Universidad Zaragoza
1542

Trabajo de Fin de Grado

Implementación de un robot colaborativo guiado por visión artificial para una operación flexible de atornillado

Implementation of a machine vision-guide collaborative robot for flexible screwdriving operation

Autor

Miguel Ruiz Molinos

Directores

José J. Guerrero Campo

Rubén De La Rubia

Grado de Ingeniería de
Tecnologías Industriales
2024/2025

AGRADECIMIENTOS

En primer lugar, me gustaría destacar la gran labor del departamento de I+D de ASAI, que me ha ayudado en todas y cada una de las dudas que he tenido en el tiempo de realización del trabajo.

Agradecer también a Josechu Guerrero, mi director desde la universidad, su atención y comprensión durante estos meses. Así como a Rubén de la Rubia, mi director en la empresa, su ayuda y acogida desde el primer día a su grupo de trabajo.

Y, por último, a mi tía Bea, que en la parte final del proceso ha sabido transmitirme ánimo, paciencia y confianza cuando más lo necesitaba.

RESUMEN

Este Trabajo Fin de Grado se dedica al diseño, desarrollo e implementación de un sistema de robótica colaborativa respaldado por visión artificial, con la finalidad de automatizar de manera flexible una operación de atornillado en un contexto industrial.

Actualmente, existen herramientas colaborativas de atornillado, pero tienen ciertos límites a la hora de acceder a lugares poco accesibles. También existen puestos de trabajo con el atornillado automatizado, pero no colaborativo. Esto no presenta ningún inconveniente, programar diferentes trayectorias, suministrar tornillos o comandar la operación de atornillado mediante señales, pero no es el proceso final que en este trabajo se busca.

El sistema propuesto integra técnicas de visión artificial, con la capacidad de evaluar la zona peligrosa de trabajo localizando manos y parar si fuera necesario. Asimismo, se ha diseñado para ofrecer flexibilidad al proceso, con restricciones mínimas, lo que exige un reconocimiento eficaz y adaptable al entorno.

La comunicación fluida entre el robot y la computadora resulta esencial, debido a que las trayectorias del robot son variables, y dependen únicamente de lo que ocurra en la mesa de trabajo y de la interacción con el operario.

Los instrumentos empleados comprenden sistemas de visión industrial, métodos de calibración entre cámara y robot, simuladores, ambientes de programación colaborativa y programas de inteligencia artificial destinados a la percepción y al diseño robótico. El trabajo se desarrolla en el marco de una compañía enfocada en la investigación y desarrollo de tecnologías emergentes para la automatización industrial, lo que facilita la validación de la solución en un contexto real y con usos industriales específicos.

ÍNDICE

ÍNDICE DE ILUSTRACIONES

1. INTRODUCCIÓN

La tendencia actual en la industria se orienta hacia una creciente demanda de personalización. Los consumidores optan por artículos personalizados a sus gustos, esto obliga a la industria a ofrecer la posibilidad de elegir entre un catálogo amplio y variado, evitando la dependencia de un producto único.

La flexibilización industrial parte de esta necesidad y uno de sus puntos clave para poder obtener esto es la visión artificial. Esta tecnología facilita realizar cambios que, anteriormente, cambiaban la distribución de una planta industrial al completo. Esta sistemática elimina la dependencia de programas de código complejo, además, de tener que ubicar los productos en una posición determinada para cada una de las tareas para su fabricación, con el gasto que esto conlleva en diseño y fabricación de utillajes, además de pérdidas de tiempo.

Trataré un proyecto en el que está trabajando ASAI, una empresa de Ingeniería especializada en diseño, automatización, control, robótica colaborativa, supervisión de procesos y visión artificial.

El presente documento recoge todo el trabajo, estudio e investigación realizada para diseñar y desarrollar un sistema de robótica colaborativa que permita la interacción segura y eficiente entre un robot UR10e y operario en un proceso de trabajo industrial. Se buscarán las mejores estrategias para encontrar objetos en tiempo real y reconocer el entorno de trabajo para conseguir el guiado por visión.

2. ENTORNO DE TRABAJO

2.1 ESTADO DEL ARTE

Hasta ahora los atornilladores colaborativos que existen son muy limitantes a la hora de acceder a lugares complicados o estrechos, un ejemplo de ello es el OnRobot Screwdriver. Esta opción está al alcance de todo el mundo y cumple con todas las normativas.

Ilustración 1. Atornillador OnRobot Screwdriver

Tiene una sencilla implementación en cualquier principal marca de robot colaborativo, rápido intercambio de tornillos, control preciso del par y detección de errores de calidad, además de que asegura un rápido tiempo de ciclo [1].

Otro atornillador colaborativo en el mercado es el Fixtured Screwdriver SEV-C de Webber, tiene la ligereza necesaria para trabajar en aplicaciones colaborativas, posee una tecnología de vacío para alcanzar zonas con difícil acceso [2].

Ilustración 2. Atornillador colaborativo Webber

Ambas opciones tienen en común la aparición o desaparición de algún elemento en su punta atornillada cuando detectan un objeto inusual cerca de ella.

En el apartado de las lentes, existen cámaras aprobadas en Asia, no todavía en Europa con una normativa más restrictiva. Son capaces de detectar manos y personas en todo momento, por lo que en un futuro podría llegar a sustituir a la cámara entrenada con *deep learning* que en este trabajo se va a estudiar.

Una de estas cámaras es la JAKA VPS, este sistema de visión sofisticado de alto rendimiento es capaz de realizar tareas de inteligencia artificial y aprendizaje automático, reconocimiento de objetos, análisis de la postura corporal humana y comprensión del comportamiento. Tiene la posibilidad de realizar todo lo anterior y procesar la información a alta velocidad [3].

Ilustración 3. Cámara JAKA VPS

2.2 DESCRIPCIÓN PUESTO DE TRABAJO

El proyecto en el inicio se trataba de un proceso muy manual, en el que llegaba la luminaria a la cadena de producción y cada operario realizaba una serie de acciones sobre ella.

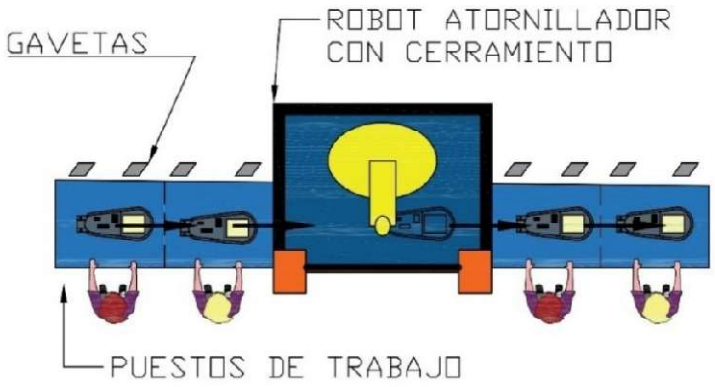


Ilustración 4. Estación de trabajo inicial

El proceso de montaje se divide en dos partes, en la primera se ensamblan los LED y las ópticas, así como el precableado de una placa de circuito impreso. La segunda etapa consiste en integrar dicha placa al resto de la luminaria, además de otras operaciones y el testeo final de la luminaria.

