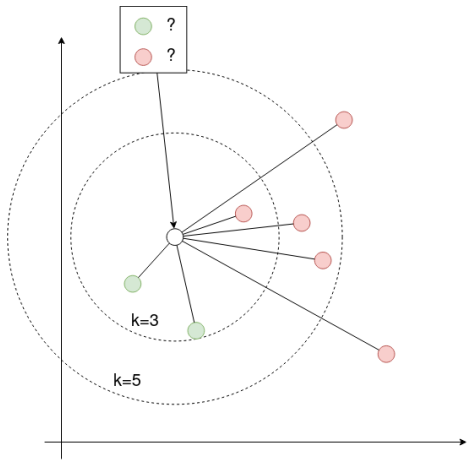


Figura 2.7: Ejemplo de 3-NN y 5-NN para clasificar entre la clase “verde” y clase “roja”

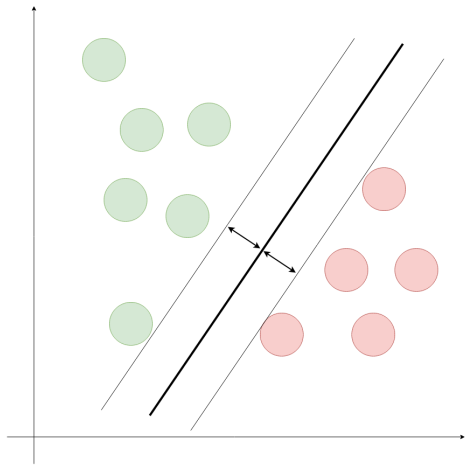


Figura 2.8: Hiperplano generado por una SVM para diferenciar las clases “roja” y “verde” maximizando el margen entre ellas.

estadístico que se utilizan principalmente para clasificación y regresión. Los hiperplanos se construyen en un espacio de dimensionalidad muy alta o incluso infinita para separar los datos de entrenamiento en dos clases.La Figura 2.8 muestra un ejemplo de un hiperplano para el caso de un espacio 2D que separa las clases “roja” y “verde” maximizando el margen que hay entre ellas.

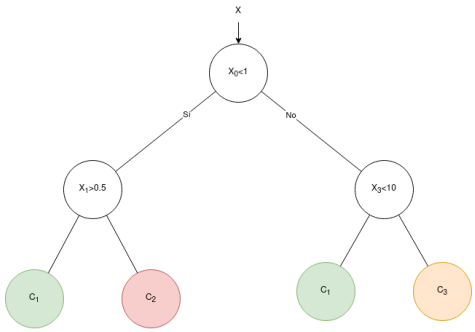


Figura 2.9: Un árbol de decisión para clasificación de tres clases.

Como el clasificador SVM es binario, se utilizan estrategias como uno contra todos (*one-vs-all*) [84] o uno contra uno (*one-vs-one*) [85] para transformar un problema multiclase en múltiples problemas binarios. SVM es un clasificador muy utilizado para clasificar descriptores HOG (anteriormente mencionados), como por ejemplo para el reconocimiento de caras o detección de objetos.

Los árboles de decisión buscan fabricar un sistema de predicción basado en una serie de condiciones de forma recursiva para la resolución de un problema. Este tipo de estructuras se parecen a las que se generan con los sistemas basados en agentes o en reglas. La Figura 2.9 muestra un árbol de decisión que tiene en cuenta 3 de las N variables (X_0, X_1 y X_3) y 3 clases (C_1,C_2 y C_3). Se puede apreciar en el ejemplo que no es necesario que el árbol utilice todas las variables disponibles, porque no son necesarias o porque se ha realizado una poda.

También existen métodos que utilizan múltiples algoritmos de aprendizaje en conjunto para refinar el resultado que da cada algoritmo por separado. Estos métodos se les conoce como *ensemble* y algunas de esas técnicas son: bagging [86], boosting [87], stacking [88] y voting [89].

2.3. Técnicas 3D

Dentro de la visión por computador, existen muchos sensores que capturan nubes de puntos o imágenes de profundidad en vez de imágenes 2D de color. Estos sensores pueden ser cámaras por triangulación láser, de luz estructurada, estéreo o basadas en tiempo de vuelo. La adquisición de esos sensores produce una nube de puntos o un mapa de profundidad. Con los parámetros intrínsecos de los sensores se pueden obtener las nubes de puntos con los mapas de profundidad.

Trabajar con nubes de puntos en vez de con imágenes 2D tiene mayor complejidad.

Esto se debe a diversas razones. La primera es que la adquisición de la nube de puntos genera datos dispersos, es decir, no es capaz de adquirir información de toda la escena 3D, la resolución del sensor limita los datos obtenidos. Segundo, las técnicas para trabajar con imágenes son más eficientes que las que trabajan con nubes de puntos. Añadir una dimensión a los datos incrementa su complejidad para tratarlos. Debido a la complejidad de los datos, se han abordado la estimación de pose 6D desde diferentes perspectivas.

2.3.1. Métodos locales basados en parches

Estos métodos utilizan sistemas de votos basados en árboles (como random forest) sobre parches locales para la detección, localización y estimación de pose. Entre estos métodos nos podemos encontrar el método diseñado por Brachmann et al. que estima directamente las coordenadas de los objetos [90], Tejani et al. que utilizan un parche invariante a la escala para predecir la localización y la pose [91] o Bonde et al. que utilizan una ventana deslizante volumétrica [92].

2.3.2. Métodos basados en matcheo de nubes de puntos

Estos métodos son los más utilizados para la detección, localización y registro en nubes de puntos. Tienen bastantes similitudes con los descriptores 2D, pero en vez de describir el color describen la geometría de los objetos.

- Spin Images [93]: es un descriptor de la forma a nivel de dato para matchear superficies junto con un esquema de compresión para reconocer simultaneamente varios objetos.
- Point Feature Histogram (PFH) [94]: es un descriptor que captura información sobre la geometría al rededor de un punto analizando las direcciones de las normales de su vecindario.
- Fast Point Feature Histogram (FPFH) [95]: una versión más rápida de PFH manteniendo prácticamente todo su poder discriminativo.
- Radius-based Surface Descriptor (RSD) [96]: busca describir la forma de la superficie alrededor de un punto. Por cada de punto de interés con su vecino se calcula la diferencia entre sus normales y la distancia entre ellos. Se ajusta una esfera de forma que encaje con las normales y la distancia entre los puntos. De esta manera si la superficie es plana el radio de la esfera será infinito, y en caso contrario, el radio de la esfera será parecido a la de la superficie curva.
- 3D Shape Context (3DSC) [97]: es un descriptor que se basa en el Shape Context [73] pero en 3D.

- Unique Shape Context (USC) [98]: extiende el descriptor 3DSC definiendo un frame de referencia para tener orientaciones únicas reduciendo la comp-lejidad del descriptor y mejorando su precisión.
- Signature of Histograms of Orientations (SHOT) [99]: encapsula la información topologica de la superficie de cada punto de interés con una estructura de soporte parecida a la usada en 3DSC. Las divisiones de la esfera son fijas (32) y se calcula un histograma por cada volumen. Es un método invariante a la rotación y robusto ante el ruido y al desorden.
- Point Pair Features (PPF) [100]: es un descriptor formado por información a pares de puntos que incluye la distancia entre los puntos, el ángulo que forman cada normal con el vector que resulta de la diferencia entre las normales y el ángulo que forman las dos normales. Este descriptor es invariante a transformaciones rígidas.
- Iteratite Closest Point (ICP) [101]: es un método iterativo que se utiliza para alinear dos nubes de puntos. ICP necesita una estimación inicial por lo que se suele utilizar para refinar las poses obtenidas por otros métodos.

La mayoría de estos métodos se pueden encontrar en la librería PCL [102], la cual es una librería de código abierto C++ para el procesamiento de nubes de puntos.

2.3.3. Métodos basados en plantillas

Estos métodos primero crean unas plantillas de los objetos a reconocer que capturan las diferentes formas de los objetos desde diferentes perspectivas. Los objetos son detectados cuando una plantilla encaja en la imagen y su pose es la que da esa plantilla. Uno de los primeros métodos se llama Snakes [103] que consiste en una Spline que se ajusta a la forma del objeto. A raíz de este método han surgido muchos otros que siguen la misma idea [104, 105]. Uno de los autores que más ha trabajado en este campo es Stefan Hinterstoisser. En 2010 [106], propuso un método basado en plantillas que buscaba las orientaciones de gradiente dominantes en un pequeño subconjunto de píxeles. Otros métodos parecidos propuestos por el mismo autor [107] han dado pie a uno de los métodos basado en plantillas más conocidos LINEMOD [108], el cual posteriormente también ha sido refinado [109]. Otros métodos que utilizan plantillas son [110] y [111].

2.4. Aprendizaje profundo

Las bases de las redes neuronales se han afianzado en el siglo pasado [112, 113], pero no ha sido hasta esta última década cuando se han podido utilizar de manera eficiente. El aprendizaje profundo ha ido ganando cada vez más popularidad debido a los méritos

conseguidos con ellas. En 2012, el modelo de aprendizaje profundo AlexNet [114] ganó el challenge ImageNet [115] con una ventaja del $10,8\,\%$ del top-5 error respecto al siguiente método. Esto fue posible gracias a la profundidad de la red y a que se utilizaron unidades de procesamiento gráfico (GPUs) durante el entrenamiento. Este momento ha sido un punto de inflexión para las redes neuronales profundas. A partir de ese momento y gracias la facilidad de obtener una GPU, han surgido una infinidad de arquitecturas y modelos que se han utilizado en diversos sectores [116, 117, 118].

El challenge ImageNet ha sido bastante importante en la generación de nuevas arquitecturas de aprendizaje profundo. Los métodos que conseguian liderar el podium han conseguido relevancia y popularidad. Entre ellos se encuentran: VGG [119], Inception [120, 121], ResNet [122], Inception-ResNet [123], ResNeXt [124], NASNET [125], EfficientNet [126] y ViT [127]. ImageNet es un dataset para clasificación, por lo que la mayoría de métodos que se han presentado son para clasificación. Aun así, la parte de extracción de características de esos modelos se puede utilizar para otras tareas.

En cuanto a estimación de pose 6D, el BOP challenge [128] ha definido una referencia común para comparar técnicas. La mayoría de las técnicas 2D de aprendizaje profundo primero hacen la detección 2D y obtienen la pose 6D de esa detección [129, 130, 131] o directamente buscan correspondencias 2D a 3D para obtener la pose 6D [132, 133, 134, 135, 136]. Profundizando más en estos métodos, los métodos como [135] o [136] siguen un planteamiento muy parecido a los métodos tradicionales.

Existen también muchos métodos semi-supervisados que utilizan datos no etiquetados para mejorar el rendimiento de las redes neuronales. Por ejemplo, Pseudo Labels [137], Noisy Student [138] o Meta Pseudo Labels [139]. O incluso métodos que permiten manejar entradas y salidas arbitrarias de datos [140, 141].

Para entender el funcionamiento de las redes neuronales, se van a explicar las partes más comunes que las componen y las arquitecturas más utilizadas. Las partes más comunes son las siguientes:

- Capa densa (*Fully-connected*): es la capa más básica. Consiste en un conjunto de funciones no lineales que encapsulan una neurona (perceptron). Estas neuronas aplican una transformación lineal a una entrada. A esta salida es a la que se le aplica una transformación no lineal a traves de la anteriormente mencionada función de no linealidad. Por ello, la salida de esta capa y es el resultado de aplicar la función de activación f (no linear) al producto escalar entre la matriz de pesos W y el vector de entrada x, más el vector de bias b (Ecuación 2.5).

$$y = f(Wx + b) \tag{2.5}$$

  Las funciones de activación posibles se explican más adelante.

- Capa convolucional: las capas convolucionales son capas que aplican una convolución a la entrada. La convolución es una operación matemática que consiste en un producto escalar, donde el núcleo (*kernel*) se desplaza a lo largo de la matriz de entrada, y tomamos el producto escalar entre ambos como si fueran vectores (Figura 2.10).

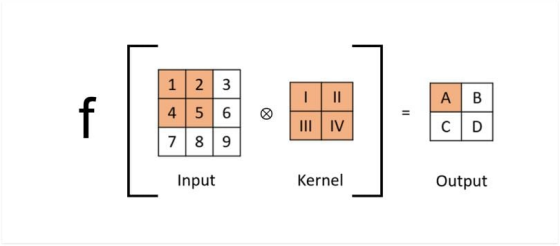


Figura 2.10: Ilustración del comportamiento de una capa convolucional. Imagen: Diego Unzueta.

  Como en las capas densas, a la salida de la convolución se le aplica una función de activación no lineal.

- Capa de pooling: Las capas de pooling sirven para reducir la dimensionalidad de un mapa de características (matriz de salida de una capa convolucional). Para un kernel de tamaño k y un stride s, el tamaño de mapa de características de $n_h \times n_w \times n_c$ se reduce a un tamaño de $(n_h - f + 1)/s \times (n_w - f + 1)/s \times n_c$, donde n_h es la altura, n_w es el ancho y n_c es el número de canales del mapa de características. Existen diferentes tipos de pooling, como por ejemplo el max pooling (selecciona el valor máximo para cada desplazamiento del kernel), el average pooling (computa la media de los valores que abarca el kernel) o el global pooling (hace un max o average pooling que reduce el mapa de características a $1 \times 1 \times n_c$).

- Capa de normalización: Son capas que permiten reducir el desplazamiento interno de covariables [142] que es un error que se da en la actualización de los pesos de los modelos cuando se asume que los pesos anteriores a la actualización son fijos. Para ello se existen métodos como el Batch normalization [142] o el Layer normalization [143] que ayudan a coordinar la actualización de pesos en multiples capas.

- Función de activación: Las funciones de activación son funciones no lineales que se aplican a la salida de una capa para introducir no linealidad en la red. Algunas de las funciones de activación más utilizadas son: ReLU [144], Leaky ReLU [145], ELU [146], SELU [147], tanh [148], sigmoid [148] y softmax [149].

- Dropout: Para evitar que las redes neuronales se sobreajusten (*overfitting*) a los datos de entrenamiento, se utilizan técnicas de regularización. Una de las técnicas más utilizadas es el dropout [150]. Esta técnica consiste en desactivar aleatoriamente un porcentaje de neuronas de la red durante el entrenamiento. De esta manera, la red no puede depender de una neurona en concreto para realizar la predicción.

- Loss: Las redes neuronales se tratan como un problema de optimización en la que los pesos de las capas son los parámetros a optimizar. Para ello se utiliza una función de coste que se minimiza mediante un algoritmo de optimización (por ejemplo, *Stochastic Gradient Descent*). Esta función de coste se conoce como función de pérdida (*loss function*). Existen diferentes tipos de funciones de pérdida en función del problema que se quiera resolver. Por ejemplo, para problemas de clasificación se suele utilizar la entropía cruzada (*cross entropy*) y para problemas de regresión se suele utilizar el error cuadrático medio (*mean squared error*).

Los grupos de arquitecturas más relevantes (aunque existen muchos más) son: Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), Autoencoders (AE), Generative Adversarial Networks (GAN), Transformers y Modelos de Difusión (Difussion models). En las siguientes secciones se explican brevemente cada uno de ellos.

2.4.1. Convolutional Neural Networks (CNN)

Las CNN son las redes neuronales más utilizadas hasta la fecha en el aprendizaje profundo para visión por computador. Estas consisten en una serie de capas convolucionales y capas de pooling que extraen características de la imagen. La Figura 2.11 muestra un ejemplo de una arquitectura convolucional.

Algunas de las CNN más conocidas y utilizadas son: VGG [119], Inception [120, 121], ResNet [122], Inception-ResNet [123], ResNeXt [124], NASNET [125] y EfficientNet [126]. Muchas de estas arquitecturas han definido unos bloques básicos que se pueden repetir para crear arquitecturas más complejas. Por ejemplo, en ResNet se definen el bloque residual, el cual añade una conexión de salto entre bloques y Inception module el bloque de Inception, que combina varias capas de multiples tamaños de kernel que luego pasa a la siguiente capa concatenados.

2.4.2. Recurrent Neural Networks (RNN)

Las redes neuronales recurrentes son redes neuronales que tienen una memoria interna que les permite recordar información de pasos anteriores. Por ello, son muy útiles para procesar datos secuenciales como texto o audio, aunque también se pueden utilizar

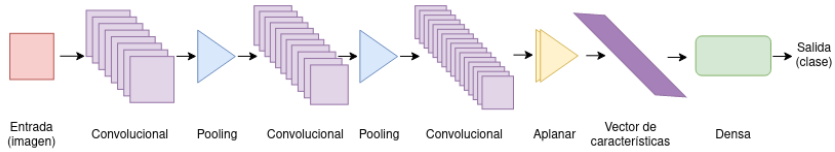


Figura 2.11: Ejemplo de una arquitectura convolucional que toma como entrada una imagen 2D, y está compuesta por 3 capas convolucionales y 2 capas de pooling intercalados. Su salida se aplana para formar el vector de características, que se pasa a la capa densa para su clasificación.

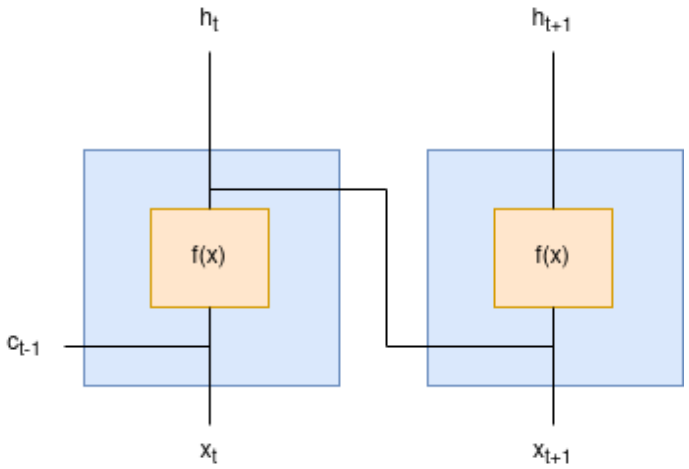


Figura 2.12: Ejemplo simplificado de una arquitectura recurrente.

combinadas con arquitecturas convolucionales para procesar vídeos. La Figura 2.12 muestra un ejemplo de una arquitectura recurrente. Algunas de las RNN más conocidas y utilizadas son: LSTM [151] y GRU [152].

2.4.3. Autoencoders (AE)

Los Autoencoders [153] son un tipo de arquitectura que se utiliza para aprender representaciones eficientes de datos. Estas arquitecturas se componen de dos partes: una codificadora y una decodificadora. La codificadora es una red neuronal que reduce la dimensionalidad de los datos de entrada y la decodificadora es una red neuronal que reconstruye los datos de entrada a partir de la codificación. La Figura 2.13 muestra un ejemplo de una arquitectura Autoencoder. Estas arquitecturas son muy utilizadas para tareas de clasificación (Sparse, Denoising y Contractive autoencoders) que incluyen

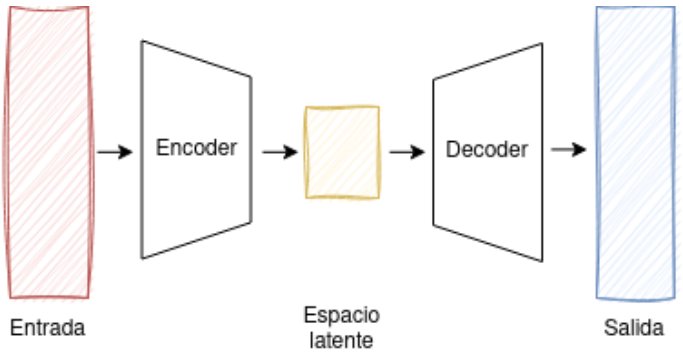


Figura 2.13: Ejemplo simplificado de una rquitectura Autoencoder que se compone de una primera parte de encoding y su posterior decoding.

reconocimiento de caras, detección de características o detección de anomalias, o como modelos generativos (Variational Autoencoders) para generar datos aleatorios similares a los datos de entrada.

Los Sparse autoencoders como [154, 155] añaden restricciones de dispersión en las unidades ocultas. Los Denoising Autoencoders [156, 157], en cambio, corrompen la entrada de datos a propósito con ruido para evitar que la red aprenda la función identidad y consiga reducir la dimensionalidad o aprender representaciones útiles. Los Contractive Autoencoders [158, 159] añaden una penalización de en la función de coste de reconstrucción. Los Variational Autoencoders [160] son una variante de los autoencoders que permiten generar muestras aleatorias de los datos de entrada, el cual se suele utilizar como modelo generativo.

2.4.4. Generative Adversarial Networks (GAN)

Los modelos generativos adversariales o Generative Adversarial Network [161, 162, 163] son un tipo de arquitectura generativa que entrenan simultaneamente dos modelos uno que genera nuevos ejemplos y otro que discrimina entre ejemplos reales y generados. La Figura 2.14 muestra un ejemplo de una arquitectura GAN. Algunos ejemplos de modelos GAN para generar imagenes 2D son [164], [165] y [166]; también se han utilizado para generar datos 3D como [167] y [168].

2.4.5. Transformers

Los Transformers [169, 170] son un tipo de arquitectura de red neuronal que se basa en la atención. La atención es un mecanismo que permite a una red neuronal prestar atención a diferentes partes de la entrada para realizar una tarea. La Figura 2.15 muestra

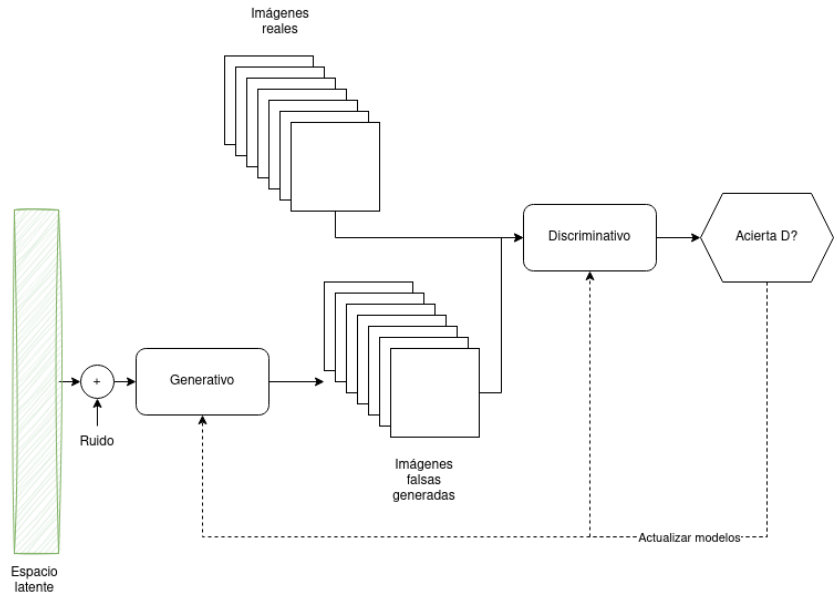


Figura 2.14: Representación de un sistema generativo GAN.

un ejemplo de una arquitectura Transformer y la Figura 2.16 muestra la arquitectura de la capa de atención. En la arquitectura Transformer de [170] se utilizan capas de atención multi-cabeza, las cuales consisten en concatenar varias capas de atención en paralelo con pesos diferentes. Estas arquitecturas son muy utilizadas en tareas de NLP (*Natural Language Processing*) como traducción, resumen de texto o generación de texto [171]. Algunas de estas arquitecturas son genéricas y se pueden utilizar para resolver muchas de las tareas de NLP, como por ejemplo, BERT [172], GPT [173, 174, 175] y XLNet [176]. Aun así estos últimos años también se estan utilizando en tareas de visión por computador [177, 178, 127].

2.4.6. Modelos de difusión

Los modelos de Difusión son unos modelos generativos que han ganado mucho interés en los últimos años [179, 180, 181]. Los modelos difusos se basan en la difusión de una distribución inicial para generar una distribución final. Es decir, en añadir ruido iterativamente a una imagen para generar datos de entrenamiento y entrenar un modelo (el modelo de difusión) el cual sea capaz de deshacer cada paso de ruido añadido a las imagenes de entrenamiento. La Figura 2.17 muestra un esquema del proceso de entrenamiento de una arquitectura de difusión. Estos modelos se pueden utilizar para

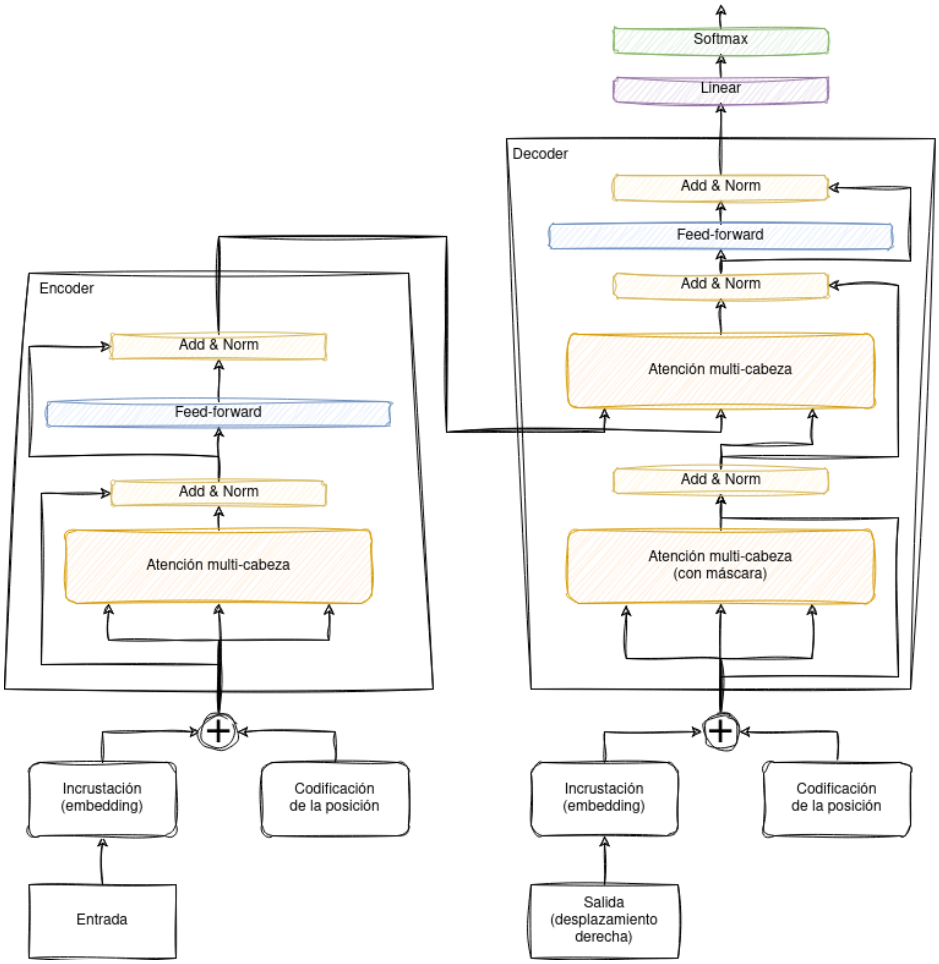


Figura 2.15: Ejemplo de una arquitectura Transformer compuesto de capas de atención multi-cabeza.

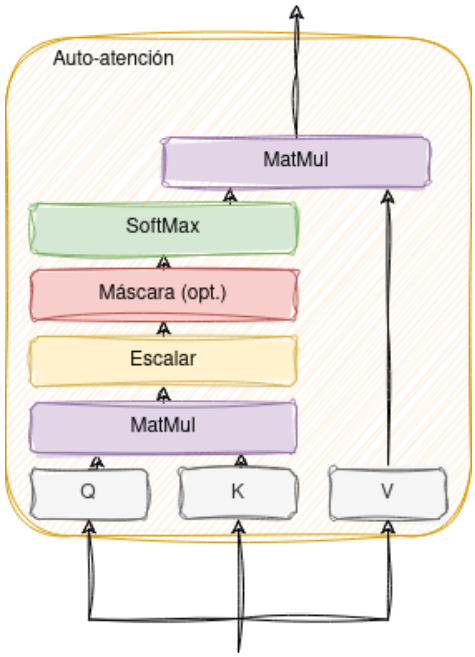


Figura 2.16: Capa auto-atención.

generar imágenes [181, 182], vídeos [183] o texto [184]. Algunos de los modelos de difusión más conocidos son: GLIDE [185], DALL-E [186], Midjourney [187] y Stable Diffusion [188] De hecho estos últimos modelos son capaces de generar imágenes a partir de texto, es decir, son modelos generativos de imágenes condicionados por texto, de esta forma se mezcla NLP con visión por computador.

2.4.7. Transfer learning

El aprendizaje transferido o *Transfer Learning* [189] es una herramienta del aprendizaje profundo para reutilizar conocimientos previos aprendidos por una red neuronal para resolver nuevos problemas. Un uso muy común del Transfer Learning es cuando no se disponen de suficientes datos para entrenar una red neuronal desde cero. Se puede ajustar una red pre-entrenada en un problema similar al problema que se quiere resolver para obtener un buen rendimiento con un número mucho menor de datos. La Figura 2.18 muestra un esquema de Transfer Learning. Existen muchos tipos de Transfer Learning [190], pero se podrían clasificar en 3 categorias: las que se basan en estrategias para cambiar los pesos de la red con nuevas instancias [191], las que hacen un mapeo del

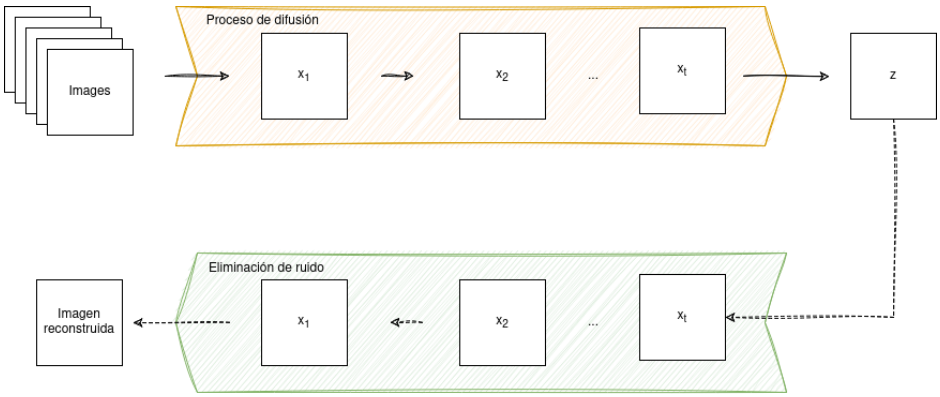


Figura 2.17: Esquema de entrenamiento de una arquitectura de difusión.

dominio de origen al dominio objetivo [192, 193] y las que utiizan partes de la red pre-entrenada como capas de la nueva red [194].

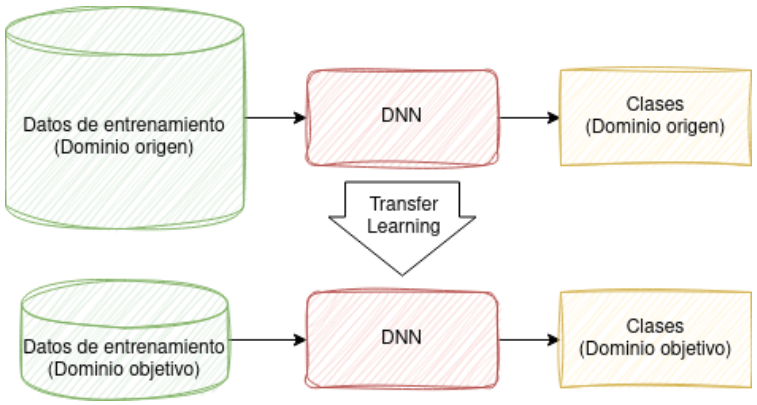


Figura 2.18: Esquema de técnicas de Transfer Learning.

2.5. Datasets y Benchmarks

Los datasets son una parte esencial de la visión por computador. Los datasets son lo que nos permite aprender el conocimiento necesario para resolver las tareas de visión. Métodos tanto clásicos como de aprendizaje profundo necesitan de datos para aprender o para buscar correspondencias de un modelo conocido. Para entrenar métodos clásicos se necesitan menos datos que para entrenar métodos de aprendizaje profundo. En esta

sección se va a presentar el benchmark de estimación de pose 6D de mayor relevancia en la actualidad y métodos para generar datos sintéticos para entrenar redes neuronales, los cuales son necesarios por la dificultad que supone etiquetar datos 3D.

2.5.1. BOP Challenge

Un benchmark importante para la estimación de pose es el BOP Challenge [128]. Este challenge ha sido creado para evaluar los algoritmos de estimación de pose. Por ello, los datasets que se recogen en él y las métricas de evaluación han servido como benchmark para la mayoría de los algoritmos de estimación de pose. El BOP challenge recoge los 11 datasets para estimación de pose más utilizados en la literatura (Linemod [195], Linemod-Oclussion [196], TLESS [197], MVTec ITODD [198], HomebrewedDB [199], YCB-Video [200, 130], Rutgers APC [201], IC-BIN [202], IC-MI [91], TUD Light [128] and Toyota Light [128]). De ellos solo se utilizan 7 para la competición. Para la evaluación se utilizan métricas conocidas como el Visible Surface Discrepancy (VSD) [128, 203], el Maximum Symmetry-Aware Surface Distance (MSSD) [198] y el Maximum Symmetry-Aware Projection Distance (MSPD) [204]. Además, desde 2020 el BOP challenge ha incluido datos sintéticos para el entrenamiento en todos los datasets. El set de test de los datasets solo incluye datos reales. Gracias a esos datos sintéticos utilizados en el entrenamiento, las métricas obtenidas han mejorado notablemente, incluso para los casos en que sólo se utilizan datos sintéticos para entrenar. Las Figuras 2.19a, 2.19b y 2.19c muestran ejemplos de imágenes reales de 3 de los datasets incluidos en el BOP challenge (LMO, YCB-V y T-Less) y la Figura 2.19d una imagen sintética generadas con las herramientas del BOP challenge para el dataset de YCBV.

2.5.2. Generación de datos sintéticos

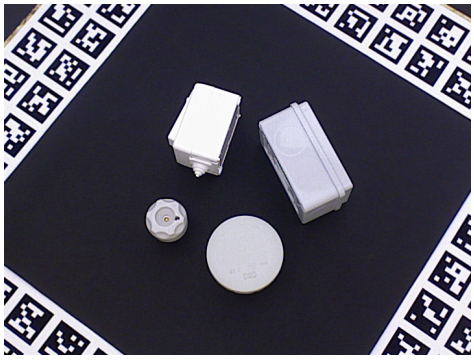
Gracias al avance en los motores gráficos y sistemas de generación de modelos 3D, las imágenes o escenas que se pueden generar mediante ellas son cada vez más realistas. Dado la gran cantidad de datos etiquetados que necesitan las redes neuronales, se está tendiendo al uso de datos sintéticos para entrenar las redes de aprendizaje profundo [205].

Entre los motores gráficos más utilizados están Unity [206] y Unreal Engine [207]. Estos motores gráficos principalmente se utilizan en la industria del videojuego, pero cada vez son más utilizados en diferentes sectores. En el caso de la generación de imágenes sintéticas para entrenar redes neuronales existen múltiples plug-ins para los motores gráficos [208, 209]. Estos plug-ins permiten automatizar la generación procedural de escenas (*Domain randomization*) y la forma en la que se van a capturar (imágenes RGB, mapas de profundidad, segmentación semántica…).

La Figura 2.20 muestra un ejemplo de escena generada en Unity. Esta imagen se ha obtenido de [210].

(a) Imagen real del dataset LMO perteneciente al BOP challenge.

(b) Imagen real del dataset YCBV perteneciente al BOP challenge.



(c) Imagen real del dataset TLESS perteneciente al BOP challenge.

(d) Imagen sintética generada con la herramienta del BOP challenge para el dataset YCBV.

Figura 2.19: Ejemplos de imágenes reales y sintéticas de los datasets del BOP challenge.

Además de Unity y Unreal Engine, existen otros motores gráficos que se pueden utilizar para generar datos sintéticos. Este es el caso de Nvidia Omniverse [211]. Nvidia Omniverse es una plataforma que engloba un conjunto de herramientas para la generación de datos 3D, muy orientado al metaverso. Nvidia Isaac Sim [212] es una herramienta integrada en Nvidia Omniverse que permite la simulación de aplicaciones robóticas y la generación de datasets sintéticos fotorealistas y físicamente precisos.

Otra herramienta que se puede utilizar para generar datos sintéticos es Blender [213]. Blender es un software de modelado, animación y renderizado 3D. Se puede utilizar para generar datos sintéticos de forma procedural. Para ello se utiliza *BlenderProc* [214], un pipeline que permite renderizados fotorealistas de escenas procedurales generadas en Blender. Varios benchmarks de visión por computador como BOP [128] o ShapeNet [215]

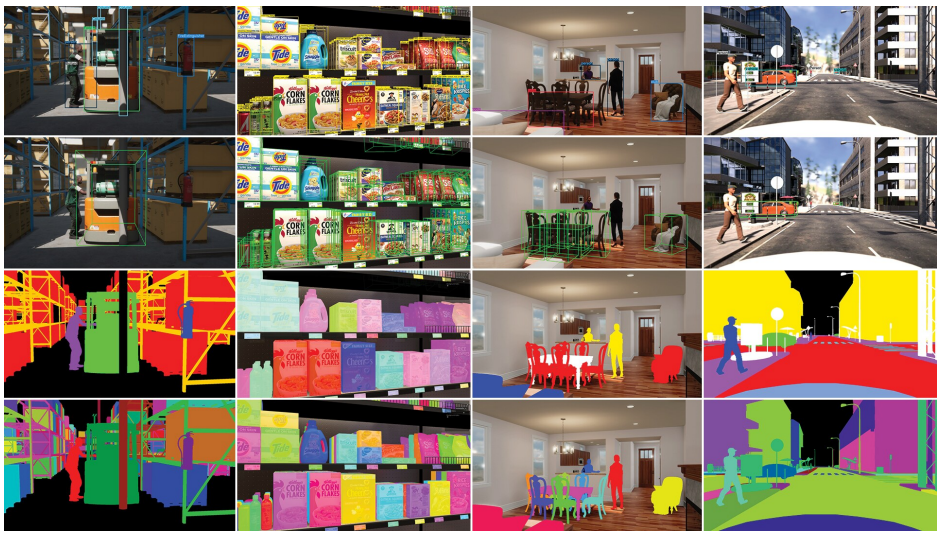


Figura 2.20: Ejemplo de escenas generadas en Unity. La primera fila muestra las imagen RGB con los bounding boxes 2D, la segunda fila muestra las imagenes RGB con los bounding boxes 3D, la tercera fila muestra la segmentación semántica y la cuarta fila muestra la segmentación de instancia.

utilizan BlenderProc para generar sus datos sintéticos.

CAPÍTULO 3

Contribuciones

Para dar respuesta a las hipótesis planteadas en la sección 1.4, se han publicado estas contribuciones en diferentes revistas y conferencias. Este capítulo recoge todos los resultados de este proyecto de tesis y se explicarán las contribuciones de cada uno de los artículos.

3.1. Resultados de la investigación

A continuación se listan los artículos en los que el autor de esta investigación ha contribuido. Los artículos en negrita son contribuciones que dan respuesta a la investigación realizada durante esta memoria y están incluidos en la Parte II.

- [216] Merino, I., Azpiazu, J., Remazeilles, A., Sierra, B., 2019. **2d Image Features Detector and Descriptor Selection Expert System**. Computer Science & Information Technology (CS & IT), AIRCC Publishing Corporation. https://doi.org/10.5121/csit.2019.91206

- [217] Merino, I., Azpiazu, J., Remazeilles, A., Sierra, B., 2019. **2D Features-based detector and descriptor selection system for hierarchical recognition of industrial parts**. International Journal of Artificial Intelligence & Applications (IJAIA), AIRCC Publishing Corporation. https://doi.org/10.5121/ijaia.2019.10601

- [218] Merino, I., Azpiazu, J., Remazeilles, A., Sierra, B., 2020. **Histogram-Based Descriptor Subset Selection for Visual Recognition of Industrial Parts**. Applied science 10, 3701. MDPI. https://doi.org/10.3390/app10113701

- [219] J.L. Outón, I. Merino, I. Villaverde, A. Ibarguren, H. Herrero, P. Daelman, B. Sierra, 2021. A Real Application of an Autonomous Industrial Mobile Manipulator within Industrial Context. Electronics 10, 1276. MDPI. https://doi.org/10.3390/electronics10111276

- [220] Merino, I., Azpiazu, J., Remazeilles, A., Sierra, B., 2021. **3D Convolutional Neural Networks Initialized from Pretrained 2D Convolutional Neural Networks for Classification of Industrial Parts**. Sensors 21, 1078. MDPI. https://doi.org/10.3390/s21041078

- [221] Merino, I., Azpiazu, J., Remazeilles, A., Sierra, B, 2023. **Ensemble of 6 dof pose estimation from state-of-the-art deep methods**. Neurocomputing, Volume 541. Elsevier. https://doi.org/10.1016/j.neucom.2023.126270

3.2. Contribuciones a la visión por computador clásica

La visión por computador clásica 2D no ha tenido avances significativos en la última década. Esto se debe al gran éxito que han tenido los algoritmos de aprendizaje profundo. Aun así, como se refleja en la hipótesis **H1**, creemos que este tipo de algoritmos aún no está obsoleto y para ciertos casos es mejor utilizar modelos de detección o clasificación clásicos en vez de algoritmos de aprendizaje profundo. Muchos de los algoritmos de detección como SIFT o SURF han sido líderes entre este tipo de métodos durante muchos años para buscar correspondencias entre texturas; por otro lado HOG es muy utilizado para detectar y clasificar caras. Es por eso que muchos de estos algoritmos tienen un uso concreto, es decir, que funcionan muy bien para ciertos casos de uso mientras que para otros no tan bien. Por eso hasta el auge de los algoritmos de aprendizaje profundo era necesario un experto o experta en visión para poder determinar cuál era el algoritmo que se tenía que usar para el caso de uso concreto que se quería resolver. Los algoritmos de aprendizaje profundo han resuelto esta problemática aprendiendo cómo codificar o filtrar la imagen para luego clasificarla. Por lo tanto, existen algunos casos de uso concretos en los que el modelo de aprendizaje profundo requiere menor presencia del experto o experta en visión. Aun así, para poder configurar bien el modelo de aprendizaje profundo sigue siendo necesario que alguien con conocimientos en el tema lo realice.

Partiendo de esta premisa nos preguntamos si existía alguna forma de replicar este comportamiento de los algoritmos de aprendizaje profundo: buscar un método que aprendiese cuál es el algoritmo de visión clásica o qué combinación de ellos es la mejor para el caso concreto al que nos enfrentámos.

Para ello desarrollamos el método presentado en [216, 217]. El caso de uso de esos artículos es clasificar imágenes de ciertas piezas industriales. Para ello se define un clasificador jerárquico que se entrena en dos fases.