En las líneas de producción donde esta operación se ha decidido automatizar, el concepto ha sido el de instalar un robot en uno de los puestos de trabajo y realizar un cerramiento alrededor con elementos físicos (paredes rígidas de metacrilato) y barreras de seguridad.

Una variante sería la de robot asistente de operaciones de montaje. En este caso el atornillador lo seguiría manejando el operario, como en la situación actual, y el robot sería el que manipularía las piezas para posicionarlas frente al operario. Colocándolas en la mejor posición y ángulo para que éste lleve a cabo los distintos atornillados, pudiendo interactuar y secuenciar una serie de movimientos y posiciones que faciliten atornillados u otras operaciones de ensamblaje en el mismo puesto de trabajo, haciendo el robot la función de utillaje adaptativo.

El atornillado automático colaborativo habilitaría la automatización de la operación de atornillado en convivencia con operarios realizando otras tareas de montaje en el entorno inmediato donde el robot está operando. Esto permitiría solapar varias operaciones, reduciendo así el tiempo de ciclo.

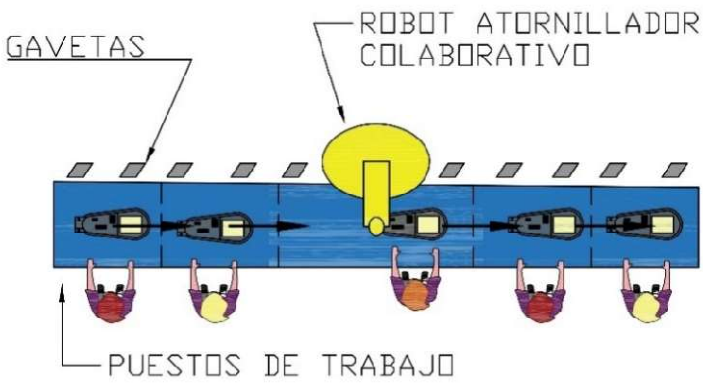


Ilustración 5. Puesto de trabajo con robot colaborativo

Tras un análisis del puesto de trabajo a automatizar se decidió colocar una zona de prueba en la empresa con la distribución que podemos observar en la *Ilustración 6.* El entorno de trabajo se ha intentado simular de la forma más cercana posible a lo que llegaría a ser en la puesta en marcha.

Ilustración 6. Puesto de trabajo de pruebas

Las pruebas se realizan sobre una mesa de 2,1 x 1 a 1,1 metros del suelo, con su parte superior de color blanco. El color de la mesa en este caso va a provocar una detección con mayor precisión de los objetos del entorno, debido al contraste de colores.

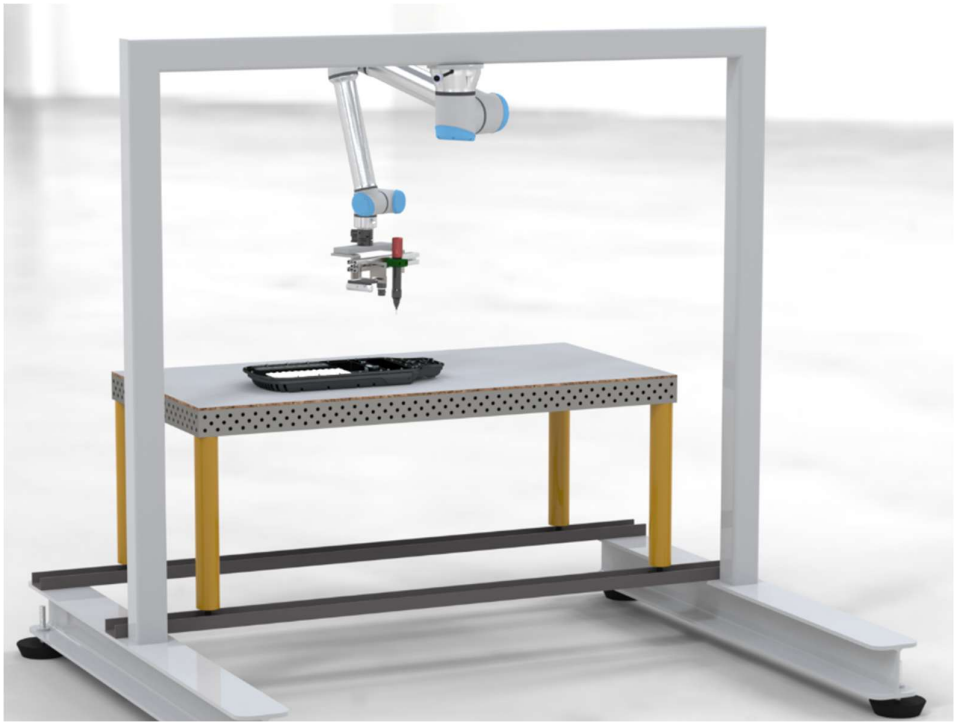


Ilustración 7. Espacio de trabajo renderizado

El robot instalado es un UR10e y la decisión de su disposición en el techo es debida a la tarea a realizar, facilitará el trabajo al robot y evita el contacto con el operario en la medida de lo posible. Se valoró la opción de instalarlo cara a cara con el operario, y sería también factible.

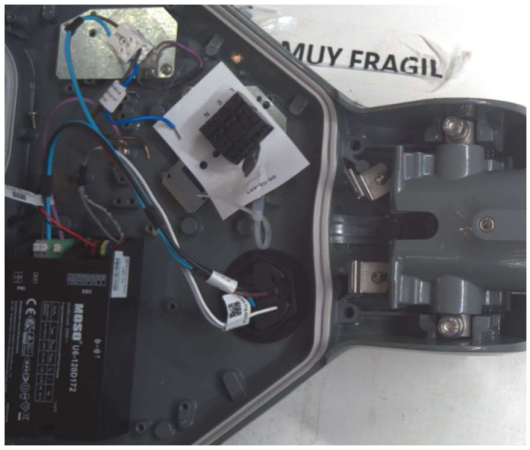


Ilustración 8: Luminaria

La *Ilustración 8* muestra una foto desde el punto de vista de una cámara matricial, 2D, a color. Podemos observar el color de la luminaria y los distintos componentes que la forman. La imagen proporcionada por esta cámara nos permitirá tanto realizar el guiado por visión artificial como un control de calidad final al acabar cada proceso.

2.3 ELECCIÓN ROBOT COLABORATIVO

2.3.1 ROBOTS COLABORATIVOS

Desde 2010, la industria 4.0 impulsa la implementación de tecnologías inteligentes en el lugar de trabajo industrial para aumentar la productividad. En 2020, la Comisión Europea lanzó la industria 5.0, este nuevo tipo de tendencia industrial insta a los actores a volver a situar al ser humano en el centro de la mayoría de los procesos. Promueve la colaboración entre humanos y cobots lo que conlleva su implementación, cada vez más, en las fábricas de todo tipo de sectores [4].

El término cobot se refiere a un dispositivo robótico que manipula objetos en colaboración con un operador humano [4]. Su uso podría ser una forma de reducir las limitaciones, y aumentar el rendimiento, sin sustituir a las personas. Los robots colaborativos trabajan en estrecha colaboración con los seres humanos, por lo tanto, tienen que estar equipados con características para evitar lesiones, trabajan bajo una estricta normativa de seguridad. Los autores han identificado un efecto positivo de la colaboración, como la reducción de los factores de riesgo de trastornos musculoesqueléticos. Sin embargo, la implementación de un sistema cobótico en el lugar de trabajo puede tener otros efectos negativos, supone otras tensiones para el operador y puede disminuir la calidad del resultado. Se han dado casos de pérdida de identidad laboral, aumento del tiempo de realización de las tareas y falta de adaptabilidad de los cobots a las limitaciones humanas.

Para su implementación en la industria, la colaboración cobótica debe ser eficaz en términos de resultados (mayor calidad de producción, menor tiempo de ciclo, menor número de gestos y menor número de errores). Además, realizar la tarea con menos gestos y con una menor carga de trabajo mental por parte del ser humano podría ser interesante para las empresas.

Todas estas variables están relacionadas con el concepto de usabilidad, el grado en que un producto puede ser utilizado por usuarios específicos para alcanzar objetivos específicos con eficacia, eficiencia y satisfacción en un contexto de uso particular. La eficacia (grado de finalización y precisión), puede medirse por el número de errores y el éxito de la tarea. La eficiencia (mínimo de recursos utilizados) puede medirse con el tiempo de ciclo y el número de gestos y la carga de trabajo. Por último, la satisfacción (cumplir las expectativas del usuario) puede medirse con la aceptabilidad.

Los robots colaborativos están restringidos mediante las normativas básicas que aplican a todos los robots industriales, como son la ISO 10218-1:2011 *Robots industriales - Requisitos de seguridad - Parte 1: Robots* y la ISO 10218-2:2011 *Robots industriales - Requisitos de seguridad - Parte 2: Sistemas robotizados y su integración.* Pero, tienen unas reglas específicas a cumplir por pertenecer al grupo de robots colaborativos y tener que compartir espacio con operarios humanos ISO/TS 15066:2016 *Robots colaborativos - Requisitos de seguridad.* Regla la cuál no aplica a nivel internacional todavía, regula los límites de fuerza y presión al chocar o contactar con algún elemento físico, limitación de velocidades o requisitos a diferentes alturas, debido a la diferencia de peligrosidad dependiendo de la zona de impacto con el operario, siendo la altura de trabajo de la cabeza la más restringida. Requisitos de sensores, monitoreo de velocidad entre otras.

2.3.2 UR10e

Universal Robots es una empresa fundada en 2005 en Dinamarca, cuyo objetivo principal era ofrecer robots accesibles para todo tipo de empresas, grandes, medianas o pequeñas. En la actualidad, es uno de los líderes en el mercado de los cobots, gracias a su gran adaptabilidad y su sencillez [5].

Su catálogo incluye dos líneas principales, la CB3-series y la e-Series. La primera de ellas fue con la que empezó la marca y hasta día de hoy siguen en el mercado, es robusta, buena disponibilidad de recambios y correcto funcionamiento probado en el tiempo. Los robots colaborativos de la e-Series, son más novedosos, más precisos, mayor seguridad por la posibilidad del control del par y mayor facilidad de uso [5] [6].

El modelo utilizado en este proyecto es el UR10e, dentro de la e-Series. Es ideal para nuestro puesto de trabajo, debido a que cumple con todas las especificaciones requeridas. Posee una carga útil de 12,5 kg, suficiente para funcionar correctamente con el atornillador, y la cámara Lucid, además de los soportes. Tiene un alcance operativo de 1300mm, superior a los 1100 mm a los que están situados los tornillos. Tiene un peso de 33,5 kg, capaz de trabajar en el techo. Programación sencilla mediante el intuitivo entorno de desarrollo PolyScope, colocado en el Teach Pendant. Además de cumplir con el requisito más importante de tener la capacidad de trabajar junto a personas, llegando a tener la interfaz necesaria para configurar los protocolos de seguridad mediante los que se rige el robot para cumplir con cada uno de los requisitos de seguridad de cada instalación.

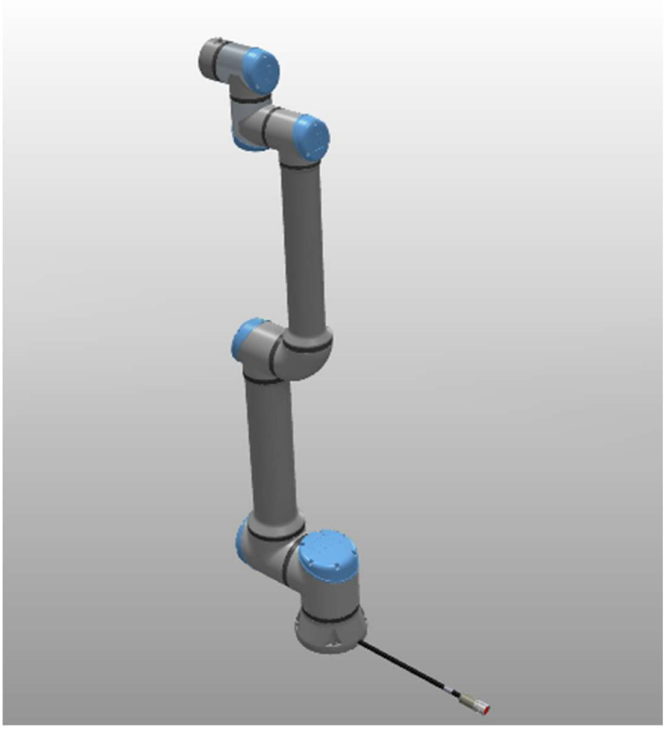


Ilustración 9. Robot UR10e

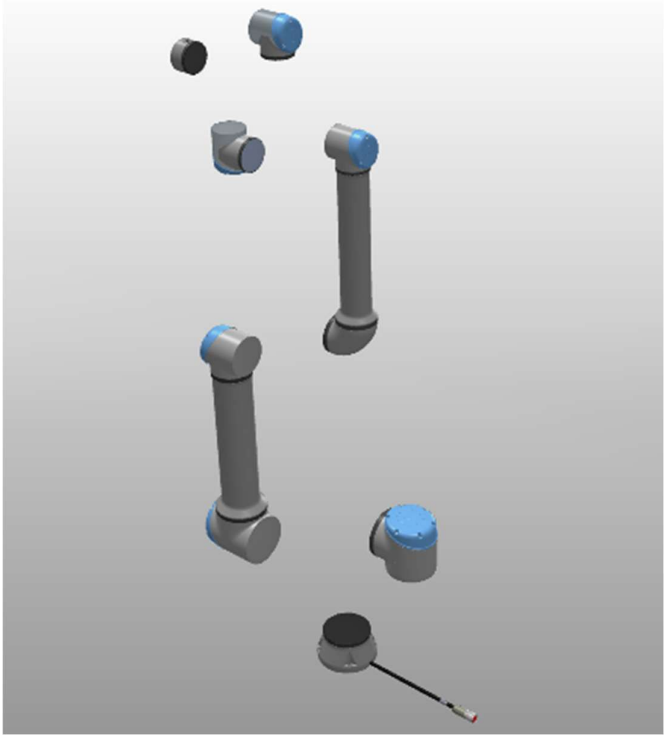


Ilustración 10. Robot UR10e segmentado por partes

El diseño del UR10e está orientado hacia una amplia variedad de tareas industriales, como lo son el paletizado, la soldadura o el control de calidad. Entre los casos de éxito aparece Raymath, un fabricante de metales, donde su implementación incrementó la producción [5].