En la primera fase se busca separar las piezas en K subgrupos diferentes en función de cómo se comportan con cada descriptor. Para ello, se calculan los valores F_1, por objeto para cada pipeline (detector + descriptor + matching). El F_1 es la media armónica entre la precisión o *precision* y la exhaustividad o *recall* (Ecuación 3.1). El pipeline que obtenga mayor F_1 es el que se utilizará como clasificador de tipologías. Una vez obtenidos los F_1 por objeto y pipeline, clusterizamos estos objetos en K ($K <$ número de objetos) tipologías utilizando K-means. Una vez definidas las llamadas tipologías, para cada una de ellas se utiliza el descriptor que mejor funciona.

$$F_1 = \frac{2}{recall^{-1} + precision^{-1}} \tag{3.1}$$

Con la primera fase terminada, hemos definido 3 elementos:

- El pipeline que se utiliza para separar las tipologías. Este es el pipeline que obtiene mayor media de valores F_1.
- Las diferentes tipologías. Estas tipologias pueden estar compuestas por uno o varios objetos.
- Los pipelines que mejor funcionan para cada tipología. Esos pipelines son los que mayor media tengan para cada tipología y servirán para diferenciar cada objeto dentro de la tipología.

En la segunda fase se entrena el pipeline que clasifica las tipologías y los pipelines de cada una de las tipologías.

Para evaluar el clasificador jerárquico se ha utilizado el valor F_1 obtenido con un número de imágenes y número de objetos variable. Se han utilizado sets de 10, 20, 30, 40 y 50 imágenes por objeto y 3, 4, 5, 6 y 7 objetos, dando un total de 25 casos diferentes. Debido al bajo número de imágenes que hay en el dataset y que para algunas de las combinaciones solo se tienen 10 imágenes por objeto, se utiliza un *Nested Leave-One-Out Cross-Validation* (Nested LOOCV [222, 223]). Primero se calculan los vectores de características para todas las imágenes con cada pipeline (utilizando solo el detector y descriptor sin hacer el matching). El LOOCV interior sirve para definir los sets de entrenamiento y validación de la primera fase. El LOOCV exterior, en cambio, sirve para entrenar los clasificadores en la segunda fase y testearlos.

Los resultados obtenidos con el clasificador jerárquico para nuestro dataset son mejores que con los pipelines definidos. Se puede apreciar que a mayor cantidad de imágenes y número de objetos, la mejora del clasificador jerárquico es más significativa.

Otro método que sigue la premisa que se ha presentado al inicio y da respuesta a la hipótesis **H1** es el que se presenta en [218]. En este artículo se presenta una técnica de selección de características de descriptores globales para clasificación de piezas industriales. Este método define la forma de buscar qué conjunto de descriptores globales funciona mejor para un caso de uso concreto. Para ello, se utilizan técnicas de selección de características (*Feature Selection*), más concretamente una selección de características secuencial hacia delante y hacia atrás (*Sequential Forward Subset Selection* y *Sequential Backward Subset Selection*, respectivamente). La SFSS, de sus siglas en ingles, parte del conjunto vacío y va incluyendo en cada iteración el descriptor que maximice la métrica a evaluar hasta que no se obtenga mejora. La SBSS, en cambio, parte del conjunto de todos los descriptores y va retirando descriptores siempre que mejore la métrica. En el caso del artículo la métrica es el F_1.

En el artículo se realizan experimentos con diferentes descriptores y formas de aplicar los descriptores tanto con SFSS y SBSS. Los experimentos muestran que al utilizar SFSS aplicando el descriptor a toda la imagen, en vez de a trozos uniformes y concatenarlos, se obtienen los mejores resultados. De hecho esta versión es también la más rápida en computar y la más sencilla de implementar. Debido al bajo número de imágenes que se tiene en el dataset (mismo dataset que en [216, 217]) este método supera a los resultados obtenidos con redes neuronales como Xception [224] o Siamese [225].

Con todo esto, se ha dado una respuesta a la hipótesis **H1**. Se puede confirmar que existen formas de mejorar los métodos clásicos actuales y que todavía tienen un gran margen de mejora. Aplicar refinamientos o fusión de características a los métodos clásicos permite incrementar la precisión de estos. En nuestros casos, conseguimos unas mejoras en torno al $10\,\%$ respecto de los métodos base.

3.3. Contribuciones a la transferencia de aprendizaje 2D-3D

Como se ha explicado en la Sección 2.2 los métodos de aprendizaje profundo han revolucionado la visión por computador. Las redes convolucionales, los datasets con una gran cantidad de imagenes y los avances en hardware han permitido desarrollar técnicas de visión muy precisas y, aunque lentas entrenando, rápidas en la inferencia. Este gran avance es especialmente notorio para imágenes 2D pero los métodos que incorporan datos 3D como nubes de puntos o vóxeles 3D no obtienen resultados tan buenos como los obtenidos con los métodos de aprendizaje profundo 2D. Esto se puede deber a la mayor dificultad de adquisición de este tipo de datos (sensores más caros si se quiere una mayor precisión) y la dificultad de etiquetarlos (por ejemplo definir las poses 6D de los objetos). En la Sección 3.4 se profundizará más sobre esto último.

El artículo [220] presenta un método para dar respuesta a este problema (**H2**). Este método consiste en transformar los pesos de redes convolucionales 2D preentrenadas para inicializar los pesos de redes convolucionales 3D. Para ello se presentan dos

transformaciones: la extrusión de pesos y la rotación de pesos. La extrusión consiste en replicar los pesos 2D a lo largo de un eje (X, Y o Z). En la rotación se rotan los pesos respecto de un eje como al generar un cuerpo de revolución. Las Figuras 3.1 y 3.2, extraidas de [220], muestran un ejemplo de extrusión y rotación, respectivamente.

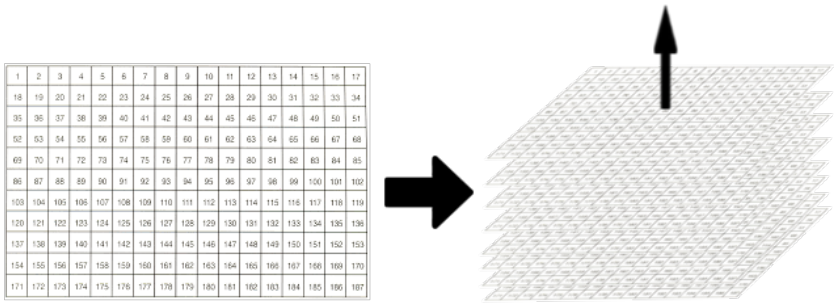


Figura 3.1: Extrusión de pesos 2D para obtener los pesos 3D: se replica una la matriz de pesos 2D a lo largo de nueva dimensión añadida para inicializar arquitecturas 3D.

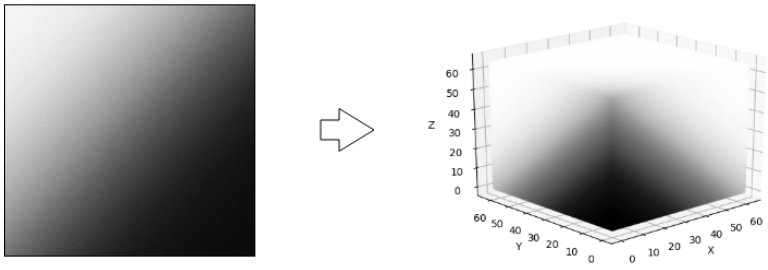


Figura 3.2: Rotación de pesos 2D para obtener los pesos 3D: para cada valor de la matriz de dimensionalidad agrandada se le asigna un valor de la matriz de pesos 2D siguiendo el mapeo de la Ecuación 3.2 para inicializar arquitecturas 3D.

$$T(x, y, z) = M(x, min(\lfloor\sqrt{y^2 + z^2}\rfloor, H)), \tag{3.2}$$

donde T es la matriz de pesos 3D, M es la matriz de pesos 2D, H es la altura de la matriz de pesos 2D y x, y y z son las indices de la posición de las matrices de pesos.

Se compara el comportamiento de las versiones 3D de 4 arquitecturas de redes convolucionales (VGG16 [119], ResNet [122], Inception-ResNet v2 [123] y EfficientNet [126]) inicializadas con los pesos de sus versiones 2D preentrenados con ImageNet [114] transformados con las técnicas antes mencionadas. Las versiones 3D de estas arquitecturas sustituyen todas las capas 2D por sus respectivas versiones 3D, y se ajusta el *padding* () y las dimensiones de entrada:

- VGG16: 224 x 224 x 3 (input versión 2D) a 96 x 96 x 96 x 3 (input versión 3D).
- ResNet: 224 x 224 x 3 (input versión 2D) a 96 x 96 x 96 x 3 (input versión 3D).
- Inception-ResNet v2: 299 x 299 x 3 (input versión 2D) a 139 x 139 x 139 x 3 (input versión 3D).
- EfficientNet: sigue la siguiente ecuación: $r_{3D} = \lfloor\sqrt[3]{r_{2D}^2}\rfloor$, donde r_{2D} es la resolución en 2D y r_{3D} es la resolución en 3D. Por ejemplo, para EfficientNet-B0 pasa de 224 x 224 x 3 a 36 x 36 x 36 x 3.

Estas arquitecturas inicializadas con los pesos 2D transformados consiguen obtener un F_1 mejor que sus versiones sin preentrenamiento. En la Tabla 3.1 se puede ver que ambos preentrenamientos funcionan mejor para todos los casos que la versión sin preentrenar. Para algunos casos funciona mejor la extrusión y para otros la rotación por lo que no es posible decir con los experimentos actuales que uno de los dos preentrenamientos sea mejor que el otro. Además, se han comparado con una arquitectura (PointNet [226]), que toma como entrada nubes de puntos, perteneciente al estado del arte y el método propuesto obtiene mejores resultados.

Arquitectura	**Sin preentrenamiento**	**Extrusión**	**Rotación**
ResNet	0,8272	**0,8612**	0,8512
Inception ResNet	0,7558	0,8887	**0,8939**
EfficientNet B0	0,8605	**0,9217**	0,9052
EfficientNet B1	0,8372	0,8420	**0,8422**
PointNet	0,9048	-	-

Tabla 3.1: Comparativa de los F_1 obtenidos en los experimentos con cada arquitectura y preentrenamiento

Como se ha podido apreciar sigue existiendo un margen de mejora para redes convolucionales 3D. En comparación con sus versiones 2D los datasets que se disponen para entrenar son mucho más grandes, debido a la complejidad de adquisición y etiquetado de los datos 3D. Aun así, con la transferencia de conocimiento 2D a 3D se ha conseguido hasta un 6 % de mejora para algunas arquitecturas, validando la hipótesis **H2**.

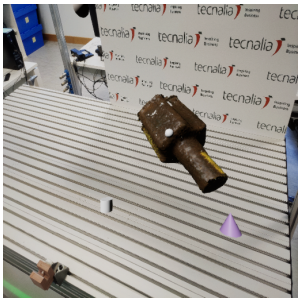
3.4. Contribuciones a la generación de datasets sintéticos

Como hemos comentado en la anterior sección, uno de los mayores problemas para trabajar con datos 3D es la dificultad de definir las posiciones reales de los objetos. En imágenes 2D, normalmente, se suelen dar como posición de los objetos el bounding box 2D de estos. Definir este bounding box es bastante sencillo y existen muchas herramientas que facilitan su etiquetado [227, 228, 229]. En el caso 3D este proceso es bastante más complicado ya que añadimos una dimensión extra. Equivocarse a la hora de definir la posición de los objetos es más común y puede complicar el entrenamiento.

Para intentar solucionar esta problemática, en los últimos años se han empezado a utilizar datos sintéticos ultra-realistas basados en físicas. Las herramientas de modelado 3D, diseño de videojuegos y entornos virtuales han avanzado mucho tanto que muchas veces cuesta diferenciar una imagen real de una generada por ordenador. Es por ello que este tipo de herramientas se han utilizado en la generación de datasets sintéticos para entrenar diferentes métodos.

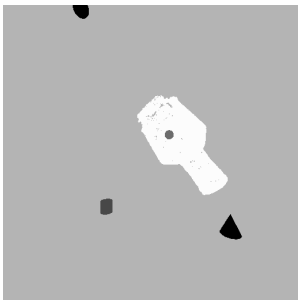
El método propuesto en [220] ha sido entrenado con un dataset sintético para dar respuesta a **H3**. Este dataset se ha generado usando Unreal Engine 4 (UE4) [207] y el plugin NVIDIA Deep Learning Dataset Synthesizer (NDDS) [208]. Este dataset está compuesto por 7 piezas industriales que pertenecen a varios proyectos en los que el autor ha participado. Se ha hecho una reconstrucción de esas piezas utilizando una cámara 3D de luz proyectada y CloudCompare [230].

Una vez generados los modelos, éstos se importan a UE4. Gracias al plugin NDDS, se definen unos distractores, objetos geométricos aleatorios, cuya posición se va adaptando en cada iteración; se define una iluminación con cambios en la intensidad y dirección; se definen cambios en el fondo y se aleatorizan las posiciones de los modelos. Por cada iteración se realizan cambios en todos aspectos antes comentados y se obtiene una captura RGBD junto con su segmentación semántica, la segmentación de instancia y por cada objeto la pose 6D, la visibilidad, el bounding box 2D, el bounding box 3D y la proyección 2D del bounding box 3D. Las Figuras 3.3a, 3.3b, 3.3c y 3.3d muestran un ejemplo de una captura de una iteración de UE4 con el plugin NDDS. La Figura 3.3a es la captura RGB, la Figura 3.3b es la imagen de profundidad, la Figura 3.3c es la segmentación semántica (Ground truth) y la Figura 3.3d es la segmentación de instancia (Ground truth).

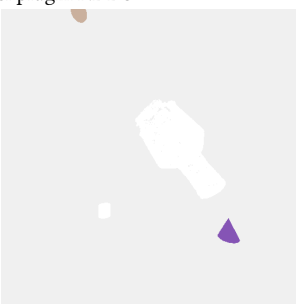


(a) Captura RGB obtenida mediante UE4 y el plugin NDDS

(b) Imagen de profundidad obtenida mediante UE4 y el plugin NDDS



(c) Segmentación semántica obtenida mediante UE4 y el plugin NDDS



(d) Segmentación de instancia obtenida mediante UE4 y el plugin NDDS

En [221] también se propone el uso de imágenes sintéticas para mejorar los resultados obtenidos con redes neuronales. En este caso la herramienta utilizada es BlenderProc [214]. BlenderProc es un pipeline procedural de Blender [213], un entorno abierto y gratuito para la creación de contenido 3D, para generar renderizados fotorealistas. El pipeline utilizado para generar las imágenes sintéticas para este caso es el que define el BOP challenge [231].

Las imágenes obtenidas con BlenderProc (Figura 3.4) son mucho más realistas que las obtenidas con UE4 y además la forma de definir el pipeline es mucho más sencilla. De hecho gracias a BlenderProc muchas de las técnicas de aprendizaje profundo que han participado en el BOP challenge que previamente no lograban superar a los métodos PPF (Point Pair Features), han conseguido liderar la tabla de clasificación. En el caso de nuestro artículo ocurre lo mismo, entrenar los métodos utilizados en él (PVN3D, FFB6D y Cosypose) con solo datos reales obtiene peores resultados que entrenando con datos sintéticos.

(a) Imagen sintética del dataset Linemod **(b)** Imagen sintética del dataset YCBV **(c)** Imagen sintética del dataset TLESS

Figura 3.4: Imagenes sintéticas generadas con BlenderProc para los datasets Linemod, YCBV y TLESS

Utilizando este tipo de herramientas de generación de datasets sintéticos, se consigue disminuir la cantidad de datos reales necesarios para entrenar modelos de aprendizaje profundo que generalicen bien y permitan clasificar, localizar o detectar objetos de forma eficiente. Esto es especialmente necesario para datos 3D debido a su complejidad de adquisición y etiquetado. Por eso, la **H3** es correcta y no solo ayuda a disminuir los datos reales necesarios, sino que también mejora los resultados que se obtienen utilizando solo imágenes reales.

3.5. Contribuciones de ensamblaje de métodos (*ensemble*)

Al igual que se han explicado las contribuciones realizadas en visión clásica mediante fusión o mejora de estos algoritmos en la Sección 3.2, en esta Sección vamos a hacer lo mismo para algoritmos de aprendizaje profundo. Una de las grandes diferencias entre los algoritmos clásicos y los de aprendizaje profundo es que los primeros por lo general estaban integrados por diferentes herramientas dentro del pipeline (detectores, descriptores y matching), las cuales se definen en función del caso del uso y del objetivo que se quiere lograr. Mientras que en los algoritmos de aprendizaje profundo se intenta evitar esto y por lo general se buscan o diseñan métodos *end-to-end*, es decir, que una única arquitectura permita realizar todo lo que antes hacia cada parte y entrenarlo en su totalidad. Por un lado, esto permite que el entrenamiento de las diferentes fases se retroalimente, esto es, que las características o *features* que se extraen estén pensadas para que el clasificador o estimador de pose que viene después sea capaz de utilizarlas de forma eficiente para maximizar su propósito, o lo que es lo mismo minimizar el la función de perdida o *Loss function*. Esta función sirve para evaluar la desviación entre las predicciones de la red neuronal y los valores reales. Por otro lado, los métodos clásicos permiten definir exactamente qué es lo que queremos extraer (bordes, esquinas, puntos de interés...). Aun así definir estos extractores de características y descriptores es mucho más laborioso y se necesita un experto o experta en visión para poder definir

que extractor y descriptor concreto funciona bien para el tipo de datos que tenemos (formas de los objetos, tipo de cámara, iluminación...). Como se puede apreciar cada uno tiene sus ventajas y desventajas por eso queremos extraer el potencial de ambos tipos de métodos fusionando métodos de aprendizaje profundo con técnicas más clásicas.

En [221] hemos presentado varios métodos de *ensemble* para fusionar varias técnicas de aprendizaje profundo para estimar poses 6D y de esta forma dar respuesta a la hipotesis **H4**. Para ello, tomamos como base 3 modelos de aprendizaje profundo: PVN3D[135], FFB6D[136] y Cosypose[131]. Los dos primeros métodos se basan en encontrar ciertos puntos de interés de los objetos (parte de aprendizaje profundo) y luego se aplica un ajuste de mínimos cuadrados (*Least Square Fitting*) con los puntos de interés del modelo. Los dos métodos difieren en la parte de aprendizaje profundo. El primero, PVN3D, saca información geométrica de la nube de puntos e información de color de la imágen RGB y luego fusiona esta información. Luego estos *features* se pasan a 3 módulos que sirven para obtener los centroides, la segmentación semántica y los puntos de interés. Gracias a los centroides y la segmentación semántica se pueden obtener los puntos de interés para cada instancia presente en la escena. La principal diferencia con el segundo modelo, FFB6D, es la parte de extracción de *features*. En vez de obtener las características geométricas y de color por separado, en cada capa de la red se fusionan ambos tipos de características para aprovechar la información geométrica a la hora de extraer *features* de color y viceversa. Cosypose, en cambio, es un método que solo utiliza imágenes RGB para obtener la pose 6D. Primero hace una detección gruesa de las poses 6D con una red convolucional gruesa (*CNN coarse*) y después refina esas poses 6D con otra red convolucional (*CNN refiner*). Cosypose incluye un refinamiento multivista que no hemos utilizado.

Una vez definidos los 3 modelos, vamos a definir cómo se han fusionado. Se han propuesto 2 estrategias: la estrategia de unión (*merge*) y la estrategia de apilado (*stacking*).

Por un lado, la estrategia de unión permite juntar los resultados de los modelos base de forma geométrica. Esta estrategia consiste en hacer una media de las poses de los modelos bases. La forma más sencilla es hacer la media sin contemplar nada más. Esto es lo que llamamos *simple merge*. Otra propuesta es añadir la confianza que dan los métodos sobre sus predicciones. De forma que se hace una media ponderada de las poses (*weighted merge*). Esto sigue teniendo sus inconvenientes cuando las poses 6D que da alguno de los modelos sea de alguna otra instancia. Para solucionar esto, se propone la unión por agrupación (*clustering merge*). Este método pretende primero agrupar las poses 6D obtenidas de los modelos base en grupos de poses cercanas y luego aplicar un *simple merge* o *weighted merge* al grupo más grande. En el caso de que tengan la misma cantidad de elementos se elige el grupo con mayor confianza media. De esta forma tenemos 4 métodos dentro de la estrategia de unión: *simple merge*, *weighted merge*, *clustering simple merge* y *clustering weighted merge*.

Por otro lado, la estrategia de apilado utiliza un modelo de *machine learning* para fusionar las poses 6D que estiman los modelos base. Estos modelos tienen como objetivo tomar como entrada los valores de las poses 6D y hacer una regresión de la nueva pose refinada. Se han probado 6 diferentes modelos: SVR, Arboles de decisión, Regresión lineal de Ridge, Regresión lineal de mínimos cuadrados ordinarios, Regresor KNN y MLP.

Los resultados obtenidos demuestran que la estrategia de unión funciona mejor para regresión que las estrategias de apilado. Además, se logra mejorar para algunos datasets las poses 6D obtenidas con los modelos base. Esto valida la hipótesis **H4**.

CAPÍTULO 4

Conclusiones

4.1. Conclusiones por hipótesis

Al inicio de este proyecto de tesis se plantearon 4 hipótesis que se han ido validando a lo largo de este trabajo. A continuación se resumen las conclusiones obtenidas para cada una de ellas.

4.1.1. Hipótesis 1: La fusión de técnicas clásicas permite mejorar los resultados obtenidos de manera individual.

Dado el gran éxito que han tenido las técnicas de aprendizaje profundo en visión por computador, el desarrollo de técnicas no profundas ha disminuido estos últimos años. Verificar si esta hipótesis es válida permite saber si todavía es necesario seguir investigando en técnicas no profundas o si por el contrario se puede dar por finalizada esta línea de investigación. La realidad es que el uso de un tipo de técnica u otra depende del problema que se quiera resolver. En el caso de la estimación de pose 6D utilizar técnicas clásicas permite disminuir notablemente el tiempo necesario para generar el modelo: preparación del dataset y entrenamiento de los modelos. Además, los resultados obtenidos son similares a los obtenidos con técnicas de aprendizaje profundo para algunos casos. Por ello, se puede concluir que la hipótesis es válida y que la fusión de técnicas clásicas permite mejorar los resultados obtenidos de manera individual y obtener resultados similares o incluso mejores, en algunos casos, que utilizando métodos de aprendizaje profundo.

4.1.2. Hipótesis 2: Transferencia de aprendizaje de algoritmos de aprendizaje profundo 2D a su versión 3D mejora la precisión obtenida.

Las técnicas de aprendizaje profundo para imágenes 2D han alcanzado un gran nivel de madurez debido al gran éxito que han tenido esta última década. Las técnicas de aprendizaje profundo que tienen como entrada datos 3D todavía no han alcanzado el mismo nivel de madurez. Por ello, se ha querido transferir el conocimiento de los modelos 2D a su versión 3D para ver si se puede obtener una mejora en la precisión obtenida. Los resultados obtenidos demuestran que se puede obtener una precisión similar utilizando datos 3D. Esto es debido a que los datos 3D contienen más información geométrica que los datos 2D, por lo que se puede obtener una mejor estimación de la pose 6D. Además, los datos 3D son más robustos a cambios de iluminación y de oclusión. Aún así, existe mucho trabajo por delante en este ámbito, ya que la representación de datos 3D como nubes de puntos o vóxels no es la óptima para que los modelos de aprendizaje profundo puedan aprender de ellos.

4.1.3. Hipótesis 3: Entrenar algoritmos con datasets sintéticos permite obtener mejores resultados que entrenarlos con sólo datos reales limitados

La generación de datos sintéticos es una técnica muy utilizada en visión por computador para entrenar algoritmos de aprendizaje profundo. Esto es debido a que etiquetar datos reales es muy laborioso y costoso. Por ello, se ha querido comprobar si entrenando algoritmos con datos sintéticos se pueden obtener mejores resultados que entrenándolos con sólo datos reales limitados. Los resultados obtenidos demuestran que con un conjunto de datos sintéticos se puede obtener una precisión similar a la obtenida con un conjunto de datos reales. Esto es debido a que los datos sintéticos generados son cada vez más realistas. Además, en estos últimos años los métodos que lideraban el BOP challenge han obtenido resultados muy satisfactorios entrenando con solo datos sintéticos, llegando incluso a igualar a los métodos que entrenaban con datos reales. De hecho, las herramientas actuales de generación de datos sintéticos permiten generar datos sintéticos de manera muy sencilla y rápida. Esto permite prácticamente integrar la generación de datos sintéticos en el pipeline de entrenamiento de los modelos de aprendizaje profundo, permitiendo entrenar estimadores de objetos nuevos a partir de sólo su modelo 3D.

4.1.4. Hipótesis 4: La combinación de técnicas clásicas y de aprendizaje profundo aumenta la precisión y la generalización

Como se ha comentado con anterioridad, para cada caso concreto hay que evaluar si es más rentable utilizar técnicas clásicas o de aprendizaje profundo. Esto se debe a diversos factores como la cantidad de datos que disponemos, la forma y colores de los objetos que se quieren detectar, el tiempo que se dispone de entrenamiento y de inferencia, etc. Además se ha detectado que las debilidades de uno son las fortalezas del otro, y viceversa. Por ello, se ha querido comprobar si fusionando técnicas de ambos tipos se puede obtener mejores resultados que utilizando cada técnica por separado. De hecho, los resultados obtenidos han podido validar esta hipótesis. En algunos casos se ha obtenido una mejora en la precisión y en otros se ha obtenido una mejora en la generalización. La dificultad de este tipo de técnicas reside en elegir correctamente los parámetros de cada técnica y en la forma de combinarlas. Además, en visión por computador aparecen nuevas técnicas cada poco tiempo. Esto dificulta esta elección aún más. Aun así, gracias a la diversidad de métodos que existen en visión por computador, se puede elegir la combinación de técnicas que mejor se adapte a cada caso concreto.

4.2. Conclusiones generales

En estos más de 4 años de trabajo, se ha explorado el campo de la visión por computadora aplicada a la estimación de pose 6D en robótica y se ha analizado su importancia en el desarrollo de sistemas robóticos precisos y confiables. A través de una revisión exhaustiva de la literatura y la experimentación práctica, se ha llegado a las siguientes conclusiones clave:

- El deep learning ha tomado la delantera en la mayoría de problemas de visión por computador, pero no siempre es la mejor opción. En este trabajo se ha demostrado que la fusión de técnicas clásicas y de aprendizaje profundo puede mejorar los resultados obtenidos de manera individual. Esto es más notorio cuando trabajamos con datos 3D, ya que el deep learning no ha alcanzado el mismo nivel de madurez que en el caso de imágenes 2D. Existe todavía mucho trabajo por hacer en este campo.

- El actual enfoque de la comunidad científica en el ámbito del deep learning para obtener pose 6D de objetos es utilizar modelos que aprenden de entradas de datos 2D y luego hacen una inferencia 3D. Sin embargo, en este trabajo se ha demostrado que se puede obtener una precisión similar utilizando datos 3D. Esto es debido a que los datos 3D contienen más información que los datos 2D, por lo que se puede obtener una mejor estimación de la pose 6D. Además, los datos 3D son más robustos a cambios de iluminación y de oclusión.

- Una línea muy arraigada en visión por computador se centra en poner el foco en el modelo (en su arquitectura o forma de entrenarlo), pero no en los datos. Cuando trabajamos con datos 3D etiquetar esos datos es muy laborioso y costoso. Es en este punto donde entra la generación de datos sintéticos. Gracias a la mejora de las técnicas de renderizado y a la potencia de las GPU, las imágenes generadas son cada vez más realistas. Por ello, estas imágenes se pueden utilizar para entrenar algoritmos de aprendizaje profundo, con los cuales se pueden mejorar los resultados obtenidos con datos reales cuando estos son muy limitados. De hecho en este trabajo se ha demostrado que con un conjunto de datos sintéticos se puede obtener una precisión similar a la obtenida con un conjunto de datos reales.

- A pesar de los avances logrados, todavía existen desafíos significativos en la estimación de pose 6D. La eficiencia computacional y el tiempo de respuesta en entornos en tiempo real son áreas en las que se requiere más investigación. Además, la capacidad de lidiar con objetos deformables, cambios en la geometría del objeto y la estimación precisa en entornos no estructurados siguen siendo desafíos que deben abordarse.

En conclusión, la estimación de pose 6D basada en la visión por computador es un componente esencial en la robótica moderna y tiene un gran potencial para impulsar el desarrollo de robots más inteligentes y versátiles. A medida que se superen los desafíos restantes y se mejoren las técnicas existentes, la estimación de pose 6D seguirá desempeñando un papel vital en aplicaciones robóticas cada vez más complejas y en una amplia gama de campos, desde la fabricación automatizada hasta la navegación autónoma y la asistencia en la atención médica.

Por último, cabe destacar que gracias a realizar la tesis doctoral en el contexto de un centro de investigación, ha sido posible trabajar otros aspectos que no se han podido incluir en este trabajo, como por ejemplo la colaboración con otros investigadores, la participación en proyectos de investigación tanto europeos como a nivel nacional, y trabajar otro tipo de técnicas o contextos como el análisis de imágenes hiperespectrales o la robótica. Todo ello ha contribuido a mi formación como investigador y a mi desarrollo personal y profesional.

4.3. Trabajo futuro

Como trabajo futuro se podría estudiar la posibilidad de utilizar datos sintéticos generados por modelos de difusión. Estos modelos generan cada vez imágenes más realistas. Existen también algunas investigaciones [232, 233, 234] que están utilizando este tipo de modelos para generar datos 3D con resultados prometedores.

Otros posibles trabajos futuros podrían centrarse en mejorar los resultados obtenidos con datos 3D o a una mejor representación de los datos 3D, para que los modelos de aprendizaje profundo puedan aprender de ellos. Por ejemplo, se podría aprender una representación neuronal del espacio tridimensional (Neural Radiance Fields - NeRF [235]). Esta representación permite generar imágenes de objetos 3D desde cualquier punto de vista. Normalmente, se utiliza para generar imágenes de objetos desde diferentes puntos de vista, pero se podría aprovechar el conocimiento aprendido de la representación neuronal como encoding de los datos 3D. Esto permitiría que los modelos de aprendizaje profundo pudieran aprender de los datos 3D de una manera más eficiente.

Cabe destacar que muchos de los modelos y técnicas utilizadas en este trabajo se han desarrollado en el contexto de la robótica, pero que no todos ellos se han probado en un robot real. Por ello, sería interesante aplicar todo este conocimiento en aplicaciones reales donde la complejidad es mayor y donde se pueden encontrar nuevos desafíos. Uno de los posibles trabajos futuros podría ser el integrar la información que se tiene del robot en un caso "eye-in-hand" (la cámara en el brazo del robot) como pueden ser los datos de odometría o de sensores de fuerza para reducir la incertidumbre en la precisión de la estimación de pose 6D o incluso poder refinar la estimación de pose 6D obtenida con técnicas de visión por computador. De esta forma, si se saca una captura desde una posición y se obtiene la pose del objeto, se mueve el robot a una nueva posición, se vuelve a capturar y se detecta el objeto, se puede verificar si ambas poses coinciden para refinar la pose final obtenida. Esto permitiría obtener una estimación de pose 6D más precisa y robusta.

Bibliografía

[1] Heiner Lasi, Peter Fettke, Hans-Georg Kemper, Thomas Feld, and Michael Hoffmann. Industry 4.0. *Business & information systems engineering*, 6(4):239–242, 2014. Ver página 3.

[2] European Commission, Directorate-General for Research, Innovation, Maija Breque, Lars De Nul, and Athanasios Petridis. *Industry 5.0 : towards a sustainable, human-centric and resilient European industry*. Publications Office, 2021. Ver página 3.

[3] Samir Yitzhak Gadre, Mitchell Wortsman, Gabriel Ilharco, Ludwig Schmidt, and Shuran Song. Clip on wheels: Zero-shot object navigation as object localization and exploration. *arXiv preprint arXiv:2203.10421*, 2022. Ver página 4.

[4] Zhengxue Zhou, Leihui Li, Alexander Fürsterling, Hjalte Joshua Durocher, Jesper Mouridsen, and Xuping Zhang. Learning-based object detection and localization for a mobile robot manipulator in sme production. *Robotics and Computer-Integrated Manufacturing*, 73:102229, 2022. Ver página 4.

[5] Guoguang Du, Kai Wang, Shiguo Lian, and Kaiyong Zhao. Vision-based robotic grasping from object localization, object pose estimation to grasp estimation for parallel grippers: a review. *Artificial Intelligence Review*, 54(3):1677–1734, 2021. Ver página 4.

[6] E Shreyas, Manav Hiren Sheth, and Mohana. 3d object detection and tracking methods using deep learning for computer vision applications. In *2021 International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT)*, pages 735–738. IEEE, 2021. Ver página 4.

[7] Zhuoqi Cheng and Thiusius Rajeeth Savarimuthu. A novel robot-assisted electrical impedance scanning system for subsurface object detection. *Measurement Science and Technology*, 32(8):085902, 2021. Ver página 4.

[8] Márton Szemenyei and Vladimir Estivill-Castro. Fully neural object detection solutions for robot soccer. *Neural Computing and Applications*, 34(24):21419–21432, 2022. Ver página 4.

[9] Imran Ahmed, Sadia Din, Gwanggil Jeon, Francesco Piccialli, and Giancarlo Fortino. Towards collaborative robotics in top view surveillance: A framework for multiple object tracking by detection using deep learning. *IEEE/CAA Journal of Automatica Sinica*, 8(7):1253–1270, 2021. Ver página 4.

[10] Jiayao Shan, Sifan Zhou, Zheng Fang, and Yubo Cui. Ptt: Point-track-transformer module for 3d single object tracking in point clouds. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1310–1316. IEEE, 2021. Ver página 4.

[11] Yizhe Wu, Oiwi Parker Jones, Martin Engelcke, and Ingmar Posner. Apex: Unsupervised, object-centric scene segmentation and tracking for robot manipulation. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3375–3382. IEEE, 2021. Ver página 4.

[12] Andrea Bonci, Pangcheng David Cen Cheng, Marina Indri, Giacomo Nabissi, and Fiorella Sibona. Human-robot perception in industrial environments: A survey. *Sensors*, 21(5):1571, 2021. Ver página 4.

[13] Janis Arents, Valters Abolins, Janis Judvaitis, Oskars Vismanis, Aly Oraby, and Kaspars Ozols. Human–robot collaboration trends and safety aspects: A systematic review. *Journal of Sensor and Actuator Networks*, 10(3):48, 2021. Ver página 4.

[14] Óscar G Hernández, Vicente Morell, José L Ramon, and Carlos A Jara. Human pose detection for robotic-assisted and rehabilitation environments. *Applied Sciences*, 11(9):4183, 2021. Ver página 4.

[15] Matheus Zorawski Silva, Thadeu Brito, José L Lima, and Manuel F Silva. Industrial robotic arm in machining process aimed to 3d objects reconstruction. In *2021 22nd IEEE International Conference on Industrial Technology (ICIT)*, volume 1, pages 1100–1105. IEEE, 2021. Ver página 4.

[16] Mohamed Tahoun, Omar Tahri, Juan Antonio Corrales Ramón, and Youcef Mezouar. Visual-tactile fusion for 3d objects reconstruction from a single depth view and a single gripper touch for robotics tasks. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6786–6793. IEEE, 2021. Ver página 4.

[17] William Agnew, Christopher Xie, Aaron Walsman, Octavian Murad, Yubo Wang, Pedro Domingos, and Siddhartha Srinivasa. Amodal 3d reconstruction for robotic manipulation via stability and connectivity. In *Conference on Robot Learning*, pages 1498–1508. PMLR, 2021. Ver página 4.

[18] Aleksandar Jokić, Milica Petrović, and Zoran Miljković. Semantic segmentation based stereo visual servoing of nonholonomic mobile robot in intelligent manufacturing environment. *Expert Systems with Applications*, 190:116203, 2022. Ver página 4.

[19] Jiehao Li, Yingpeng Dai, Junzheng Wang, Xiaohang Su, and Ruijun Ma. Towards broad learning networks on unmanned mobile robot for semantic segmentation. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 9228–9234. IEEE, 2022. Ver página 4.

[20] Rubén Rodríguez Abril. Segmentación panóptica. https://lamaquinaoraculo.com/computacion/segmentacion-panoptica/. Accessed: 2023-02-13. Ver página 21.

[21] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Ep n p: An accurate o (n) solution to the p n p problem. *International journal of computer vision*, 81:155–166, 2009. Ver página 22.

[22] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. Ieee, 2005. Ver página 24.

[23] Di Huang, Chao Zhu, Yunhong Wang, and Liming Chen. Hsog: a novel local image descriptor based on histograms of the second-order gradients. *IEEE Transactions on Image Processing*, 23(11):4680–4695, 2014. Ver página 24.

[24] Aude Oliva and Antonio Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International journal of computer vision*, 42(3):145–175, 2001. Ver página 24.

[25] Bangalore S Manjunath, J-R Ohm, Vinod V Vasudevan, and Akio Yamada. Color and texture descriptors. *IEEE Transactions on circuits and systems for video technology*, 11(6):703–715, 2001. Ver página 24.

[26] Thomas Sikora. The mpeg-7 visual standard for content description-an overview. *IEEE Transactions on circuits and systems for video technology*, 11(6):696–702, 2001. Ver página 24.

[27] Antonio Fernández, Marcos X Álvarez, and Francesco Bianconi. Texture description through histograms of equivalent patterns. *Journal of mathematical imaging and vision*, 45(1):76–102, 2013. Ver página 24.

[28] Timo Ojala, Matti Pietikainen, and Topi Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on pattern analysis and machine intelligence*, 24(7):971–987, 2002. Ver página 24.

[29] Francisco J Madrid-Cuevas, R Medina Carnicer, M Prieto Villegas, NL García, and A Carmona Poyato. Simplified texture unit: A˜ new descriptor of the local texture in gray-level images. In *Iberian Conference on Pattern Recognition and Image Analysis*, pages 470–477. Springer, 2003. Ver página 24.

[30] Bing Xu, Peng Gong, Edmund Seto, and Robert Spear. Comparison of gray-level reduction and different texture spectrum encoding methods for land-use classification using a panchromatic ikonos image. *Photogrammetric Engineering & Remote Sensing*, 69(5):529–536, 2003. Ver página 26.

[31] Zhenhua Guo, Lei Zhang, and David Zhang. A completed modeling of local binary pattern operator for texture classification. *IEEE transactions on image processing*, 19(6):1657–1663, 2010. Ver página 26.

[32] Xiaoyang Tan and Bill Triggs. Enhanced local texture feature sets for face recognition under difficult lighting conditions. *IEEE transactions on image processing*, 19(6):1635–1650, 2010. Ver página 26.

[33] Jing-Hua Yuan, Hao-Dong Zhu, Yong Gan, and Li Shang. Enhanced local ternary pattern for texture classification. In *International conference on intelligent computing*, pages 443–448. Springer, 2014. Ver página 26.

[34] Subrahmanyam Murala, RP Maheshwari, and R Balasubramanian. Local tetra patterns: a new feature descriptor for content-based image retrieval. *IEEE transactions on image processing*, 21(5):2874–2886, 2012. Ver página 26.

[35] Antonio Fernández, Marcos X Álvarez, and Francesco Bianconi. Image classification with binary gradient contours. *Optics and Lasers in Engineering*, 49(9-10):1177–1184, 2011. Ver página 26.

[36] Wenchao Zhang, Shiguang Shan, Wen Gao, Xilin Chen, and Hongming Zhang. Local gabor binary pattern histogram sequence (lgbphs): A novel non-statistical model for face representation and recognition. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 1, pages 786–791. IEEE, 2005. Ver página 26.

[37] Sibt Ul Hussain, Thibault Napoléon, and Fréderic Jurie. Face recognition using local quantized patterns. In *British machive vision conference*, pages 11–pages, 2012. Ver página 26.

[38] Robert M Haralick, Karthikeyan Shanmugam, and Its' Hak Dinstein. Textural features for image classification. *IEEE Transactions on systems, man, and cybernetics*, (6):610–621, 1973. Ver página 26.

[39] Jie Chen, Shiguang Shan, Chu He, Guoying Zhao, Matti Pietikäinen, Xilin Chen, and Wen Gao. Wld: A robust local image descriptor. *IEEE transactions on pattern analysis and machine intelligence*, 32(9):1705–1720, 2009. Ver página 26.

[40] Gustav Theodor Fechner. *Elemente der psychophysik*, volume 2. Breitkopf u. Härtel, 1860. Ver página 26.

[41] William T. Freeman and Edward H. Adelson. The design and use of steerable filters. *IEEE Transactions on Pattern analysis and machine intelligence*, 13(9):891–906, 1991. Ver página 27.

[42] Irwin Sobel and Gary Feldman. A 3x3 isotropic gradient operator for image processing. *a talk at the Stanford Artificial Project in*, pages 271–272, 1968. Ver página 27.

[43] John Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986. Ver página 27.

[44] Ramesh Jain, Rangachar Kasturi, and Brian G. Schunck. *Machine vision*, volume 5. McGraw-hill New York, 1995. Ver página 27.

[45] Emanuele Trucco and Alessandro Verri. *Introductory techniques for 3-D computer vision*, volume 201. Prentice Hall Englewood Cliffs, 1998. Ver página 27.

[46] Hans Peter Moravec. *Obstacle avoidance and navigation in the real world by a seeing robot rover*. Stanford University, 1980. Ver página 28.

[47] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244, 1988. Ver página 28.

[48] Jianbo Shi and Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994. Ver página 28.

[49] Stephen M Smith and J Michael Brady. Susan—a new approach to low level image processing. *International journal of computer vision*, 23(1):45–78, 1997. Ver página 28.

[50] Edward Rosten and Tom Drummond. Fusing points and lines for high performance tracking. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 2, pages 1508–1515. Ieee, 2005. Ver página 28.

[51] Elmar Mair, Gregory D Hager, Darius Burschka, Michael Suppa, and Gerhard Hirzinger. Adaptive and generic corner detection based on the accelerated segment test. In *European conference on Computer vision*, pages 183–196. Springer, 2010. Ver página 28.

[52] Stefan Leutenegger, Margarita Chli, and Roland Y Siegwart. Brisk: Binary robust invariant scalable keypoints. In *2011 International conference on computer vision*, pages 2548–2555. Ieee, 2011. Ver páginas 28, 29.

[53] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 International conference on computer vision*, pages 2564–2571. Ieee, 2011. Ver páginas 28, 29.

[54] James L Crowley and Alice C Parker. A representation for shape based on peaks and ridges in the difference of low-pass transform. *IEEE transactions on pattern analysis and machine intelligence*, (2):156–170, 1984. Ver página 28.

[55] Tony Lindeberg. Detecting salient blob-like image structures and their scales with a scale-space primal sketch: A method for focus-of-attention. *International Journal of Computer Vision*, 11(3):283–318, 1993. Ver página 28.

[56] D.G. Lowe. Object recognition from local scale-invariant features. In *Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157 vol.2, 1999. Ver páginas 28, 29.

[57] Tony Lindeberg. Scale-space theory: A basic tool for analyzing structures at different scales. *Journal of applied statistics*, 21(1-2):225–270, 1994. Ver página 28.

[58] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006. Ver páginas 28, 29.

[59] Motilal Agrawal, Kurt Konolige, and Morten Rufus Blas. Censure: Center surround extremas for realtime feature detection and matching. In *European conference on computer vision*, pages 102–115. Springer, 2008. Ver página 29.

[60] Krystian Mikolajczyk and Cordelia Schmid. An affine invariant interest point detector. In *European conference on computer vision*, pages 128–142. Springer, 2002. Ver página 29.

[61] Krystian Mikolajczyk and Cordelia Schmid. Scale & affine invariant interest point detectors. *International journal of computer vision*, 60(1):63–86, 2004. Ver página 29.

[62] Jiri Matas, Ondrej Chum, Martin Urban, and Tomás Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and vision computing*, 22(10):761–767, 2004. Ver página 29.