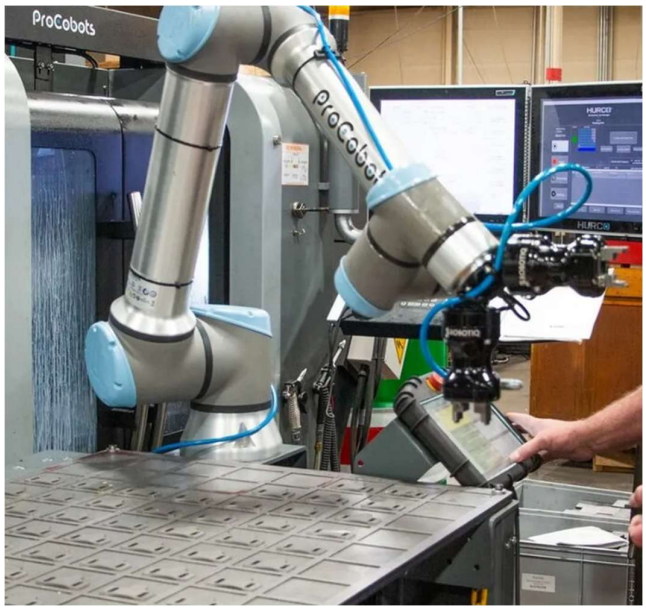


Ilustración 11. UR10e en Raymath

DeAngelo Marine Exhaust, fabricante de piezas de escape, casi había renunciado a automatizar soldaduras TIG manuales complejas hasta que la empresa se encontró con Universal Robots. Ahora ha podido aumentar diez veces la velocidad de soldadura, reducir los plazos de entrega y mejorar la calidad con su nuevo cobot [5].

Ilustración 12. Universal Robots en DeAngelo Marine Exhaust

2.3.3 ALTERNATIVAS

El robot UR10e podría haberse sustituido por muchos cobots del mercado que también cumplían las características necesarias. El mercado de los robots colaborativos está en auge y, por ello, en la actualidad, se ofrece una amplia gama de productos entre los que elegir. Además de Universal Robots, hay empresas como FANUC, ABB Robotics, Jaka y Kuka, las cuáles, tienen robots interesantes para esta aplicación, a continuación, se van a analizar algunos de ellos.

- **FANUC CRX-10iA**, es un cobot con una carga útil de 10 kg y un alcance de 1249 mm, adecuado para medianas y pequeñas empresas. Presenta un impresionante movimiento colocado en techo y una gran repetibilidad de hasta ±0,04mm. La serie CRX incorpora una nueva interfaz de programación intuitiva de FANUC que se puede usar mediante arrastrar y soltar en una tableta programadora portátil. El precio está alrededor de 40000€.

Ilustración 13. FANUC CRX-10iA en un proceso productivo

- **GoFa 12** de la serie GoFa CRB 15000, el robot colaborativo con mayor precisión de trayectoria del mercado, hasta 0,02 mm. Esto lo convierte en ideal para el atornillado. Pesa 48 kg y está diseñado para manipular cargas de hasta 14 kg. Cuenta con sensores de torque integrados en cada una de las articulaciones. Configuración fácil y programación con interfaz intuitiva. Con un alcance de 1370 mm y la posibilidad de colocarse en el techo. El precio ronda los 34000€, similar al resto, aunque más económico.

Ilustración 14. GoFa12 en una estación de despaletizado

- **JAKA Zu12**, de JAKA, una empresa dedicada al desarrollo de cobots, que comenzó en el año 2014 en Shanghái. Ofrece soluciones económicas y flexibles para todo tipo de industrias. Robot con un alcance de 1327 mm, una carga de 12kg, una repetibilidad de 0.03 mm y un peso de 41kg. Tiene un precio más económico que el resto, pero da menos seguridad trabajando en el techo, debido a su pérdida de calidad en los componentes empleados en su fabricación.

Ilustración 15. JAKA Zu12 en proceso productivo

Tabla 1. Comparación robots colaborativos

Robot	Alcance (mm)	Precisión (mm)	Peso (kg)	Techo	Precio (€)
UR10e	1300	±0,05	33,5	SI	38000
CRX10iA	1249	±0,04	40	SI	40000
GoFA12	1370	±0,02	48	SI	34000
Zu12	1327	±0,03	41	NO	24000

En la Tabla 1 se puede observar una comparativa de los 4 robots que se han estudiado. Llegando a la conclusión de que el robot definitivo para el proyecto sería el GoFA12 de ABB, sustituyendo al cobot UR10e con el que se realiza la investigación. Esta elección es debida a su precio, el más bajo de todos, su alta precisión y su largo alcance.

2.4 HERRAMIENTAS NECESARIAS

La función principal del robot es la de atornillado, por lo que la herramienta que no puede faltar en la mano del robot será un destornillador automático controlado por el robot.

Un tema interesante para tener en cuenta de las aplicaciones cobóticas es la normativa aplicada sobre las herramientas. Para que un robot sea tratado como robot colaborativo, y pueda trabajar como tal, necesita cumplir leyes que regulan el funcionamiento de las herramientas colocadas en él. Esto es lo que se denomina herramientas colaborativas y están reguladas bajo la norma ISO 10218-2 + ISO/TS 15066. Es importante tener en cuenta estas normativas antes de dar por terminado un proyecto colaborativo.

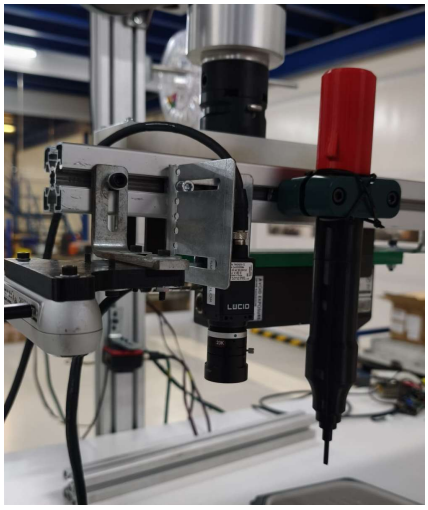


Ilustración 16. Herramienta de pruebas

En un caso típico el destornillador no sería una herramienta que cumpliría las normas por sí solo. El objetivo de este proyecto es realizar una investigación acerca de si acompañado de una cámara, también en el brazo del robot, se pueda detectar en todo momento al usuario, y restringir el movimiento, e incluso parar, si fuera necesario.

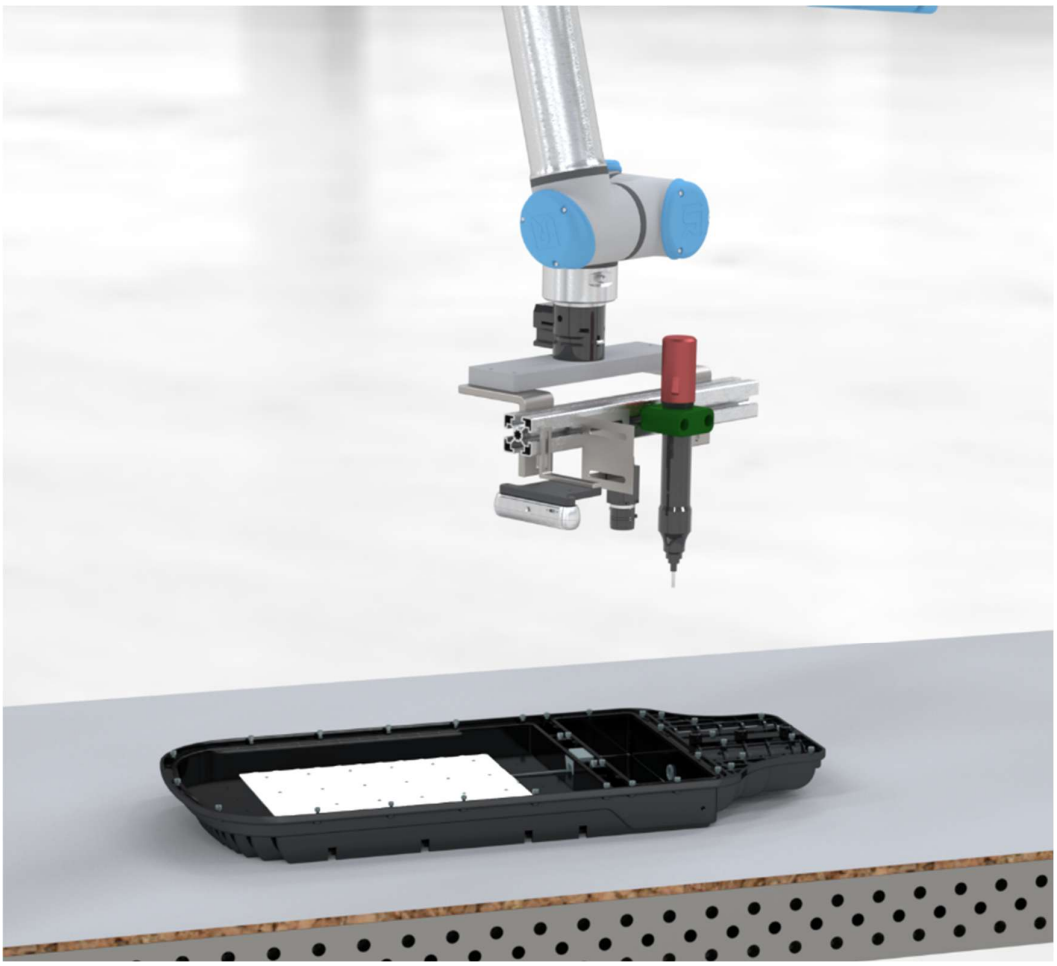


Ilustración 17. Herramienta actual renderizada

Otro de los temas a tratar es la cámara que utilizar, parte importante del proyecto sobre la cual se va a sustentar. Elegir una buena opción, aunque sea solo para realizar las pruebas, facilita mucho el trabajo.

Se han hecho pruebas de reconocimiento de tornillos con la cámara Mech-Eye NANO, una cámara 3D industrial compacta y de alta precisión, diseñada para aplicaciones robóticas. Con un rango de trabajo entre 300 y 600 mm y una resolución de 1,3 MP. Genera nubes de puntos detalladas mediante luz azul [7].

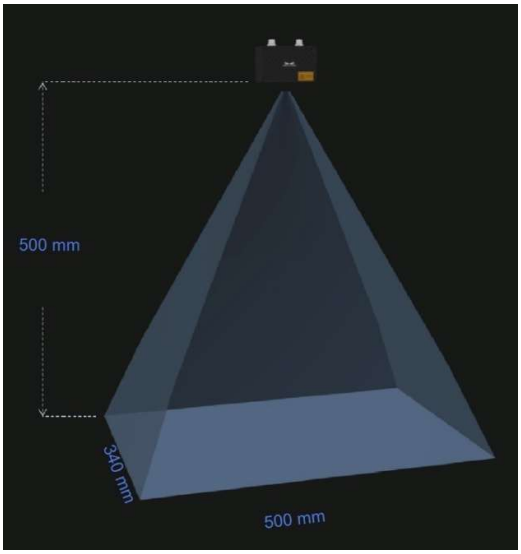


Ilustración 18. Rango Mech-Eye NANO

Esta cámara permite una calibración *eye in hand* sencilla y las imágenes que se obtienen son perfectas para detectar tornillos. Ha sido descartada debido a su elevado precio comparado con cualquier cámara 2D.

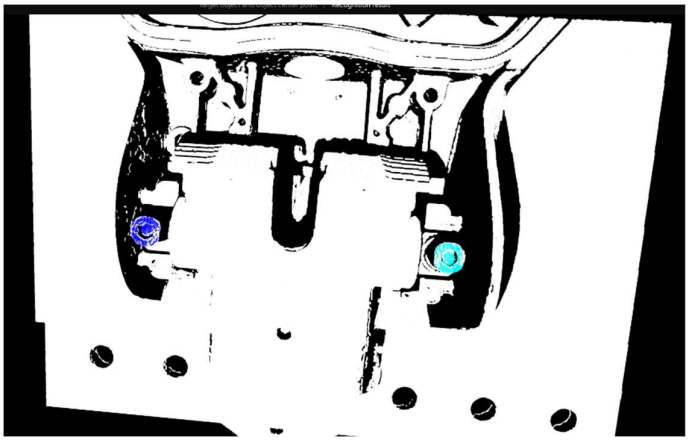


Ilustración 19. Captura ejemplo Mech-Eye NANO

La cámara elegida para el proyecto es una Lucid Triton 5 MP. Se trata de una cámara industrial 2D compacta, 29x29x45 mm, con protección IP67, ideal para ambientes exigentes como la industria. Es capaz de conectarse por código al ordenador mediante cable ethernet PoE, que nos permitirá pasar datos y alimentación por un mismo cable, un tema para tener en cuenta a la hora de la instalación, ya que, se facilita mucho. Cuenta con un sensor Sony IMX264 CMOS que captura imágenes de 2448 x 2048 píxeles, a unos 24 fps. Este sensor captura la imagen y el procesador interno de la cámara nos permite elegir la salida que nosotros vemos en distintos formatos de color. La óptica incorporada a la cámara es una Compact 8 mm ½.8 2/3”, cuyas características son adecuadas para la aplicación.