[63] Tai Sing Lee. Image representation using 2d gabor wavelets. *IEEE Transactions on pattern analysis and machine intelligence*, 18(10):959–971, 1996. Ver página 29.

[64] John Illingworth and Josef Kittler. A survey of the hough transform. *Computer vision, graphics, and image processing*, 44(1):87–116, 1988. Ver página 29.

[65] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In *European conference on computer vision*, pages 778–792. Springer, 2010. Ver página 29.

[66] Yan Ke and Rahul Sukthankar. Pca-sift: A more distinctive representation for local image descriptors. In *2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 2, pages II–II. IEEE, 2004. Ver página 29.

[67] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. A sparse texture representation using local affine regions. *IEEE transactions on pattern analysis and machine intelligence*, 27(8):1265–1278, 2005. Ver página 30.

[68] Bin Fan, Fuchao Wu, and Zhanyi Hu. Rotationally invariant descriptors using intensity order pooling. *IEEE transactions on pattern analysis and machine intelligence*, 34(10):2031–2045, 2011. Ver página 30.

[69] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *IEEE transactions on pattern analysis and machine intelligence*, 27(10):1615–1630, 2005. Ver página 30.

[70] Engin Tola, Vincent Lepetit, and Pascal Fua. Daisy: An efficient dense descriptor applied to wide-baseline stereo. *IEEE transactions on pattern analysis and machine intelligence*, 32(5):815–830, 2009. Ver página 30.

[71] Zhenhua Wang, Bin Fan, and Fuchao Wu. Local intensity order pattern for feature description. In *2011 International Conference on Computer Vision*, pages 603–610. IEEE, 2011. Ver página 30.

[72] Matthew Brown, Richard Szeliski, and Simon Winder. Multi-image matching using multi-scale oriented patches. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 510–517. IEEE, 2005. Ver página 30.

[73] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *IEEE transactions on pattern analysis and machine intelligence*, 24(4):509–522, 2002. Ver páginas 30, 35.

[74] Eli Shechtman and Michal Irani. Matching local self-similarities across images and videos. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007. Ver página 30.

[75] Jingneng Liu, Guihua Zeng, and Jianping Fan. Fast local self-similarity for describing interest regions. *Pattern Recognition Letters*, 33(9):1224–1235, 2012. Ver página 30.

[76] Alexandre Alahi, Raphael Ortiz, and Pierre Vandergheynst. Freak: Fast retina keypoint. In *IEEE conference on computer vision and pattern recognition*, pages 510–517. Ieee, 2012. Ver página 30.

[77] Li Fei-Fei and Pietro Perona. A bayesian hierarchical model for learning natural scene categories. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 524–531. IEEE, 2005. Ver página 31.

[78] Ajmal S Mian, Mohammed Bennamoun, and Robyn Owens. Three-dimensional model-based object recognition and segmentation in cluttered scenes. *IEEE transactions on pattern analysis and machine intelligence*, 28(10):1584–1601, 2006. Ver página 31.

[79] Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, 2(11):559–572, 1901. Ver página 31.

[80] Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417, 1933. Ver página 31.

[81] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000. Ver página 31.

[82] Aapo Hyvärinen and Erkki Oja. Independent component analysis: algorithms and applications. *Neural networks*, 13(4-5):411–430, 2000. Ver página 31.

[83] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995. Ver página 32.

[84] Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *The Journal of Machine Learning Research*, 5:101–141, 2004. Ver página 34.

[85] Stefan Knerr, Léon Personnaz, and Gérard Dreyfus. Single-layer learning revisited: a stepwise procedure for building and training a neural network. In *Neurocomputing: algorithms, architectures and applications*, pages 41–50. Springer, 1990. Ver página 34.

[86] Leo Breiman. Bagging predictors. *Machine learning*, 24:123–140, 1996. Ver página 34.

[87] Robert E Schapire. A brief introduction to boosting. In *Ijcai*, volume 99, pages 1401–1406, 1999. Ver página 34.

[88] David H Wolpert. Stacked generalization. *Neural networks*, 5(2):241–259, 1992. Ver página 34.

[89] Artittayapron Rojarath, Wararat Songpan, and Chakrit Pong-inwong. Improved ensemble learning for classification techniques based on majority voting. In *2016 7th IEEE international conference on software engineering and service science (ICSESS)*, pages 107–110. IEEE, 2016. Ver página 34.

[90] Eric Brachmann, Alexander Krull, Frank Michel, Stefan Gumhold, Jamie Shotton, and Carsten Rother. Learning 6d object pose estimation using 3d object coordinates. In *Computer Vision – ECCV 2014*, pages 536–551. Springer, 2014. Ver página 35.

[91] Alykhan Tejani, Danhang Tang, Rigas Kouskouridas, and Tae-Kyun Kim. Latent-class hough forests for 3d object detection and pose estimation. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland*, pages 462–477. Springer, 2014. Ver páginas 35, 46.

[92] Ujwal Bonde, Vijay Badrinarayanan, and Roberto Cipolla. Robust instance recognition in presence of occlusion and clutter. In *Computer Vision – ECCV 2014*, pages 520–535. Springer, 2014. Ver página 35.

[93] Andrew E Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on pattern analysis and machine intelligence*, 21(5):433–449, 1999. Ver página 35.

[94] Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow, and Michael Beetz. Persistent point feature histograms for 3d point clouds. In *10th International Conference on Intelligent Autonomous Systems (IAS-10), Baden-Baden, Germany*, pages 119–128, 2008. Ver página 35.

[95] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In *IEEE international conference on robotics and automation*, pages 3212–3217. IEEE, 2009. Ver página 35.

[96] Zoltan-Csaba Marton, Dejan Pangercic, Nico Blodow, Jonathan Kleinehellefort, and Michael Beetz. General 3d modelling of novel objects from a single view. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3700–3705. IEEE, 2010. Ver página 35.

[97] Andrea Frome, Daniel Huber, Ravi Kolluri, Thomas Bülow, and Jitendra Malik. Recognizing objects in range data using regional point descriptors. In *Computer Vision-ECCV 2004: 8th European Conference on Computer Vision, Prague, Czech Republic*, pages 224–237. Springer, 2004. Ver página 35.

[98] Federico Tombari, Samuele Salti, and Luigi Di Stefano. Unique shape context for 3d data description. In *ACM workshop on 3D object retrieval*, pages 57–62, 2010. Ver página 36.

[99] Samuele Salti, Federico Tombari, and Luigi Di Stefano. Shot: Unique signatures of histograms for surface and texture description. *Computer Vision and Image Understanding*, 125:251–264, 2014. Ver página 36.

[100] Bertram Drost, Markus Ulrich, Nassir Navab, and Slobodan Ilic. Model globally, match locally: Efficient and robust 3d object recognition. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 998–1005. Ieee, 2010. Ver página 36.

[101] Paul J. Besl and Neil D. McKay. Method for registration of 3-D shapes. In Paul S. Schenker, editor, *Sensor Fusion IV: Control Paradigms and Data Structures*, volume 1611, pages 586 – 606. International Society for Optics and Photonics, SPIE, 1992. Ver página 36.

[102] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011. IEEE. Ver página 36.

[103] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International journal of computer vision*, 1(4):321–331, 1988. Ver página 36.

[104] Daniel P Huttenlocher, Gregory A. Klanderman, and William J Rucklidge. Comparing images using the hausdorff distance. *IEEE Transactions on pattern analysis and machine intelligence*, 15(9):850–863, 1993. Ver página 36.

[105] Carsten Steger. Similarity measures for occlusion, clutter, and illumination invariant object recognition. In *Pattern Recognition: 23rd DAGM Symposium Munich, Germany*, pages 148–154. Springer, 2001. Ver página 36.

[106] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Pascal Fua, and Nassir Navab. Dominant orientation templates for real-time detection of texture-less objects. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2257–2264. IEEE, 2010. Ver página 36.

[107] Stefan Hinterstoisser, Cedric Cagniart, Slobodan Ilic, Peter Sturm, Nassir Navab, Pascal Fua, and Vincent Lepetit. Gradient response maps for real-time detection of textureless objects. *IEEE transactions on pattern analysis and machine intelligence*, 34(5):876–888, 2011. Ver página 36.

[108] Stefan Hinterstoisser, Stefan Holzer, Cedric Cagniart, Slobodan Ilic, Kurt Konolige, Nassir Navab, and Vincent Lepetit. Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. In *International conference on computer vision*, pages 858–865. IEEE, 2011. Ver página 36.

[109] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary Bradski, Kurt Konolige, and Nassir Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In *Asian conference on computer vision*, pages 548–562. Springer, 2013. Ver página 36.

[110] Reyes Rios-Cabrera and Tinne Tuytelaars. Discriminatively trained templates for 3d object detection: A real time scalable approach. In *IEEE international conference on computer vision*, pages 2048–2055, 2013. Ver página 36.

[111] Paul Wohlhart and Vincent Lepetit. Learning descriptors for object recognition and 3d pose estimation. In *IEEE conference on computer vision and pattern recognition*, pages 3109–3118, 2015. Ver página 36.

[112] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5:115–133, 1943. Ver página 36.

[113] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958. Ver página 36.

[114] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. Ver páginas 37, 54.

[115] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015. Ver página 37.

[116] Andreas Kamilaris and Francesc X Prenafeta-Boldú. Deep learning in agriculture: A survey. *Computers and electronics in agriculture*, 147:70–90, 2018. Ver página 37.

[117] Juan Izquierdo Tomás. Reliability engineering advancements to enhance fleet asset management in service-oriented business models. 2020. Ver página 37.

[118] Francesco Piccialli, Vittorio Di Somma, Fabio Giampaolo, Salvatore Cuomo, and Giancarlo Fortino. A survey on deep learning in medicine: Why, how and when? *Information Fusion*, 66:111–137, 2021. Ver página 37.

[119] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. Ver páginas 37, 39, and 54.

[120] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015. Ver páginas 37, 39.

[121] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016. Ver páginas 37, 39.

[122] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. Ver páginas 37, 39, and 54.

[123] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first AAAI conference on artificial intelligence*, 2017. Ver páginas 37, 39, and 54.

[124] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017. Ver páginas 37, 39.

[125] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018. Ver páginas 37, 39.

[126] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019. Ver páginas 37, 39, and 54.

[127] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. Ver páginas 37, 42.

[128] Tomáš Hodaň, Frank Michel, Eric Brachmann, Wadim Kehl, Anders Glent Buch, Dirk Kraft, Bertram Drost, Joel Vidal, Stephan Ihrke, Xenophon Zabulis, Caner Sahin, Fabian Manhardt, Federico Tombari, Tae-Kyun Kim, Jiří Matas, and Carsten Rother. BOP: Benchmark for 6D object pose estimation. *European Conference on Computer Vision (ECCV)*, 2018. Ver páginas 37, 46, and 47.

[129] Mahdi Rad and Vincent Lepetit. Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth. In *IEEE international conference on computer vision*, pages 3828–3836, 2017. Ver página 37.

[130] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199*, 2017. Ver páginas 37, 46.

[131] Yann Labbé, Justin Carpentier, Mathieu Aubry, and Josef Sivic. Cosypose: Consistent multi-view multi-object 6d pose estimation. In *European Conference on Computer Vision*, pages 574–591. Springer, 2020. Ver páginas 37, 58.

[132] Sida Peng, Yuan Liu, Qixing Huang, Xiaowei Zhou, and Hujun Bao. Pvnet: Pixel-wise voting network for 6dof pose estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4561–4570, 2019. Ver página 37.

[133] Sergey Zakharov, Ivan Shugurov, and Slobodan Ilic. Dpod: 6d pose object detector and refiner. In *IEEE/CVF international conference on computer vision*, pages 1941–1950, 2019. Ver página 37.

[134] Giorgia Pitteri, Slobodan Ilic, and Vincent Lepetit. Cornet: generic 3d corners for 6d pose estimation of new objects without retraining. In *IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019. Ver página 37.

[135] Yisheng He, Wei Sun, Haibin Huang, Jianran Liu, Haoqiang Fan, and Jian Sun. Pvn3d: A deep point-wise 3d keypoints voting network for 6dof pose estimation. In *IEEE/CVF conference on computer vision and pattern recognition*, pages 11632–11641, 2020. Ver páginas 37, 58.

[136] Yisheng He, Haibin Huang, Haoqiang Fan, Qifeng Chen, and Jian Sun. Ffb6d: A full flow bidirectional fusion network for 6d pose estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3003–3013, 2021. Ver páginas 37, 58.

[137] Dong-Hyun Lee et al. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, page 896, 2013. Ver página 37.

[138] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. Self-training with noisy student improves imagenet classification. In *IEEE/CVF conference on computer vision and pattern recognition*, pages 10687–10698, 2020. Ver página 37.

[139] Hieu Pham, Zihang Dai, Qizhe Xie, and Quoc V Le. Meta pseudo labels. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11557–11568, 2021. Ver página 37.

[140] Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. Perceiver: General perception with iterative attention. In *International conference on machine learning*, pages 4651–4664. PMLR, 2021. Ver página 37.

[141] Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, et al. Perceiver io: A general architecture for structured inputs & outputs. *arXiv preprint arXiv:2107.14795*, 2021. Ver página 37.

[142] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015. Ver página 38.

[143] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016. Ver página 38.

[144] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *27th international conference on machine learning (ICML-10)*, pages 807–814, 2010. Ver página 38.

[145] Andrew L Maas, Awni Y Hannun, Andrew Y Ng, et al. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3. Atlanta, Georgia, USA, 2013. Ver página 38.

[146] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015. Ver página 38.

[147] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. *Advances in neural information processing systems*, 30, 2017. Ver página 38.

[148] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015. Ver página 38.

[149] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006. Ver página 38.

[150] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014. Ver página 39.

[151] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. Ver página 40.

[152] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014. Ver página 40.

[153] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006. Ver página 40.

[154] Andrew Ng et al. Sparse autoencoder. *CS294A Lecture notes*, 72(2011):1–19, 2011. Ver página 41.

[155] Ian Goodfellow, Honglak Lee, Quoc Le, Andrew Saxe, and Andrew Ng. Measuring invariances in deep networks. *Advances in neural information processing systems*, 22, 2009. Ver página 41.

[156] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *25th international conference on Machine learning*, pages 1096–1103, 2008. Ver página 41.

[157] Shixiang Gu and Luca Rigazio. Towards deep neural network architectures robust to adversarial examples. *arXiv preprint arXiv:1412.5068*, 2014. Ver página 41.

[158] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *28th international confe-*

rence on international conference on machine learning, pages 833–840, 2011. Ver página 41.

[159] Salah Rifai, Xavier Muller, Xavier Glorot, Grégoire Mesnil, Yoshua Bengio, and Pascal Vincent. Learning invariant features through local space contraction. *arXiv preprint arXiv:1104.4153*, 2011. Ver página 41.

[160] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. Ver página 41.

[161] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. Ver página 41.

[162] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *IEEE conference on computer vision and pattern recognition*, pages 2536–2544, 2016. Ver página 41.

[163] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020. Ver página 41.

[164] Chao Yang, Xin Lu, Zhe Lin, Eli Shechtman, Oliver Wang, and Hao Li. High-resolution image inpainting using multi-scale neural patch synthesis. In *IEEE conference on computer vision and pattern recognition*, pages 6721–6729, 2017. Ver página 41.

[165] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. Globally and locally consistent image completion. *ACM Transactions on Graphics (ToG)*, 36(4):1–14, 2017. Ver página 41.

[166] Guilin Liu, Fitsum A Reda, Kevin J Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. Image inpainting for irregular holes using partial convolutions. In *European conference on computer vision (ECCV)*, pages 85–100, 2018. Ver página 41.

[167] Jacob Varley, Chad DeChant, Adam Richardson, Joaquín Ruales, and Peter Allen. Shape completion enabled robotic grasping. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 2442–2447. IEEE, 2017. Ver página 41.

[168] Weiyue Wang, Qiangui Huang, Suya You, Chao Yang, and Ulrich Neumann. Shape inpainting using 3d generative adversarial network and recurrent convolutional networks. In *IEEE international conference on computer vision*, pages 2298–2306, 2017. Ver página 41.

[169] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014. Ver página 41.

[170] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. Ver páginas 41, 42.

[171] Diksha Khurana, Aditya Koli, Kiran Khatter, and Sukhdev Singh. Natural language processing: State of the art, current trends and challenges. *Multimedia tools and applications*, 82(3):3713–3744, 2023. Ver página 42.

[172] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. Ver página 42.

[173] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018. Ver página 42.

[174] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019. Ver página 42.

[175] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020. Ver página 42.

[176] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32, 2019. Ver página 42.

[177] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *International conference on machine learning*, pages 4055–4064. PMLR, 2018. Ver página 42.

[178] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK*, pages 213–229. Springer, 2020. Ver página 42.

[179] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015. Ver página 42.

[180] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019. Ver página 42.

[181] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020. Ver páginas 42, 44.

[182] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR, 2021. Ver página 44.

[183] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *arXiv preprint arXiv:2204.03458*, 2022. Ver página 44.

[184] Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. Diffusion-lm improves controllable text generation. *Advances in Neural Information Processing Systems*, 35:4328–4343, 2022. Ver página 44.

[185] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*, 2021. Ver página 44.

[186] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR, 2021. Ver página 44.

[187] David Holz. Midjourney. Website, 2022. Ver página 44.

[188] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022. Ver página 44.

[189] Lorien Y Pratt, Jack Mostow, Candace A Kamm, Ace A Kamm, et al. Direct transfer of learned information among neural networks. In *Aaai*, volume 91, pages 584–589, 1991. Ver página 44.

[190] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. A survey on deep transfer learning. In *Artificial Neural Networks and Machine Learning–ICANN 2018: 27th International Conference on Artificial Neural Networks, Rhodes, Greece*, pages 270–279. Springer, 2018. Ver página 44.

[191] Xiaobo Liu, Zhentao Liu, Guangjun Wang, Zhihua Cai, and Harry Zhang. Ensemble transfer learning algorithm. *Ieee Access*, 6:2389–2396, 2017. Ver página 44.

[192] Sinno Jialin Pan, Ivor W Tsang, James T Kwok, and Qiang Yang. Domain adaptation via transfer component analysis. *IEEE transactions on neural networks*, 22(2):199–210, 2010. Ver página 45.

[193] Jing Zhang, Wanqing Li, and Philip Ogunbona. Joint geometrical and statistical alignment for visual domain adaptation. In *IEEE conference on computer vision and pattern recognition*, pages 1859–1867, 2017. Ver página 45.

[194] Jui-Ting Huang, Jinyu Li, Dong Yu, Li Deng, and Yifan Gong. Cross-language knowledge transfer using multilingual deep neural network with shared hidden layers. In *IEEE international conference on acoustics, speech and signal processing*, pages 7304–7308. IEEE, 2013. Ver página 45.

[195] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary Bradski, Kurt Konolige, and Nassir Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In *Asian conference on computer vision*, pages 548–562. Springer, 2012. Ver página 46.

[196] Eric Brachmann, Alexander Krull, Frank Michel, Stefan Gumhold, Jamie Shotton, and Carsten Rother. Learning 6d object pose estimation using 3d object coordinates. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 536–551, Cham, 2014. Springer International Publishing. Ver página 46.

[197] Tomáš Hodaň, Pavel Haluza, Štěpán Obdržálek, Jiří Matas, Manolis Lourakis, and Xenophon Zabulis. T-LESS: An RGB-D dataset for 6D pose estimation of texture-less objects. *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2017. Ver página 46.

[198] Bertram Drost, Markus Ulrich, Paul Bergmann, Philipp Hartinger, and Carsten Steger. Introducing mvtec itodd-a dataset for 3d object recognition in industry. In *IEEE international conference on computer vision workshops*, pages 2200–2208, 2017. Ver página 46.

[199] Roman Kaskman, Sergey Zakharov, Ivan Shugurov, and Slobodan Ilic. Homebreweddb: Rgb-d dataset for 6d pose estimation of 3d objects. In *IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019. Ver página 46.

[200] Berk Calli, Arjun Singh, Aaron Walsman, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. The ycb object and model set: Towards common benchmarks for manipulation research. In *2015 international conference on advanced robotics (ICAR)*, pages 510–517. IEEE, 2015. Ver página 46.

[201] Colin Rennie, Rahul Shome, Kostas E Bekris, and Alberto F De Souza. A dataset for improved rgbd-based object detection and pose estimation for warehouse pick-and-place. *IEEE Robotics and Automation Letters*, 1(2):1179–1185, 2016. Ver página 46.

[202] Andreas Doumanoglou, Rigas Kouskouridas, Sotiris Malassiotis, and Tae-Kyun Kim. Recovering 6d object pose and predicting next-best-view in the crowd. In *IEEE conference on computer vision and pattern recognition*, pages 3583–3592, 2016. Ver página 46.

[203] Tomáš Hodaň, Jiří Matas, and Štěpán Obdržálek. On evaluation of 6d object pose estimation. In *Computer Vision–ECCV 2016*, pages 606–619. Springer, 2016. Ver página 46.

[204] Eric Brachmann, Frank Michel, Alexander Krull, Michael Ying Yang, Stefan Gumhold, and Carsten Rother. Uncertainty-driven 6d pose estimation of objects and scenes from a single rgb image. In *IEEE conference on computer vision and pattern recognition*, pages 3364–3372, 2016. Ver página 46.

[205] Celso M de Melo, Antonio Torralba, Leonidas Guibas, James DiCarlo, Rama Chellappa, and Jessica Hodgins. Next-generation deep learning based on simulators and synthetic data. *Trends in cognitive sciences*, 2021. Ver página 46.

[206] John K Haas. A history of the unity game engine. 2014. Ver página 46.

[207] Epic Games. Unreal engine. Ver páginas 46, 55.

[208] Thang To, Jonathan Tremblay, Duncan McKay, Yukie Yamaguchi, Kirby Leung, Adrian Balanon, Jia Cheng, William Hodge, and Stan Birchfield. NDDS: NVIDIA deep learning dataset synthesizer, 2018. https://github.com/NVIDIA/Dataset_Synthesizer. Ver páginas 46, 55.

[209] Unreal gt. https://unrealgt.github.io/. Accessed: 2022-12-20. Ver página 46.

[210] James Fort Anthony Navarro. Supercharge your computer vision models with synthetic datasets built by unity. https://blog.unity.com/technology/supercharge-your-computer-vision-models-with-synthetic-datasets-built-by-unity, 2021. Accessed: 2023-01-24. Ver página 46.

[211] Nvidia. Nvidia omniverse. https://www.nvidia.com/es-es/omniverse/, 2021. Accessed: 2023-01-26. Ver página 47.

[212] Nvidia. Nvidia isaac sim. https://developer.nvidia.com/isaac-sim, 2021. Accessed: 2023-01-26. Ver página 47.

[213] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. Ver páginas 47, 56.

[214] Maximilian Denninger, Martin Sundermeyer, Dominik Winkelbauer, Youssef Zidan, Dmitry Olefir, Mohamad Elbadrawy, Ahsan Lodhi, and Harinandan Katam. Blenderproc. *arXiv preprint arXiv:1911.01911*, 2019. Ver páginas 47, 56.

[215] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015. Ver página 47.

[216] Ibon Merino, Jon Azpiazu, Anthony Remazeilles, and Basilio Sierra. 2d features-based detector and descriptor selection system for hierarchical recognition of industrial parts. *international Journal of Artificial Intelligence and Applications (IJAIA)*, 10:1–13, 2019. Ver páginas 49, 51, and 52.

[217] Ibon Merino, Jon Azpiazu, Anthony Remazeilles, and Basilio Sierra. 2d image features detector and descriptor selection expert system. *arXiv preprint arXiv:2006.02933*, 2020. Ver páginas 49, 51, and 52.

[218] Ibon Merino, Jon Azpiazu, Anthony Remazeilles, and Basilio Sierra. Histogram-based descriptor subset selection for visual recognition of industrial parts. *Applied Sciences*, 10(11):3701, 2020. Ver páginas 50, 52.

[219] Jose Luis Outón, Ibon Merino, Iván Villaverde, Aitor Ibarguren, Héctor Herrero, Paul Daelman, and Basilio Sierra. A real application of an autonomous industrial mobile manipulator within industrial context. *Electronics*, 10(11):1276, 2021. Ver página 50.

[220] Ibon Merino, Jon Azpiazu, Anthony Remazeilles, and Basilio Sierra. 3d convolutional neural networks initialized from pretrained 2d convolutional neural networks for classification of industrial parts. *Sensors*, 21(4):1078, 2021. Ver páginas 50, 52, 53, and 55.

[221] Ibon Merino, Jon Azpiazu, Anthony Remazeilles, and Basilio Sierra. Ensemble of 6 dof pose estimation from state-of-the-art deep methods. *Neurocomputing*, page 126270, 2023. Ver páginas 50, 56, and 58.

[222] Mervyn Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the royal statistical society: Series B (Methodological)*, 36(2):111–133, 1974. Ver página 51.

[223] Sudhir Varma and Richard Simon. Bias in error estimation when using cross-validation for model selection. *BMC bioinformatics*, 7(1):1–8, 2006. Ver página 51.

[224] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017. Ver página 52.

[225] Iaroslav Melekhov, Juho Kannala, and Esa Rahtu. Siamese network features for image matching. In *2016 23rd international conference on pattern recognition (ICPR)*, pages 378–383. IEEE, 2016. Ver página 52.

[226] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. Ver página 54.

[227] Kentaro Wada. labelme: Image polygonal annotation with python. https://github.com/wkentaro/labelme, 2018. Ver página 55.

[228] Boris Sekachev, Nikita Manovich, Maxim Zhiltsov, Andrey Zhavoronkov, Dmitry Kalinin, Ben Hoff, TOsmanov, Dmitry Kruchinin, Artyom Zankevich, DmitriySidnev, Maksim Markelov, Johannes222, Mathis Chenuet, a andre, telenachos, Aleksandr Melnikov, Jijoong Kim, Liron Ilouz, Nikita Glazov, Priya4607, Rush Tehrani, Seungwon Jeong, Vladimir Skubriev, Sebastian Yonekura, vugia truong, zliang7, lizhming, and Tritin Truong. opencv/cvat: v1.1.0, August 2020. Ver página 55.

[229] Maxim Tkachenko, Mikhail Malyuk, Andrey Holmanyuk, and Nikolai Liubimov. Label Studio: Data labeling software, 2020-2022. Open source software available from https://github.com/heartexlabs/label-studio. Ver página 55.

[230] Cloudcompare. https://www.danielgm.net/cc/. Accedido: 21-12-2022. Ver página 55.

[231] Tomáš Hodaň, Martin Sundermeyer, Bertram Drost, Yann Labbé, Eric Brachmann, Frank Michel, Carsten Rother, and Jiří Matas. BOP challenge 2020 on 6D object localization. *European Conference on Computer Vision Workshops (ECCVW)*, 2020. Ver página 56.

[232] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv*, 2022. Ver página 64.

[233] Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. Magic3d: High-resolution text-to-3d content creation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. Ver página 64.

[234] Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot one image to 3d object, 2023. Ver página 64.

[235] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, dec 2021. Ver página 65.

Parte II

Publicaciones obtenidas

CAPÍTULO 5

2d Image Features Detector and Descriptor Selection Expert System

Título:	2d Image Features Detector and Descriptor Selection Expert System
Autores:	I. Merino, J. Azpiazu, A. Remazeilles, B. Sierra
Conferencia:	Computer Science & Information Technology (CS & IT)
Editor:	AIRCC Publishing Corporation
DOI:	10.5121/csit.2019.91206
Año:	2019
Cuartil (Scimago/WoS):	- / -

2D Image Features Detector And Descriptor Selection Expert System

Ibon Merino[1], Jon Azpiazu[1], Anthony Remazeilles[1], and Basilio Sierra[2]

[1]Industry and Transport, Tecnalia Research and Innovation, Donostia-San Sebastian, Spain
{ibon.merino, jon.azpiazu, anthony.remazeilles}@tecnalia.com
[2]Computer Science and Artificial Intelligence, University of the Basque Country UPV/EHU, Donostia-San Sebastian, Spain
b.sierra@ehu.eus

Abstract

Detection and description of keypoints from an image is a well-studied problem in Computer Vision. Some methods like SIFT, SURF or ORB are computationally really efficient. This paper proposes a solution for a particular case study on object recognition of industrial parts based on hierarchical classification. Reducing the number of instances leads to better performance, indeed, that is what the use of the hierarchical classification is looking for. We demonstrate that this method performs better than using just one method like ORB, SIFT or FREAK, despite being fairly slower.

Keywords

Computer vision, Descriptors, Feature-based object recognition, Expert system

1. Introduction

Object recognition is an important branch of computer vision. Its main idea is to extract important data or features from images in order to recognize which object is present on it. Many different techniques are used in order to achieve this. In recent computer vision literature, it has been a widely spread tendency to use deep learning due to their benefits throwing out many techniques of previous literature that, actually, have a good performance in many cases. Our aim is to recover those techniques in order to boost them and increase their performance or use their benefits that neural networks may not have.

The classical methods in computer vision are based in pure mathematical operations were images are used as matrices. These methods look for gradient changes, patterns... and try to find similarities in different images or build a machine learning model to try to predict the objects that are present in the image.

Our use case is the industrial area were many similar parts are to be recognized. Those parts vary a lot from one to another (textures, size, color, reflections,...) so an expert is needed for choosing which method is better for recognizing the objects. We propose a method that simulates the expert role. This is achieved learning a model that classifies the objects in groups that behave similarly to different recognition methods. This leads to a hierarchical classification that first classifies the object to be recognized in one of the previously obtained groups and inside the group the method that works better in that group is used to recognize the object.

The paper is organized as follows. In Section 2 we present a state of art of the most used 2D feature-based methods, including detectors, descriptors and matchers. The purpose of Section 3 is to present the method that we propose and how we evaluate it. The experiments done and their

Natarajan Meghanathan et al. (Eds) : NLP, ARIA, JSE, DMS, ITCS - 2019
pp. 51-61, 2019. © CS & IT-CSCP 2019 DOI: 10.5121/csit.2019.91206

results are shown in section 4. Section 5 summarizes the conclusions that can be drawn from our work.

2. BACKGROUND

There are several methods for object recognition. In our case, we have focused on feature-based methods. These methods look for points of interest of the images (detectors), try to describe them (descriptors) and match them (matchers). The combination of different detectors, descriptors and matchers vary the perfomance of the whole system. This is a fast growing area in image processing field. The following short and chronologically ordered review presents the gradual improvements in feature detection (Subsection 2.1), description (Subsection 2.2) and matching (Subsection 2.3).

2.1. 2D features detectors

One of the most used methods was proposed in 1999 by Lowe [13]. This method is called SIFT, which stands for Scale Invariant Feature Transform. The main idea is to use the Difference-of-Gaussian function (a close approximation to the Laplacian-of-Gaussian proposed by Lowe) to search for extrema in the scale space. Even if SIFT was relatively fast, a new method, SURF (Speeded Up Robust Features) [3], outperforms it in terms of repeatability, distinctiveness and robustness, although it can be computed and compared much faster.

In addition, FAST (Features from Accelerated Segment Test) [24] proposed by Rosten and Drummond introduce a fast detector. FAST outperforms previous algorithms (like SURF and SIFT) in both computational performance and repeatability. AGAST [16] is based on the FAST, but it is more efficient as well as generic. BRISK [11] is a novel method for keypoint detection, description and matching which has a low computational cost (as stated in the corresponding article, an order of magnitude faster than SURF in some cases). Following the same line of FAST based mehods, we find ORB Rublee et al. [25], an efficient alternative to SIFT or SURF. This method's detector is based on FAST but it adds orientation in order to obtain better results. In fact, this method performs at two orders of magnitude faster than SIFT, in many situations.

2.2. 2D features descriptors

Lowe also proposed a descriptor called SIFT. As mentioned above, is one of the most popular feature detector and descriptor. The descriptor is a position-dependent histogram of local image gradient directions around the interest point and is also scale invariant. It has numerous extensions such as PCA-SIFT [9], that mixes PCA with SIFT; CSIFT [1], Color invariant SIFT; GLOH [17]; DAISY [26], a dense descriptor inspired in SIFT and GLOH; and so on. SURF descriptor [3] relies on integral images for image convolutions in order to obtain its speed.

BRIEF [4] is a highly discriminative feature descriptor that is fast both to build and to match. BRISK [11] descriptor is composed as a binary string by concatenating the results of simple brightness comparison tests. ORB descriptor is BRIEF-based and adds rotation invariance and resistance to noise.

LBP (Local Binary Patterns) [21] is a two-level version of the texture spectrum method [27]. This methods has been really popular and many derivatives has been proposed. Based on this, the CS-LBP (Center-Symmetric Local Binary Pattern) [7] combines the strengths of SIFT and LBP. Later in 2010, the LTP (Local Ternary Pattern) [12] appeared, a generalization of the LBP that is more discriminant and less sensitive to noise in uniform regions. Same year, ELTP (Extended local ternary pattern) [20] improved this by attempting to strike a balance by using a clustering method to group the patterns in a meaningful way. In 2012, LTrP (Local Tetra Patterns) [19] encoded the relationship between the referenced pixel and its neighbors, based on the directions that are

calculated using the first-order derivatives in vertical and horizontal directions. In [22] there are gathered other methods that are based on the LBP.

Other descriptor called FREAK [2] is a keypoint descriptor inspired by the human visual system and more precisely the retina. It is faster, usess less memory and more robust than SIFT, SURF and BRISK. They are thus competitive alternatives to existing descriptors in particular for embedded applications.

2.3. Matchers

The most widely used method for matching is Nearest Neighbor (NN). Many algorithms follow this method. One of the most used is the kd-tree [23] which works well with low dimensionality. For dealing with higher dimensionalities many researchers have proposed diverse methods such as the Approximate Nearest Neighbor (ANN) by Indyk and Motwani [8] or the Fast Approximate Nearest Neighbors of Muja and Lowe [18] which is implemented in the well known open source library FLANN (Fast Library for Approximate Nearest Neighbors).

3. PROPOSED APPROACH

As we have stated before, the issue we are dealing with is the recognition of industrial parts for pick-and-placing. The main problem is that the accurate recognition of some kind of parts are highly dependant on the recognition pipeline used. This is because parts' characteristics like texture (presence or absence), forms, colors, brightness; make some detectors or descriptors work differently. We are thus proposing a systematic approach for selecting the best recognition pipeline for a given object (Subsection 3.2). We also propose in Subsection 3.3 an expert system that identifies groups of parts that are recognized similarly to improve the overall accuracy. The recognition pipeline is explained in Subsection 3.1.

We start defining some notations. An industrial part, or object, is named **instance**. The images captured of each part are named **views**. Given the set of views $\boldsymbol{X}$, the set of instance labels $\boldsymbol{Y}$ and the set of recognition pipelines $\boldsymbol{\Psi}$, the function $\omega^{\Psi}_{X,Y}(y)$ returns for each $y \in \boldsymbol{Y}$ the best pipeline $\psi* \in \boldsymbol{\Psi}$ according to a metric F_1 that is later discussed. We call ψ^{**} to the pipeline that on average performs better according to the evaluation metric, this is, that maximizes the average of the scores per instance (2).

$$\omega^{\Psi}_{X,Y}(y) = \underset{\psi \in \Psi}{\operatorname{argmax}} F_1{}^{\psi}_{y}(X, Y) = \psi^{*} \tag{1}$$

$$\psi^{**} = \underset{\psi \in \Psi}{\operatorname{argmax}} \frac{\sum_{y \in Y} F_1{}^{\psi}_{y}(X, Y)}{|Y|} \tag{2}$$

3.1. Recognition Pipeline

A recognition pipeline Ψ is composed of 3 steps: detection, description and matching. Detectors, $\boldsymbol{\Gamma}$, localize interesting keypoints in the view (gradient changes, changes in illumination,...). Descriptors, $\boldsymbol{\Phi}$, are used to represent those keypoints in order to locate them in other views. Matchers, $\boldsymbol{\Omega}$, find the closest features between views. So, a pipeline ψ is composed by a keypoint detector γ, a feature descriptor ϕ and a matcher ω. Figure 1 shows the structure of the recognition pipeline.

The keypoints detection and description are described previously in the background section. In the matching, are two groups of features: the ones that form the model (train) and the ones that

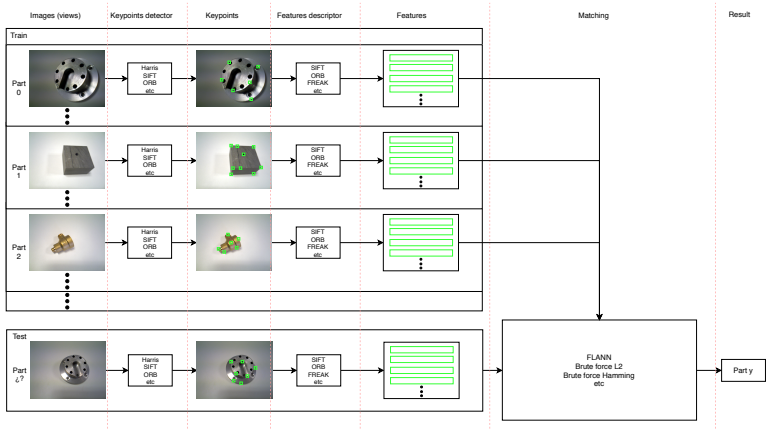


Figure 1: Recognition pipeline

need to be recognized (test). Different kind of methods could be used to match features, but, mainly, distance based techniques are used. This techniques make use of different distances (L2, hamming,...) to find the closest feature to the one that needs to be labeled. Those two features (the test feature and the closest to this one) are considered a match. In order to discard ambiguous features, we use the Lowe's ratio test [14] to define whether two features are a "good match". Assuming f_t is the feature to be recognized, and f_{l1} and f_{l2} its two closest features from the model, then (f_t, f_{l1}) is a good match if:

$$\frac{d(f_t, f_{l_1})}{d(f_t, f_{l_2})} < r \tag{3}$$

where $d(f_A, f_B)$ is the distance (L2, Hamming,...) between features A and B, and r is a threshold that is used to validate if two features are similarly close to the test feature and discard it. This threshold is set at 0.8. Now a simple voting system is used for labeling the view. For each view from the model (train) the number of good matches are counted. The good matches of each instances are summed and the test view is labeled as the instance with more good matches.

3.2. Recognition Evaluation

As we have said, we have the input views X, the instance labels Y and the pipelines Ψ. To evaluate the pipelines we have to separate the views in train and test. The evaluation method used for it is Leave-One-Out Cross-Validation (LOOCV) [10]. It consists of $|X|$ iterations, that for each iteration i, the train dataset is $(X - x_i)$ and the test sample is x_i. With this separation train-test we can generate the confusion matrix. Table 1 is an example of a confusion matrix for 3 instances.

As mentioned in the introduction of Section 3, we use the metric F_1 value [6] for scoring the performance of the system. The score is calculated for the tests views from the LOOCV. F_1 score, or value, is calculated per each instance (4). This metric is an harmonic mean between the

Table 1: Example of a confusion matrix for 3 instances.

		Actual instance			
		object 1	object 2	object 3	
Predicted instance	object 1	40	10	0	50
	object 2	0	30	25	55
	object 3	10	10	25	45
		50	50	50	150

precision and the recall. The mean of all the F_1's, $\bar{F}_1$ (5) is used for calculating ψ^{**}.

$$F_1(y) = 2 \cdot \frac{precision_y * recall_y}{precision_y + recall_y} \tag{4}$$

$$\bar{F}_1 = \frac{\sum_{y \in Y} F_1(y)}{|Y|} \tag{5}$$

The precision (Equation 6) is the ratio between the correctly predicted views with label y (tp_y) and all predicted views for that given instance ($|\psi(X) = y|$). The recall (Equation 7), instead, is the relation between correctly predicted views with label y (tp_y) and all views that should have that label ($|label(X) = y|$).

$$precision_y = \frac{tp_y}{|\psi(X) = y|} \tag{6}$$

$$recall_y = \frac{tp_y}{|label(X) = y|} \tag{7}$$

3.3. Expert system

The function ω gives a lot of information about objects but it needs the instance to return the best pipeline for that instance which is not available a priori. Indeed, this is what we want to identify. We use the information that would provide ω to build a hierarchical classification based in a clustering of similar objects.

Since some parts work better with some particular pipelines because of their shape, color or texture, we try to take advantage of this and make clusters of objects that are classified similarly well by each pipeline. For example, two parts that have textures may be better recognized by pipelines that use descriptors like SIFT or SURF rather than non textured parts. We call these clusters typologies. This clustering is made using the algorithm K-means [15], that aims to partition the objects into K clusters (where $K < |Y|$) in which each object belongs to the cluster with the nearest centroids. The input is a matrix with the instances as rows and for each row the F_1 value of each pipeline. The inputs for this algorithm are for each instance an array of the F_1 value obtained with every pipeline. The election of a good K may highly vary the result since if almost all the clusters are composed by 1 instance the result would be close to just using ψ^{**}. After obtaining the K typologies, the ψ^*_T's (8) are calculated, i.e., the best pipeline for each typology.

$$\psi^*_T = \underset{\psi \in \Psi}{\operatorname{argmax}} \frac{\sum_{y \in T} F1^{\psi}_y(X, Y)}{|T|} \tag{8}$$

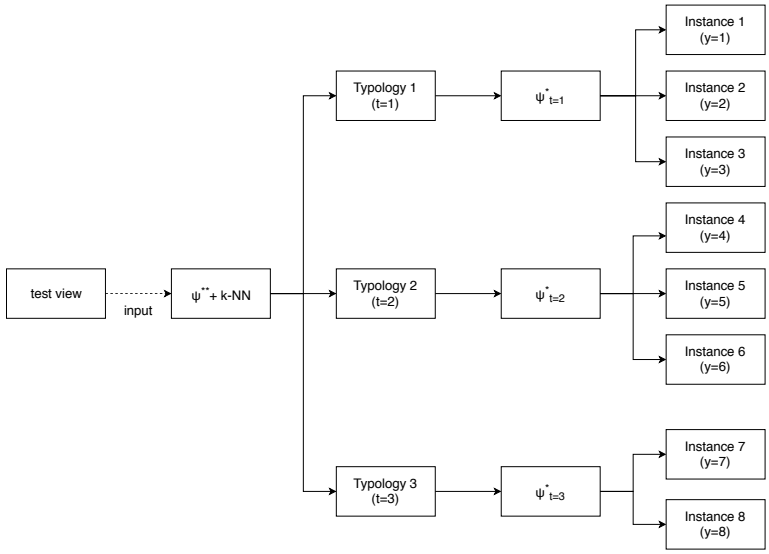


Figure 2: Hierarchical classification

The first step of the hierarchical recognition is to recognize the typology with the ψ^{**}. Given the typology t as the typology predicted, the ψ_t^* is used to recognize the instance y of the object. We call the hierarchical recognition Υ. The Figure 2 shows an scheme of the hierarchical recognition for clarification.

4. EXPERIMENTS AND RESULTS

Our initial hypothesis is that Υ has a better performance than ψ^{**}. In order to demonstrate this hypothesis we conducted some experiments. Moreover, we want to know in which way does the number of parts and the number of views per part affect the result.

The pipelines used (detector, descriptor and matcher) are defined in Subsection 4.1. In Subsection 4.2, we explain the dataset we have created to evaluate the proposed method under the use case that is the industrial area and the results obtained. In order to compare these results with a well-known dataset in Subsection 4.3 we present the Caltech dataset [5] and the results obtained.

4.1. Pipelines

The pipelines we have selected are shown in Table 2. Many combination could be done but it is not consistent to match binary descriptors with a L2 distance. The combinations chosen are compatible and may not be the best combination. LBP does not need a detector because it is a global descriptor.