Ante la dificultad de programación y configuración de esta cámara, para realizar pruebas a lo largo del proyecto, se ha colocado también una cámara RealSense D455C, cámara sencilla de utilizar y perfecta para realizar todo tipo de experimentos con los distintos *softwares*.

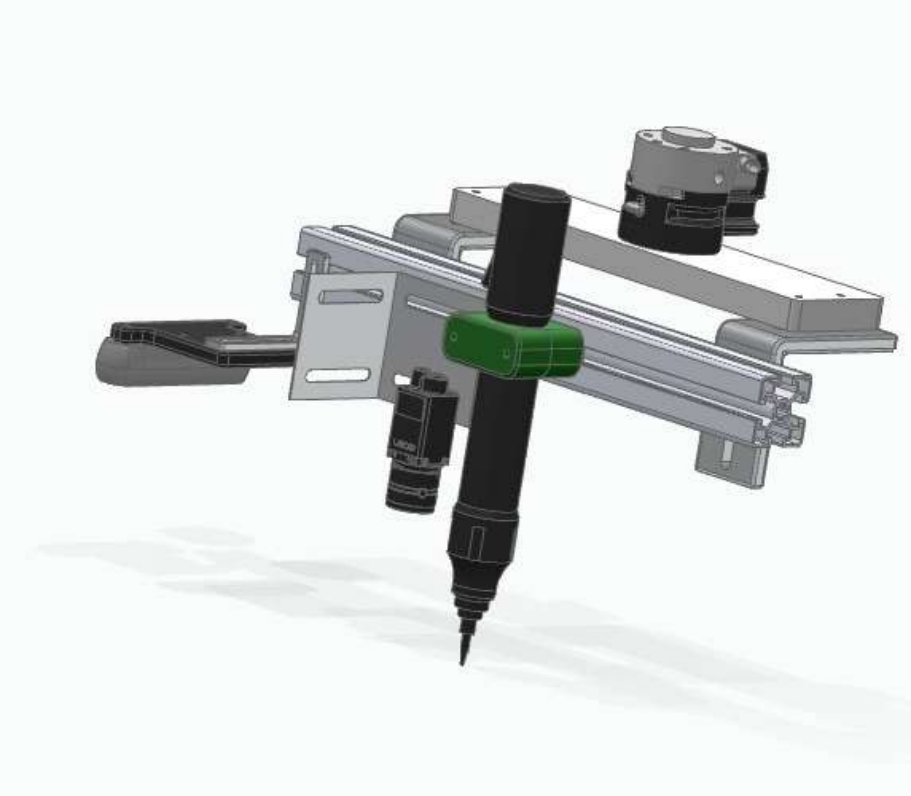


Ilustración 20. CAD de la herramienta actual de pruebas

A partir de la herramienta de pruebas, la cual se puede observar en la *Ilustración 13.* Diseñada con objetos de fácil acceso e intercambiables de manera rápida y sencilla, sin ningún tipo de diseño propio. Se ha diseñado la herramienta final en base al cumplimiento de la norma de herramientas colaborativas, que permitiría su adaptación a una versión final más cercana a la aceptación y certificación del proyecto.

La idea comienza por eliminar la cámara RealSense, ya que es una cámara utilizada solo para facilitar las pruebas, no una cámara definitiva. Por tanto, al eliminar una cámara el soporte Bosch pasa a tener una longitud menor, con el fin de aligerar el *tool*.

El soporte que une el cambio rápido con el resto de la herramienta será de Polietileno de alta densidad y sus dimensiones cambian para ajustarse a la aparición de una caja distribuidora Phoenix Contact SACB-4/ 8-L-10,0PUR SCOP.

Ilustración 21. Phoenix Contact 1452615

La caja distribuidora es indispensable en estos casos para mejorar la apariencia de la herramienta y facilitar su montaje. Necesitaremos cables cortos para conectar el destornillador automático, la cámara LUCID y el cambio rápido. Después se sacará un solo cable del *tool,* demostrando sencillez y buen aspecto, además de evitar posibles enganchones y nudos.

Las fundas en este tipo de herramientas son necesarias, debido a que se utilizan materiales resistentes para sujetar el peso total, y estos no son precisamente blandos si golpean al operario.

Estas fundas son de polietileno de baja densidad, creadas en impresoras 3D, y ocupan las zonas que más sobresalen, por lo que, si alguna de ellas choca con algo o alguien, la fuerza, el impacto y el daño causado, será mucho menor.

Por último, se ha rediseñado la placa de acero AISI 1045 estirado en frío que sujeta la cámara LUCID. Con un grosor de 3 mm se le ha aplicado una función para poder inclinarse, por si fuera necesario en la colocación de la cámara en el punto final del proyecto, para poder ver de mejor manera la punta del atornillador.

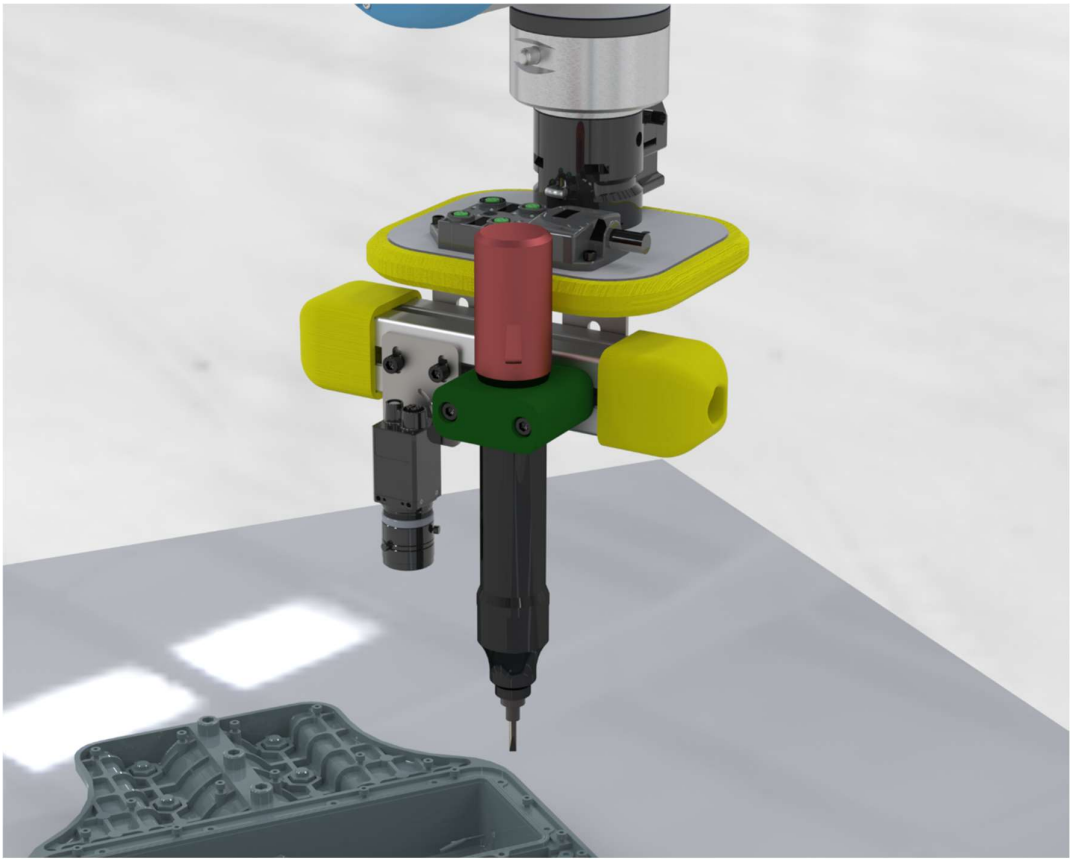


Ilustración 22. Herramienta colaborativa final CAD renderizado

Una vez se tiene la garra diseñada hay que obtener su *payload*. El *payload* es la carga útil que puede transportar o manipular en su extremo un robot. Para obtenerlo es necesario conocer el peso total de la herramienta, que tras un análisis en Solid Edge 2024, es de 3,457 kg. Una vez conocida la masa, se debe obtener el TCP o centro de masas, todos estos datos es posible obtenerlos mediante una buena aproximación de todos los pesos de cada una de las partes que conformarán el *tool* final. Como se puede ver en la *Ilustración 18,* el centro de masas está a -74 mm en X, 0,6 mm en Y, -106,05 mm desplazado en Z.

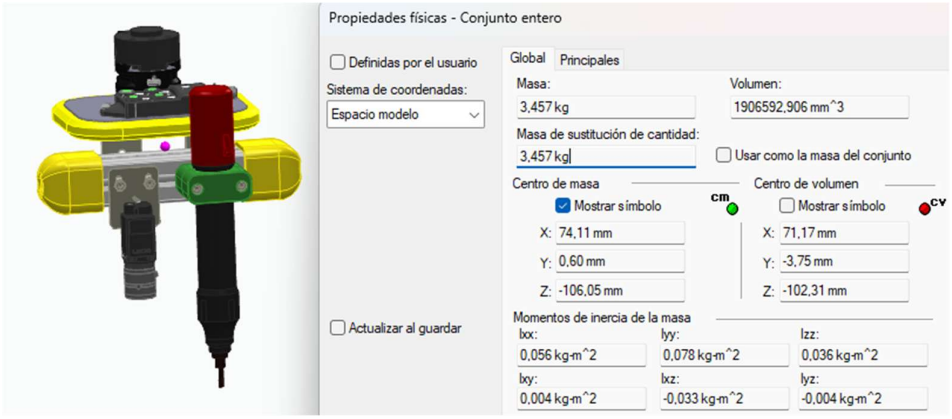


Ilustración 23. Centro de masas garra colaborativa

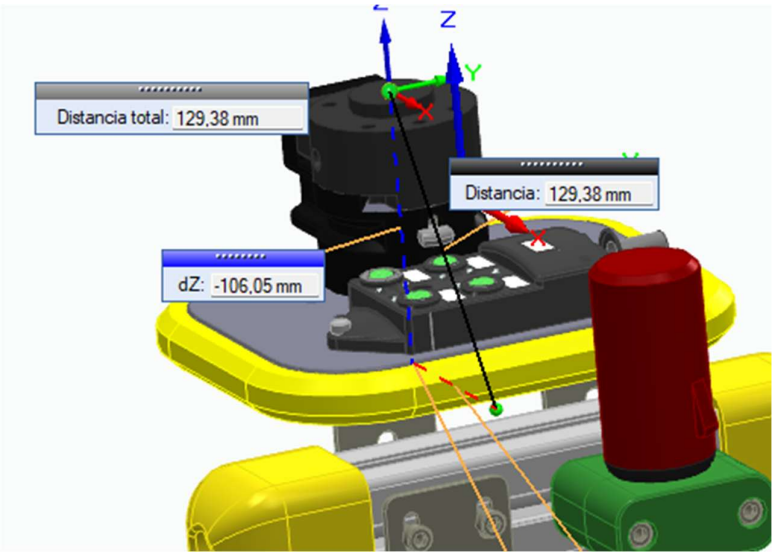


Ilustración 24. Offset del centro de masas

Los datos tanto del TCP como del peso se incluyen en esta gráfica propia del UR10e para validar esta herramienta, en la que se puede comprobar que la garra es adecuada para este robot.

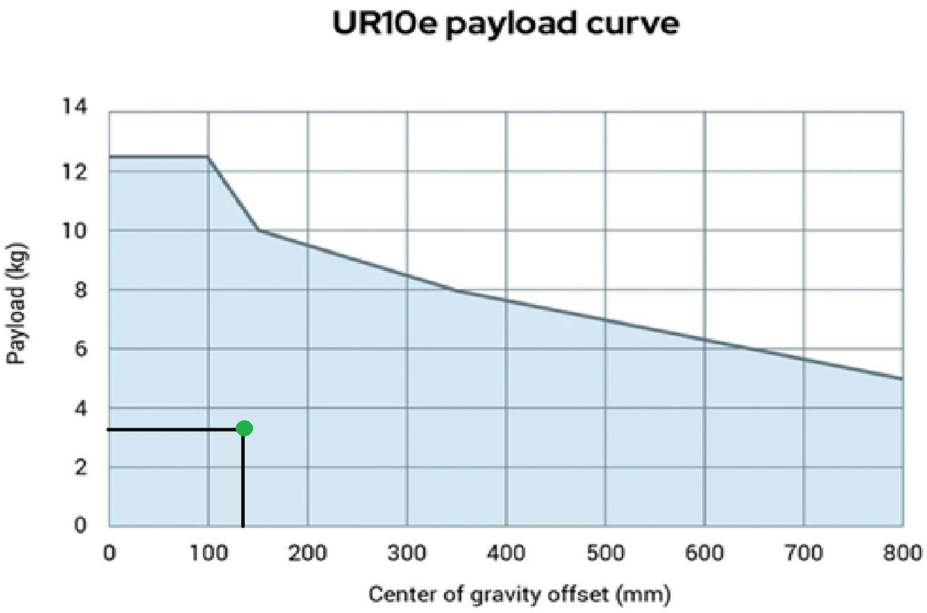


Ilustración 25. Validación del payload de la herramienta colaborativa

3. CALIBRACIÓN CÁMARA 2D

3.1 HALCON

MVTec es la compañía puntera en *software* de visión artificial, los productos que ofrece son utilizados en una amplia gama de sectores como el automovilístico, agricultura, metalurgia, alimentación, logística, medicina. Permiten aplicaciones como la inspección de superficies, el control de calidad óptico, el guiado de robots, la identificación, la medición, la clasificación, calibración, lectura de códigos de barras. El software de MVTec facilita nuevas soluciones para la automatización de la industria, proporciona tecnologías modernas como visión 3D, aprendizaje profundo y visión integrada. Con sede en Múnich (Alemania) y oficinas en Boston, Massachusetts (EE. UU.), Lyon (Francia), Utrecht (Países Bajos), Kunshan, cerca de Shanghái (China), Zhubei (Taiwán) y Bundang (Corea), así como una red consolidada de socios comerciales internacionales, MVTec está presente en más de 35 países de todo el mundo [8] [9].