4.2. Our dataset

We select 7 random industrial parts and on a white background we make 50 pictures per part from different angles randomly. That way, we have a dataset with 350 pictures. In Figure 3 are shown zoomed in examples of the pictures taken to the parts.

Table 2: Pipelines composition.

Pipeline	Detector	Descriptor	Matcher
ψ_0	SIFT	SIFT	FLANN
ψ_1	SURF	SURF	FLANN
ψ_2	ORB	ORB	Brute force Hamming
ψ_3	—-	LBP	FLANN
ψ_4	SURF	BRIEF	Brute force Hamming
ψ_5	BRISK	BRISK	Brute force Hamming
ψ_6	AGAST	DAISY	FLANN
ψ_7	AGAST	FREAK	Brute force Hamming

Figure 3: Parts used in our dataset.

Table 3: F_1's of the ψ^{**}'s and Υ for each subset of our dataset. p stands for number of parts and t for number of pictures per part.

p \ t	10		20		30		40		50	
	ψ^{**}	Υ	ψ^{**}	Υ	ψ^{**}	Υ	ψ^{**}	Υ	ψ^{**}	Υ
3	**0.935**	0.862	0.967	**0.983**	0.989	**1**	0.992	**1**	0.993	**1**
4	**0.899**	0.854	0.924	**0.962**	0.932	**0.966**	**0.944**	0.801	**0.91**	0.865
5	**0.859**	0.843	**0.868**	0.863	**0.883**	0.818	0.876	**0.901**	0.87	**0.912**
6	0.865	**0.967**	0.873	**0.992**	**0.891**	0.87	0.88	0.88	0.856	**0.901**
7	0.872	**0.9**	0.886	**0.986**	**0.894**	0.891	0.88	**0.876**	0.845	**0.94**

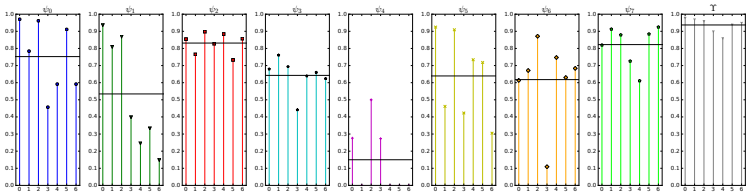


Figure 4: F_1 score for each instance and algorithm.

We use subsets of the dataset to evaluate if changing the number of views per instance and the number of instance vary the performance. This subsets have from 3 to 7 parts and from 10 to 50 views (10 views step). In Table 3 are gathered the results for all the subsets using ψ^{**} and Υ. The highest score for each subset is in bold. On average the hierarchical recognition performs better. The more parts or views per part, the better that performs the hierarchical recognition comparing with the best pipeline.

Now we focus on the whole dataset. In Figure 4 are shown the F_1's of each instance using each pipeline for this particular case. The horizontal lines mark the $\bar{F}_1$ for that pipeline. The score we obtain with our method (last column) is higher (0.94) than the best pipeline which is ψ_2 that corresponds to the pipeline that uses ORB (0.845).

A truthful evaluation of the time performance of the hierarchical classificator is a bit cumbersome since it directly depends on the clustering phase and on which are the best pipelines for each cluster. At least, it needs more time than just using a single pipeline. Given $t(\psi)$ the time need by the pipeline ψ, the time needed by Υ is approximately $t(\psi^{**}) + t(\psi^*_{T'})$ where T' is the typology guessed by the ψ^{**}. In Table 4 is shown the time in seconds that each pipeline and the Υ need to recognize a view.

4.3. Caltech-101 dataset

Caltech-101 dataset [5] is a known dataset for object recognition than could be similar to our dataset. This dataset has been tested like our dataset making subsets of the same characteristics.

Table 4: Time in seconds that needs each pipeline in recognize a piece.

ψ_0	ψ_1	ψ_2	ψ_3	ψ_4	ψ_5	ψ_6	ψ_7	Υ
0.276	0.861	0.976	0.001	0.106	0.111	0.296	1.099	1.948

Figure 5: 6 random examples of images from the Caltech-101 dataset. The classes are: Face, Leopard, Motorbike, Airplane, Accordion and Anchor.

Table 5: F_1's of the ψ^{**}'s for each test (Caltech-101). p stands for number of parts and t for number of pictures per part.

p \ t	10		20		30		40		50	
	ψ^{**}	Υ	ψ^{**}	Υ	ψ^{**}	Υ	ψ^{**}	Υ	ψ^{**}	Υ
3	0.967	0.967	0.983	0.983	0.989	0.989	0.992	0.992	0.993	0.993
4	0.975	0.975	0.987	0.987	0.975	0.975	0.969	0.969	0.963	**0.97**
5	0.98	0.98	0.99	0.99	0.967	0.967	0.96	0.96	0.956	0.956
6	0.883	0.883	0.907	0.907	0.883	0.883	0.848	0.848	0.851	**0.936**
7	0.776	0.776	0.794	**0.831**	0.78	**0.84**	0.768	**0.849**	0.783	**0.843**

Some randomly picked images from the dataset are shown in Figure 5.

The results obtained for the subsets of this datasets are shown in Table 5. Same conclusions are obtained for this dataset.

5. CONCLUSION

We proposed a hierarchical recognition method based in clustering similar behaviour by the recognition pipelines. It has been demonstrated that on average works better than just recognizing with classical feature-based methods achieving high F_1 Scores (in the biggest case, 0.94 for our dataset and 0.843 for the Caltech-101).

As we stated, once we recognize a piece we need its pose to tell the robot where to pick it. This has been let for future work. The use of local features enables the possibility to estimate objects pose using methods such as Hough voting schema, RANSAC or PnP. Additionally, including new feature-based methods may lead to better performance or at least more repeatability and scalability of the hierarchical recognition.

6. ACKNOWLEDGMENT

This paper has been supported by the project SHERLOCK under the European Unions Horizon 2020 Research Innovation programme, grant agreement No. 820689.

7. REFERENCES

[1] A. E. Abdel-Hakim and A. A. Farag. Csift: A sift descriptor with color invariant characteristics. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1978–1983. Ieee, 2006.

[2] A. Alahi, R. Ortiz, and P. Vandergheynst. FREAK: Fast Retina Keypoint. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 510–517. IEEE, June 2012.

[3] H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded Up Robust Features. In *Computer Vision ECCV 2006*, pages 404–417. 2006.

[4] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. Brief: Binary robust independent elementary features. In *European conference on computer vision*, pages 778–792, 2010.

[5] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer Vision and Image Understanding*, 106(1):59 – 70, 2007.

[6] C. Goutte and E. Gaussier. A probabilistic interpretation of precision, recall and f-score, with implication for evaluation. In *European Conference on Information Retrieval*, pages 345–359, 2005.

[7] M. Heikkil, M. Pietikinen, and C. Schmid. Description of interest regions with local binary patterns. *Pattern Recognition*, 42(3):425–436, March 2009.

[8] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*.

[9] Y. Ke and R. Sukthankar. PCA-SIFT: a more distinctive representation for local image descriptors. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 2, pages 506–513. IEEE, 2004.

[10] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145. Montreal, Canada, 1995.

[11] S. Leutenegger, M. Chli, and R. Y. Siegwart. BRISK: Binary Robust invariant scalable keypoints. In *2011 International Conference on Computer Vision*, pages 2548–2555, November 2011.

[12] W. Liao. Region Description Using Extended Local Ternary Patterns. In *2010 20th International Conference on Pattern Recognition*, pages 1003–1006, August 2010.

[13] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157 vol.2, September 1999.

[14] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004.

[15] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, pages 281–297. University of California Press, 1967.

[16] E. Mair, G. D. Hager, D. Burschka, M. Suppa, and G. Hirzinger. Adaptive and generic corner detection based on the accelerated segment test. In *European conference on Computer vision*, pages 183–196. Springer, 2010.

[17] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, October 2005.

[18] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP*, 2(331-340):2, 2009.

[19] S. Murala, R. P. Maheshwari, and R. Balasubramanian. Local Tetra Patterns: A New Feature Descriptor for Content-Based Image Retrieval. *IEEE Transactions on Image Processing*, 21 (5):2874–2886, May 2012.

[20] L. Nanni, S. Brahnam, and A. Lumini. A local approach based on a local binary patterns

variant texture descriptor for classifying pain states. *Expert Systems with Applications*, 37 (12):7888–7894, 2010.

[21] T. Ojala, M. Pietikinen, and D. Harwood. A comparative study of texture measures with classification based on featured distributions. *Pattern Recognition*, 29(1):51–59, January 1996.

[22] M. Pietikinen, A. Hadid, G. Zhao, and T. Ahonen. Local Binary Patterns for Still Images. In *Computer Vision Using Local Binary Patterns*, Computational Imaging and Vision, pages 13–47. Springer London, 2011.

[23] John T. Robinson. The k-d-b-tree: A search structure for large multidimensional dynamic indexes. In *Proceedings of the 1981 ACM SIGMOD International Conference on Management of Data*, SIGMOD '81, pages 10–18, 1981.

[24] E. Rosten and T. Drummond. Fusing points and lines for high performance tracking. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, pages 1508–1515 Vol. 2. IEEE, 2005.

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

[26] E. Tola, V. Lepetit, and P. Fua. DAISY: An Efficient Dense Descriptor Applied to Wide-Baseline Stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(5): 815–830, May 2010.

[27] L. Wang and D.C. He. Texture classification using texture spectrum. *Pattern Recognition*, 23(8):905–910, 1990.

CAPÍTULO 6

2D Features-based detector and descriptor selection system for hierarchical recognition of industrial parts

Título:	2D Features-based detector and descriptor selection system for hierarchical recognition of industrial parts
Autores:	I. Merino, J. Azpiazu, A. Remazeilles, B. Sierra
Revista:	International Journal of Artificial Intelligence & Applications (IJAIA)
Volumen:	10
Número:	6
Editor:	AIRCC Publishing Corporation
DOI:	10.5121/ijaia.2019.10601
Año:	2019
Cuartil (Scimago/WoS):	- / -

International Journal of Artificial Intelligence & Applications (IJAIA) Vol.10, No.6, November 2019

2D Features-Based detector and Descriptor Selection System for Hierarchical Recognition of Industrial Parts

Ibon Merino[1], Jon Azpiazu[1], Anthony Remazeilles[1], and Basilio Sierra[2]

[1]Industry and Transport, Tecnalia Research and Innovation, Donostia-San Sebastian, Spain
{ibon.merino, jon.azpiazu, anthony.remazeilles}@tecnalia.com
[2]Computer Science and Artificial Intelligence, University of the Basque Country UPV/EHU, Donostia-San Sebastian, Spain
b.sierra@ehu.eus

Abstract

Detection and description of keypoints from an image is a well-studied problem in Computer Vision. Some methods like SIFT, SURF or ORB are computationally really efficient. This paper proposes a solution for a particular case study on object recognition of industrial parts based on hierarchical classification. Reducing the number of instances leads to better performance, indeed, that is what the use of the hierarchical classification is looking for. We demonstrate that this method performs better than using just one method like ORB, SIFT or FREAK, despite being fairly slower.

Keywords

Computer vision, Descriptors, Feature-based object recognition, Expert system

1. Introduction

Object recognition is an important branch of computer vision. Its main idea is to extract important data or features from images in order to recognize which object is present on it. Many different techniques are used in order to achieve this. In recent computer vision literature, it has been a widely spread tendency to use deep learning due to their benefits throwing out many techniques of previous literature that, actually, have a good performance in many cases. Our aim is to recover those techniques in order to boost them and increase their performance or use their benefits that neural networks may not have.

The classical methods in computer vision are based in pure mathematical operations were images are used as matrices. These methods look for gradient changes, patterns... and try to find similarities in different images or build a machine learning model to try to predict the objects that are present in the image.

Our use case is the industrial area were many similar parts are to be recognized. Those parts vary a lot from one to another (textures, size, color, reflections,...) so an expert is needed for choosing which method is better for recognizing the objects. We propose a method that simulates the expert role. This is achieved learning a model that classifies the objects in groups that behave similarly to different recognition methods. This leads to a hierarchical classification that first classifies the object to be recognized in one of the previously obtained groups and inside the group the method that works better in that group is used to recognize the object.

DOI: 10.5121/ijaia.2019.10601

The paper is organized as follows. In Section 2 we present a state of art of the most used 2D feature-based methods, including detectors, descriptors and matchers. The purpose of Section 3 is to present the method that we propose and how we evaluate it. The experiments done and their results are shown in section 4. Section 5 summarizes the conclusions that can be drawn from our work.

2. BACKGROUND

There are several methods for object recognition. In our case, we have focused on feature-based methods. These methods look for points of interest of the images (detectors), try to describe them (descriptors) and match them (matchers). The combination of different detectors, descriptors and matchers vary the perfomance of the whole system. This is a fast growing area in image processing field. The following short and chronologically ordered review presents the gradual improvements in feature detection (Subsection 2.1), description (Subsection 2.2) and matching (Subsection 2.3).

2.1. 2D features detectors

One of the most used methods was proposed in 1999 by Lowe [14]. This method is called SIFT, which stands for Scale Invariant Feature Transform. The main idea is to use the Difference-of-Gaussian function (a close approximation to the Laplacian-of-Gaussian proposed by Lowe) to search for extrema in the scale space. Even if SIFT was relatively fast, a new method, SURF (Speeded Up Robust Features) [3], outperforms it in terms of repeatability, distinctiveness and robustness, although it can be computed and compared much faster.

In addition, FAST (Features from Accelerated Segment Test) [25] proposed by Rosten and Drummond introduce a fast detector. FAST outperforms previous algorithms (like SURF and SIFT) in both computational performance and repeatability. AGAST [17] is based on the FAST, but it is more efficient as well as generic. BRISK [12] is a novel method for keypoint detection, description and matching which has a low computational cost (as stated in the corresponding article, an order of magnitude faster than SURF in some cases). Following the same line of FAST based mehods, we find ORB Rublee et al. [26], an efficient alternative to SIFT or SURF. This method's detector is based on FAST but it adds orientation in order to obtain better results. In fact, this method performs at two orders of magnitude faster than SIFT, in many situations.

Figure 1 shows some detectors and the relation between them chronologically ordered.

2.2. 2D features descriptors

Lowe also proposed a descriptor called SIFT. As mentioned above, is one of the most popular feature detector and descriptor. The descriptor is a position-dependent histogram of local image gradient directions around the interest point and is also scale invariant. It has numerous extensions such as PCA-SIFT [10], that mixes PCA with SIFT; CSIFT [1], Color invariant SIFT; GLOH [18]; DAISY [27], a dense descriptor inspired in SIFT and GLOH; and so on. SURF descriptor [3] relies on integral images for image convolutions in order to obtain its speed.

BRIEF [4] is a highly discriminative feature descriptor that is fast both to build and to match. BRISK [12] descriptor is composed as a binary string by concatenating the results of simple brightness comparison tests. ORB descriptor is BRIEF-based and adds rotation invariance and resistance to noise.

LBP (Local Binary Patterns) [22] is a two-level version of the texture spectrum method [28]. This methods has been really popular and many derivatives has been proposed. Based on this, the CS-LBP (Center-Symmetric Local Binary Pattern) [8] combines the strengths of SIFT and LBP. Later

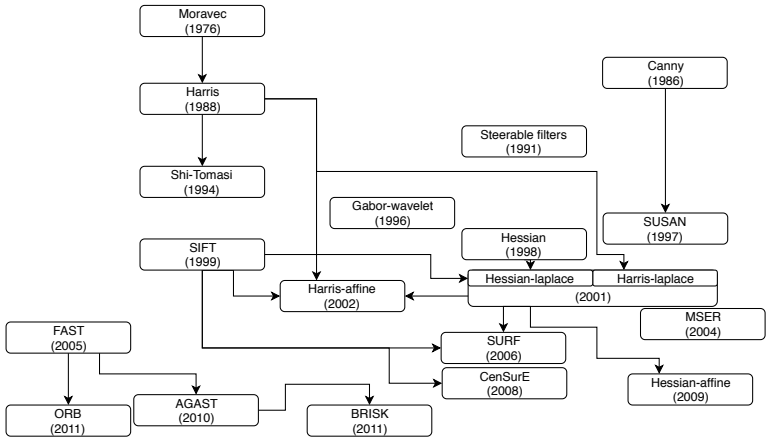


Figure 1: Recognition pipeline

in 2010, the LTP (Local Ternary Pattern) [13] appeared, a generalization of the LBP that is more discriminant and less sensitive to noise in uniform regions. Same year, ELTP (Extended local ternary pattern) [21] improved this by attempting to strike a balance by using a clustering method to group the patterns in a meaningful way. In 2012, LTrP (Local Tetra Patterns) [20] encoded the relationship between the referenced pixel and its neighbors, based on the directions that are calculated using the first-order derivatives in vertical and horizontal directions. In [23] there are gathered other methods that are based on the LBP.

MTS (Modified texture spectrum) proposed by Xu et al. [29] can be considered as a simplified version of LBP, where only a subset of the peripheral pixels (up-left, up, up-right and right) is considered.

The Binary Gradient Contours (BGC) [6] is a binary 8-tuple proposed by Fernndez et al. The simple loop form (BGC1) makes a closed path around the central pixel computing a set of eight binary gradients between pairs of pixels.

Other descriptor called FREAK [2] is a keypoint descriptor inspired by the human visual system and more precisely the retina. It is faster, usess less memory and more robust than SIFT, SURF and BRISK. They are thus competitive alternatives to existing descriptors in particular for embedded applications.

Figure 2 shows some detectors and the relation between them chronologically ordered.

2.3. Matchers

The most widely used method for matching is Nearest Neighbor (NN). Many algorithms follow this method. One of the most used is the kd-tree [24] which works well with low dimensionality. For dealing with higher dimensionalities many researchers have proposed diverse methods such as the Approximate Nearest Neighbor (ANN) by Indyk and Motwani [9] or the Fast Approximate Nearest Neighbors of Muja and Lowe [19] which is implemented in the well known open source library FLANN (Fast Library for Approximate Nearest Neighbors).

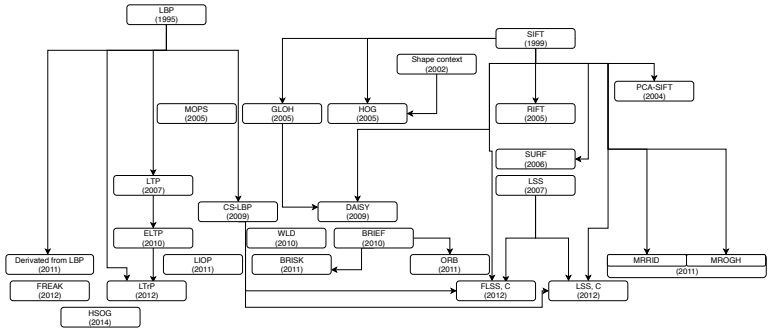


Figure 2: Recognition pipeline

3. PROPOSED APPROACH

As we have stated before, the issue we are dealing with is the recognition of industrial parts for pick-and-placing. The main problem is that the accurate recognition of some kind of parts are highly dependant on the recognition pipeline used. This is because parts' characteristics like texture (presence or absence), forms, colors, brightness; make some detectors or descriptors work differently. We are thus proposing a systematic approach for selecting the best recognition pipeline for a given object (Subsection 3.2). We also propose in Subsection 3.3 an expert system that identifies groups of parts that are recognized similarly to improve the overall accuracy. The recognition pipeline is explained in Subsection 3.1.

We start defining some notations. An industrial part, or object, is named **instance**. The images captured of each part are named **views**. Given the set of views $\boldsymbol{X}$, the set of instance labels $\boldsymbol{Y}$ and the set of recognition pipelines $\boldsymbol{\Psi}$, the function $\omega^{\boldsymbol{\Psi}}_{X,Y}(y)$ returns for each $y \in \boldsymbol{Y}$ the best pipeline $\psi* \in \boldsymbol{\Psi}$ according to a metric F_1 that is later discussed. We call ψ^{**} to the pipeline that on average performs better according to the evaluation metric, this is, that maximizes the average of the scores per instance (2).

$$\omega^{\boldsymbol{\Psi}}_{X,Y}(y) = \underset{\psi \in \boldsymbol{\Psi}}{\operatorname{argmax}} F1^{\psi}_{y}(X,Y) = \psi^{*} \tag{1}$$

$$\psi^{**} = \underset{\psi \in \boldsymbol{\Psi}}{\operatorname{argmax}} \frac{\sum_{y \in Y} F1^{\psi}_{y}(X,Y)}{|Y|} \tag{2}$$

3.1. Recognition Pipeline

A recognition pipeline Ψ is composed of 3 steps: detection, description and matching. Detectors, $\boldsymbol{\Gamma}$, localize interesting keypoints in the view (gradient changes, changes in illumination,...). Descriptors, $\boldsymbol{\Phi}$, are used to represent those keypoints in order to locate them in other views. Matchers, $\boldsymbol{\Omega}$, find the closest features between views. So, a pipeline ψ is composed by a keypoint detector γ, a feature descriptor ϕ and a matcher ω. Figure 3 shows the structure of the recognition pipeline.

The keypoints detection and description are described previously in the background section. In the matching, are two groups of features: the ones that form the model (train) and the ones that

International Journal of Artificial Intelligence & Applications (IJAIA) Vol.10, No.6, November 2019

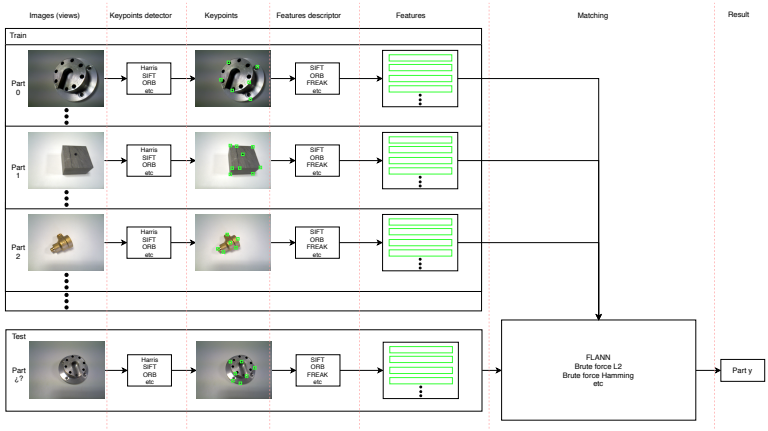


Figure 3: Recognition pipeline

need to be recognized (test). Different kind of methods could be used to match features, but, mainly, distance based techniques are used. This techniques make use of different distances (L2, hamming,...) to find the closest feature to the one that needs to be labeled. Those two features (the test feature and the closest to this one) are considered a match. In order to discard ambiguous features, we use the Lowe's ratio test [15] to define whether two features are a "good match". Assuming f_t is the feature to be recognized, and f_{l1} and f_{l2} its two closest features from the model, then (f_t, f_{l1}) is a good match if:

$$\frac{d(f_t, f_{l_1})}{d(f_t, f_{l_2})} < r \tag{3}$$

where $d(f_A, f_B)$ is the distance (Euclidean or L2 distance: Equation 4, Hamming distance: Equation 5, where δ is the kronecker delta (Equation 6),...) between features A and B, and r is a threshold that is used to validate if two features are similarly close to the test feature and discard it. This threshold is set at 0.8. Now a simple voting system is used for labeling the view. For each view from the model (train) the number of good matches are counted. The good matches of each instances are summed and the test view is labeled as the instance with more good matches.

$$d_E(P,Q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2} \tag{4}$$

$$d_H(P,Q) = \sum_{i=1}^{n}\delta(p_i, q_i) \tag{5}$$

$$\delta(p_i, q_i) = \begin{cases} 0 & if \quad x_i = y_i \\ 1 & if \quad x_i \neq y_i \end{cases} \tag{6}$$

Table 1: Example of a confusion matrix for 3 instances.

		Actual instance			
		object 1	object 2	object 3	
Predicted instance	object 1	40	10	0	50
	object 2	0	30	25	55
	object 3	10	10	25	45
		50	50	50	150

3.2. Recognition Evaluation

As we have said, we have the input views X, the instance labels Y and the pipelines Ψ. To evaluate the pipelines we have to separate the views in train and test. The evaluation method used for it is Leave-One-Out Cross-Validation (LOOCV) [11]. It consists of $|X|$ iterations, that for each iteration i, the train dataset is $(X - x_i)$ and the test sample is x_i. With this separation train-test we can generate the confusion matrix. Table 1 is an example of a confusion matrix for 3 instances.

As mentioned in the introduction of Section 3, we use the metric F_1 value [7] for scoring the performance of the system. The score is calculated for the tests views from the LOOCV. F_1 score, or value, is calculated per each instance (7). This metric is an harmonic mean between the precision and the recall. The mean of all the F_1's, $\bar{F_1}$ (8) is used for calculating ψ^{**}.

$$F_1(y) = 2 \cdot \frac{precision_y * recall_y}{precision_y + recall_y} \tag{7}$$

$$\bar{F_1} = \frac{\sum_{y \in Y} F_1(y)}{|Y|} \tag{8}$$

The precision (Equation 9) is the ratio between the correctly predicted views with label y (tp_y) and all predicted views for that given instance ($|\psi(X) = y|$). The recall (Equation 10), instead, is the relation between correctly predicted views with label y (tp_y) and all views that should have that label ($|label(X) = y|$).

$$precision_y = \frac{tp_y}{|\psi(X) = y|} \tag{9}$$

$$recall_y = \frac{tp_y}{|label(X) = y|} \tag{10}$$

3.3. Expert system

The function ω gives a lot of information about objects but it needs the instance to return the best pipeline for that instance which is not available a priori. Indeed, this is what we want to identify. We use the information that would provide ω to build a hierarchical classification based in a clustering of similar objects.

Since some parts work better with some particular pipelines because of their shape, color or texture, we try to take advantage of this and make clusters of objects that are classified similarly well by each pipeline. For example, two parts that have textures may be better recognized by pipelines that use descriptors like SIFT or SURF rather than non textured parts. We call these clusters typologies. This clustering is made using the algorithm K-means [16], that aims to partition the

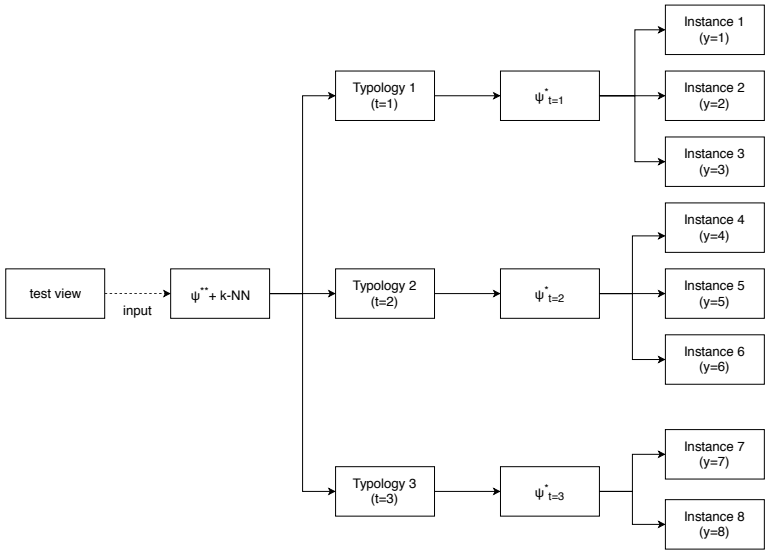


Figure 4: Hierarchical classification

objects into K clusters (where $K < |Y|$) in which each object belongs to the cluster with the nearest centroids. The input is a matrix with the instances as rows and for each row the F_1 value of each pipeline. The inputs for this algorithm are for each instance an array of the F_1 value obtained with every pipeline. The election of a good K may highly vary the result since if almost all the clusters are composed by 1 instance the result would be close to just using ψ^{**}. After obtaining the K typologies, the ψ_T^*'s (11) are calculated, i.e., the best pipeline for each typology.

$$\psi_T^* = \underset{\psi \in \Psi}{\operatorname{argmax}} \frac{\sum_{y \in T} F_1{}_y^{\psi}(X, Y)}{|T|} \tag{11}$$

The first step of the hierarchical recognition is to recognize the typology with the ψ^{**}. Given the typology t as the typology predicted, the ψ_t^* is used to recognize the instance y of the object. We call the hierarchical recognition Υ. The Figure 4 shows an scheme of the hierarchical recognition for clarification.

4. Experiments and results

Our initial hypothesis is that Υ has a better performance than ψ^{**}. In order to demonstrate this hypothesis we conducted some experiments. Moreover, we want to know in which way does the number of parts and the number of views per part affect the result.

The pipelines used (detector, descriptor and matcher) are defined in Subsection 4.1. In Subsection 4.2, we explain the dataset we have created to evaluate the proposed method under the use case that is the industrial area and the results obtained. In order to compare these results with a well-known dataset in Subsection 4.3 we present the Caltech dataset [5] and the results obtained.

Table 2: Pipelines composition.

Pipeline	Detector	Descriptor	Matcher
ψ_0	SIFT	SIFT	FLANN
ψ_1	SURF	SURF	FLANN
ψ_2	ORB	ORB	Brute force Hamming
ψ_3	—-	LBP	FLANN
ψ_4	SURF	BRIEF	Brute force Hamming
ψ_5	BRISK	BRISK	Brute force Hamming
ψ_6	AGAST	DAISY	FLANN
ψ_7	AGAST	FREAK	Brute force Hamming

Table 3: F_1's of the ψ^{**}'s and Υ for each subset of our dataset. p stands for number of parts and t for number of pictures per part.

p \ t	10		20		30		40		50	
	ψ^{**}	Υ	ψ^{**}	Υ	ψ^{**}	Υ	ψ^{**}	Υ	ψ^{**}	Υ
3	**0.935**	0.862	0.967	**0.983**	0.989	**1**	0.992	**1**	0.993	**1**
4	**0.899**	0.854	0.924	**0.962**	0.932	**0.966**	**0.944**	0.801	**0.91**	0.865
5	**0.859**	0.843	**0.868**	0.863	**0.883**	0.818	0.876	**0.901**	0.87	**0.912**
6	0.865	**0.967**	0.873	**0.992**	**0.891**	0.87	0.88	0.88	0.856	**0.901**
7	0.872	**0.9**	0.886	**0.986**	**0.894**	0.891	0.88	**0.876**	0.845	**0.94**

4.1. Pipelines

The pipelines we have selected are shown in Table 2. Many combination could be done but it is not consistent to match binary descriptors with a L2 distance. The combinations chosen are compatible and may not be the best combination. Global descriptors, such as LBP, does not need a detector.

4.2. Our dataset

We select 7 random industrial parts and on a white background we make 50 pictures per part from different angles randomly. That way, we have a dataset with 350 pictures. In Figure 5 are shown zoomed in examples of the pictures taken to the parts.

We use subsets of the dataset to evaluate if changing the number of views per instance and the number of instance vary the performance. This subsets have from 3 to 7 parts and from 10 to 50 views (10 views step). In Table 3 are gathered the results for all the subsets using ψ^{**} and Υ. The highest score for each subset is in bold. On average the hierarchical recognition performs better. The more parts or views per part, the better that performs the hierarchical recognition comparing with the best pipeline.

Now we focus on the whole dataset. In Figure 6 are shown the F_1's of each instance using each pipeline for this particular case. The horizontal lines mark the $\bar{F}_1$ for that pipeline. The score we obtain with our method (last column) is higher (0.94) than the best pipeline which is ψ_2 that corresponds to the pipeline that uses ORB (0.845).

Figure 5: Parts used in our dataset.

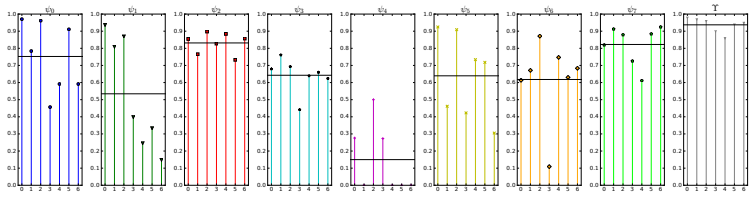


Figure 6: F_1 score for each instance and algorithm.

Table 4: Time in seconds that needs each pipeline in recognize a piece.

ψ_0	ψ_1	ψ_2	ψ_3	ψ_4	ψ_5	ψ_6	ψ_7	Υ
0.276	0.861	0.976	0.001	0.106	0.111	0.296	1.099	1.948

Figure 7: 6 random examples of images from the Caltech-101 dataset. The classes are: Face, Leopard, Motorbike, Airplane, Accordion and Anchor.

Table 5: F_1's of the ψ^{**}'s for each test (Caltech-101). p stands for number of parts and t for number of pictures per part.

p \ t	10		20		30		40		50	
	ψ^{**}	Υ	ψ^{**}	Υ	ψ^{**}	Υ	ψ^{**}	Υ	ψ^{**}	Υ
3	0.967	0.967	0.983	0.983	0.989	0.989	0.992	0.992	0.993	0.993
4	0.975	0.975	0.987	0.987	0.975	0.975	0.969	0.969	0.963	**0.97**
5	0.98	0.98	0.99	0.99	0.967	0.967	0.96	0.96	0.956	0.956
6	0.883	0.883	0.907	0.907	0.883	0.883	0.848	0.848	0.851	**0.936**
7	0.776	0.776	0.794	**0.831**	0.78	**0.84**	0.768	**0.849**	0.783	**0.843**

A truthful evaluation of the time performance of the hierarchical classificator is a bit cumbersome since it directly depends on the clustering phase and on which are the best pipelines for each cluster. At least, it needs more time than just using a single pipeline. Given $t(\psi)$ the time need by the pipeline ψ, the time needed by Υ is approximately $t(\psi^{**}) + t(\psi^*_{T'})$ where T' is the typology guessed by the ψ^{**}. In Table 4 is shown the time in seconds that each pipeline and the Υ need to recognize a view.

4.3. Caltech-101 dataset

Caltech-101 dataset [5] is a known dataset for object recognition than could be similar to our dataset. This dataset has been tested like our dataset making subsets of the same characteristics. Some randomly picked images from the dataset are shown in Figure 7.

The results obtained for the subsets of this datasets are shown in Table 5. Same conclusions are obtained for this dataset.

4.4. Adding more descriptors

Due to the great performance that global descriptors have, we tried other experiments on the previous 2 datasets adding new descriptors that had been considered from the beginning. These experiments mix global descriptors and local descriptors. Local descriptors make use of the matching technique to recognise the objects in the images. Global descriptors, instead, are usually used

Table 6: F1s per instance and mean using added descriptors for our dataset

	0	1	2	3	4	5	6	mean
SIFT	0.962	0.785	0.951	0.455	0.598	0.906	0.598	0.751
SURF	0.857	0.5	0.8	0.196	0.667	0.25	0.5	0.539
ORB	0.6	0.53	0.723	0.715	0.8	0.9	0.8	0.845
BRIEF	0.287	0	0.5	0.287	0	0	0	0.153
BRISK	0.931	0.475	0.912	0.422	0.735	0.726	0.301	0.643
DAISY	0.623	0.675	0.871	0.121	0.75	0.637	0.687	0.623
FREAK	0.823	0.911	0.872	0.738	0.617	0.878	0.927	0.824
LBP	0.687	0.767	0.694	0.432	0.647	0.667	0.635	0.647
LBPU	0.106	0.4	0.5	0.333	0.75	0	0.25	0.334
BGC1	0.72	0.4	0.462	0.182	1	0.333	0.5	0.514
GRAD	0.8	0.667	0.5	0.546	0.8	1	0.889	0.743
GABORGB	0.909	1	0.667	0.8	0.889	0.889	0.8	0.851
LTP	0.667	0.572	0.6	0.285	0.8	0.25	0	0.453
MTS	0.261	0	0.445	0	0.75	0.286	0.364	0.301
TPLBP	0.667	0.667	0.889	0.667	0.857	0.8	0.933	0.783

among machine learning techniques to identify the objects. In this case, we have used a Support Vector Machine to learn the model that recognise the objects using the global descriptors.

The newly added global descriptors are: LBPU (Uniform LBP), BGC1, Gradded histogram, Gabor Filters, LTP, MTS and TPLBP (Three-Patch LBP). In Figure 6, are shown the F1s obtained using all the descriptors (the first 8 and the newly added global descriptors) for our dataset. The hierarchical classificator using all the descriptors achieves a F1 of 0.937. In Figure 7, instead are the results for the Caltech-101 dataset. The hierarchical classificator with this dataset achieves a F1 of 0.95.

Adding more descriptors without taking out others does not improve the hierarchical classificator. Even more, in some experiments done, the results are worse than just using the firstly proposed 8 methods. This is due to the fact that the typologies are chosen by clustering the instances. That does not provide the best combination of instances, but the ones that behave similarly to all the descriptors. So the more descriptors the more variability in clustering.

5. Conclusion

We proposed a hierarchical recognition method based in clustering similar behaviour by the recognition pipelines. It has been demonstrated that on average works better than just recognizing with classical feature-based methods achieving high F_1 Scores (in the biggest case, 0.94 for our dataset and 0.843 for the Caltech-101). The performance of the hierarchical method is highly dependant of the feature-based method used to build it. In order to improve its performance new combinations of descriptor may be proposed. Not just adding and increasing the algorithm bag, but selecting the best combination of algorithms to obtain the best performance.

As we stated, once we recognize a piece we need its pose to tell the robot where to pick it. This has been let for future work. The use of local features enables the possibility to estimate objects pose using methods such as Hough voting schema, RANSAC or PnP.

6. Acknowledgment

This paper has been supported by the project SHERLOCK under the European Unions Horizon 2020 Research Innovation programme, grant agreement No. 820689.

Table 7: F1s per instance and mean using added descriptors for Caltech-101

	0	1	2	3	4	5	6	mean
SIFT	0	0	0	0.5	0.667	0.461	0.2	0.261
SURF	0	0.25	0.428	0.706	0	0	0	0.197
ORB	0.693	0.753	0.723	0.715	0.859	0.944	0.8	0.783
BRIEF	0	0.125	0.286	0.8	0	0.333	0.182	0.246
BRISK	0.167	0	0.25	0.889	0.445	0.222	0.182	0.307
DAISY	0	0	0	0.25	0	0	0.222	0.067
FREAK	0	0.182	0.545	0.909	0	0.461	0.5	0.371
LBP	0.909	1	1	0.833	1	0.667	0.889	0.899
LBPU	1	1	0.889	0.8	0.923	0.933	0.857	0.914
BGC1	0.667	0.923	1	0.75	0.889	0.923	0.857	0.858
GRAD	0.75	0.857	1	0.909	1	0.857	0.667	0.862
GABORGB	1	0.72	1	0.744	1	1	0.85	0.902
LTP	0.727	0.445	1	1	0.889	1	0.286	0.763
MTS	0.857	1	0.8	1	1	1	0.5	0.879
TPLBP	1	1	1	0.667	0.857	0.857	0.667	0.864

7. REFERENCES

[1] A. E. Abdel-Hakim and A. A. Farag. Csift: A sift descriptor with color invariant characteristics. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1978–1983. Ieee, 2006.

[2] A. Alahi, R. Ortiz, and P. Vandergheynst. FREAK: Fast Retina Keypoint. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 510–517. IEEE, June 2012.

[3] H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded Up Robust Features. In *Computer Vision ECCV 2006*, pages 404–417. 2006.

[4] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. Brief: Binary robust independent elementary features. In *European conference on computer vision*, pages 778–792, 2010.

[5] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer Vision and Image Understanding*, 106(1):59 – 70, 2007.

[6] A. Fernndez, M. X. lvarez, and F. Bianconi. Image classification with binary gradient contours. *Optics and Lasers in Engineering*, 49(9-10):1177–1184, 2011.

[7] C. Goutte and E. Gaussier. A probabilistic interpretation of precision, recall and f-score, with implication for evaluation. In *European Conference on Information Retrieval*, pages 345–359, 2005.

[8] M. Heikkil, M. Pietikinen, and C. Schmid. Description of interest regions with local binary patterns. *Pattern Recognition*, 42(3):425–436, March 2009.

[9] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*.

[10] Y. Ke and R. Sukthankar. PCA-SIFT: a more distinctive representation for local image descriptors. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 2, pages 506–513. IEEE, 2004.

[11] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145. Montreal, Canada, 1995.

[12] S. Leutenegger, M. Chli, and R. Y. Siegwart. BRISK: Binary Robust invariant scalable keypoints. In *2011 International Conference on Computer Vision*, pages 2548–2555, November

2011.

[13] W. Liao. Region Description Using Extended Local Ternary Patterns. In *2010 20th International Conference on Pattern Recognition*, pages 1003–1006, August 2010.

[14] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157 vol.2, September 1999.

[15] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004.

[16] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, pages 281–297. University of California Press, 1967.

[17] E. Mair, G. D. Hager, D. Burschka, M. Suppa, and G. Hirzinger. Adaptive and generic corner detection based on the accelerated segment test. In *European conference on Computer vision*, pages 183–196. Springer, 2010.

[18] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, October 2005.

[19] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP*, 2(331-340):2, 2009.

[20] S. Murala, R. P. Maheshwari, and R. Balasubramanian. Local Tetra Patterns: A New Feature Descriptor for Content-Based Image Retrieval. *IEEE Transactions on Image Processing*, 21 (5):2874–2886, May 2012.

[21] L. Nanni, S. Brahnam, and A. Lumini. A local approach based on a local binary patterns variant texture descriptor for classifying pain states. *Expert Systems with Applications*, 37 (12):7888–7894, 2010.

[22] T. Ojala, M. Pietikinen, and D. Harwood. A comparative study of texture measures with classification based on featured distributions. *Pattern Recognition*, 29(1):51–59, January 1996.

[23] M. Pietikinen, A. Hadid, G. Zhao, and T. Ahonen. Local Binary Patterns for Still Images. In *Computer Vision Using Local Binary Patterns*, Computational Imaging and Vision, pages 13–47. Springer London, 2011.

[24] John T. Robinson. The k-d-b-tree: A search structure for large multidimensional dynamic indexes. In *Proceedings of the 1981 ACM SIGMOD International Conference on Management of Data*, SIGMOD '81, pages 10–18, 1981.

[25] E. Rosten and T. Drummond. Fusing points and lines for high performance tracking. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, pages 1508–1515 Vol. 2. IEEE, 2005.

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

[27] E. Tola, V. Lepetit, and P. Fua. DAISY: An Efficient Dense Descriptor Applied to Wide-Baseline Stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(5): 815–830, May 2010.

[28] L. Wang and D.C. He. Texture classification using texture spectrum. *Pattern Recognition*, 23(8):905–910, 1990.

[29] B. Xu, P. Gong, E. Seto, and R. Spear. Comparison of Gray-Level Reduction and Different Texture Spectrum Encoding Methods for Land-Use Classification Using a Panchromatic Ikonos Image. *Photogrammetric Engineering & Remote Sensing*, 69(5):529–536, May 2003.

CAPÍTULO 7

Histogram-Based Descriptor Subset Selection for Visual Recognition of Industrial Parts

Título:	Histogram-Based Descriptor Subset Selection for Visual Recognition of Industrial Parts
Autores:	I. Merino, J. Azpiazu, A. Remazeilles, B. Sierra
Revista:	Applied sciences
Volumen:	10
Número:	3701
Editor:	MDPI
DOI:	10.3390/app10113701
Año:	2020
Cuartil (Scimago/WoS):	Q2 (Computer Science Applications) / Q2 (Engineering, multidisciplinary)

applied sciences

MDPI

Article

Histogram-Based Descriptor Subset Selection for Visual Recognition of Industrial Parts

Ibon Merino [1,2,*], Jon Azpiazu [1], Anthony Remazeilles [1] and Basilio Sierra [2]

[1] TECNALIA, Basque Research and Technology Alliance (BRTA), Paseo Mikeletegi 7, 20009 Donostia-San Sebastian, Spain; jon.azpiazu@tecnalia.com (J.A.); anthony.remazeilles@tecnalia.com (A.R.)

[2] Department of Computer Science and Artificial Intelligence, University of the Basque Country UPV/EHU, 20018 Donostia-San Sebastian, Spain; b.sierra@ehu.eus

* Correspondence: ibon.merino@tecnalia.com

Received: 1 April 2020; Accepted: 25 May 2020; Published: 27 May 2020

Abstract: This article deals with the 2D image-based recognition of industrial parts. Methods based on histograms are well known and widely used, but it is hard to find the best combination of histograms, most distinctive for instance, for each situation and without a high user expertise. We proposed a descriptor subset selection technique that automatically selects the most appropriate descriptor combination, and that outperforms approach involving single descriptors. We have considered both backward and forward mechanisms. Furthermore, to recognize the industrial parts a supervised classification is used with the global descriptors as predictors. Several class approaches are compared. Given our application, the best results are obtained with the Support Vector Machine with a combination of descriptors increasing the F1 by 0.031 with respect to the best descriptor alone.

Keywords: computer vision; feature descriptor; histogram; feature subset selection; industrial objects

1. Introduction

Computer vision, in the last years, has gained much interest in many fields, such as autonomous driving [1], medical [2], face recognition [3], object detection [4], and object segmentation [5]. Perception is also regarded as one of the key enabling technologies for extending the robot capabilities, preferentially targeting flexibility, adaptation, and robustness, as required for fulfilling the industry 4.0 paradigm [6]. Although in most fields large and complex datasets can be obtained, detection of industrial parts has a lack of datasets. One of the reasons is that most of the time in industrial context, the aim is to detect an object from which usually the CAD is available. However, sometimes there is a need of detecting diverse, complex, and tiny objects [7] and lack of time to generate a robust dataset (taking pictures and labeling). One of the solutions is to generate simulated data to train the models but usually there is a significant gap transferring that learned knowledge to reality.

To make matter worse, industrial parts are usually texture-less. This means that many of the most used recognition methods cannot deal with them. One of the methods to deal with texture-less objects are Convolutional Neural Networks. Nowadays, computer vision researches are mainly focused on using Convolutional Neural Networks (CNN) [8–10]. One of the disadvantages of the CNNs is the need of a large dataset to train them. Even if it is possible to use the CNN trained on other fields in industry [11], there is still a need of a large enough training dataset to obtain good results. Feature descriptors based on classical methods have been very useful and thoroughly spread in the literature previous to CNN. One of the benefits of using this approach is that there is no need of a large training set to obtain good results. Actually, there are many image descriptors and each of them has its advantages and disadvantages.

Our approach is based in the idea that the combination of different descriptors leads to a better performance, taking advantage of the benefits of each descriptor to deal with the two problems mentioned before (lack of a large dataset and texture-less objects). The crux of the matter is to select the descriptors that contribute to achieve a better result and discard those that do not provide any improvement. Our method achieves a classification quality similar to state-of-the-art methods on the experiments done.

In Section 2, we present a background of the description methods, classifiers, and features subset selection techniques. In Section 3, we explain the combination of the descriptors and the image classification. The experiments done and their results are gathered in Section 4. Finally, in Section 5, the conclusions are summarized.

2. Background

The analysis of images usually relies on the extraction of visual features. Such an approach can be observed in classification [12], object detection [4], and segmentation [5]. In this section, we provide an overview of the main feature descriptors, together with some of the related classification techniques.

2.1. Features Descriptors

Local features extractors are characteristic local primitives as points focusing on a close neighborhood. Some examples of those features are SIFT [13], SURF [14], and LBP [15]. Global descriptors, instead, extract information directly from the whole image by computing histograms for example. Local features are good for image recognition as each point is independent from the rest and the features are more discriminant. Global features instead are more used for classification and object detection as they achieve a more global representation. Nevertheless, small changes have a larger impact on global features and a better preprocessing is needed when using them. Extracting global features and their classification is usually faster.

As a matter of a fact, combining both local and global features usually performs better [16]. Many researchers use histograms of local features to obtain benefits of both types. Doing so, we obtain a global representation of the local features. [16] present a taxonomy called Histogram of Equivalent Patterns (HEP) that gathers those histograms of local features. In order for a feature to be part of this framework, it needs to have a delimited quantification, that is, the number of possible values of the extracted feature must be small enough to obtain a relevant histogram. For example, LBP [15] is part of this framework as the possible values are 256 so the resulting histogram is of length 256, while HOG or SIFT are not part of the HEP framework as the number of possible values is high and the resulting histogram is not relevant. In [17], a combination of descriptors was also used, but limited to local descriptors.

One of the first HEP methods was introduced in 1973. This method, called Gray Level Co-occurrences Matrices (GLCM) [18] measures the joint probability of the gray levels of two pixels standing in some predefined relative positions. Since 1973, it has been widely used in many texture analysis applications as a feature extractor in this context.

In 1990, [19] proposed the texture spectrum (TS), which inspired many HEP methods. This texture descriptor is based in decomposing the image into a set of essential small units, called Texture Units (TUs). The occurrence distribution of TU is the TS. One of the first and most used TU-based descriptors is the Local Binary Pattern (LBP) [15]. This last one is a two-level TU, gray-scale invariant and easily combined with a simple contrast measure. One of the main characteristics is its robust invariant to light changes.

Another method based in the TU is the Simplified Texture Unit (STU) [20]. This method use a more reduced range of values without a significant loss of the characterization power. This way, there are two options of STU: using the crosswide neighbors (up, right, down, and left) and using diagonal neighbors (up-left, up-right, down-right, and down-left); its reduced length is commonly used in real-time applications obtaining similar performance to LBP.

The modified texture spectrum (MTS) [21] can be considered as a simplified version of LBP, where only a subset of the peripheral pixels (up-left, up, up-right, and right) are considered. Its TS is 16 elements in length, significantly improving the computation efficiency on classification. Similarly to STU, the reduction on the TS length leads to a faster classification while achieving similar performance.

The GaborLBP [22] considers the advantages of the Gabor filters in computer vision and exploits them. It first applies a Gabor transformation and encodes the magnitude values with the LBP operator. Fusing both tools enables handling of illumination changes, viewpoint angle changes, and non-rigid bodies. Usually this combination is used for face recognition or person identification.

The Local Ternary Pattern (LTP) [23] is a generalization of the LBP and it is more discriminant and less sensitive to noise in uniform regions. It is a local texture descriptor that uses a 3-value coding that thresholds around zero. Comparing to the LBP, LTP is more resistant to noise but no longer invariant to gray-level transformations.

The Binary Gradient Contours (BGC) [24] is a binary 8-tuple. It relies on computing a set of eight binary gradients between pairs of pixels all along a closed path around the central pixel of a 3×3 grayscale image patch. They defined the closed path in three different ways: single-loop (BGC1), double-loop (BGC2), and triple-loop (BGC3).

Another HEP descriptor, is the Local Quantized Patterns (LQP) [25]. This is a generalization of local pattern features that makes use of vector quantization. It uses large local neighbourhoods and/or deeper quantization with domain-adaptative vector quantization.

The Weber's Law Descriptor (WLD) [26] was proposed in 2010 as a simple, yet very powerful and robust descriptor. It is based on the fact that human pattern perception also depends on the original intensity of the stimulus and not only on the change of a stimulus (such as sound and lighting). It is composed of two components: differential excitation and orientation.

The Histogram of oriented gradients (HOG) [27] is a feature descriptor that counts the occurrences of gradient orientation in localized portions of an image. Operating on local cells provides invariation to geometric and photometric transformations. The HOG descriptor is particularly suited for human detection in images. Even if HOG is not part of HEP, the way it generates the descriptor (calculating a histogram of gradients) works similar to HEP methods so it can be used similarly.

2.2. *Classifiers*

Descriptors are used to obtain features from images. Those features are then used by the classifiers to predict which object is on each image. Many machine learning algorithms are used for classifying images, but some of the most popular ones are K-Nearest Neighbors, Naive Bayes, Random Forest, Support Vector machine, Random Committee, Bagging, and Multiclass Classifier.

The Nearest Neighbor Rule is a well-known algorithm and the simplest nonparametric decision procedure that assigns to the uncategorized object the label of the closest sample of the training set. In 1967, a modification of this algorithm led to one of the most used classification algorithms, the K-Nearest Neighbors (KNN) [28]. It is based on looking for closest points and classifying them as the majority class. For a given set of n pairs $(x_1, \theta_1), ..., (x_n, \theta_n)$, where x_i is in a metric space X and θ_i is the category that x_i belongs to from a subset $\{1, 2, ..., M\}$, a new arriving instance x is analyzed to estimate its corresponding class θ. This estimation is done by looking for the nearest neighbor $X'_n \in (x_1, x_2, ..., x_n)$:

$$min\, d(x_i, x) = d(x'_n, x) i = 1, 2, ..., n$$

where d is a distance metric according to the space X. The new instance x will be assigned to the category θ'_n. This is the basic 1-NN. In general, KNN rule decides x belongs to the category of majority vote of the nearest k neighbors.

The Naive Bayes [29], the simplest Bayesian classifier, is another classification algorithm that is often used for its simplicity. It is based on the Bayesian Rule and assumes that variables are independent

given the class. Despite this unrealistic assumption, it is successful in practice. The Bayesian rule states that the probability that a instance x belongs to class C_k is

$$P(C_k|\mathbf{x}) = \frac{P(C_k)P(\mathbf{x}|C_k)}{P(\mathbf{x})} \tag{1}$$

where C_k is the class between the K possible classes and x the instance to be classified. Taking into account the independence assumption, the conditional distribution over the class variable C is

$$p(C_k|x_1, ..., x_n) = \frac{1}{Z}p(C_k)\prod_{i=1}^{n} p(x_i|C_k) \tag{2}$$

The instance is classified as the class with more $p(C_k|x_1, ..., x_n)$.

The Random Forest (RF) [30] is a combination of decision trees that use random subsets of the features to be built. Figure 1 shows an example of RF.

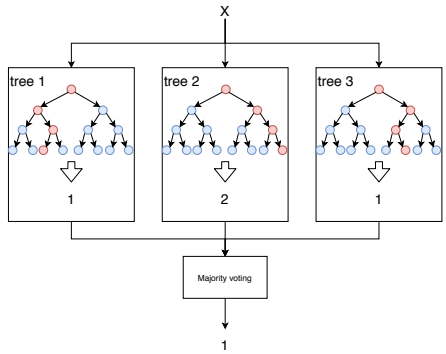


Figure 1. Random Forest example where each tree classifies the new instance and the resulting class is decided by majority voting.

Support Vector Machines (SVM) [31] are supervised learning models that look for optimal hyperplanes that separates classes. An optimal hyperplane is defined as the linear decision function with maximal margin between the vectors of the two classes (Figure 2).

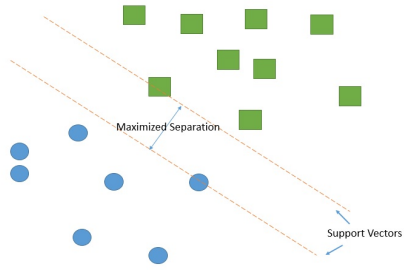


Figure 2. Support vector machine: maximum separation between two classes.

Random Committee (RC) [32] is a committee of random classifiers. The base randomizable classifiers (that form the committee members) are built using different random number seeds based in the same data. The final prediction is a straight average of the predictions generated by the individual base classifiers.

The Bagging [33] technique is called after Bootstrap aggregating. This machine learning ensemble that can be used to improve the stability of a model by improving the accuracy and reducing variance in order to reduce overfitting.

2.3. Feature Selection

As stated before, the crux of the matter in this paper relies on how to select the different visual features to improve the individual score of each descriptor. Some authors have used different techniques to do this [34,35]. Feature Selection is a machine learning technique that is used in many fields and usually improves the accuracy of the model. In [34], the authors uses different feature selection techniques to improve the score in the Quantitative Structure–Activity Relationship (QSAR). In [35], instead, they use a similar approach for hand pose recognition. In [36], a view over the different feature selection techniques and its variations is described. Our approach is based in those methods and is used in a completely different context.

3. Proposed Approach

In order to achieve a better performance than just using a single global descriptor, we propose using a Descriptor Subset Selector. That is, we try to find the combination of global descriptors that scores a better result. Among all available options of subset selection, we have used 2 for their greedy approach which achieve a significant performance: forward selection and backward selection. First, we present the classification of a single image, given a descriptor and a classifier. After that, we explain the feature selection techniques to choose the combination of descriptor to use. Next, we present the evaluation methods, in order to decide which is the best solution. Finally, we present the whole pipeline of the proposed approach.

3.1. Classification

The first step in the pipeline is to classify a picture into the C different classes. Given a descriptor and a classifier, the classifier is trained with features obtained from the description of the set of images for training. Given a new image to be classified, the descriptor extracts the feature from the image and that feature is classified by the classifier (Figure 3).

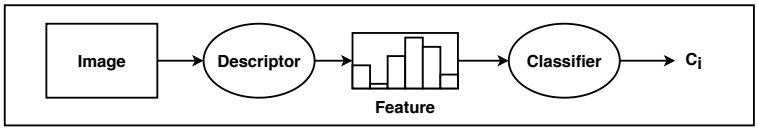


Figure 3. Classification of a new image given a descriptor and a classifier.

3.2. Feature Selection Techniques

The feature selection techniques are used to chose the descriptors for the classification. An exhaustive search of best combination of descriptors is computationally inefficient, while it guarantees that the optimal solution is achieved. Nevertheless, a suboptimal solution can be achieved using a sequential search. This is an iterative search that once a stage of the search is reached, is impossible to go back. The complexity of the exhaustive search is exponential ($O(2^n)$), while the sequential search remains polynomial ($O(n^{k+1})$), where k is the number of evaluated subsets in each stage. This last one does not guarantee an optimal solution.

Another important consideration in the feature selection techniques is the generation of the successors, i.e., how to select the next candidates for the following stage. The simplest and most used methods are Forward and Backward generation [36]. In forward generation, on each stage the element which makes J (the evaluation measure) greater is selected and added to the selected subset. For example, the first descriptor added to the subset would be the one with the best individual score. The next stage would add to the subset the one that concatenated with the previous one makes the score greater. We refer to this method as Sequential Forward Subset Selection (SFSS) [36], and its pseudocode is described in Algorithm 1. The backwards is the opposite behavior. The subset is initialized with all the elements and on each stage the element that that makes J greater when removed is done so. The stopping criteria in both cases can be that J is not increased in j steps or the subset achieves a desired length. We refer to this method as Sequential Backward Subset Selection (SBSS) [36], and its pseudocode is described in Algorithm 2.

Algorithm 1: Sequential Forward Subset Selection

Input :
X—Set of elements
J—evaluation metric
Output:
X'—solution found

$X' = \emptyset$
repeat
  $x' := argmax\{J(X' \cup x) | x \in (X \setminus X')\}$
  $X' := X' \cup \{x'\}$
until *not improvement in* J ***OR*** $X' = X$;

where $\cup$ stands for union between two sets or an element and a set and $\setminus$ operator stands for difference.

Algorithm 2: Sequential Backward Subset Selection

Input :
X—Set of elements
J—evaluation metric
Output:
X'—solution found

$X' = X$
repeat
  $x' := argmax\{J(X' \setminus x) | x \in X'\}$
  $X' := X' \setminus \{x'\}$
until *not improvement in* J ***OR*** $X' = \emptyset$;

3.3. Evaluation Measure

A classification quality can be quantified using measures such the one of Equation (3). This measure, named F-value [37] or F-score, is an evaluation measure that takes into account the precision and the recall. More precisely, the metric used is a particular case of the F-value where the precision and the recall are balanced. This is called F_1, an harmonic mean between the precision and the recall.

$$F_1(y) = 2 \cdot \frac{precision_y * recall_y}{precision_y + recall_y} \tag{3}$$

where y refers to a class (also referred in this paper as C_i). F_1 is class-dependent, so for each class, y, the precision and the recall are computed for that class. The precision (Equation (4)) is the ratio between the correctly predicted views with label y (tp_y or true positive) and all predicted views for that given instance ($|\psi(X) = y|$). The recall (Equation (5)), instead, is the relation between correctly predicted views with label y (tp_y or true positive) and all views that should have that label ($|label(X) = y|$).

$$precision_y = \frac{tp_y}{|\psi(X) = y|} \tag{4}$$

$$recall_y = \frac{tp_y}{|label(X) = y|} \tag{5}$$

To evaluate each stage of the feature selection we use the averaged F_1. This is the mean of the F_1's of all the classes (Equation (6)).

$$F_1 = \frac{1}{|Y|} \sum_{y \in Y} F_1(y) \tag{6}$$

3.4. Full Pipeline

The dataset is divided in two sets: training and test. During the search of the best combination of descriptors, training set is used for training the classifiers and validate the feature selection technique. This separation is made by a Leave-One-Out Cross-Validation (LOOCV) [38]. Each image of the set is used as validation while the rest of the set is used to train the model. Figure 4 shows the whole process. Given a descriptor and a classifier, both are tested using the LOOCV to set the training and validation sets. Once the best combination of descriptors is found, to test the quality of this combination, we use the test set to obtain a general evaluation metric.

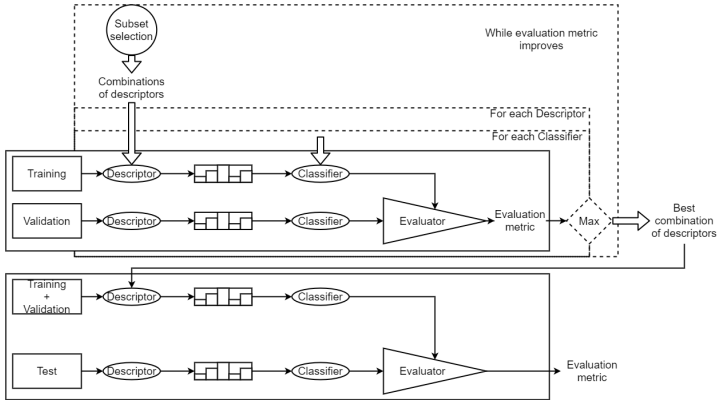


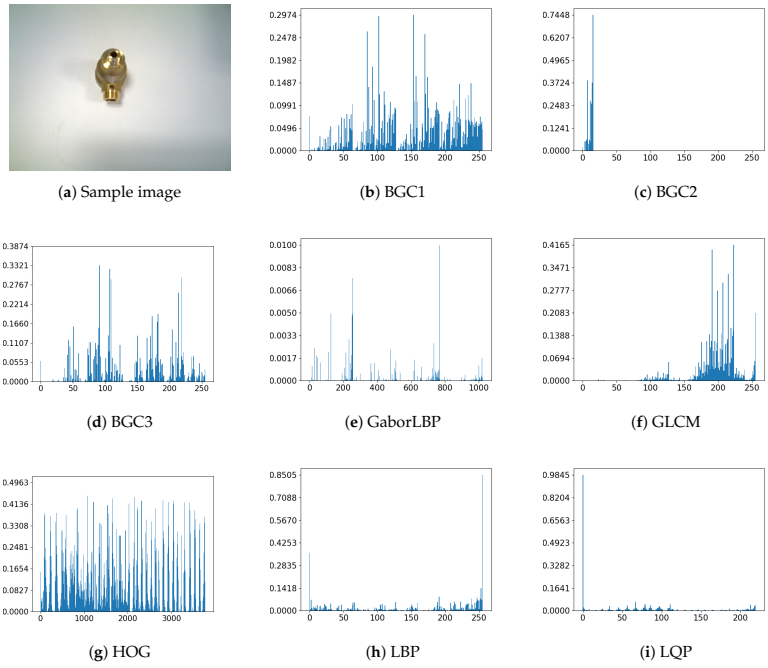
Figure 4. Full pipeline of the proposed method, including training, validation, and evaluation.

4. Experiments and Results

As stated before, the aim of this paper is to present a method to improve the accuracy on reduced datasets of texture-less objects. In order to prove that our method improves the score of the descriptors by their own, we have created a small dataset composed by seven different random industrial parts (Figure 5). We took 50 pictures of each industrial part taken from different viewpoints and different illumination conditions. Objects are rotated and translated but all images are free from occlusion, and with an empty and white background.

Figure 5. Pictures of the parts used in the experiment.

Our pool of descriptors D for discovering the best combination is made up of BGC1 BGC2, BGC3, LBP, GaborLBP, GLCM, HOG, LQP, LTP, MTS, STU+ (or STU1), STU $\times$ (or STU2), and WLD. All descriptors but HOG are computed on grids of different sizes: 1×1, 4×4, and 8×8. The length of gridded histograms is the length of the descriptor multiplied by the number of grids. The HOG is applied to the whole image directly. Figure 6 shows a sample image from our database that has been described by each of the descriptors.

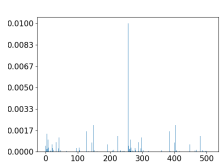


(**a**) Sample image (**b**) BGC1 (**c**) BGC2

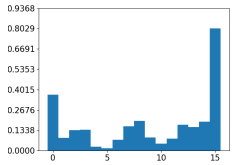
(**d**) BGC3 (**e**) GaborLBP (**f**) GLCM

(**g**) HOG (**h**) LBP (**i**) LQP

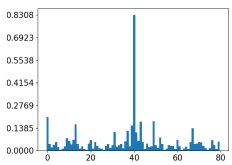
Figure 6. *Cont.*



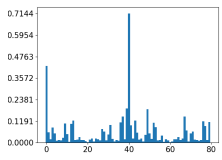
(**j**) LTP



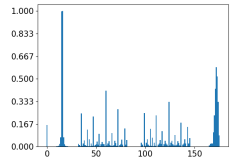
(**k**) MTS



(**l**) STU1



(**m**) STU2



(**n**) WLD

Figure 6. Histogram of all the used descriptors applied to a sample image. The vertical axis represents the number of occurrences of each texture unit normalized and the horizontal axis represents each of the texture units of the histograms. The descriptors are the ones that are part of D described at the beginning of Section 4.

The classifiers used are KNN, NB, SVM 1-vs-1 trained with SMO (Sequential Minimal Optimization [39]), SVM 1-againt-all trained with SGD (Stochastic Gradient Descent [40]), RC, RF, and Bagging. To distinguish between the two SVM implementations, we call SVM to the one trained with SMO and SVM-SGD to the other one. In terms of performance, some of the classifiers are drastically affected by the parameters, but tuning the parameters makes a complex casuistry which is not the aim of this paper. Used parameters are standards and those are given in the Appendix A. The results are obtained for a Intel Xeon CPU of 3GHz and 16GB of RAM, and no GPU acceleration has been used. The following subsections explain the results obtained in the experiments.

4.1. Forward Subset Selection

Forwards Subset Selection of descriptors applied to the whole image (from now on, $FSS1 \times 1$) experiments results are shown in Table 1. In Table 1, the classifier that is between brackets is the one that achieves the highest mean score. If we would use the best descriptor alone, the F_1 would be 0.94 with WLD. By combining it with BGC2 and MTS, and using SVM as classifier, we are able to augment quality of 3% to reach 0.971. On first iteration WLD outperforms the other descriptors with a difference of 0.1 comparing to the next best descriptor. The second iteration increases the overall accuracy and in almost all the cases improves the accuracy of the previous iteration best case.

Table 2 shows the results of the Forwards Subset Selection of descriptors applied to a 4×4 grid ($FSS4 \times 4$). On average, the first iteration performs better than the non-gridded version $FSS4 \times 4$, but the last iteration does not improve the results obtained with $FSS4 \times 4$. The first iteration achieves an F_1 of 0.934 and the final iteration 0.969. Therefore, an improvement of 3.5% is obtained. The final combination of descriptors, the one which achieves the highest score, is composed by STU1 and WLD.

Table 3 shows the results for the 8×8 gridded version ($FSS8 \times 8$). The results are similar to the ones obtained in $FSS4 \times 4$. The first iteration achieves an F_1 of 0.94, while the last one achieves a score of 0.96. In this case, the improvement is 2%.

The performance of the 3 options of the parameters are similar but the speed of the classification is much faster with the $FSS1 \times 1$ version because the length of the final descriptor is shorter. Therefore, the

value of the grid parameter makes not a significant difference in the performance. The recommendation is to use the $FSS1 \times 1$.

Table 1. Forward Subset Selection of descriptors applied to the whole image (also known as $FSS4 \times 4$). Level 1 uses only one descriptor. The following levels concatenate the best descriptor from the previous level to the rest of the descriptors. The algorithm stops on level 4 because the evaluation measure is not improved from level 3 to 4.

Descriptor	Level 1	Level 2	Level 3	Level 4
		WLD +	WLD + BGC2 +	WLD + BGC2 + MTS +
BGC1	0.66 (RF)	0.931 (RF)	0.937 (SVM)	0.934 (SVM/RF)
BGC2	0.489 (SVM)	**0.969 (SVM)**	—	—
BGC3	0.611 (SVM)	0.937 (RC)	0.929 (RF)	0.929 (RF)
GaborLBP	0.671 (RF)	0.931 (RF)	0.903 (RF)	0.906 (RF)
GLCM	0.811 (RF)	0.951 (RF)	0.903 (RF)	0.963 (SVM)
HOG	0.84 (SVM)	0.923 (SVM)	0.923 (SVM)	0.923 (SVM)
LBP	0.611 (RF)	0.946 (RF)	0.917 (SVM)	0.923 (RF)
LQP	0.697 (RF)	0.949 (SVM)	0.954 (SVM)	0.949 (SVM)
LTP	0.563 (RF)	0.966 (SVM)	0.969 (SVM)	**0.966 (SVM)**
MTS	0.666 (KNN)	0.966 (SVM)	**0.971 (SVM)**	—
STU1	0.746 (SVM)	0.951 (SVM)	0.951 (SVM)	0.948 (SVM)
STU2	0.74 (SVM)	0.951 (SVM)	0.957 (SVM)	0.957 (SVM)
WLD	**0.94 (RF)**	—	—	—

Table 2. Forward Subset Selection of descriptors applied to 4×4 gridded image (also known as $FSS4 \times 4$).

Descriptor	Level 1	Level 2	Level 3
		STU1 +	STU1 + WLD +
BGC1	0.877 (SVM)	0.908 (SVM-SGD)	0.934 (SVM)
BGC2	0.903 (SVM)	0.931 (SVM)	**0.969 (SVM)**
BGC3	0.857 (SVM)	0.906 (SVM)	0.931 (SVM)
GaborLBP	0.834 (SVM)	0.883 (SVM)	0.903 (SVM)
GLCM	0.923 (RF)	0.957 (SVM)	0.966 (SVM)
HOG	0.846 (KNN)	0.917 (SVM)	0.94 (SVM)
LBP	0.874 (SVM)	0.906 (SVM)	0.929 (SVM)
LQP	0.911 (SVM)	0.94 (SVM)	**0.969 (SVM)**
LTP	0.889 (SVM)	0.931 (SVM)	0.96 (SVM)
MTS	0.909 (SVM)	0.94 (SVM)	0.96 (SVM)
STU1	**0.934 (SVM)**	—	—
STU2	0.914 (SVM)	0.931 (SVM)	0.96 (SVM)
WLD	0.931 (SVM)	**0.969 (SVM)**	—

Table 3. Forward Subset Selection of descriptors applied to 8×8 gridded image (also known as $FSS8 \times 8$).

Descriptor	Level 1	Level 2	Level 3
		WLD +	WLD + MTS +
BGC1	0.845 (SVM)	0.877 (SVM)	0.897 (SVM)
BGC2	0.911 (SVM)	0.954 (SVM)	0.957 (SVM)
BGC3	0.843 (SVM)	0.863 (RC)	0.877 (SVM)
GaborLBP	0.783 (SVM)	0.849 (Bagging)	0.869 (Bagging)
GLCM	0.909 (SVM)	0.92 (SVM)	0.923 (SVM)
HOG	0.846 (KNN)	0.923 (SVM)	0.909 (SVM)
LBP	0.831 (SVM)	0.886 (Bagging)	0.889 (Bagging)
LQP	0.897 (SVM)	0.929 (SVM)	0.931 (SVM)
LTP	0.9 (SVM)	0.951 (SVM)	**0.954 (SVM)**
MTS	0.903 (SVM)	**0.96 (SVM)**	—
STU1	0.921 (SVM)	0.94 (SVM)	0.949 (SVM)
STU2	0.917 (SVM)	0.929 (SVM)	0.937 (SVM)
WLD	**0.94 (RF)**	—	—

Regarding the classifiers, in almost all the cases the best classifier is the SVM, which is more evident as the number of descriptors concatenated raises. This is because SVM works well with high dimensionality. Our recommendation is to use SVM trained with SMO.

4.2. Backward Subset Selection

The Backward Subset Selection has a similar behavior. Table 4 shows the results of the Backward Subset Selection of descriptors applied to the whole image (*BSS*1 × 1). This is the case with more iterations. It increases the accuracy from 0.917 to 0.937. The resulting descriptor set is composed by BGC2, BGC3, GLCM, LQP, LTP, MTS, STU1, STU2, and WLD.

Table 4. Backward Subset Selection of descriptors applied to 1 × 1 gridded image (also known as *BSS*1 × 1). Level 1 uses all the descriptors in set D concatenated. Level 2 uses the concatenation of the descriptors in D without each of the descriptors. The following levels use the concatenation of the descriptors in D without the descriptor that makes score higher of the previous level.

Descriptor	Level 1	Level 2	Level 3	Level 4	Level 5	Level 6
	D	**D**	**D\ GaborLBP +**	**D\ GaborLBP + HOG +**	**D\ GaborLBP + HOG + BGC1 +**	**D\ GaborLBP + HOG + BGC1 + LBP +**
BGC1		0.92 (SVM)	0.923 (SVM)	**0.929 (SVM)**	—	—
BGC2		0.914 (SVM)	0.92 (SVM)	0.92 (RF)	0.937 (SVM)	0.931 (SVM)
BGC3		0.917 (SVM)	0.92 (SVM)	0.929 (SVM)	0.934 (SVM)	**0.937 (SVM)**
LBP		0.914 (SVM)	0.92 (SVM)	0.926 (RF)	**0.937 (SVM)**	—
GaborLBP		**0.92 (SVM)**	—	—	—	—
GLCM		0.914 (SVM)	0.914 (SVM)	0.92 (SVM)	0.929 (SVM)	0.923 (SVM)
HOG	0.917 (SVM)	0.903 (SVM)	**0.923 (SVM)**	—	—	—
LQP		0.917 (RF)	0.92 (SVM)	0.923 (SVM)	0.923 (SVM)	0.934 (SVM)
LTP		0.909 (SVM)	0.92 (SVM)	0.92 (RF)	0.926 (SVM)	0.929 (SVM)
MTS		0.914 (SVM)	0.92 (SVM)	0.923 (SVM)	0.929 (SVM)	0.931 (SVM)
STU1		0.917 (SVM)	0.92 (SVM)	0.923 (RF)	0.923 (SVM)	0.926 (SVM)
STU2		0.914 (SVM)	0.92 (SVM)	0.926 (RF)	0.931 (SVM)	0.926 (SVM)
WLD		0.914 (SVM)	0.891 (SVM)	0.82 (RF)	0.834 (RF)	0.934 (RF)

Table 5 shows the results of the Backward Subset Selection of descriptors applied to 4 × 4 gridded images (*BSS*4 × 4). This time, the improvement is from 0.943 to 0.954. The resulting descriptor set is BGC1 BGC2, BGC3, GaborLBP, GLCM, HOG, LQP, LTP, MTS, STU2, and WLD.

Table 5. Backward Subset Selection of descriptors applied to 4 × 4 gridded image (also known as *BSS*4 × 4).

Descriptor	Level 1	Level 2	Level 3	Level 4
	D	**D **	**D \ LBP +**	D \ **LBP + STU1 +**
BGC1		0.95 (SVM)	0.951 (SVM)	0.949 (SVM)
BGC2		0.946 (SVM)	0.951 (SVM)	**0.954 (SVM)**
BGC3		0.949 (SVM)	0.95 (SVM)	0.945 (SVM)
LBP		**0.951 (SVM)**	—	—
GaborLBP		0.946 (SVM)	0.951 (RF)	0.949 (RF)
GLCM		0.937 (SVM)	0.937 (SVM)	0.94 (SVM)
HOG	0.943 (SVM)	0.94 (SVM)	0.943 (SVM)	0.946 (SVM)
LQP		0.946 (RF)	0.946 (SVM)	0.946 (SVM)
LTP		0.946 (SVM)	0.951 (SVM)	0.949 (SVM)
MTS		0.946 (SVM)	0.951 (SVM)	**0.954 (SVM)**
STU1		0.946 (SVM)	**0.954 (SVM)**	—
STU2		0.949 (SVM)	0.95 (SVM)	0.949 (SVM)
WLD		0.946 (SVM)	0.946 (SVM)	0.946 (SVM)

Appl. Sci. **2020**, *10*, 3701

Table 6, instead, shows the results of the Backward Subset Selection of descriptors applied to 8×8 gridded images ($BSS8 \times 8$). The first iteration achieves and score of 0.92, while in the last iteration the score is 0.946. The improvement is 0.026. The resulting descriptor set is BGC1 BGC2, GLCM, HOG, LQP, LTP, MTS, STU1, STU2, and WLD.

Table 6. Backward Subset Selection of descriptors applied to 8×8 gridded image (also known as $BSS8 \times 8$).

Descriptor	Level 1	Level 2	Level 3	Level 4	Level 5
	D	**D**	**D\ LBP +**	**D\ LBP + GaborLBP +**	**D\ LBP + GaborLBP + BGC3 +**
BGC1		0.914 (SVM)	0.923 (SVM)	0.943 (SVM)	0.931 (SVM)
BGC2		0.909 (SVM)	0.929 (SVM)	0.934 (SVM)	0.943 (SVM)
BGC3		0.917 (SVM)	0.926 (SVM)	**0.946 (SVM)**	—
LBP		**0.926 (SVM)**	—	—	—
GaborLBP		0.92 (SVM)	**0.934 (RF)**	—	—
GLCM		0.894 (SVM)	0.9 (SVM)	0.917 (SVM)	0.929 (SVM)
HOG	0.92 (SVM)	0.903 (SVM)	0.914 (SVM)	0.934 (SVM)	**0.946 (SVM)**
LQP		0.906 (RF)	0.917 (SVM)	0.931 (SVM)	0.934 (SVM)
LTP		0.909 (SVM)	0.914 (SVM)	0.929 (SVM)	0.934 (SVM)
MTS		0.909 (SVM)	0.937 (SVM)	0.929 (SVM)	**0.946 (SVM)**
STU1		0.906 (SVM)	0.929 (SVM)	0.934 (SVM)	0.937 (SVM)
STU2		0.914 (SVM)	0.929 (SVM)	0.937 (SVM)	0.943 (SVM)
WLD		0.906 (SVM)	0.923 (SVM)	0.931 (SVM)	0.943 (SVM)

4.3. Comparative between Methods

Figure 7 shows a comparative of the highest scores of each iteration of the different selection techniques. The maximum of each technique is obtained in the previous to the last level.

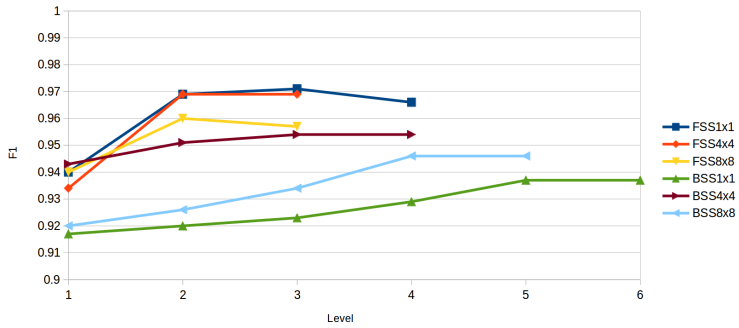


Figure 7. Comparative of highest F1 of each iteration of the Subset Selection techniques. FSS stands for Forward Subset Selection and BSS stands for Backward Subset Selection. The numbers after each selection technique stand for the number of windows the descriptor has been applied to.

On general, Forward Subset Selection achieves better results than Backward Subset Selection. Backward selection computation time is higher than forward so is preferable to use a forward selection since computation time is shorter and the performance is better.

We have compared our method with two known CNN methods: Xception [41] and Siamese [42]. Xception is a Deep learning network inspired by Inception [43], where Inception modules, treated as

intermediate step in-between regular convolutions, are replaced by depthwise separable convolutions. Siamese network, instead, is a Convolutional network that inputs two images and classifies if the two images are the same object. One of its advantages is that it gives good results even with small datasets.

Table 7 shows a comparison between the proposed method and the two previous described methods. In the case of the Xception, the results are not as good as Siamese or our proposal as Xception works better for large datasets. Even if Siamese works better than Xception, it does not give better results than our proposal.

Table 7. Comparison between standard DL methods and our proposal.

Method	F1
Xception	0.35
Siamese	0.89
Our proposal ($FSS1 \times 1$)	0.97

In terms of speed, Figure 8 shows a comparative of the test time for each of the methods. The time shown is the average of the different descriptors and classification techniques for testing one image. $FSS1 \times 1$, $FSS4 \times 4$, and $BSS1 \times 1$ have a low computation time, and $BSS8 \times 8$ version performs much slower than the rest of versions due to the high dimensionallity of the data.

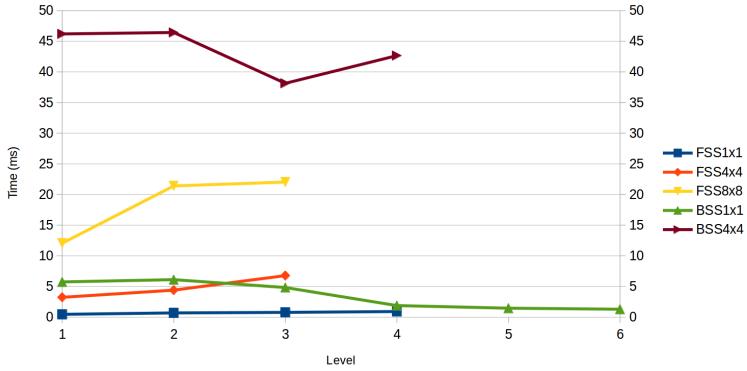


Figure 8. Times to classify an image with the different Subset selection methods on each level. The times on this Figure correspond to the average time that the classifier needs to classify an image using each descriptor on that level. The times of the $BSS8 \times 8$ are not shown since its values are around 200 ms and distorts the plot.

Taking into account the speed and score the best option is to use $FSS1 \times 1$. Although score is similar to the rest of the versions, it outperforms remaining in terms of speed.

5. Conclusions

The main two problems we have to deal with in computer vision in an industrial context is the complexity of the objects, that is, their unusual shape and texture-less objects, and the lack of large datasets to train CNNs that can handle the previous problem. To manage this situation, we proposed in this paper an approach for selecting the best combination of descriptor that, together, provides a better classification. Even if more than one descriptors has to been calculated, this method is still fast enough for real-time applications.

The proposed method is a greedy approach that iteratively adds (Forward Subset Selection) or removes (Backward Subset Selection) descriptors to the solution until performance is not improved. The resulting descriptor set always improves the quality of the classification comparing to the best descriptor by its own. This selection techniques can be extended to different datasets and contexts as proved within this paper and previous ones [34–36].

The used dataset for the experiments is composed by seven typical industrial texture-less objects. The proposed method achieves a state-of-the-art classification quality for that given dataset. Our method achieves a F1 of 0.971, 3% more than the best descriptor alone. Description and classification of a new image can be achieved in real-time applications, given its low processing time (between 10 and 50 ms).

The next steps will include a larger set of descriptors and DL networks in order to mix both classical and Neural Network methods. As this particular application is within a bigger industry 4.0 set-up, the following works will include not only a visual approach, but the application as a whole.

Author Contributions: Conceptualization, I.M., J.A. A.R., and B.S.; methodology, B.S.; software, I.M.; validation, J.A., A.R., and B.S.; formal analysis, I.M.; investigation, I.M.; resources, I.M.; data curation, I.M.; writing—original draft preparation, I.M.; writing—review and editing, I.M., J.A., A.R., and B.S.; visualization, I.M.; supervision, B.S.; project administration, J.A.; funding acquisition, J.A. All authors have read and agreed to the published version of the manuscript.

Funding: This paper has been supported by the project SHERLOCK under the European Union's Horizon 2020 Research & Innovation programme, grant agreement No. 820689.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A. Parameters

Table A1. Parameters of the methods.

Algorithm	Parameter	Description of the Parameter	Value
KNN	K	number of neighbors	1
SVM SMO	C	parameter C	1.0
	L	tolerance	0.001
	P	epsilon for round-off error	1.0×10^{-12}
	N	Normalization	true
	V	calibration folds	-1
	K	Kernel	PolyKernel
	C PolyKernel	Cache size of the kernel	250,007
	E PolyKernel	Exponent value of the kernel	2.0
SVM SGD	M	Multiclass type	1-against-all
	F	Loss function	hinge loss
	L	Learning rate	0.001
	R	Regulation constant	0.0001
	E	Number of epochs to perform	500
	C	Epsilon threshold for loss function	0.001
RC	W	The base classifier to be used	RandomTree
	K	Number of choosen attributes in the RandomTree	$int(log_2(predictors) + 1)$
	M RandomTree	Minimum total weight in a leaf	1.0
	V RandomTree	Minimum proportion of variance	0.001

Table A1. *Cont.*

Algorithm	Parameter	Description of the Parameter	Value
RF	P	Size of each bag	100
	I	Number of iterations	100
	K	Number of randomly choosen attributes	$int(log_2(predictors)+1)$
	M RandomTree	Minimum total weight in a leaf	1.0
	V RandomTree	Minimum proportion of variance	0.001
Bagging	P	Size of each bag	100
	I	Number of iterations	10
	W	The base classifier to be used	REPTree (Fast Decision Tree)
	M REPTree	Minimum total weight in a leaf	2
	V REPTree	Minimum proportion of variance	0.001
	N REPTree	Amount of data used for prunning	3
	L REPTree	Maximum depth of the tree	-1 (no restriction)
	I REPTree	Initial class value count	0.0

References

1. Teichmann, M.; Weber, M.; Zoellner, M.; Cipolla, R.; Urtasun, R. MultiNet: Real-time Joint Semantic Reasoning for Autonomous Driving. *arXiv* **2016**, arXiv:cs.CV/1612.07695.
2. Ronneberger, O.; Fischer, P.; Brox, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. *arXiv* **2015**, arXiv:cs.CV/1505.04597.
3. Yan, M.; Zhao, M.; Xu, Z.; Zhang, Q.; Wang, G.; Su, Z. VarGFaceNet: An Efficient Variable Group Convolutional Neural Network for Lightweight Face Recognition. *arXiv* **2019**, arXiv:cs.CV/1910.04985.
4. Liu, Y.; Wang, Y.; Wang, S.; Liang, T.; Zhao, Q.; Tang, Z.; Ling, H. CBNet: A Novel Composite Backbone Network Architecture for Object Detection. *arXiv* **2019**, arXiv:cs.CV/1909.03625.
5. Yuan, Y.; Chen, X.; Wang, J. Object-Contextual Representations for Semantic Segmentation. *arXiv* **2019**, arXiv:cs.CV/1909.11065.
6. Gómez, A.; de la Fuente, D.C.; García, N.; Rosillo, R.; Puche, J. A vision of industry 4.0 from an artificial intelligence point of view. In Proceedings of the 18th International Conference on Artificial Intelligence, Varna, Bulgaria, 25–28 July 2016; p. 407.
7. Chen, C.; Liu, M.Y.; Tuzel, O.; Xiao, J. R-CNN for Small Object Detection. In *Computer Vision—ACCV 2016*; Lai, S.H., Lepetit, V., Nishino, K., Sato, Y., Eds.; Springer International Publishing: Cham, Switzerland, 2017; pp. 214–230.
8. Xie, Q.; Luong, M.T.; Hovy, E.; Le, Q.V. Self-training with Noisy Student improves ImageNet classification. *arXiv* **2020**, arXiv:1911.04252.
9. Cubuk, E.D.; Zoph, B.; Mane, D.; Vasudevan, V.; Le, Q.V. AutoAugment: Learning Augmentation Policies from Data. *arXiv* **2019**, arXiv:1805.09501.
10. Kolesnikov, A.; Beyer, L.; Zhai, X.; Puigcerver, J.; Yung, J.; Gelly, S.; Houlsby, N. Large Scale Learning of General Visual Representations for Transfer. *arXiv* **2019**, arXiv:1912.11370.
11. Wang, J.; Chen, Y.; Yu, H.; Huang, M.; Yang, Q. Easy Transfer Learning By Exploiting Intra-domain Structures. *arXiv* **2019**, arXiv:cs.LG/1904.01376.
12. Nannia, L.; Ghidonia, S.; Brahnamb, S. Handcrafted vs. non-handcrafted features for computer vision classification. *Pattern Recognit.* **2017**, *71*, 158–172.
13. Lowe, D.G. Object recognition from local scale-invariant features. In Proceedings of the Seventh IEEE International Conference on Computer Vision, Kerkyra, Greece, 20–27 September 1999; Volume 2, pp. 1150–1157.
14. Bay, H.; Tuytelaars, T.; Van Gool, L. SURF: Speeded Up Robust Features.In Proceedings of European Conference on Computer Vision, Graz, Austria, 7–13 May 2006; pp. 404–417.
15. Ojala, T.; Pietikäinen, M.; Harwood, D. A comparative study of texture measures with classification based on featured distributions. *Pattern Recognit.* **1996**, *29*, 51–59. [CrossRef]

16. Fernández, A.; Álvarez, M.X.; Bianconi, F. Texture Description Through Histograms of Equivalent Patterns. *J. Math. Imaging Vis.* **2013**, *45*, 76–102. [CrossRef]
17. Merino, I.; Azpiazu, J.; Remazeilles, A.; Sierra, B. 2D Features-based Detector and Descriptor Selection System for Hierarchical Recognition of Industrial Parts. *IJAIA* **2019**, *10*, 1–13. [CrossRef]
18. Haralick, R.M.; Shanmugam, K.; Dinstein, I. Textural Features for Image Classification. *IEEE Trans. Syst. Man Cybern.* **1973**, *SMC-3*, 610–621. [CrossRef]
19. Wang, L.; He, D.C. Texture classification using texture spectrum. *Pattern Recognit.* **1990**, *23*, 905–910.
20. Madrid-Cuevas, F.J.; Medina, R.; Prieto, M.; Fernández, N.L.; Carmona, A. Simplified Texture Unit: A New Descriptor of the Local Texture in Gray-Level Images. In *Pattern Recognition and Image Analysis*; Springer Berlin Heidelberg: Berlin/Heidelberg, Germany, 2003; Volume 2652, pp. 470–477.
21. Xu, B.; Gong, P.; Seto, E.; Spear, R. Comparison of Gray-Level Reduction and Different Texture Spectrum Encoding Methods for Land-Use Classification Using a Panchromatic Ikonos Image. *Photogramm. Eng. Remote Sens.* **2003**, *69*, 529–536.
22. Zhang, W.; Shan, S.; Gao, W.; Chen, X.; Zhang, H. Local Gabor binary pattern histogram sequence (LGBPHS): A novel non-statistical model for face representation and recognition. In Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1, Beijing, China, 17–21 October 2005; Volume 1, pp. 786–791.
23. Tan, X.; Triggs, W. Enhanced local texture feature sets for face recognition under difficult lighting conditions. *IEEE Trans. Image Process.* **2010**, *19*, 1635–1650.
24. Fernández, A.; Álvarez, M.X.; Bianconi, F. Image classification with binary gradient contours. *Opt. Lasers Eng.* **2011**, *49*, 1177–1184.
25. Hussain, S.U.; Napoléon, T.; Jurie, F. Face Recognition using Local Quantized Patterns. In *Procedings of the British Machine Vision Conference 2012*; British Machine Vision Association: Guildford, UK, 2012; pp. 99.1–99.11.
26. Chen, J.; Shan, S.; He, C.; Zhao, G.; Pietikainen, M.; Chen, X.; Gao, W. WLD: A Robust Local Image Descriptor. *IEEE Trans. Pattern Anal. Mach. Intell.* **2010**, *32*, 1705–1720. [CrossRef]
27. Dalal, N.; Triggs, B. Histograms of oriented gradients for human detection. In Proceedings of the 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05), San Diego, CA, USA, 20–25 June 2005; IEEE: Piscataway, NJ, USA, 2005; Volume 1, pp. 886–893.
28. Cover, T.; Hart, P. Nearest neighbor pattern classification. *IEEE Trans. Inf. Theory* **1967**, *13*, 21–27.
29. Rish, I. An empirical study of the naive Bayes classifier. In *IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence*; IBM: New York, NY, USA, 2001; Volume 3, pp. 41–46.
30. Breiman, L. Random forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
31. Cortes, C.; Vapnik, V. Support-vector networks. *Mach. Learn.* **1995**, *20*, 273–297. [CrossRef]
32. Lira, M.M.S.; de Aquino, R.R.B.; Ferreira, A.A.; Carvalho, M.A.; Neto, O.N.; Santos, G.S.M. Combining Multiple Artificial Neural Networks Using Random Committee to Decide upon Electrical Disturbance Classification. In Proceedings of the 2007 International Joint Conference on Neural Networks, Orlando, FL, USA, 12–17 August 2007; pp. 2863–2868.
33. Breiman, L. Bagging predictors. *Mach. Learn.* **1996**, *24*, 123–140. [CrossRef]
34. Shahlaei, M. Descriptor Selection Methods in Quantitative Structure–Activity Relationship Studies: A Review Study. *Chem. Rev.* **2013**, *113*, 8093–8103. [CrossRef] [PubMed]
35. Rasines, I.; Remazeilles, A.; Bengoa, P.M.I. Feature selection for hand pose recognition in human-robot object exchange scenario. In Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA), Barcelona, Spain, 16–19 September 2014; pp. 1–8.
36. Molina, L.; Belanche, L.; Nebot, A. Feature selection algorithms: A survey and experimental evaluation. In Proceedings of the 2002 IEEE International Conference on Data Mining, Maebashi City, Japan, 9–12 December 2002; pp. 306–313. [CrossRef]
37. Chinchor, N. MUC-4 Evaluation Metrics. In *Proceedings of the 4th Conference on Message Understanding*; Association for Computational Linguistics: Stroudsburg, PA, USA, 1992; pp. 22–29. [CrossRef]
38. Forman, G.; Scholz, M. Apples-to-Apples in Cross-Validation Studies: Pitfalls in Classifier Performance Measurement. *SIGKDD Explor. Newsl.* **2010**, *12*, 49–57. [CrossRef]
39. Platt, J.C. *Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines*; MIT Press: Cambridge, MA, USA, 1998.

40. Robbins, H.; Monro, S. A Stochastic Approximation Method. *Ann. Math. Statist.* **1951**, *22*, 400–407. [CrossRef]
41. Chollet, F. Xception: Deep Learning With Depthwise Separable Convolutions. In Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009; pp. 248–255.
42. Melekhov, I.; Kannala, J.; Rahtu, E. Siamese network features for image matching. In Proceedings of the 2016 23rd International Conference on Pattern Recognition (ICPR), Cancun, Mexico, 4–8 December 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 378–383.
43. Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, Z. Rethinking the Inception Architecture for Computer Vision. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26 June–1 July 2016; pp. 2818–2826.

CAPÍTULO 8

3D Convolutional Neural Networks Initialized from Pretrained 2D Convolutional Neural Networks for Classification of Industrial Parts

Título:	3D Convolutional Neural Networks Initialized from Pretrained 2D Convolutional Neural Networks for Classification of Industrial Parts
Autores:	I. Merino, J. Azpiazu, A. Remazeilles, B. Sierra
Revista:	Sensors
Volumen:	21
Número:	1078
Editor:	MDPI
DOI:	10.3390/s21041078
Año:	2021
Cuartil (Scimago/WoS):	Q2 (Electrical and electronic engineering) / Q2 (Engineering, electrical & electronic)

sensors

MDPI

Article

3D Convolutional Neural Networks Initialized from Pretrained 2D Convolutional Neural Networks for Classification of Industrial Parts

Ibon Merino [1,2,*], Jon Azpiazu [1], Anthony Remazeilles [1] and Basilio Sierra [2]

[1] TECNALIA, Basque Research and Technology Alliance (BRTA), Mikeletegi Pasealekua 7, 20009 Donostia-San Sebastián, Spain; jon.azpiazu@tecnalia.com (J.A.); anthony.remazeilles@tecnalia.com (A.R.)
[2] Robotics and Autonomous Systems Group, Universidad del País Vasco/Euskal Herriko Unibertsitatea, 48940 Basque, Spain; b.sierra@ehu.eus
* Correspondence: ibon.merino@tecnalia.com

Abstract: Deep learning methods have been successfully applied to image processing, mainly using 2D vision sensors. Recently, the rise of depth cameras and other similar 3D sensors has opened the field for new perception techniques. Nevertheless, 3D convolutional neural networks perform slightly worse than other 3D deep learning methods, and even worse than their 2D version. In this paper, we propose to improve 3D deep learning results by transferring the pretrained weights learned in 2D networks to their corresponding 3D version. Using an industrial object recognition context, we have analyzed different combinations of 3D convolutional networks (VGG16, ResNet, Inception ResNet, and EfficientNet), comparing the recognition accuracy. The highest accuracy is obtained with EfficientNetB0 using extrusion with an accuracy of 0.9217, which gives comparable results to state-of-the art methods. We also observed that the transfer approach enabled to improve the accuracy of the Inception ResNet 3D version up to 18% with respect to the score of the 3D approach alone.

Keywords: computer vision; deep learning; transfer learning; object recognition

Citation: Merino, I.; Azpiazu, J.; Remazeilles, A.; Sierra, B. 3D Convolutional Neural Networks Initialized from Pretrained 2D Convolutional Neural Networks for Classification of Industrial Parts. *Sensors* **2021**, *21*, 1078. https://doi.org/10.3390/s21041078

Academic Editor: Sheryl Berlin Brahnam
Received: 28 December 2020
Accepted: 2 February 2021
Published: 4 February 2021

1. Introduction

Industrial processes are continuously changing and now digitalization and smart automation are the main focuses to improve performance and productivity of the industrial plants. Robotics, combined with computer science techniques, such as machine learning, have boosted exponentially the production and security. This industrial revolution has been named *Industry 4.0*. Many fields have been integrated to this paradigm. One of them is computer vision.

Computer vision is a sub-field of machine learning consisting in acquiring, processing, analyzing and understanding images of the real world in order to generate information that a computer can deal with. As in all artificial intelligence fields, deep learning techniques are extensively used nowadays in computer vision. Deep learning approaches usually need a big dataset to obtain significant results. In order to meet this requirement, some techniques use deep learning nets that have been trained with huge datasets and transfer that learned knowledge to smaller or different datasets. Those techniques are called transfer learning [1,2].

In industrial application, and particularly in SMEs, or when small production batches are targeted, making huge datasets can be too expensive and arduous. In addition, objects to be recognized can be small and uncommon. Reducing the number of images needed for training is critical in this context. Using transfer learning methods can help to reduce computing time and the minimum dataset size that is needed to obtain significant results.

In parallel, in recent years, 3D cameras are gaining more and more popularity, specially in robotic applications. Working with 3D data is a relatively new paradigm that 2D convolutional networks cannot handle so easily. Therefore, new deep learning methods have been designed to deal with this paradigm [3,4], like the 3D convolutional networks.

Our proposal is a transfer learning technique that relies on using 2D features learned by 2D convolutional nets to train a 3D convolutional net.

The rest of this paper is organized as follows: Section 2 outlines the related works, Section 3 details the proposed approach, Section 4 describes the training phase of the network, Section 5 shows the experimental results obtained, and Section 6 summarizes the conclusions.

2. Related Works

Recent improvements in computing power and the rapid development of more affordable 3D sensors, have opened a new paradigm where 3D data, such as point clouds, are providing better understanding of the environment. Even if some advances have been done in deep learning on point clouds, this is still an underdeveloped field compared to 2D deep learning [3].

Dealing with 3D data in deep learning opens many new fronts. For example, 3D data is difficult to label, so that a significant time is required to label training data. Therefore, usually, the size of the training set for 3D approaches is notably smaller than the one used with 2D techniques.

Many different Convolutional Neural Networks (CNN) have been possible and gained a great success due to the large amount of public image repositories, such ImageNet [5,6] and high-performance computing systems, like GPUs.

Two-dimensional CNN have been widely studied, and there are many successful methods, but 3D CNN still need more research, as we will show in the next sections.

2.1. 2D CNN

The most important deep learning architectures are identified through the ImageNet Large Scale Visual Recognition Competition (ILSVRC) [6]. One of the first ones winning this competition is Alexnet [5]. ZFNet [7] and OverFeat [8] followed Alexnet, improving the results they obtain for the ImageNet dataset.

Understanding of convolutional layers is improved by Reference [7], thanks to their visualization. The following architectures focused on extracting features on low spatial resolutions. One of them is VGG [9], which is still being used as a base to many other architectures because of its simple and homogeneous topology. VGG scored the second place in the ILSVRC 2014. The first place was achieved by GoogLeNet [10], also known as Inception Network. This network was an improvement of the AlexNet, reducing the number of parameters while being much deeper. They introduced the Inception module, which enabled to recognize patterns of different sizes within the same layer, concurrently performing several convolutions of different receptive fields and combining the results.

Another influential architecture was introduced by Reference [11] named the Residual blocks. The architecture called ResNet introduced those Residual blocks which include a skip connection on a convolution block that is merged by summation with the output of that block. This network won the ILSVRC 2015 localization and classification contests and also the COCO detection and segmentation challenges [12].

A modification of the GoogLeNet called Inception-v4 [13] included an improvement on the inception module and three different kinds of inception modules. In addition, this paper also presents a combination of the inception module with the residual connection, named Inception-ResNet, resulting in a more efficient network. Another network similar to the previous one, the ResNeXts [14], achieved the second place in the 2016 ILSVRC classification challenge. The first place in classification, localization, and detection challenges was achieved by ResNet101, Inception-v4, and Inception-ResNet-v1, respectively.

Due to the success of the Inception and Residual modules, many subsequent networks have been derived from them. For example, DenseNets [15] combine the output of the residual connection and the output of the residual block by depth wise filter concatenation. The 2017 ILSVRC localization challenge's first place and the top 3 in classification and detection categories were won by Dual Path Network (DPN) [16], a network that combines the architectures of DenseNets and ResNet.

Since previous networks focus on achieving the highest possible accuracy, they are not prepared for real-time applications with restricted hardware, like mobile platforms. MobileNets [17] tackles this problem by replacing standard convolutions with Depthwise Separable Convolutions.

Recently, Reference [18] proposed a novel scaling method that uniformly scales network's depth, width, and resolution, obtaining a new family of models called EfficientNet. This family achieves much better accuracy with a 6.1× gain factor in computation time and a 8.4× factor in size reduction compared to previous ConvNets.

2.2. 3D CNN

Some researchers have taken advantage of the fact that 2D deep learning is more mature than 3D deep learning, trying to obtain a solution to 3D based on 2D deep learning.

Recently, the arrival of RGB-D sensors, such as the Microsoft's Kinect or the Intel's Realsense, has enabled to acquire at a low cost 3D information. These sensors provide a 2D color image (RGB), along with a depth map (D), which provides the 3-dimensional information. Since both RGB and D are 2D images, 2D deep learning methods can be adapted to receive as input two images instead of one. Even if this representation is quite simple, they are effective for different tasks, such as human pose regression [19], 6D pose estimation [20], or object detection [21].

Despite representing 3D data, RGB-D images are composed by 2D data and no transformation is needed. One possible transformation as proposed in Reference [22,23], consists of projecting the 3D data into another 2D space while keeping some of the original 3D shape key properties.

To keep 3D data without transforming it to 2D, some works, like Reference [24,25], propose a Voxel-based method. Voxels are used to describe how the 3D object is distributed in the three dimensions of the space. This representation is not always the best option since it stores both the occupied and non-occupied parts of the scene. Voxel-based methods are not recommended for high-resolution data since they store a huge unnecessary amount of data. To deal with this problem, octree-based methods with varying sized voxels [26,27] are proposed.

In order to reduce the number of parameters, which is too high in voxel-based methods, some methods propose point-based methods that include point cloud as an unordered set of points as input [28,29].

Our proposal changes this perspective. We adapt the 2D deep learning architecture to 3D and transform the weights from 2D to 3D as initial weights for the newly generated 3D Convolutional model. This approach makes it possible to leverage on existing nets trained on 2D data and apply them on 3D data while maintaining the original data structure.

3. Proposed Approach

Due to the great success of 2D CNN in computer vision, our proposal uses those nets as the base to train a 3D CNN for classification. Figure 1 shows the overview of the proposed architecture. First, the weights of a pre-trained 2D CNN are transformed to 3D to be, therefore, used as the weights of the 3D CNN. The input point cloud is discretized by computing a voxel grid. That grid is the input tensor to the 3D CNN, which is an adapted form of the 2D CNN using 3D layers instead of 2D layers. That 3D CNN computes the 3D features that are then passed on to the classifier.

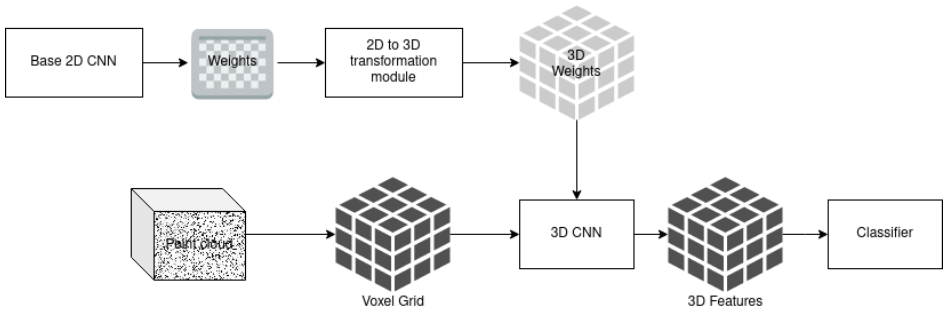


Figure 1. Overall architecture of the proposed method.

The following subsections explains the different modules of the architecture.

3.1. 2D to 3D Transformations

CNN weights can be represented as 2D matrices. Thus, we need to transform a 2D matrix into a 3D tensor, i.e., map the function $M(x, y) = (r, g, b)$ to $T(x, y, z) = (r, g, b)$, where $x, y \in \mathbb{N}$, and $r, g, b \in \mathbb{R}$. For each value of x and y of the 2D matrix and each of the new possible z values the transformation function is: $h(x, y, z, M(x, y)) = T(x', y', z') = (r, g, b)$. We have proposed 2 different transformation functions, the extrusion and the rotation.

3.1.1. Extrusion

Extrusion of the plane consists of filling the tensor copying the RGB values along one axis. Given a matrix M of size $(W \times H)$ and the resulting tensor T of size $(W \times H \times D)$ and the fact that we use inputs that have all the dimensions of the same length, this is, $W = H = D$, the Extrusion mapping is defined as:

$$\forall x, y, z \leq W : T(x, y, z) = M(x, y).$$

The extrusion can be done along the three main axes:

- Z axis: $T(x, y, z) = M(x, y)$,
- Y Axis: $T(x, z, y) = M(x, y)$,
- X Axis: $T(z, x, y) = M(x, y)$.

Figure 2 shows how the extrusion along the Z axis is done for a matrix.

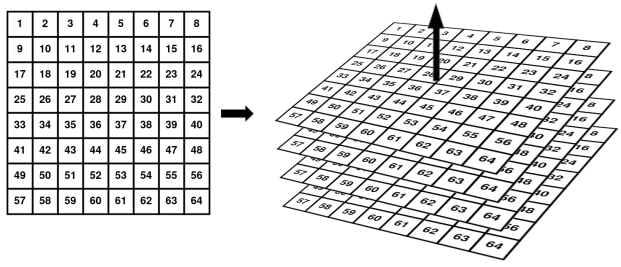


Figure 2. Extrusion along Z axis of a 2D matrix to generate a 3D tensor.

3.1.2. Rotation

To add curvature to the 2D weights, a rotation from 0 to 90 degrees with respect to the Z axis is applied to the 2D weights. The mapping from 2D to 3D is defined as:

$$T(x, y, z) = M(x, min(\left\lfloor \sqrt{y^2 + z^2} \right\rfloor, W)).$$

Figure 3 shows an example of the 2D to 3D rotation transformation.

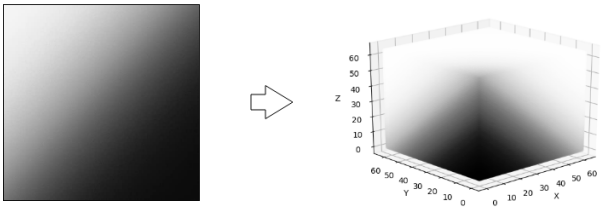
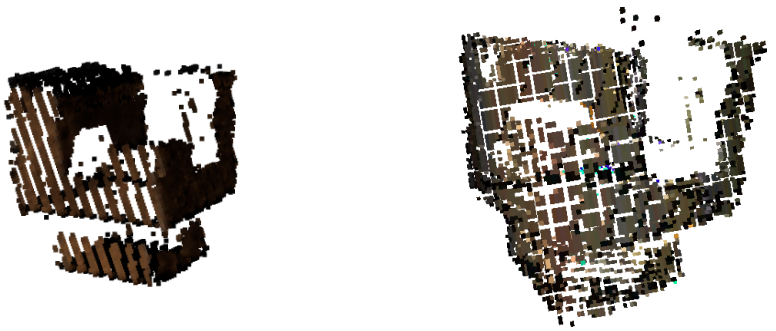


Figure 3. Rotation on Z axis of a 2D matrix to generate a 3D tensor.

3.2. Discretization of the Point Cloud

The input of the architecture is a point cloud. In order to use a CNN, we have to discretize/sample the point cloud to a gridded structure (tensor). The representation used is the voxel grid, in which a voxel is the three-dimensional equivalent of a pixel. This method generates a three-dimensional grid of shape (nx, ny, nz), where each point of the point cloud is assigned to a voxel. If more than one point is assigned to the same cell, an interpolation is used to calculate the RGB value of that voxel. As an illustration, Figure 4a shows a point cloud and Figure 4b presents its voxelization. The number of voxels on each dimension (nx, ny, nz) depends on the architecture used, and it is explained in the subsection of each architecture.



(**a**) Point cloud example

(**b**) Voxelization of the point cloud

Figure 4. Transformation from a point cloud to a voxel grid.

3.3. Feature Extractor CNN

The most used CNN feature extractors in the literature are ResNet, Inception, VGG16, Xception, and Inception-ResNet v2. All of them can be transformed to 3D CNN by replacing the 2D convolutional layers with 3D convolutional layers and 2D MaxPooling layers with 3D MaxPooling layers. The weights of those nets trained with ImageNet are transformed to 3D to train the 3D version of those nets. The structure of each of the nets are described in the following subsections.

3.3.1. VGG16

The 2D version of the VGG16 requires as input a 224×224 RGB image. In the 3D case, the input is a $96 \times 96 \times 96 \times 3$ (RGB) tensor. Figure 5 shows the architecture of the 3D version of the VGG16 feature extractor. This is composed of 2 Convolutional layers of 64 filters, MaxPooling layer, 2 Convolutional layers of 128 filters, MaxPooling layer, 3 Convolutional layers of 512 filters, MaxPooling layer, 3 Convolutional layers of 512 filters, and a MaxPooling layer. The convolution receptive field is 3×3, the stride is fixed to 1, padding to 1, and MaxPooling is performed over a $2\times2\times2$ window with a stride of 2. The 2D Convolutional layers have been replaced by 3D Convolutional layers and 2D MaxPooling by 3D MaxPooling.

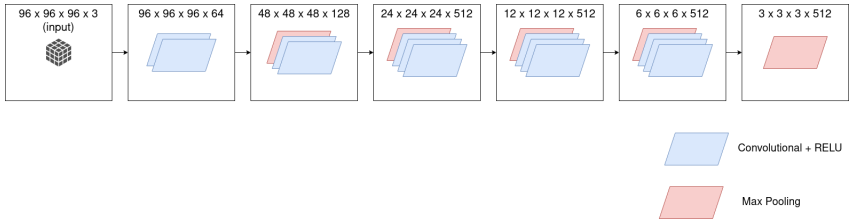


Figure 5. VGG16 architecture.

3.3.2. ResNet

The Residual Neural Network's (or ResNet) main contribution is what is called *skip connections*, which allows output from previous layers to bypass the next layers in order to propagate residual information (Residual Blocks). The ResNet can be tuned by changing the number of blocks that each group has. Table 1 presents different configurations of the 3D ResNet that we have used in the experiments.

Figure 6 shows the architecture of the ResNet50. Figure 7a–d are the building blocks that form the ResNet50. In order to transform this network to 3D, Convolutional and MaxPooling layers have been changed by their respective 3D version. AveragePooling also has its 3D version. The input 2D image size is 224×224, while the 3D input tensor is $96 \times 96 \times 96$.

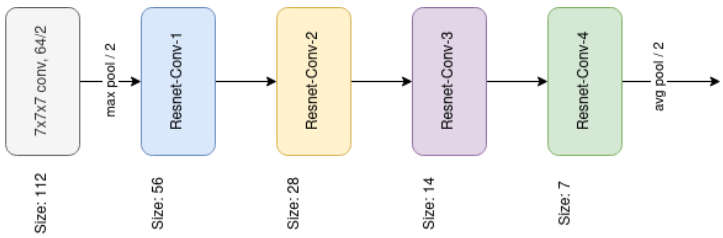
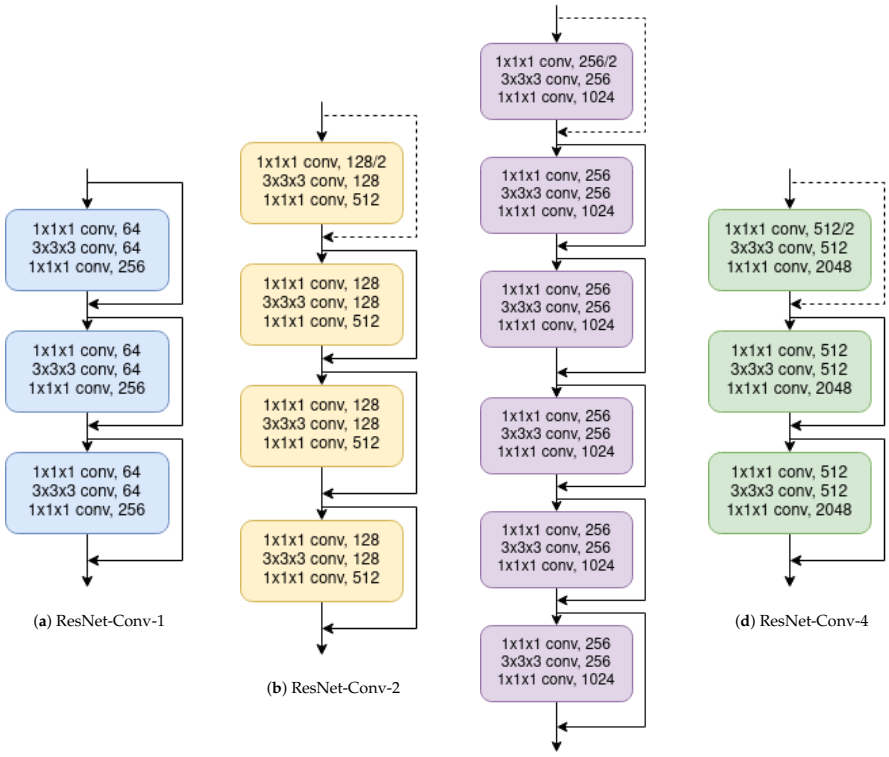


Figure 6. ResNet architecture. Building blocks are shown in brackets.



(**a**) ResNet-Conv-1

(**b**) ResNet-Conv-2

(**c**) ResNet-Conv-3

(**d**) ResNet-Conv-4

Figure 7. Building blocks of the Resnet50.

Table 1. Architectures of ResNet.

Layer Group	Output Size	18-Layer	34-Layer	50-Layer	101-Layer	152-Layer
conv1	$48 \times 48 \times 48$	$7 \times 7 \times 7$, 64, stride 2				
conv2	$24 \times 24 \times 24$	$3 \times 3 \times 3$ max pool, stride 2				
		$\begin{bmatrix} 3\times3\times3, 64 \\ 3\times3\times3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3\times3, 64 \\ 3\times3\times3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1\times1, 64 \\ 3\times3\times3, 64 \\ 1\times1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1\times1, 64 \\ 3\times3\times3, 64 \\ 1\times1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1\times1, 64 \\ 3\times3\times3, 64 \\ 1\times1\times1, 256 \end{bmatrix} \times 3$
conv3	$12 \times 12 \times 12$	$\begin{bmatrix} 3\times3\times3, 128 \\ 3\times3\times3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3\times3, 128 \\ 3\times3\times3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1\times1, 128 \\ 3\times3\times3, 128 \\ 1\times1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1\times1, 128 \\ 3\times3\times3, 128 \\ 1\times1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1\times1, 128 \\ 3\times3\times3, 128 \\ 1\times1\times1, 512 \end{bmatrix} \times 8$
conv4	$6 \times 6 \times 6$	$\begin{bmatrix} 3\times3\times3, 256 \\ 3\times3\times3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3\times3, 256 \\ 3\times3\times3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1\times1, 256 \\ 3\times3\times3, 256 \\ 1\times1\times1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1\times1, 256 \\ 3\times3\times3, 256 \\ 1\times1\times1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1\times1\times1, 256 \\ 3\times3\times3, 256 \\ 1\times1\times1, 1024 \end{bmatrix} \times 36$
conv5	$3 \times 3 \times 3$	$\begin{bmatrix} 3\times3\times3, 512 \\ 3\times3\times3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3\times3, 512 \\ 3\times3\times3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1\times1, 512 \\ 3\times3\times3, 512 \\ 1\times1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1\times1, 512 \\ 3\times3\times3, 512 \\ 1\times1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1\times1, 512 \\ 3\times3\times3, 512 \\ 1\times1\times1, 2048 \end{bmatrix} \times 3$
	$1 \times 1 \times 1$	average pool				

3.3.3. Inception-ResNet v2

This network combines the idea of the two previous networks: Inception blocks and Residual blocks. Authors of this architecture [13] propose two different versions: v1 and v2. Inception-ResNet-v2 is a wider version which is similar to Inception-v4 but adding Residual blocks. Figure 8 shows the 3D architecture of this second version of the network. The Inception-ResNet blocks, the Reduction blocks, and Stem block are described in Reference [13].

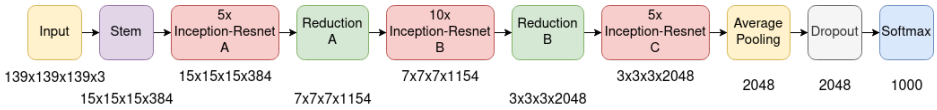


Figure 8. Inception-ResNet-v2 architecture.

To adapt this 2D Convolutional Network to 3D, we have changed the 2D Convolutional layers to 3D Convolutional layers,the 2D Average Pooling layers to 3D Average Pooling layers, the 2D MaxPooling layers to 3D MaxPooling layers, and we use 3D vector as stride instead of 2D vector and 3D kernel size instead of 2D kernel size. The input size of this architecture is $139 \times 139 \times 139 \times 3$ (RGB) instead of $299 \times 299 \times 3$ so that the resulting outputs of the blocks are $15 \times 15 \times 15 \times 384$, $7 \times 7 \times 7 \times 1154$, and $3 \times 3 \times 3 \times 2048$, respectively, for the Stem block, the Reduction A block, and the Reduction B block.

3.3.4. EfficientNet

Given a base network, it is typical to scale its depth, width and resolution in order to improve accuracy. In Reference [18], researchers stated that *"Scaling up any dimension of network width, depth or resolution improves accuracy, but the accuracy gain diminishes for bigger models"* and that *"In order to pursue better accuracy and efficiency, it is critical to balance all dimensions of network width, depth and resolution during ConvNet scaling"*. Consequently, they proposed a compound scaling method that uses a compound coefficient to uniformly scale the 3 components.

The base network from which they start scaling is called EfficientNet-B0 (see Figure 9). This architecture is similar to the MnasNet [30]. The main building block is a mobile inverted bottleneck MBConv [30,31], to which is added a squeeze-and-excitation optimization [32]. This base network is scaled up applying the compound scaling method to obtain EfficientNet-B1 to B7.

The 3D transformation of this architecture is done by changing the 2D Convolutional layers to 3D convolutional layers, the 2D Depthwise Convolutional layers to 3D Depthwise Convolutional layers, the 2D Global Average Pooling layers to 3D Global Average Pooling layers, and by adjusting the 2 dimensional paddings to 3 dimensional paddings.

Table 2 presents the different architectures that we have been experimenting, by using different configurations for the width and depth factors, with the information of the initial and transformed input size. The 3D resolution adjustment follows the Equation (1)

$$w_{3D} \times h_{3D} \times d_{3D} = w_{2D} \times h_{2D}, \tag{1}$$

where $w_{3D} = h_{3D} = d_{3D}$ are, respectively, the width, height, and depth of the 3D input. All the three values are equal, since we want regular grids. $w_{2D} = h_{2D}$ are the width and height of the 2D resolution that Reference [18] proposes for EfficientNet, and which are are also equal. We name the 3D resolution r_{3D} (Equation (2)) and the 2D resolution r_{2D} (Equation (3)).

$$r_{3D} = w_{3D} = h_{3D} = d_{3D}, \tag{2}$$

$$r_{2D} = w_{2D} = h_{2D}. \tag{3}$$

The transformation from 2D to 3D resolution expressed with Equation (1) can, therefore, be simplified as:

$$r_{3D} = \left\lfloor \sqrt[3]{r_{2D}^2} \right\rfloor. \tag{4}$$

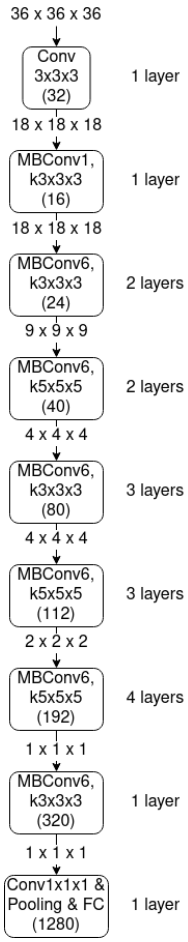


Figure 9. EfficientNet base architecture (EfficientNetB0).

Table 2. Parameters of each of the EfficientNet architectures (width factor, depth factor, 2D resolution, and 3D resolution).

EfficientNet Architecture	w	d	r_{2D}	r_{3D}
B0	1.0	1.0	224	36
B1	1.0	1.1	240	38
B2	1.1	1.2	260	40
B3	1.2	1.4	300	44
B4	1.4	1.8	380	52
B5	1.6	2.2	456	60
B6	1.8	2.6	528	66
B7	2.0	3.1	600	72

3.4. Classifier

The final layers of the Deep Neural Networks, or top of the network, depend on the purpose of the network. In our case, the network is aimed for classification. If the architecture has not its own top layers defined, the top is composed by a Fully Connected (FC) layer with a number of filters equal to the number of classes that we want to classify and a softmax activation layer. This softmax layer squashes a vector in the range $[0, 1]$, and the resulting elements add up to 1. During training a dropout layer is introduced before the FC layer.

4. Networks Training

All networks have been trained using a custom dataset composed of 7 industrial parts and 500 RGB-D images per object. The dataset is artificially generated using the Unreal Engine 4 (UE4) plugin NVIDIA Deep learning Dataset Synthesizer (NDDS) [33]. The creation of the dataset is described in Section 4.1.

We have trained all networks with stochastic gradient using Tensorflow [34]. We used Adam with a learning rate of 0.0001 in the first 20 epochs and 0.00001 in the last 10 epochs. We have only trained networks for 30 epochs to prevent network to saturate as the dataset is small. Since the aimed task is multi-class classification, in which there is only one element in the target vector which is not zero (the positive class), we use Categorical Cross-Entropy loss (Equation (5)). This loss is used because it is a very good measure of how distinguishable two discrete probability distributions are from each other.

$$CE = -log\left(\frac{e^{s_p}}{\sum_j^C e^{s_j}}\right), \tag{5}$$

where s_p is the score obtained from the net for the positive class.

4.1. Dataset generation

As stated before, the dataset has been artificially generated using UE4 and NDDS plugin. All industrial parts are first reconstructed from point clouds obtained with high accuracy structured light sensor. Figure 10 shows captures of the reconstructed models of the 7 industrial parts. The obtained meshes are imported to UE4 and are randomly rotated and scaled, and they are rendered with a virtual RGB-D camera. For each image, we obtain the color image and the depth image, as well as a *json* file containing a set of relevant information, such as the 3D bounding box enclosing the object and its 3D location. For each object, 500 images are rendered, which produce a dataset of 3500 RGB-D images. This dataset is relatively small compared to other deep learning datasets. This is because the focus of this paper is to work with learning datasets of limited size as opposed to other methods in literature. Finally, the validation split is set to 10% of the instances of each class.

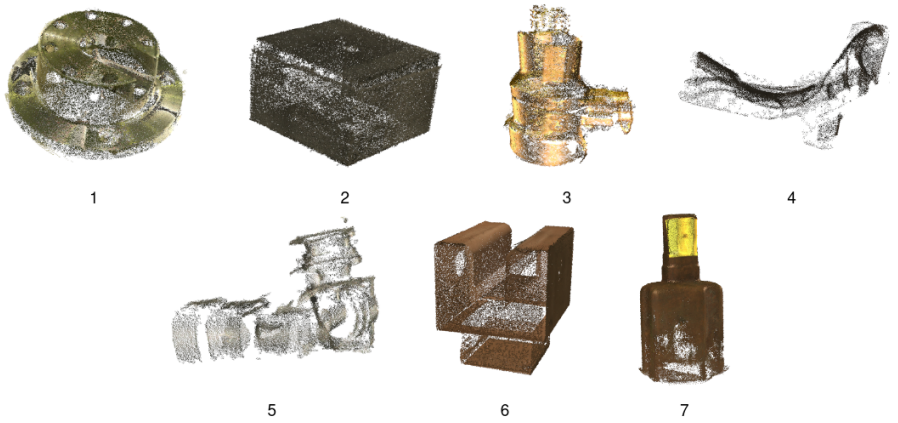


Figure 10. Reconstructed models of the 7 industrial parts.

4.2. Data Preprocessing

The RGB-D image is first projected into a point cloud using the parameters of the camera used for capturing the RGB-D images in UE4. That point cloud is then segmented using the bounding box information and then discretized using a voxel grid. The size of the grid is different for each of the architectures. Table 3 shows the number of grids of each of the architectures used in this paper.

Table 3. Number of grids of each architecture used by the Voxel grid.

Architecture	Number of Grids
VGG16	$96 \times 96 \times 96$
ResNet	$96 \times 96 \times 96$
Inception-ResNet v2	$139 \times 139 \times 139$
EfficientNet B0	$36 \times 36 \times 36$
EfficientNet B1	$38 \times 38 \times 38$
EfficientNet B2	$40 \times 40 \times 40$
EfficientNet B3	$44 \times 44 \times 44$
EfficientNet B4	$52 \times 52 \times 52$
EfficientNet B5	$60 \times 60 \times 60$
EfficientNet B6	$66 \times 66 \times 66$
EfficientNet B7	$72 \times 72 \times 72$

In order to avoid overfitting, different transformations are applied to the dataset. Each training sample is used unchanged, applied a flipping operation or rotated in any of its axis. This data augmentation introduces variability in the dataset in order to reduce the risk of bias.

4.3. Evaluation Metric

The aim of our proposal is multiclass classification, this is, each sample can only belong to one class. Therefore, the metric used for evaluating our model is categorical accuracy. This metric checks how many samples have been correctly labeled. To extend this evaluation, the precision (Equation (6)), the recall (Equation (7)), the F1 score (Equation (8)),

and the macro-F1 (Equation (9)) of the highest accuracy model are going to be evaluated to analyze the behavior of the model for each class.

$$precision_i = \frac{TP_i}{TP_i + FP_i}, \tag{6}$$

$$recall_i = \frac{TP_i}{TP_i + FN_i}, \tag{7}$$

$$F1_i = 2 \times \frac{precision_i \times recall_i}{precision_i + recall_i}, \tag{8}$$

$$\text{macro-F1} = \frac{\sum_i^C F1_i}{|C|}, \tag{9}$$

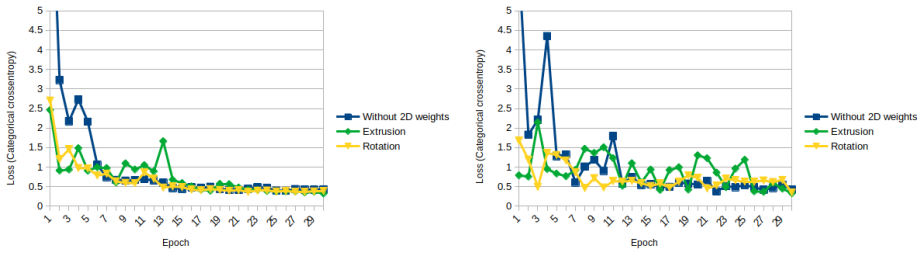
where TP_i is the count of true positive instances of the class i (correctly classified instances), FP_i is the count of false positive instances of the class i (the count of instances that are uncorrectly classified as class i), and FN_i is the count of false negative instances of the class i (count of instances that should be classified as class i and they are classified as another class).

5. Experimental Results

The experiment was conducted using VGG16, ResNet, Inception ResNet v2 (from now on just Inception ResNet), and EfficientNet architectures. For each of the architectures, we have compared the results (i) without initializing the weights and (ii) using the pretrained weights from 2D architectures trained on Imagenet and transformed to 3D using extrusion and rotation.

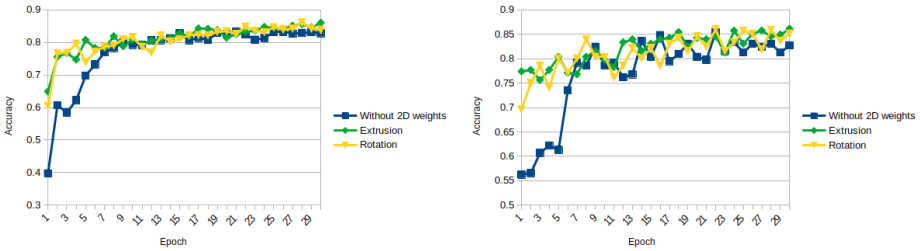
For VGG16, results are not as good as expected. Therefore, no Figures are going to be included. The net classifies all instances the same class; therefore, the net is saturated. This happens for all the experiments (without weights, extrusion, and rotation).

For ResNet, the obtained results are more like expected. Figure 11a,b show the evolution of the losses using the 3 different weight initializations on training and on validation, respectively. The aforementioned figures show that initializing from 2D weights makes initial loss lower and converges earlier on both stages, training and validation. Regarding the accuracy, a similar behavior is observed: the initial and final accuracy are higher with the initialized weights. Figure 12a,b show the evolution of the accuracy on training and on validation, respectively.



(**a**) ResNet 3D Loss on training (**b**) ResNet 3D Loss on validation.

Figure 11. ResNet 3D Loss (Categorical Cross-Entropy).

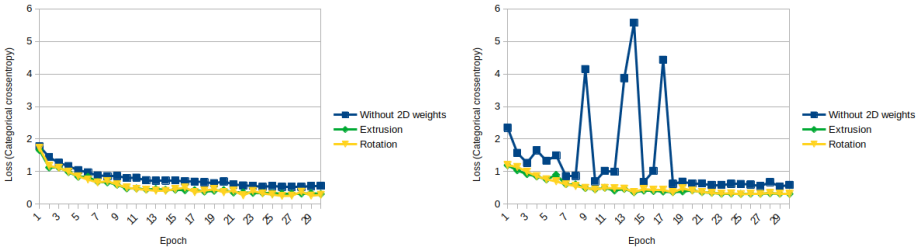


(**a**) ResNet 3D accuracy on training. (**b**) ResNet 3D accuracy on validation.

Figure 12. ResNet 3D accuracy.

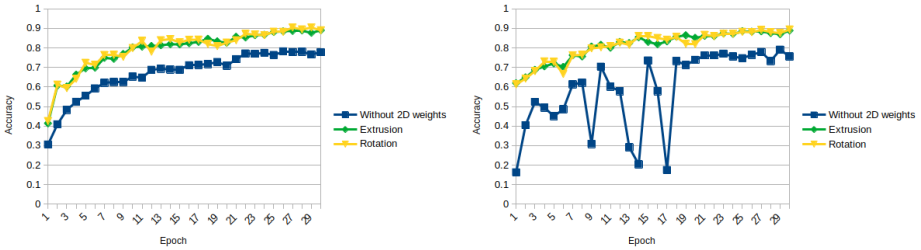
The results of Inception ResNet architecture are similar to the ones obtained with ResNet architecture, but the differences between the results obtained with the initialization and without initialization are even higher. Figures 13a,b and 14a,b show, respectively, the evolution of the loss on training and validation, and the accuracy on training and validation.

Among the architectures tested, the Inception ResNet is the one that takes more profit from using the 2D weights, increasing its validation accuracy from 0.7558 to 0.8887 on extrusion and 0.8939 on rotation (increase of around 17 to 18%).



(**a**) Inception ResNet 3D Loss on training. (**b**) Inception ResNet 3D Loss on validation

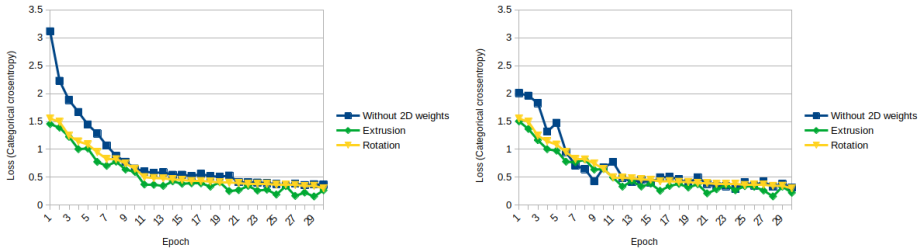
Figure 13. Inception ResNet 3D Loss (Categorical Cross-Entropy).



(**a**) Inception ResNet 3D accuracy on training (**b**) Inception ResNet 3D accuracy on validation.

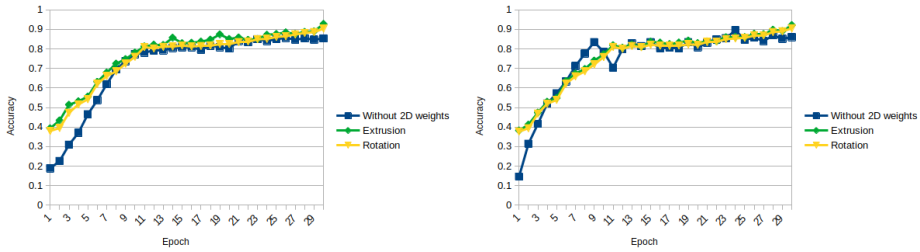
Figure 14. Inception ResNet 3D accuracy.

EfficientNet is tested only from B0 up to B4 since our computing capabilities are limited to those set-ups. We observed that the performance of the network decreases from B0 to B4. This could be because the more complex the net is, the more epochs it needs to obtain similar results. This is even more evident in small datasets as the one we used. Figures 15a,b and 16a,b show the evolution of the loss and accuracy only for B0 version, which performs better for our experimental set-up.



(**a**) EfficientNetB0 3D Loss on training (**b**) EfficientNetB0 3D Loss on validation

Figure 15. EfficientNetB0 3D Loss (Categorical Cross-Entropy).



(**a**) EfficientNetB0 3D accuracy on training (**b**) EfficientNetB0 3D accuracy on validation

Figure 16. EfficientNetB0 3D accuracy.

As shown in Figures 11–16, almost in every epoch of the training of each model, the accuracy is higher using the initialized weights. For some of them, like the Inception ResNet, the difference of accuracy is notable.

We compared all the extrusion architectures we propose with the PointNet [28], a 3D point-based deep learning method, by looking at the obtained accuracy during training (see Figure 17).

Table 4 shows the obtained results for all the architectures with all the weights initialized for the last epoch.

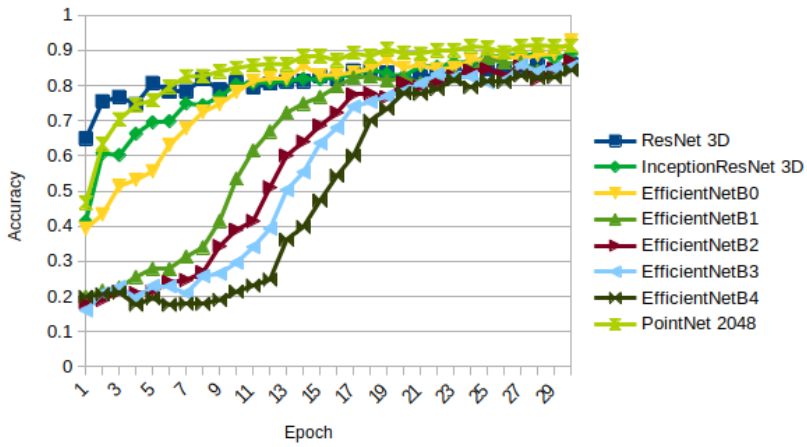


Figure 17. Accuracy on training: PointNet compared to all the architectures initialized with extrusion.

Table 4. Accuracy and loss comparative between all the used architectures and PointNet. Last column is the number of parameters each network has to train.

Method	Train		Val		Params
	Loss	acc	Loss	acc	
ResNet 3D	0.4284	0.8279	0.4279	0.8272	47M
ResNet 3D Extrusion	0.3245	0.8599	0.3312	0.8612	
ResNet 3D Rotation	0.3892	0.8372	0.3484	0.8512	
Inception ResNet 3D	0.5653	0.7777	0.592	0.7558	67M
Inception ResNet 3D Extrusion	0.3127	0.8901	0.3157	0.8887	
Inception ResNet 3D Rotation	0.2880	0.8900	0.3133	0.8939	
EfficientNetB0 3D	0.3664	0.8534	0.3137	0.8605	4.7M
EfficientNetB0 3D Extrusion	0.2663	0.9276	0.2213	0.9217	
EfficientNetB0 3D Rotation	0.0312	0.9039	0.3004	0.9052	
EfficientNetB1 3D	0.3987	0.847	0.367	0.8372	7.5M
EfficientNetB1 3D Extrusion	0.3758	0.8771	0.3610	0.8420	
EfficientNetB1 3D Rotation	0.2325	0.8485	0.3526	0.8422	
EfficientNetB2 3D	0.4393	0.8324	0.3408	0.8547	8.8M
EfficientNetB2 3D Extrusion	0.3880	0.8592	0.3384	0.8595	
EfficientNetB2 3D Rotation	0.3273	0.8362	0.3271	0.8549	
EfficientNetB3 3D	0.459	0.8289	0.3906	0.8285	12.1M
EfficientNetB3 3D Extrusion	0.3887	0.8422	0.3809	0.8305	
EfficientNetB3 3D Rotation	0.4111	0.8583	0.3856	0.8318	
EfficientNetB4 3D	0.4785	0.8176	0.4106	0.8097	19.7M
EfficientNetB4 3D Extrusion	0.3960	0.8544	0.3949	0.8123	
EfficientNetB4 3D Rotation	0.4335	0.8207	0.4039	0.8153	
PointNet	1.1802	0.9132	1.2587	0.9048	1M

The highest accuracy is obtained with EfficientNetB0 initialized using extrusion (0.9276 in training and 0.9217 in validation). The largest improvement is achieved with Inception ResNet, increasing its accuracy from 0.7777 to 0.8901 and 0.8900 using extrusion and rotation, respectively, on training and from 0.7558 to 0.8887 and 0.8939 using extrusion and rotation, respectively, on validation.

To extend our analysis, we have calculated the confusion matrix of the EfficientNetB0 initialized using extrusion on validation since this is the architecture that obtains the highest accuracy. The confusion matrix is shown in Table 5. The network works similarly for every part, and the obtained Macro-F1 is really close to the accuracy. This is because the per-class precision and recall are very similar.

Table 5. Confusion matrix and metrics of EfficientNetB0 initialized using extrusion.

		Real Class							**Total**
		1	2	3	4	5	6	7	
	1	486	1	0	20	1	10	3	
	2	7	492	0	11	5	5	2	
	3	2	7	443	1	45	3	20	
Predicted class	4	5	0	1	455	2	0	2	
	5	0	0	55	3	447	0	0	
	6	0	0	1	3	0	472	42	
	7	0	0	0	7	0	10	431	
Precision		0.972	0.984	0.886	0.91	0.894	0.944	0.862	
Recall		0.933	0.943	0.85	0.978	0.885	0.911	0.962	
F1		0.952	0963	0.868	0.943	0.89	0.927	0.909	
				Macro-F1					0.922
				Accuracy					0.9217

6. Conclusions

In this paper, we introduced a novel approach to transfer learned 2D convolutional networks to 3D convolutional networks in a reduced size dataset. We have studied several 2D architectures: VGG16, ResNet, Inception ResNet v2, and EfficientNet. The weights from pretrained 2D networks have been transformed to 3D using two approaches, the extrusion and the rotation.These transformed weights are then used to initialize the 3D version of those architectures.

On almost every architecture, the obtained accuracy is better with the 2D weights than without initialization, reaching a performance similar to state of art 3D deep learning methods.

In future works, other 2D to 3D transformations will be tested, looking for a better performance. Combinations of transformations may produce also improved results.

Author Contributions: Conceptualization, I.M., J.A., A.R., and B.S.; methodology, B.S.; software, I.M.; validation, J.A., A.R., and B.S.; formal analysis, I.M.; investigation, I.M.; resources, I.M.; data curation, I.M.; writing—original draft preparation, I.M.; writing—review and editing, I.M., J.A., A.R., and B.S.; visualization, I.M.; supervision, B.S.; project administration, J.A.; funding acquisition, J.A. All authors have read and agreed to the published version of the manuscript.

Funding: This paper has been supported by the project ELKARBOT under the Basque program ELKARTEK, grant agreement No. KK-2020/00092.

Data Availability Statement: The data presented in this study are available on demand from thecorresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Torralba, A.; Efros, A.A. Unbiased look at dataset bias. In Proceedings of the CVPR 2011, Providence, RI, USA, 20–25 June 2011; pp. 1521–1528.
2. Pan, S.J.; Yang, Q. A survey on transfer learning. *IEEE Trans. Knowl. Data Eng.* **2009**, *22*, 1345–1359. [CrossRef]
3. Guo, Y.; Wang, H.; Hu, Q.; Liu, H.; Liu, L.; Bennamoun, M. Deep Learning for 3D Point Clouds: A Survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**. [CrossRef] [PubMed]
4. Ahmed, E.; Saint, A.; Shabayek, A.; Cherenkova, K.; Das, R.; Gusev, G.; Aouada, D.; Ottersten, B. Deep learning advances on different 3D data representations: A survey. *arXiv* **2018**, arXiv:1808.01462
5. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. *Adv. Neural Inf. Process. Syst.* **2012**, *25*, 1097–1105. [CrossRef]
6. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009; pp. 248–255.
7. Zeiler, M.D.; Fergus, R. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*; Springer: Cham, Switzerland, 2014; pp. 818–833.
8. Sermanet, P.; Eigen, D.; Zhang, X.; Mathieu, M.; Fergus, R.; LeCun, Y. OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks. *arXiv* **2013**, arXiv:1312.6229.
9. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv* **2014**, arXiv:1409.1556.
10. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015, pp. 1–9.
11. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
12. Lin, T.Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft COCO: Common Objects in Context. In *Computer Vision—ECCV 2014*; Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T., Eds.; Springer International Publishing: Cham, Switzerland, 2014; pp. 740–755.
13. Szegedy, C.; Ioffe, S.; Vanhoucke, V.; Alemi, A. Inception-v4, inception-resnet and the impact of residual connections on learning. In Proceedings of the AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017; Volume 31.
14. Xie, S.; Girshick, R.; Dollár, P.; Tu, Z.; He, K. Aggregated residual transformations for deep neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 1492–1500.
15. Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K.Q. Densely connected convolutional networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 4700–4708.
16. Chen, Y.; Li, J.; Xiao, H.; Jin, X.; Yan, S.; Feng, J. Dual path networks. *arXiv* **2017**, arXiv:1707.01629.
17. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv* **2017**, arXiv:1704.04861.
18. Tan, M.; Le, Q.V. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *arXiv* **2020**, arXiv:1905.11946.
19. Fanelli, G.; Weise, T.; Gall, J.; Van Gool, L. Real time head pose estimation from consumer depth cameras. In *Joint Pattern Recognition Symposium*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 101–110.
20. Tremblay, J.; To, T.; Sundaralingam, B.; Xiang, Y.; Fox, D.; Birchfield, S. Deep object pose estimation for semantic robotic grasping of household objects. *arXiv* **2018**, arXiv:1809.10790.
21. Liu, N.; Han, J.; Zhang, D.; Wen, S.; Liu, T. Predicting eye fixations using convolutional neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 362–370.
22. Sinha, A.; Bai, J.; Ramani, K. Deep learning 3D shape surfaces using geometry images. In *European Conference on Computer Vision*; Springer: Cham, Switzerland, 2016; pp. 223–240.
23. Cao, Z.; Huang, Q.; Karthik, R. 3D object classification via spherical projections. In Proceedings of the IEEE 2017 International Conference on 3D Vision (3DV), Qingdao, China, 10–12 October 2017; pp. 566–574.
24. Zhou, Y.; Tuzel, O. Voxelnet: End-to-end learning for point cloud based 3d object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 4490–4499.
25. Gomez-Donoso, F.; Escalona, F.; Cazorla, M. Par3DNet: Using 3DCNNs for Object Recognition on Tridimensional Partial Views. *Appl. Sci.* **2020**, *10*, 3409. [CrossRef]
26. Wang, P.S.; Liu, Y.; Guo, Y.X.; Sun, C.Y.; Tong, X. O-CNN: Octree-Based Convolutional Neural Networks for 3D Shape Analysis. *ACM Trans. Graph.* **2017**, *36*. [CrossRef]
27. Muzahid, A.; Wan, W.; Sohel, F.; Khan, N.U.; Villagómez, O.D.C.; Ullah, H. 3D Object classification using a volumetric deep neural network: An efficient Octree Guided Auxiliary Learning approach. *IEEE Access* **2020**, *8*, 23802–23816. [CrossRef]
28. Qi, C.R.; Su, H.; Mo, K.; Guibas, L.J. Pointnet: Deep learning on point sets for 3d classification and segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 652–660.
29. Qi, C.R.; Yi, L.; Su, H.; Guibas, L.J. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv* **2017**, arXiv:1706.02413.

30. Tan, M.; Chen, B.; Pang, R.; Vasudevan, V.; Sandler, M.; Howard, A.; Le, Q.V. Mnasnet: Platform-aware neural architecture search for mobile. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 5–20 June 2019; pp. 2820–2828.
31. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.C. Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 4510–4520.
32. Hu, J.; Shen, L.; Sun, G. Squeeze-and-excitation networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 7132–7141.
33. To, T.; Tremblay, J.; McKay, D.; Yamaguchi, Y.; Leung, K.; Balanon, A.; Cheng, J.; Hodge, W.; Birchfield, S. NDDS: NVIDIA Deep Learning Dataset Synthesizer. 2018. Available online: https://github.com/NVIDIA/Dataset_Synthesizer (accessed on 30 December 2020).
34. Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G.S.; Davis, A.; Dean, J.; Devin, M.; et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. 2015. Available online: www.tensorflow.org (accessed on 30 December 2020).

CAPÍTULO 9

Ensemble of 6 DoF Pose estimation from state-of-the-art deep methods

Título:	Ensemble of 6 DoF Pose estimation from state-of-the-art deep methods
Autores:	I. Merino, J. Azpiazu, A. Remazeilles, B. Sierra
Revista:	Neurocomputing
Volumen:	541
Número:	26270
Editor:	Elsevier
DOI:	10.1016/j.neucom.2023.126270
Año:	2023
Cuartil (Scimago/WoS):	Q1 (Artificial Intelligence) / Q2 (Computer science, artificial intelligence)

Neurocomputing 541 (2023) 126270

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Ensemble of 6 DoF Pose estimation from state-of-the-art deep methods.

Ibon Merino [a,b,*], Jon Azpiazu [a], Anthony Remazeilles [a], Basilio Sierra [b]

[a] TECNALIA, Basque Research and Technology Alliance (BRTA), Mikeletegi Pasealekua 7, 20009 Donostia-San Sebastián, Spain
[b] Robotics and Autonomous Systems Group, Universidad del País Vasco/Euskal Herriko Unibertsitatea, San Sebastián 20009, Spain

ARTICLE INFO

Article history:
Received 2 November 2022
Revised 31 March 2023
Accepted 22 April 2023
Available online 25 April 2023
Communicated by Zidong Wang

Keywords:
Deep learning
Pose estimation
Ensemble
Stacked generalization

ABSTRACT

Deep learning methods have revolutionized computer vision since the appearance of AlexNet in 2012. Nevertheless, 6 degrees of freedom pose estimation is still a difficult task to perform precisely. Therefore, we propose 2 ensemble techniques to refine poses from different deep learning 6DoF pose estimation models. The first technique, merge ensemble, combines the outputs of the base models geometrically. In the second, stacked generalization, a machine learning model is trained using the outputs of the base models and outputs the refined pose. The merge method improves the performance of the base models on LMO and YCB-V datasets and performs better on the pose estimation task than the stacking strategy.

1. Introduction

Vision-based object pose estimation is a task with applications in different fields, such as robotics or augmented reality. Many techniques approach this task by estimating the 2D or 3D bounding box that corresponds to the pose of the object [1]. In contrast, this paper focuses on estimating the 6 degrees of freedom (6 DoF) pose of object also known as 6D localization, that is the 3 DoF translation and 3 DoF rotation that would align the object with a target object reference model. Fig. 1 shows an example of a 6 DoF pose estimation task. 6DoF pose estimation is usually represented in three diferent ways: Bounding Boxes, Axes and model overlapping. The Fig. 1 shows the 6DoF poses of the objects overlapping the models of the objects on the image.

The leaderboards of the main challenges presented in the Benchmark for 6D Object Pose estimation [2] (BOP challenge, one of the most well-known 6DoF estimation benchmarks) were led by classical non-deep learning methods based on Point Pair Features (PPF) [3–6] until 2019. Those methods were depth only methods that rely on geometrical features of pointclouds. Methods based on convolutional neural networks (CNNs) [7–10] were limited by the insufficient number of real training images, since annotation of 6DoF poses is expensive, and the large domain gap between real test images and the commonly used synthetically generated training images. On BOP 2020, 350 K physics-based rendered training images were provided to the participants in order to deal with these problems. Since that moment, deep learning methods have finally caught up and surpassed PPF-based methods. In relation with the input data used, on BOP 2020 almost all deep learning methods were trained on RGB images only. However, more recently, deep RGB-D approaches have obtained similar results to deep RGB only methods [11,12].

The main contributions of this paper are summarized as follows:

- An exhaustive analysis of existing RGB/RGB-D deep learning methods for 6DoF pose estimation.
- Two ensemble methods (merge and stacking) to combine state-of-the-art models in order to obtain more homogenous detection performance across different datasets.

In the experiments performed we compare how each base model and ensemble strategy perform for each metric, and the results show that the metrics obtained on the ensemble strategies are more homogenous than the ones obtained with the base models (or level-0). The rest of the paper is organized as follows: in Section 2 we present related works; in Section 3 we introduce the proposed ensemble method; in Section 5 we explain the conducted experiments; and, finally, we summarize the paper and the conclusions in Section 6.

* Corresponding author at: TECNALIA, Basque Research and Technology Alliance (BRTA), Mikeletegi Pasealekua 7, 20009 Donostia-San Sebastián, Spain.
E-mail addresses: ibon.merino@tecnalia.com (I. Merino), jon.azpiazu@tecnalia.com (J. Azpiazu), anthony.remazeilles@tecnalia.com (A. Remazeilles), b.sierra@ehu.eus (B. Sierra).

https://doi.org/10.1016/j.neucom.2023.126270
0925-2312/

Fig. 1. 6DoF pose estimation of Linemod objects from BOP challenge.

2. Background

For some time now, computer vision has been used to develop technologies to recognize objects. For industrial objects, many authors have designed specific methods to deal with this paradigm [13–15]. Nevertheless, generic vision techniques have also been used for recognition of industrial objects. The first approaches were based on feature matching or classification techniques: object features are detected, described and matched with known features from target objects.

Typically, features were hand-crafted to produce descriptors with convenient robustness or invariance to various conditions in order to make the matching between images and object models possible. SIFT [16] features have been one of the most popular hand-crafted features for a long time. More recently, deep neural network based methods became popular, thanks to the development of more powerful GPUs and larger datasets that enable the training of big convolutional neural network models.

Methods that **directly regress the rotation and translation** of objects are very common. The method proposed in [17] regresses 6DoF object poses using only synthetic images. Another pose estimator that directly regresses the rotation and translation is presented in [18]. This method calculates the 3D translation predicting the distance from object to camera and the rotation is estimated regressing to a quaternion representation. In [19], authors propose an end-to-end trained model that extends Mask R-CNN [20], the canonical 2D object detection network, to regress 3D translation and rotation. [21] is a dense fusion network that extracts pixel-wise dense feature embeddings from RGB-D images to predict the pose of the objects. However, this group of methods is not very scalable to regress poses of datasets with a high number of different classes.

Other approaches rely on **two-stage approaches** that first detect keypoints and later estimate 2D-3D or 3D-3D correspondences of keypoints using Perspective-n-Point (PnP) algorithms to estimate the poses. One of those methods estimates semantic keypoints and fits a deformable shape model to the 2D detections [22]. BB8 [23] predicts a 2D projection of the corners of the 3D bounding boxes as keypoints. [24] introduces a similar approach that detects the 2D projection of the 3D bounding box corners using a single shot convolutional neural network. [25] proposed a category-agnostic keypoint representation by combining what they call StarMap heatmap and their corresponding features as 3D locations in the canonical viewpoint. [26] combines global features for object segmentation and local features for coordinate regression followed by a RANSAC to optimize the object pose. [27] introduces a pixel-wise voting network (PVNet) to regress pixel-wise vectors pointing to the keypoints and uses those vectors to vote for the location of keypoints. This voting method has inspired many later methods. [28] predicts newly introduced features called NOCS that, combined with the depth map, can jointly estimate the 6D pose. Two-stage approaches do not train the entire pipeline at once. The keypoint prediction is trained while the correspondence matching to 3D and 6D pose estimation is obtained a posteiori.

Learning from **pointclouds** provides many benefits to robotics. PointNet++ [29] introduce a hierarchical neural network that applies PointNet [30] recursively. [11] estimates the 6Dof pose by detection 3D keypoints from pointclouds and RGB images and searching for correspondences similar to two-stage approaches. [12] is an improvement from the previous method that adds a bidirectional fusion layer to combine RBG information and geometric information. Those methods use the geometrical features that can be extracted from pointclouds that 2D images can not provide.

Even if the continuous development of new deep learning architectures (such as, MFDNet [31], ARHPE [32], EDMF [33], EHPE [34] or NGDNet [35]) and techniques led to significant performance improvement on many datasets, there is still a lack of a general method that works for every situation or dataset. Stacking and ensemble methods try to deal with this problem. They try to extract the ability of each individual method to obtain a more abstract method [36]. There are two critical design elements for this kind of models: the type of generalizer that is suitable to derive the higher-level model and what kind of attributes should be used as inputs [37].

Another challenge to deal with is all the hyperparameters that base learners and higher level models have to tune. To optimize this hyperparameter search, authors in [38] proposed a heuristic search-based stacking of classifiers. Authors from [39] also make

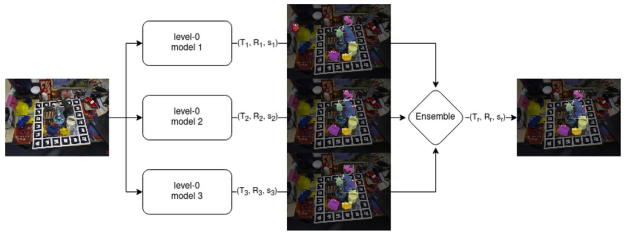


Fig. 2. Illustration of the ensemble approach, in which the output of level-0 models outcome are gathered to deduce a unique refined pose.

Table 1
Ensemble strategies and methods.

Strategy	Method
Merge	Simple
	Weighted
	Clustering
Stacking	SVR
	Decision tree
	Linear regression
	KNN
	MLP

use of meta-heuristic algorithms to configure the stacking ensemble.

Insect behavior based optimization algorithms are also a good solution to search the best configurations in a high range of possible hyperparameters [40–43]. Many more authors have used ensemble or stacking methods to improve the generalization of base learners [44,45]. Some of them even consider stacking a kind of super learning method [46].

Stacking and ensemble methods have also been used for computer vision systems. In [47], authors designed a support vector machine training method based on stacked generalization for image classification. Stacked generalization also improves accuracy for scene analysis and object recognition [48].

3. Proposed method

The ability of neural networks to retrieve abstract information from data enables them to attain competitive performances in many fields and contexts. Nevertheless, not all models are effective against every task or dataset. Indeed, the optimal selection of the

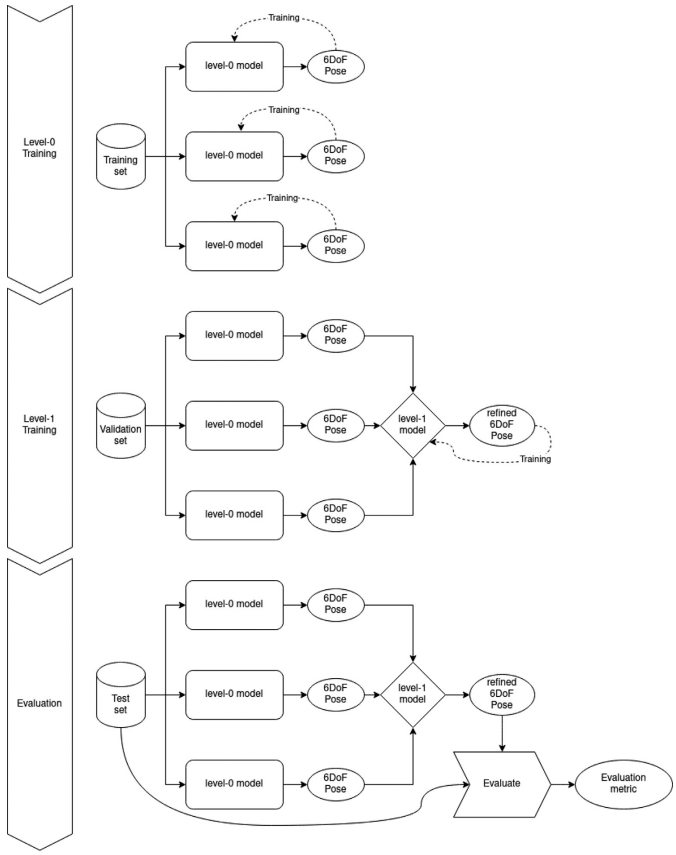


Fig. 3. Stacking poses of 3 level-0 models.

model type and structure, and the comprehension of the reasons that explain the results [49] is one of the main challenges that deep learning can still not solve.

Since looking for the correct deep learning approach is tedious, we propose a method that learns when a given algorithm is more appropriate than another one. Our method is an ensemble model that enables the fusion of the performance of different models. We have defined two different strategies: merging and stacking. Both strategies take the output of different models and refine the obtained poses using the knowledge obtained from each model. The output of each model is composed of a 3D translation T, a 3D rotation R, related to the pose of the object with respect to the camera, and a score s that is the confidence that each model has for that prediction. Given n different models, T_i, R_i and s_i represent the translation, the rotation and the score outcome of the i-th method, where $i \in \{1..n\}$. We defined the refined translation as T_r, the refined rotation as R_r and the refined score as s_r, the outcome of our fusion approach.

Fig. 2 shows the general idea of the ensemble method. Both ensemble approaches, merge and stacking, rely on a set of deep learning models (named from now on as level-0 models) that provide the initial predictions of the pose of the object.

3.1. Level-0 models

We have used as level-0 algorithms three state-of-the-art 6DoF pose estimation deep models: PVN3D [11], FFB6D [12] and Cosypose [50]. These three methods have led the BOP challenge 2020 leaderboard in different datasets.

PVN3D [11] is a novel method for 6DoF object pose estimation from a single RGBD image. This method is based on a deep Hough voting network that takes as input RGBD images and detects 3D keypoints as is performed in RGB-based 6DoF estimation. Once the 3D keypoints are detected, these are used to estimate the parameters of the 6D pose using a least-squares fitting. The method has outperformed state-of-the-art methods on several

Fig. 4. Real test images from LMO.

Fig. 5. Real test images from YCB-V.

Fig. 6. Real test images from T-LESS.

(a) Image from LM PBR training set.

(b) Image from YCB-V PBR training set.

(c) Image from T-LESS PBR training set.

Fig. 7. PBR training images.

Table 2
Level-0 results.

Dataset		PVN3D	FFB6D	COSYPOSE
LMO PBR	AR_{VSD}	0.4740	0.4450	**0.4805**
	AR_{MSSD}	**0.6523**	0.6309	0.6062
	AR_{MSPD}	0.7105	0.6704	**0.8122**
	AR	0.6123	0.5821	**0.6330**
YCB-V PBR	AR_{VSD}	**0.5888**	0.3387	0.5166
	AR_{MSSD}	**0.6955**	0.4093	0.5538
	AR_{MSPD}	0.5649	0.2758	**0.6527**
	AR	**0.6164**	0.3413	0.5744
YCB-V S + R	AR_{VSD}	0.7291	**0.7855**	0.7731
	AR_{MSSD}	0.8541	**0.8791**	0.8421
	AR_{MSPD}	0.7441	0.7856	**0.8502**
	AR	0.7757	0.8167	**0.8218**
T-LESS PBR	AR_{VSD}	0.1789	0.1291	**0.5714**
	AR_{MSSD}	0.2067	0.1268	**0.5892**
	AR_{MSPD}	0.2039	0.1165	**0.7605**
	AR	0.1965	0.1241	**0.6404**
T-LESS S + R	AR_{VSD}	0.2255	0.1985	**0.6691**
	AR_{MSSD}	0.2826	0.2396	**0.6946**
	AR_{MSPD}	0.2666	0.2291	**0.8212**
	AR	0.2582	0.2224	**0.7283**

benchmarks such as Linemod [51] and YCB-V [52]. PVN3D trains its architecture using the multi-task loss (Eq. 4) jointly training keypoints detection module (Eq. 1), semantic segmentation module (Eq. 2) and and objects center voting module (Eq. 3).

$$L_{keypoints} = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M}||of_i^j - of_i^{j*}||\mathbb{I}(p_i \in I), \tag{1}$$

where of_i^j is the translation offset, of_i^{j*} is the ground truth translation offset, M is the total number of selected target keypoints, N is the total number of seeds and $\mathbb{I}$ is an indicating function that equates to 1 only when point p_i belongs to instance I, and 0 otherwise.

$$L_{semantic} = -\alpha(1-q_i)^{\gamma}log(q_i), \quad \text{where} \quad q_i = c_i \cdot l_i \tag{2}$$

with α the α-balance parameter, γ the focusing parameter, c_i the predicted confidence for the i_{th} point belongs to each class and l_i the one-hot representation of ground true class label.

$$L_{center} = \frac{1}{N}\sum_{i=1}^{N}||\Delta x_i - \Delta x_i^*||\mathbb{I}(p_i \in I), \tag{3}$$

where N denotes the total number of seed points on the object surface and Δx_i^* is the ground truth translation offset from seed p_i to the instance center.

$$L_{multi-task} = \lambda_1 L_{keypoints} + \lambda_2 L_{semantic} + \lambda_3 L_{center}, \tag{4}$$

where λ_1, λ_2 and λ_3 are the weights for each task. In our case, we use 2.0 for λ_2 and 1.0 for λ_1 and λ_3.

FFB6D [12] or Full Flow Bidirectional Fusion Network for 6D Pose Estimation is a deep learning network that combines appearance information from RGB images and geometric information from depth images during the representation learning stage (encoding and decoding). Instead of learning each feature independently and fusing them at the end like DenseFusion [21] or PVN3D [11], the fusion is applied to each encoding and decoding layer giving complementary information from 3D to 2D and the other way round. After the representation learning stage, both features are concatenated and 6D pose parameters are estimated as in PVN3D [11]. FFB6D uses the same multi-task loss that was presented in PVN3D. The main difference between FFB6D and PVN3D is the architecture of the feature extraction module. PVN3D extracts geometric features and color features and afterwards fuses them, while FFB6D fuses the geometric and color information on each layer of the network.

Cosypose [50] is a multi-view multi-object 6D pose estimator. In the paper, a single-view single-object estimation method to generate 6D hypothesis and a method for matching the predicted hypothesis to generate a single consistent scene are introduced. With the generated scene, global refinement is applied across different objects and views. They outperform current state-of-the-art results for single-view and multi-view. The single-view 6D pose estimation of Cosypose is carried out following the focal loss adapted in [53] and handle symmetries as in [28]. The object candidate matching first removes the object candidates that are not consistent across views using a RANSAC procedure. Moreover, for the scene refinement they introduce a reconstruction loss (Eq. 5) that operates at a level of objects.

$$L(T_{P_n}, T_{C_a}|T_{C_a}O_{a,\alpha}) = \min_{S\in S(I)}\frac{1}{|\chi_I|}\sum_{x\in\chi_I}||\pi_a(T_{C_a}O_{a,\alpha}Sx) - \pi_a\left(T_{C_a}^{-1}T_{P_n}x\right)||, \tag{5}$$

where $||\cdot||$ is a truncated L2 loss, and for the rest of the parameters check the original paper [50].

The scores of PVN3D and FFB6D are the confidence of the point level instance segmentation of each object. The scores of Cosypose, instead, are the confidence of the initial 2D predictions of the objects.

All three models have their advantages and disadvantages, none of them outperforms the others on every dataset (this can be seen in the BOP Challenge leaderboard). Fig. 2 includes an example of the predictions of the 3 methods. In this example level-0 model 1 is PVN3D, model 2 is FFB6D and model 3 Cosypose. Predictions of the 3 level-0 models are similar except from the ape (red colored) which PVN3D can not locate correctly. A combination of the outputs refines the poses obtaining better results.

For simplicity, our methods deal with the SiVo (Single instance Various objects) task instead of the ViVo (Various instances Various objects) task. We detect a single instance of multiple objects.

4. Ensemble

In order to obtain the refined pose, we define two different ensemble strategies: merge and stacking. Merge methods are defined by mixing the individual results of the level-0 methods

mathematically or algorithmically. Stacking methods, instead, make use of statistical methods or machine learning methods to refine the level-0 poses.

Table 1 summarizes all the proposed strategies and methods.

4.1. Ensemble-Merge

We have defined three merge methods that take the output poses of the level-0 models and calculate a refined pose. The three defined methods are: simple merge, weighted merge and clustered merge.

4.1.1. Simple merge

The first approach is a simple merge method that outputs the mean of the input poses. Eq. 6 and Eq. 7 explains how to obtain the resulting translation and rotation, respectively. Computing a rotation mean is not trivial and can be obtained several ways. In this paper, we use the chordal L2 mean which provides the rotation that minimizes the square of the difference between rotation matrices [54].

$$T_r = \frac{1}{n}\sum_{i=1}^{n} T_i \tag{6}$$

$$R_r = \text{argmin}_{R\in SO(3)} \sum_{i=1}^{n} ||R_i - R||^2 \tag{7}$$

where R is a rotation that belongs to the three-dimensional space of rotations $SO(3)$.

4.1.2. Weighted merge

Simple merge considers each of the models equally even if a model itself outputs a pose with a low confidence. Therefore, this second approach takes into account the scores each model gives

Table 3
Merge ensemble results.

Dataset		Simple (S)	Weighted (W)	Simple clustering (SC)	Weighted clustering (WC)
LMO PBR	AR_{VSD}	0.4776	0.4820	0.4807	**0.4822**
	AR_{MSSD}	0.6473	0.6411	0.6496	0.6509
	AR_{MSPD}	0.7049	0.7338	0.7416	0.7434
	AR	0.6099	0.6189	0.6240	0.6255
YCB-V PBR	AR_{VSD}	0.4877	0.5277	0.5451	0.5502
	AR_{MSSD}	0.5565	0.6003	0.6150	0.6235
	AR_{MSPD}	0.4370	0.5228	0.5312	0.5403
	AR	0.4938	0.5503	0.5638	0.5713
YCB-V S + R	AR_{VSD}	0.7930	0.7748	**0.7958**	0.7946
	AR_{MSSD}	0.8733	0.8528	0.8733	**0.8816**
	AR_{MSPD}	0.8033	0.8473	0.8132	0.8294
	AR	0.8233	0.8249	0.8275	**0.8352**
T-LESS PBR	AR_{VSD}	0.1741	0.2730	0.2732	0.2883
	AR_{MSSD}	0.1998	0.2887	0.2921	0.3073
	AR_{MSPD}	0.1897	0.3177	0.3232	0.3394
	AR	0.1879	0.2931	0.2962	0.3117
T-LESS S + R	AR_{VSD}	0.2410	0.3578	0.3275	0.3521
	AR_{MSSD}	0.2922	0.3950	0.3699	0.3912
	AR_{MSPD}	0.2822	0.4154	0.3802	0.4055
	AR	0.2718	0.3894	0.3592	0.3830

Table 4
Results of stacking.

Dataset		Stacking Ridge (R)	Stacking SVR (SVR)	Stacking tree (T)	Stacking Linear (L)	Stacking KNN (KNN)	Stacking MLP (MLP)
LMO PBR	AR_{VSD}	0.3165 (s)	0.3146	0.1991	0.3165 (s)	0.3195	0.4408
	AR_{MSSD}	0.4375 (s)	0.5833	0.4707	0.4376 (s)	0.5789	0.6176
	AR_{MSPD}	0.6354 (s)	0.6298	0.5451	0.6355 (s)	0.6363	0.6812
	AR	0.4631 (s)	0.5092	0.4050	0.4632 (s)	0.5115	0.5799
YCB-V PBR	AR_{VSD}	0.4040 (s)	0.4479 (s)	0.2994 (s)	0.4040 (s)	0.3928	0.4462 (s)
	AR_{MSSD}	0.4896 (s)	0.5773 (s)	0.4487 (s)	0.4896 (s)	0.5117	0.5462 (s)
	AR_{MSPD}	0.3793 (s)	0.4268 (s)	0.2615 (s)	0.3792 (s)	0.3623	0.4175 (s)
	AR	0.4243 (s)	0.4840 (s)	0.3365 (s)	0.4243 (s)	0.4223	0.4700 (s)
YCB-V S + R	AR_{VSD}	0.7233 (s)	0.6024	0.3548	0.7233 (s)	0.564	0.6676
	AR_{MSSD}	0.8113 (s)	0.7967	0.6208	0.8113 (s)	0.7777	0.7979
	AR_{MSPD}	0.7218 (s)	0.6574 (s)	0.4023	0.7216 (s)	0.6446	0.6938 (s)
	AR	0.7521 (s)	0.6835 (s)	0.4593	0.7521 (s)	0.6621	0.7159
T-LESS PBR	AR_{VSD}	0.0441 (s)	0.1431 (s)	0.0513	0.0441 (s)	0.0714	0.0421
	AR_{MSSD}	0.0364 (s)	0.1301 (s)	0.0470	0.0364 (s)	0.0600 (s)	0.0535
	AR_{MSPD}	0.0521 (s)	0.1372 (s)	0.0560	0.0521 (s)	0.0685 (s)	0.0656
	AR	0.0442 (s)	0.1368 (s)	0.0514	0.0442 (s)	0.0664 (s)	0.0537
T-LESS S + R	AR_{VSD}	0.0589 (s)	0.2216 (s)	0.0747	0.0590 (s)	0.1079	0.0736
	AR_{MSSD}	0.0694 (s)	0.2500	0.0996	0.0694 (s)	0.1182	0.1198
	AR_{MSPD}	0.0789 (s)	0.2433	0.1023	0.0789 (s)	0.1208	0.1133
	AR	0.0691 (s)	0.2370 (s)	0.0922	0.0691 (s)	0.1156	0.1022

to its estimation. These scores are usually describing how confident the models are about the object detected. This score does not provide any information about how precise the pose is, but the confidence of the class of the detection.

Each of the poses is weighted accordingly to the obtained score over the total score. The weight of each model is calculated following Eq. 8 and the refined translation and rotation are obtained respectively with Eqs. 9 and 10.

$$w_i = \frac{1}{(1-s)^2 + \varepsilon} \tag{8}$$

$$T_r = \sum_{i=1}^{n} w_i \cdot T_i \tag{9}$$

Table 5
Comparative of all level-0 and ensemble methods. ∗: Cosypose, △: PVN3D, ◦:FFB6D

Dataset	Metric	Level-0	Merge	Stacking	CDPNv2	Pix2Pose
LMO PBR	VSD	0.480∗	**0.482 (WC)**	0.440 (MLP)	0.469	0.473
	MSSD	0.652△	0.650 (WC)	0.618 (MLP)	0.689	0.631
	MSPD	**0.812**∗	0.743 (WC)	0.681 (MLP)	0.731	0.659
	AR	**0.633**∗	0.626 (WC)	0.580 (MLP)	0.630	0.588
YCB-V PBR	VSD	0.589△	0.550 (WC)	0.448 (SVR-s)	0.511	-
	MSSD	0.696△	0.624 (WC)	0.577 (SVR-s)	0.603	-
	MSPD	**0.653**∗	0.540 (WC)	0.426 (SVR-s)	0.483	-
	AR	**0.616**∗	0.571 (WC)	0.484 (SVR-s)	0.532	-
YCB-V S + R	VSD	0.786◦	**0.796 (SC)**	0.723 (R-s/L-s)	0.590	0.766
	MSSD	0.879◦	**0.882 (WC)**	0.811 (R-s/L-s)	0.701	0.817
	MSPD	**0.850**∗	0.847 (W)	0.722 (R-s)	0.565	0.758
	AR	0.822∗	**0.835 (WC)**	0.752 (R-s/L-s)	0.619	0.780
T-LESS PBR	VSD	**0.571**∗	0.288 (WC)	0.143 (SVR-s)	0.368	-
	MSSD	**0.589**∗	0.307 (WC)	0.130 (SVR-s)	0.449	-
	MSPD	**0.761**∗	0.339 (WC)	0.137 (SVR-s)	0.488	-
	AR	**0.640**∗	0.312 (WC)	0.137 (SVR-s)	0.435	-
T-LESS S + R	VSD	**0.669**∗	0.358 (W)	0.222 (SVR-s)	0.385	0.438
	MSSD	**0.695**∗	0.395 (W)	0.250 (SVR)	0.489	0.548
	MSPD	**0.821**∗	0.415 (W)	0.243 (SVR)	0.516	0.549
	AR	**0.728**∗	0.389 (W)	0.237 (SVR-s)	0.464	0.512

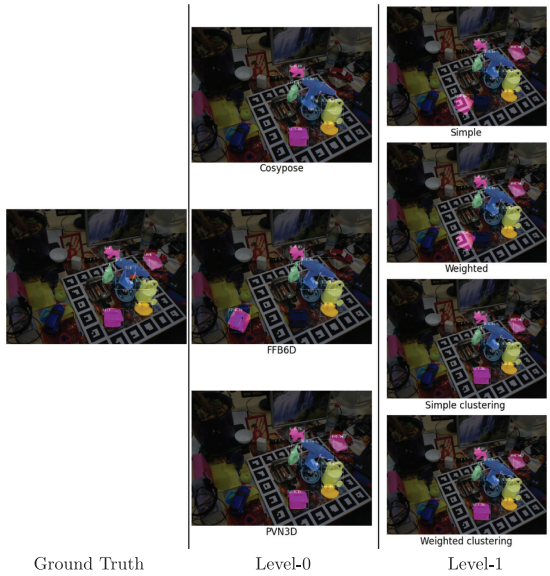


Fig. 8. Qualitative comparative of merge methods on a LMO dataset image.

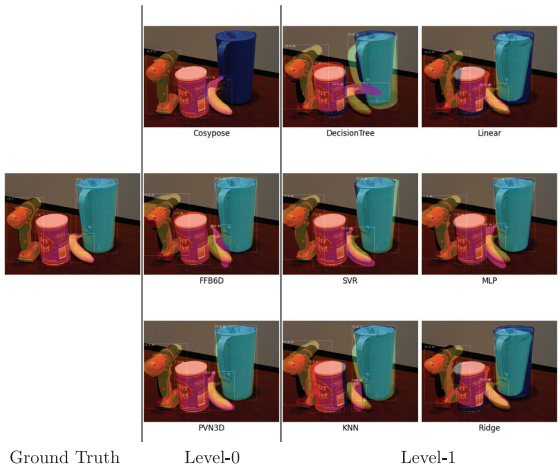


Fig. 9. Qualitative comparative of stacking methods on a YCB-V dataset image.

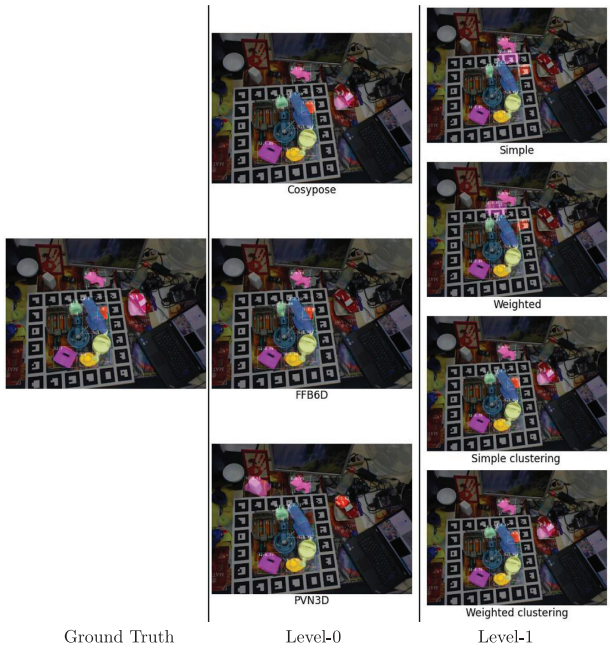


Fig. A.10. Qualitative comparative of merge methods on LMO dataset image 8 from scene 2.

Fig. A.11. Qualitative comparative of merge methods on LMO dataset image 110 from scene 2.

$$R_r = \text{argmin}_{R \in SO(3)} \sum_{i=1}^{n} w_i ||R_i - R||^2 \quad (10)$$

where ε is very small positive number close to zero used to prevent the division by zero.

4.1.3. Clustering merge

A limitation of the previous approach is that if a model makes a bad estimation, the resulting pose may be badly influenced, even if the remaining methods were quite precise. That is, the resulting pose may be worse than the best of the initial poses.

Therefore, we designed a method based on clustering the candidate poses. To reduce the impact of incorrect poses, we focus the processing on the most similar poses, which is deduced by the cluster gathering more poses. For an equal number of poses, the cluster with the highest mean confidence is chosen. Then, to obtain the refined pose, a simple merge or a weighted merge is applied only on the poses belonging to the selected cluster. Algorithm 1 explains how the clusters are created and how the resulting pose is obtained. The distance thresholds have been selected based on previous experiments. The distance function is the Euclidean distance between the two given poses. Only translation is considered, rotation is not used to calculate the distance.

Algorithm 1: Clustering merge method

```
function clusteringposes
    n_items ← len(poses)
    if n_items = 1 then
        result ← poses[0]
    else
        biggest_cluster ← []
        for i ∈ (0, len(poses)) do
            cluster ← [poses[i]]
            for j ∈ (i, len(poses)) do
                if distance(poses[i], poses[j]) < threshold then
                    cluster.append(pose[j])
                end if
            end for
            if len(biggest_cluster < cluster) then
                biggest_cluster ← cluster
            else if len(biggest_cluster == cluster)
                if score(biggest_cluster) < score(cluster) then
                    biggest_cluster ← cluster
                end if
            end if
```

(continued on next page)

(*continued*)

Algorithm 1: Clustering merge method

```
    end for
    result ← merge(biggest_cluster)   ▷simple merge or weighted merge.
  end if
  return result
end function
```

4.2. Ensemble-Stacking

Stacking or stacked generalization is an ensemble method that combines different machine learning models using another machine learning algorithm [36,37,46]. The base models or level-0 models are trained using the training data. Then, the validation data is predicted using the trained models, generating a new dataset where the independent variables are the outputs of each model. Finally, the combiner or level-1 model is trained using the validation data. Fig. 3 shows the full pipeline of training and evaluating this model.

4.2.1. Level-1 models

Level-1 methods combine the results that level-0 models have output. The following level-1 methods have been used in the proposed method:

- SVR [55] or support vector regression is based on the support vector machine concept [56]. This method assumes that the input vectors are non-linearly mapped to a high-dimension feature space. Nevertheless, the SVR computational complexity does not depend on the dimensionality of the input space, and it has excellent generalization capability with high prediction accuracy.
- Decision Trees [57] are formed by a collection of rules that are modelled according to input variables values. Variables are selected to get the best split to differentiate the values of the dependent variable. The rule splits the tree into two nodes and the process is repeated for each child node recursively. The method stops when no more gain can be made or by some pre-set stopping rules.
- The linear Regression [58] is based on the representation of a linear equation. The coefficients are learned from the training data using statistical properties(e.g., means, standard deviations, correlations or/and covariance), using Ordinary Least Squares to minimize the squared residuals or using the Gradient Descent method, for instance. We have used 2 linear regressions: Ordinary least squares Linear Regression and Ridge Linear Regression (Linear least squares with l2 regularization).
- KNN Regressor or K-Nearest Neighbors regressor [59,60] is a non-parametric model that uses the feature similarity to predict values. It uses the Euclidean, Manhattan or Minkowski distance as the KNN classification model. The target is predicted by local interpolation of the targets associated to the nearest neighbors in the training set.

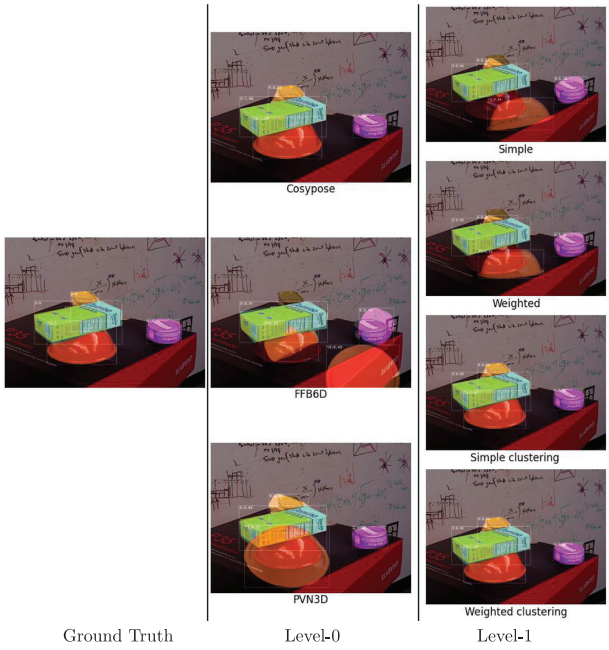


Fig. A.12. Qualitative comparative of merge methods on YCBV dataset image 1130 from scene 49.

- The MultiLayer Perceptron or MLP [61] is a type of feedforward Artificial Neural Network. MLP is usually a neural network containing a single hidden layer. We use Mean Squared Error (MSE) as loss function.

5. Experiments

We evaluate our method on a subset of the BOP challenge [2,62] benchmark. This objective of this challenge is to set a common benchmark to compare different state-of-the-art 6 DoF object pose estimation methods. The challenge defines an evaluation methodology and datasets to use. We compare the results obtained with the state-of-the-art methods (level-0) and the ensemble methods.

5.1. Datasets

BOP challenge includes 7 core datasets. We use a subset of those datasets including Linemod-Occlusion [51,63], T-LESS [64] and YCB-Video [52]. This subset is selected for simplicity and because it is relevant for our use case. Each dataset is composed of 50 k PBR (Physically-based renderer) synthetic images generated using BlenderProc for the BOP challenge for training, another training set with real and synthetic images that each of the datasets provides and test images. BOP challenge is evaluated using a subset from the test images (BOP test images). Figs. 4–7 show images of the LMO dataset, YCB-V dataset, T-LESS dataset and PBR images from all datasets, respectively.

5.1.1. Linemod

Linemod-Occlusion (LM-O) [51,63] is a dataset first proposed by [51] and later redefined by [63]. The initial dataset consists of 15 texture-less objects on a cluttered desk. The later redefinition takes only into account the scene '000002' with extra ground-truth information. This configuration is more challenging and only uses the 13 objects with proper 3D models (2 were omitted since proper 3D models were missing) and correct ground-truth poses (Some ground truth poses from the original dataset were incorrect). In the BOP challenge, the LM-O has available the 50 K PBR training images generated with BlenderProc (provided by BOP challenge), 400 k PBR training images from Microsoft Research (MSR) [65], synthetic training images of isolated objects generated in [63] and test images.

5.1.2. YCB-Video

YCB-Video [52,18] is a 6D pose estimation dataset composed of 21 objects from the YCB dataset [52] observed in 92 videos with 133827 frames. 80 videos are used for training and 12 videos for testing. In the BOP challenge, the YCB-V dataset has the 50 k PBR training images generated with BlenderProc (provided by BOP challenge), original synthetic training images, real training images and test images.

5.1.3. T-LESS

T-LESS [64] is an industry-relevant texture-less object dataset. The objects are often similar in shape and some objects are parts of others. Three different synchronized sensors have been used to

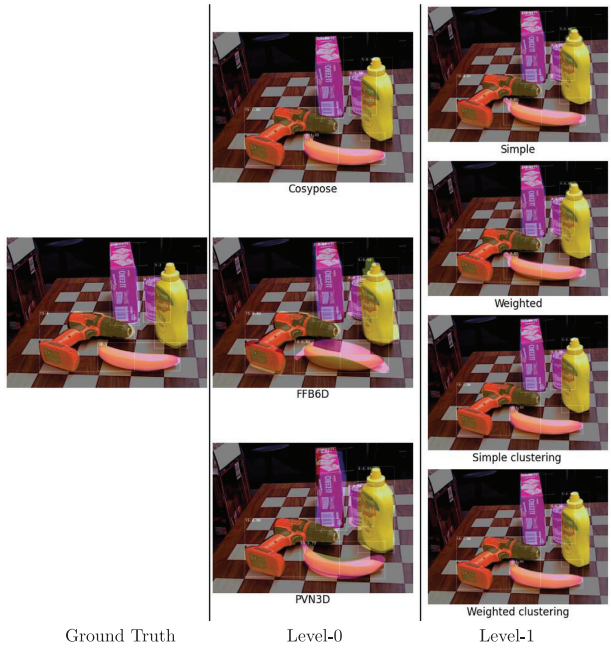


Fig. A.13. Qualitative comparative of merge methods on YCBV dataset image 1052 from scene 50.

capture the real training and test images. Real training images depict individual objects against a black background. In the BOP challenge, the T-LESS dataset has the 50 k PBR training images generated with BlenderProc (provided by BOP challenge), synthetic training images of isolated objects, real training images of isolated objects and test images.

5.2. Evaluation metrics

The BOP challenge task is 6D localization of Varying number of instances of a Varying number of objects in a single RGB-D image (ViVo task). In the ViVo task, each image can contain 0 or n instances of 1 or j objects, where j is always less or equal to J, the number of objects of the corresponding dataset. In our case, we simplify the problem, and we focus on the 6D localization of Single instance of Varying number of objects in a single RGB-D image (SiVo task). In the SiSo task, each image can contain 0 or 1 instances of 1 or j objects.

To evaluate the level-0 models and all the ensemble methods, we have followed the evaluation methodology that the BOP challenge follows. Three pose-error functions are used: Visible Surface Discrepancy (VSD) [2,66], Maximum Symmetry-Aware Surface Distance (MSSD) [67] and Maximum Symmetry-Aware Projection Distance (MSPD) [62]. An estimated pose is considered correct if the value for each specific pose-error function (VSD, MSSD or MSPD) is less than a given threshold. The recall is the ratio between the correctly predicted poses over the annotated poses. Given different settings of the threshold and misalignment tolerances, the average recall of a specific pose-error function e (AR_e) is defined as the average of all the recalls for the different configurations on that pose-error function. The Average Recall (AR) is the average of the three specific pose-error function Average Recalls ($AR_{VSD}, AR_{MSSD}, AR_{MSPD}$). Each dataset and method is evaluated by the Average Recall (AR).

5.2.1. VSD

VSD only considers visible parts of the objects. Therefore, VSD treats poses that can not be distinct from one another as equivalent. Eq. 11 shows how to calculate the VSD error.

$$e_{VSD}\left(\hat{D},\overline{D},\hat{V},\overline{V},\tau\right) = avg_{p\in\hat{V}\cup\overline{V}}\begin{cases} 0 & \text{if} p\in\hat{V}\cap\overline{V}\wedge|\hat{D}(p)-\overline{D}(p)|<\tau \\ 1 & \text{otherwise}\end{cases} \tag{11}$$

where $\hat{D}$ and $\overline{D}$ are distance maps from rendering the model in the estimated pose $\hat{P}$ and ground-truth pose $\overline{P}$, respectively; and $\hat{V}$ and $\overline{V}$ the visibility masks. AR_{VSD} is calculated from the misalignment tolerance ranging from 5% to 50% of the object diameter with a step of 5%, and the threshold from 0.05 to 0.5 with a step of 0.05.

5.2.2. MSSD

The maximum distance is relevant for robotic manipulation as it indicates if there is a chance of a successful grasp. Compared to ADD/ADI [66,51], this metric behaves more independently of

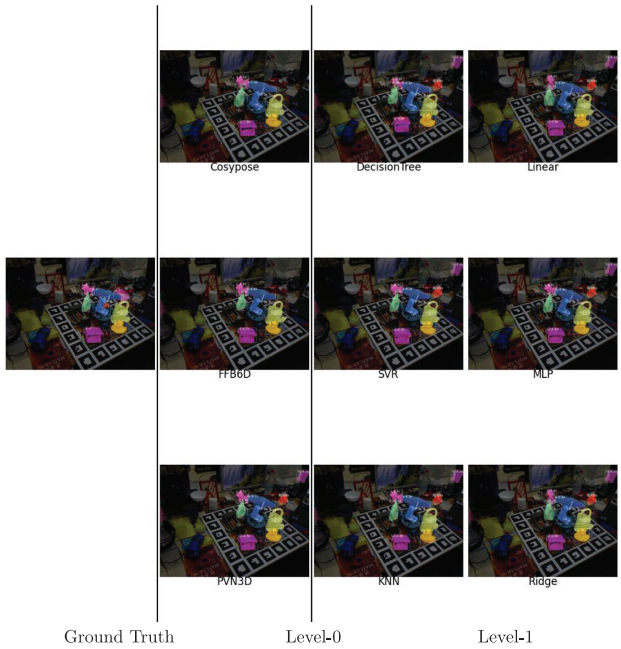


Fig. A.14. Qualitative comparative of stacking methods on LMO dataset image 47 from scene 2.

the sampling of the mesh. Eq. 12 shows how to calculate MSSD error.

$$e_{MSSD}\left(\hat{\mathbf{P}}, \bar{\mathbf{P}}, S_M, V_M\right) = min_{\mathbf{S}\in S_M} max_{\mathbf{x}\in V_M} \|\hat{\mathbf{P}}\mathbf{x} - \bar{\mathbf{P}}\mathbf{S}\mathbf{x}\|_2 \quad (12)$$

where $\widehat{P}$ is the estimated pose, $\overline{P}$ is the ground-truth pose, S_M is a set of global symmetry transformations and V_M is a set of mesh vertices of the model. AR_{MSSD} is calculated from the threshold ranging from 5% to 50% of the object diameter with a step of 5%.

5.2.3. MSPD

This measure is relevant for augmented reality applications and suitable for evaluation of RGB-only methods. Eq. 13 shows how to calculate MSPD error.

$$e_{MSPD}\left(\hat{\mathbf{P}}, \bar{\mathbf{P}}, S_M, V_M\right) = min_{\mathbf{S}\in S_M} max_{\mathbf{x}\in V_M} \|\text{proj}\left(\hat{\mathbf{P}}\mathbf{x}\right) - \text{proj}(\bar{\mathbf{P}}\mathbf{S}\mathbf{x})\|_2 \quad (13)$$

where *proj* is the 2D projection operation. AR_{MSPD} is calculated from the threshold ranging from $5r$ to $50r$ with a step of $5r$, where $r = w/640$ and w is the width of the image in pixels.

6. Results

We have defined 3 merging methods: simple, weighted and clustering; and a stacking method. First we are presenting the scores of level-0 models to compare them with all the ensemble methods defined.

Table 2 shows the results obtained for LMO PBR, YCB-V PBR, YCB-V S + R, T-LESS PBR and T-LESS S + R. S + R (SYN + REAL) is composed of the synthetic pbr images and the real images. For each dataset and metric, the highest score is set to bold (one per row). If we focus on the Average Recall(*AR*), Cosypose is the best level-0 approach on every dataset except for YCB-V PBR. The three models have similar *AR* on LMO PBR and YCB-V S + R. For YCB-V PBR, PVN3D and Cosypose behave similarly while FFB6D has lower *AR*. Results on T-LESS are significantly different: Cosypose outperforms FFB6D and PVN3D. This is because PVN3D and FFB6D work better for datasets like LMO and YCB-V since their authors designed these methods for those datasets, and the implementation is not prepared for ViVo task as Cosypose is. Indeed, the T-LESS dataset has more than one instance of the same object on some images. This leads to poor performance in many examples.

If we focus on the average recall of each metric, AR_{VSD} behaves similarly to *AR*. Even though, for this metric Cosypose is not the best for every dataset (PVN3D wins on YCB-V PBR, and FFB6D wins on YCB-V S + R).

On AR_{MSSD}, Cosypose is not the best method on any dataset, but, on AR_{MSPD}, Cosypose is the best on every dataset. On the one hand, MSSD is designed to verify if a pose has a chance of a successful grasp. On the other hand, MSPD does not evaluate the alignment along the optical axis (*Z* axis) and is designed for evaluating RGB-only methods. Therefore, MSSD is higher for PVN3D and FFB6D

Fig. A.15. Qualitative comparative of stacking methods on LMO dataset image 387 from scene 2.

since both methods use pointclouds instead of only RGB images. And for the same reason, Cosypose has higher MSPD because it is more suitable for working with RGB only input. Cosypose only uses depth if some pointcloud level refinement is performed, such as ICP.

The first strategy we are evaluating is the merge ensemble. Table 3 shows the results of the different merging methods for each dataset. The scores are underlined if, for that dataset and metric, the merge method has obtained a higher value than the corresponding highest level-0 score, and bold scores are the highest from the underlined scores for each row. On LMO PBR, results are close to Cosypose for all metrics, but Weighted clustering has the highest AR_{VSD}. On YCB-V PBR, PVN3D wins on AR_{VSD}, AR_{MSSD} and AR and none of the merge methods achieves better results. Nevertheless, weighted clustering has the highest AR_{VSD}, AR_{MSSD} and AR for YCB-V S + R. Merge methods can not deal with the T-LESS dataset since two out of three level-0 methods have very poor predictions. Even though, merge methods improve the scores of PVN3D and FFB6D.

The second strategy is stacking. Table 4 shows results of the stacking in which metrics with (s) denote that including the scores of the level-0 models improves the metric of the level-1. In this case, none of the level-1 models of the stacking improves the metrics obtained with the level-0 models. Moreover, there is not a level-1 model that stands out from the others. For each dataset, a different level-1 model obtains the highest metric.

Table 5 shows a comparative of the different level-0, all ensemble methods and two state-of-the-art methods leading the BOP challenge (CDPNv2 [7] and Pix2Pose [8]). As expected, some level-0 work better for some datasets and the leaderboard is very heterogeneous. On merge methods, the weighted clustering is by far the best approach, winning on 14 out of 20 rows. On stacking, instead, there is not a clear winner and a different level-1 works better for each dataset. Nevertheless, almost on every metric on the LMO and YCB-V datasets merge methods are better than the state-of-the-art methods.

Even if our initial hypothesis was that stacking strategy would lead to a better performance than merge strategy, since level-1 models could learn to distinguish which methods work better for each object and combine them in order to minimize the errors of the refined predictions, that is not the reality. This could be due to various reasons which may include that bad predictions of level-0 models lead to a bad level-1 model performance, or level-1 models were not able to learn a pattern. Nevertheless, if we focus on qualitative results (Figs. 8 and 9) we can see that the obtained results are valid. Fig. 8 shows a qualitative comparison of the merge method on the LMO dataset. Simple and weighted merge have some errors on the cat object but both clustering methods refine the prediction and obtain a better result. For example, on Fig. 9, even if each individual prediction of the level-0 models is not perfect, the predictions that are not present in some predictions is obtained from the others and refined. SVR and MLP outper-

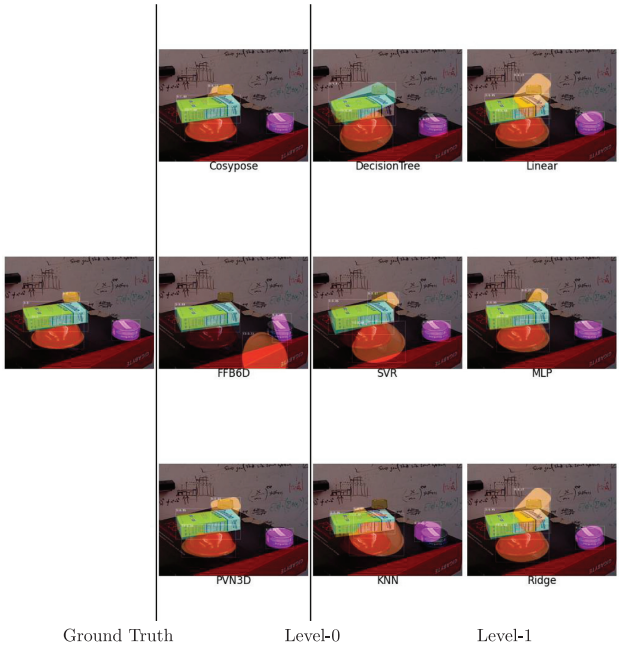


Fig. A.16. Qualitative comparative of stacking methods on YCBV dataset image 1557 from scene 49.

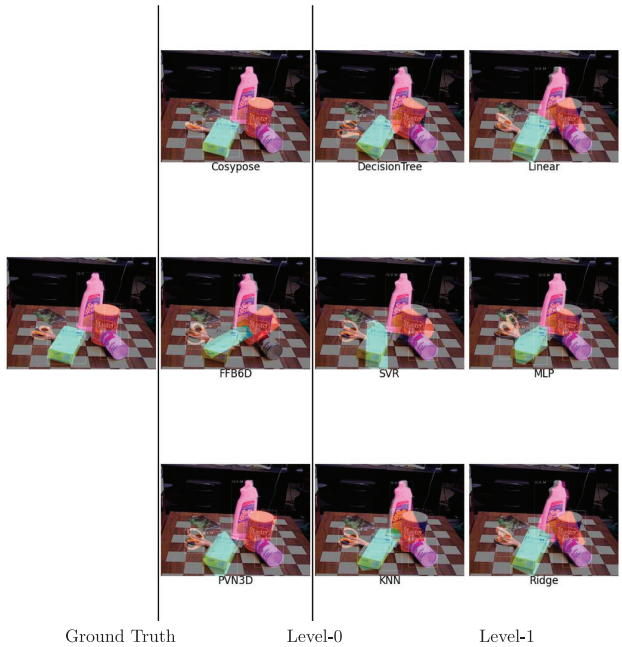


Fig. A.17. Qualitative comparative of stacking methods on YCBV dataset image 672 from scene 51.

form other level-1 models on situations like this. More examples of qualitative results can be found in the Appendix A.

7. Conclusions

To deal with 6DoF pose estimation task, we have designed some ensemble strategies. We have proposed 2 ensemble strategies for deep learning techniques: merge and stacking. On the one hand, merge strategy is based on the average of the predictions. This enables a geometrical refinement of the poses and, including the clustering technique, excludes the poses that do not represent the same instance. On the other hand, stacking trains a machine learning model (level-1) that inputs the outputs of the base models (level-0) to refine the estimated poses. Within merge strategies, the best method is weighted clustering which achieves 0.8352 AR, 0.7946 AR_{VSD}, 0.8816 AR_{MSSD} on YCB-V dataset using synthetic and real images, outperforming results obtained with FFB6d, PVN3D and Cosypose. Weighted clustering also improves AR_{VSD} on LMO. Stacking methods, instead, did not improve none of the results obtained with the level-0 models. The obtained results are for every dataset and level-1 model worse than the results obtained with level-0 models.

As stated before, qualitative results show that predictions are good enough for many applications. For example, robotic applications, where a robot must pick objects and little rotation discrepancies are not crucial, could obtain good results with the ensemble methods.

Future work will include using different and more level-0 models, discarding level-0 models that are not prepared for VIVO task and validating the obtained results in a real robotic application whose objective is to correctly grasp the maximum number of objects.

Data availability

Data will be made available on request.

Declaration of Competing Interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Ibon Merino reports financial support was provided by Basque Government.

Acknowledgements

This paper has been supported by the project PROFLOW under the Basque program ELKARTEK, grant agreement No. KK-2022/00024.

Appendix A. More examples of voting and stacking

Figs. A.10 and A.11 and Figs. A.12 and A.13 show more examples of the comparatives of merge methods used on LMO and YCBV, respectively; and Figs. A.14 and A.15 and Figs. A.16 and A.17 show more examples of comparative of stacking methods for the LMO and YCBV datasets, respectively.

References

[1] G. Marullo, L. Tanzi, P. Piazzolla, E. Vezzetti, 6d object position estimation from 2d images: a literature review, Multimedia Tools and Applications (2022) 1–39.

[2] T. Hodaň, F. Michel, E. Brachmann, W. Kehl, A. Glent Buch, D. Kraft, B. Drost, J. Vidal, S. Ihrke, X. Zabulis, C. Sahin, F. Manhardt, F. Tombari, T.-K. Kim, J. Matas, C. Rother, BOP: Benchmark for 6D object pose estimation, European Conference on Computer Vision (ECCV).

[3] B. Drost, M. Ulrich, N. Navab, S. Ilic, Model globally, match locally: Efficient and robust 3d object recognition, in: 2010 IEEE computer society conference on computer vision and pattern recognition, IEEE, 2010, pp. 998–1005.

[4] C. Raposo, J.P. Barreto, Using 2 point+ normal sets for fast registration of point clouds with small overlap, in: 2017 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2017, pp. 5652–5658.

[5] J. Vidal, C.-Y. Lin, X. Lladó, R. Martí, A method for 6d pose estimation of free-form rigid objects using point pair features on range data, Sensors 18 (8) (2018) 2678.

[6] P. Rodrigues, M. Antunes, C. Raposo, P. Marques, F. Fonseca, J.P. Barreto, Deep segmentation leverages geometric pose estimation in computer-aided total knee arthroplasty, Healthcare technology letters 6 (6) (2019) 226–230.

[7] Z. Li, G. Wang, X. Ji, Cdpn: Coordinates-based disentangled pose network for real-time rgb-based 6-dof object pose estimation, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 7678–7687.

[8] K. Park, T. Patten, M. Vincze, Pix2pose: Pixel-wise coordinate regression of objects for 6d pose estimation, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 7668–7677.

[9] S. Zakharov, I. Shugurov, S. Ilic, Dpod: 6d pose object detector and refiner, in: Proceedings of the IEEE/CVF international conference on computer vision, 2019, pp. 1941–1950.

[10] M. Sundermeyer, Z.-C. Marton, M. Durner, R. Triebel, Augmented autoencoders: Implicit 3d orientation learning for 6d object detection, International Journal of Computer Vision 128 (3) (2020) 714–729.

[11] Y. He, W. Sun, H. Huang, J. Liu, H. Fan, J. Sun, Pvn3d: A deep point-wise 3d keypoints voting network for 6dof pose estimation, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2020, pp. 11632–11641.

[12] Y. He, H. Huang, H. Fan, Q. Chen, J. Sun, Ffb6d: A full flow bidirectional fusion network for 6d pose estimation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 3003–3013.

[13] Perkins, A model-based vision system for industrial parts, IEEE Transactions on Computers C-27 (2) (1978) 126–143.

[14] Tsuji Yachida, A versatile machine vision system for complex industrial parts, IEEE Transactions on Computers C-26 (9) (1977) 882–894.

[15] D. Kragic, M. Vincze, Vision for robotics, Now Publishers Inc, 2009.

[16] D.G. Lowe, Object recognition from local scale-invariant features, in: Proceedings of the seventh IEEE international conference on computer vision, Vol. 2, IEEE, 1999, pp. 1150–1157.

[17] J. Rambach, C. Deng, A. Pagani, D. Stricker, Learning 6dof object poses from synthetic single channel images, in: 2018 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct), IEEE, 2018, pp. 164–169.

[18] Y. Xiang, T. Schmidt, V. Narayanan, D. Fox, Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes, arXiv preprint arXiv:1711.00199.

[19] W. Zou, D. Wu, S. Tian, C. Xiang, X. Li, L. Zhang, End-to-end 6dof pose estimation from monocular rgb images, IEEE Transactions on Consumer Electronics 67 (1) (2021) 87–96.

[20] K. He, G. Gkioxari, P. Dollár, R. Girshick, Mask r-cnn, in: Proceedings of the IEEE international conference on computer vision, 2017, pp. 2961–2969.

[21] C. Wang, D. Xu, Y. Zhu, R. Martín-Martín, C. Lu, L. Fei-Fei, S. Savarese, Densefusion: 6d object pose estimation by iterative dense fusion, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2019, pp. 3343–3352.

[22] G. Pavlakos, X. Zhou, A. Chan, K.G. Derpanis, K. Daniilidis, 6-dof object pose from semantic keypoints, in: 2017 IEEE international conference on robotics and automation (ICRA), IEEE, 2017, pp. 2011–2018.

[23] M. Rad, V. Lepetit, Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth, in: Proceedings of the IEEE international conference on computer vision, 2017, pp. 3828–3836.

[24] B. Tekin, S.N. Sinha, P. Fua, Real-time seamless single shot 6d object pose prediction, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 292–301.

[25] X. Zhou, A. Karpur, L. Luo, Q. Huang, Starmap for category-agnostic keypoint and viewpoint estimation, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 318–334.

[26] A. Nigam, A. Penate-Sanchez, L. Agapito, Detect globally, label locally: Learning accurate 6-dof object pose estimation by joint segmentation and coordinate regression, IEEE Robotics and Automation Letters 3 (4) (2018) 3960–3967.

[27] S. Peng, Y. Liu, Q. Huang, X. Zhou, H. Bao, Pvnet: Pixel-wise voting network for 6dof pose estimation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 4561–4570.

[28] H. Wang, S. Sridhar, J. Huang, J. Valentin, S. Song, L.J. Guibas, Normalized object coordinate space for category-level 6d object pose and size estimation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 2642–2651.

[29] C.R. Qi, L. Yi, H. Su, L.J. Guibas, Pointnet++: Deep hierarchical feature learning on point sets in a metric space, Advances in neural information processing systems 30.

[30] C.R. Qi, H. Su, K. Mo, L.J. Guibas, Pointnet: Deep learning on point sets for 3d classification and segmentation, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 652–660.

[31] H. Liu, S. Fang, Z. Zhang, D. Li, K. Lin, J. Wang, Mfdnet: Collaborative poses perception and matrix fisher distribution for head pose estimation, IEEE Transactions on Multimedia 24 (2021) 2449–2460.

[32] H. Liu, T. Liu, Z. Zhang, A.K. Sangaiah, B. Yang, Y. Li, Arhpe: Asymmetric relation-aware representation learning for head pose estimation in industrial human–computer interaction, IEEE Transactions on Industrial Informatics 18 (10) (2022) 7107–7117.

[33] H. Liu, C. Zheng, D. Li, X. Shen, K. Lin, J. Wang, Z. Zhang, Z. Zhang, N.N. Xiong, Edmf: Efficient deep matrix factorization with review feature learning for industrial recommender system, IEEE Transactions on Industrial Informatics 18 (7) (2021) 4361–4371.

[34] H. Liu, T. Liu, Y. Chen, Z. Zhang, Y.-F. Li, Ehpe: Skeleton cues-based gaussian coordinate encoding for efficient human pose estimation, IEEE Transactions on Multimedia.

[35] T. Liu, J. Wang, B. Yang, X. Wang, Ngdnet: Nonuniform gaussian-label distribution learning for infrared head pose estimation and on-task behavior understanding in the classroom, Neurocomputing 436 (2021) 210–220.

[36] D.H. Wolpert, Stacked generalization, Neural networks 5 (2) (1992) 241–259.

[37] K.M. Ting, I.H. Witten, Issues in stacked generalization, Journal of artificial intelligence research 10 (1999) 271–289.

[38] A. Ledezma, R. Aler, D. Borrajo, Heuristic Search-Based Stacking of Classifiers, IGI Global, 2002.

[39] A. Gupta, A.R. Thakkar, Optimization of stacking ensemble configuration based on various metaheuristic algorithms, IEEE (2014) 444–451.

[40] Y. Chen, M.L. Wong, An ant colony optimization approach for stacking ensemble, in, Second World Congress on Nature and Biologically Inspired Computing (NaBIC), IEEE 2010 (2010) 146–151.

[41] Y. Chen, M.-L. Wong, Applying ant colony optimization in configuring stacking ensemble, IEEE (2012) 2111–2116.

[42] P. Shunmugapriya, S. Kanmani, Optimization of stacking ensemble configurations through artificial bee colony algorithm, Swarm and Evolutionary Computation 12 (2013) 24–32.

[43] J. Kozak, Ensemble methods, in: Decision Tree and Ensemble Learning Based on Ant Colony Optimization, Springer, 2019, pp. 107–118.

[44] M.P. Sesmero, A.I. Ledezma, A. Sanchis, Generating ensembles of heterogeneous classifiers using stacked generalization, Wiley interdisciplinary reviews: data mining and knowledge discovery 5 (1) (2015) 21–34.

[45] S. Nair, A. Gupta, R. Joshi, V. Chitre, Combining varied learners for binary classification using stacked generalization, arXiv preprint arXiv:2202.08910.

[46] A.I. Naimi, L.B. Balzer, Stacked generalization: an introduction to super learning, European journal of epidemiology 33 (5) (2018) 459–464.

[47] C.-F. Tsai, Training support vector machines based on stacked generalization for image classification, Neurocomputing 64 (2005) 497–503, trends in Neurocomputing: 12th European Symposium on Artificial Neural Networks 2004.

[48] L. Peppoloni, M. Satler, E. Luchetti, C.A. Avizzano, P. Tripicchio, Stacked generalization for scene analysis and object recognition, in: IEEE 18th International Conference on Intelligent Engineering Systems INES 2014, 2014, pp. 215–220.

[49] A. Voulodimos, N. Doulamis, A. Doulamis, E. Protopapadakis, Deep learning for computer vision: A brief review, Computational intelligence and neuroscience (2018).

[50] Y. Labbé, J. Carpentier, M. Aubry, J. Sivic, Cosypose: Consistent multi-view multi-object 6d pose estimation, in: European Conference on Computer Vision, Springer, 2020, pp. 574–591.

[51] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, N. Navab, Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes, in: Asian conference on computer vision, Springer, 2012, pp. 548–562.

[52] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, A.M. Dollar, The ycb object and model set: Towards common benchmarks for manipulation research, in: 2015 international conference on advanced robotics (ICAR), IEEE, 2015, pp. 510–517.

[53] A. Simonelli, S.R. Bulo, L. Porzi, M. López-Antequera, P. Kontschieder, Disentangling monocular 3d object detection, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 1991–1999.

[54] F.L. Markley, Y. Cheng, J.L. Crassidis, Y. Oshman, Averaging quaternions, Journal of Guidance, Control, and Dynamics 30 (4) (2007) 1193–1197.

[55] H. Drucker, C.J. Burges, L. Kaufman, A. Smola, V. Vapnik, Support vector regression machines, Advances in neural information processing systems 9.

[56] C. Cortes, V. Vapnik, Support-vector networks, Machine learning 20 (3) (1995) 273–297.

[57] L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone, Classification and regression trees, Routledge, 1984.

[58] D.C. Montgomery, E.A. Peck, G.G. Vining, Introduction to linear regression analysis, John Wiley & Sons, 2021.

[59] T. Cover, P. Hart, Nearest neighbor pattern classification, IEEE transactions on information theory 13 (1) (1967) 21–27.
[60] N.S. Altman, An introduction to kernel and nearest-neighbor nonparametric regression, The American Statistician 46 (3) (1992) 175–185.
[61] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural networks 2 (5) (1989) 359–366.
[62] T. Hodaň, M. Sundermeyer, B. Drost, Y. Labbé, E. Brachmann, F. Michel, C. Rother, J. Matas, BOP challenge 2020 on 6D object localization, European Conference on Computer Vision Workshops (ECCVW).
[63] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, C. Rother, Learning 6d object pose estimation using 3d object coordinates, in: D. Fleet, T. Pajdla, B. Schiele, T. Tuytelaars (Eds.), Computer Vision – ECCV 2014, Springer International Publishing, Cham, 2014, pp. 536–551.
[64] T. Hodaň, P. Haluza, Š. Obdržálek, J. Matas, M. Lourakis, X. Zabulis, T-LESS: An RGB-D dataset for 6D pose estimation of texture-less objects, IEEE Winter Conference on Applications of Computer Vision (WACV).
[65] T. Hodaň, V. Vineet, R. Gal, E. Shalev, J. Hanzelka, T. Connell, P. Urbina, S. Sinha, B. Guenter, Photorealistic image synthesis for object instance detection, IEEE International Conference on Image Processing (ICIP).
[66] T. Hodaň, J. Matas, Š. Obdržálek, On evaluation of 6d object pose estimation, in: European Conference on Computer Vision, Springer, 2016, pp. 606–619.
[67] B. Drost, M. Ulrich, P. Bergmann, P. Hartinger, C. Steger, Introducing mvtec itodd-a dataset for 3d object recognition in industry, in: Proceedings of the IEEE international conference on computer vision workshops, 2017, pp. 2200–2208.

Ibon Merino received his B.S. degree in Computer Science from the University of the Basque Country, Donostia-San Sebastian, Spain in 2017 and his M. Sc. in Computer engineering and intelligent systems from the same university in 2018. Currently, he is working towards the Ph.D. degree at Tecnalia and the University of the Basque Country. His research focuses in 3D Computer Vision applied for robotics and merging deep learning methods and non-deep learning methods.

Jon Azpiazu received his B.S degree in Computer Engineer from the University of the Basque Country - Euskal Herriko Unibertsitatea, UPV/EHU (2003) and Certificate of Advanced Studies in the "Computer Engineering: Communications, Control and Artificial Intelligence" programme from the same university in 2005. In 2005 he joined Fatronik (later Tecnalia) as a researcher, initially participating in projects related to Artificial Intelligence and Machine Learning; later his research focused on aspects of Artificial Vision, leading the line of Artificial Vision in the field of robotics. In 2013 he joined SINTEF, the largest Scandinavian research centre, as a scientific researcher. During this stage, he participates and leads several research projects related to robotics and machine vision, both applied research projects (mainly concepts of use of robotics for the oil & gas industry), as well as basic research projects such as a project on semantic perception of the environment for snake robots. In 2017 he rejoined Tecnalia in the robotics group, leading two European projects in this field.

Dr. Anthony Remazeilles obtained a MSc in Computer Science and Artificial Intelligence from the University of Rennes 1 in 2001, and a PhD in Computer science from the University of Rennes in 2004. From 2001 to 2006, his research concerned the vision-based control of robotics systems within large navigation spaces. From 2006 to 2008, he was a Post-Doctoral researcher at the Commissariat ? l'Energie Atomique (CEA-LIST), Fontenay aux Roses, France. He was involved in the development of a semi-autonomous mobile arm providing vision-based grasping assistance to disabled people within the framework of the ITEA ANSO project that received an ITEA Appreciation Award in 2008. He has contributed to more than 25 peer-reviewed articles (conferences and journals), and holds a patent on vision-based object grasping. In 2004, he received the Best Paper Award in Pattern Recognition in the French Congress on Pattern Recognition and Artificial Intelligence (RFIA). Since 2008, he has been working as Researcher and then Project Leader in the Health Division of Tecnalia, designing intelligent robotics systems for human-centred applications. Dr. Remazeilles has been actively contributing to various project sponsored by the European Commission, such as Florence, CogLaboration, STIFF-FLOP, SARAFun, ROSIN, Robotunion, Eurobench, TraceBot.

Prof. Basilio Sierra is Full Professor at the Computer Science and Artificial Intelligence Department in the University of the Basque Country. Prof. Sierra has published more than 50 papers in journals with impact factor and more than 100 conference papers. His areas of scientific interest focus on Robotics, Machine Learning, Data Analysis and Computer Vision, having made numerous contributions in this area both at the theoretical level and in the development of new algorithms and their application in solving real problems. He has supervised 20 doctoral theses, and participated in several research projects related with Artificial Intelligence.