Una de las opciones que nos proporciona MVTec es MERLIC, una alternativa con una interfaz sin código que permite a los usuarios crear soluciones rápidamente y de forma intuitiva [9].

MVTec HALCON es el software estándar integral para visión artificial con un entorno de desarrollo integrado (HDevelop) que se utiliza en todo el mundo. Permite ahorrar costes y mejorar el tiempo de comercialización. La arquitectura flexible de HALCON facilita el desarrollo rápido de cualquier tipo de aplicación de visión artificial [9].

Ofrece un rendimiento excepcional y una compatibilidad completa con plataformas multinúcleo, así como aceleración por GPU. Da servicio a todos los sectores, con una biblioteca utilizada en cientos de miles de instalaciones en todas las áreas de la imagen, como el análisis de blobs, la morfología, la comparación, la medición y la identificación. El software proporciona las últimas tecnologías de visión artificial de vanguardia, como la visión 3D completa y los algoritmos de aprendizaje profundo.

El *software* es compatible con una amplia gama de sistemas operativos y proporciona interfaces para cientos de cámaras industriales y capturadoras de imágenes. Por defecto, MVTec HALCON se ejecuta en plataformas de visión integradas en su propia interfaz, aunque también se puede portar a diversas plataformas de destino. Por lo tanto, este programa es ideal para su uso en sistemas integrados y personalizados.

3.2 CALIBRACIÓN

Es bien sabido que las operaciones con movimiento de robot programadas en línea, con el Teach Pendant, tienen una mayor repetibilidad y precisión que las guiadas por visión artificial [10]. Las operaciones aprendidas por guiado se mueven siempre por los mismos puntos, puntos guardados a los que previamente el robot ha sido llevado manualmente. Siempre podrá diferir en décimas de milímetros entre repeticiones de la pose nominal en el controlador del robot. Conseguir una buena precisión en programas guiados por visión se sustenta gracias a una precisa calibración de la cámara.

En este tipo de casos existen 2 calibraciones distintas, dependiendo del posicionamiento de la cámara en el entorno. Si la cámara está montada en la mano del robot, se ejecutará una calibración *eye in hand,* esta es apropiada cuando se necesita precisión e imágenes cercanas. La otra opción es que la cámara esté fija en el espacio, entonces se trata de un caso de *eye to hand,* útil cuándo se necesita una imagen global y contextual. La calibración que se va a tratar en este caso es la primera ya que la cámara está embarcada en el robot.

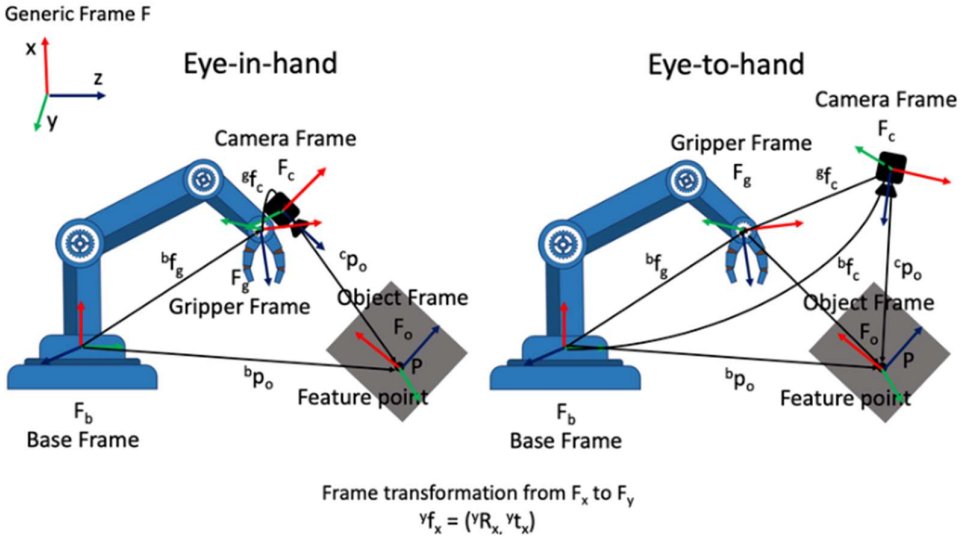


Ilustración 26. Diferencias entre eye in hand/eye to hand

La calibración mano-ojo es un campo clave para la robótica por visión artificial. Esta tecnología permite al robot realizar tareas como agarrar, posicionar y manipular objetos con mayor precisión. En la investigación de la calibración mano-ojo, los casos con cámara monocular se han convertido en un tema de estudio destacado debido a su bajo costo, facilidad de instalación y usabilidad. La velocidad, la estabilidad y la precisión del algoritmo de calibración determinan en gran medida la eficiencia del brazo robótico.

Los sistemas para la calibración mano-ojo de cámaras monoculares tienen una larga historia que se remonta a la década de 1980, cuando Tsai y Lenz construyeron un sistema de calibración mano-ojo completamente automático y eficiente. Fueron los primeros en implementarlos y sigue siendo la base de lo que es ahora. El procedimiento no solo era muy preciso, sino que también imponía muy pocos requisitos al robot. Sin embargo, dado que requería no una, sino dos cámaras, además de brazos robóticos, no era muy práctico [10].

Los investigadores han reconocido varios métodos para la calibración mano-ojo que hacen uso de una sola cámara, capaz de esto, debido al avance de las tecnologías de visión. Tomasi y Kanade presentaron su idea para un enfoque de calibración *eye in hand*, evaluar la relación posicional relativa entre la mano del robot y la cámara, se utilizaron dos placas de calibración y aproximaron la posición y actitud del objeto en relación con el sistema de coordenadas principal del robot. El sistema no era demasiado complicado de implementar, pero se necesitaban unas estrictas placas de calibración [10].

En la actualidad, la tecnología de calibración mano-ojo de cámaras monoculares también está en constante innovación. En los últimos años, los investigadores han presentado una gran cantidad de nuevos métodos para la calibración mano-ojo de cámaras monoculares. Estas técnicas incluyen técnicas basadas aprendizaje profundo, en el que entran en juego las redes convolucionales. El *deep learning* se ha convertido en un área de investigación cada vez más popular en los últimos años. Esta tecnología logra la calibración mano-ojo mediante el uso de redes neuronales profundas y no requiere placas de calibración ni escenarios especializados. Como resultado, es posible lograr una calibración mano-ojo de alta precisión en una variedad de entornos y ubicaciones.

El objetivo de calibrar una cámara es obtener la relación de transformación de la cámara respecto de la base del robot, pasando previamente por un objeto de referencia, el cuál será una placa o *grid* como el de la imagen.

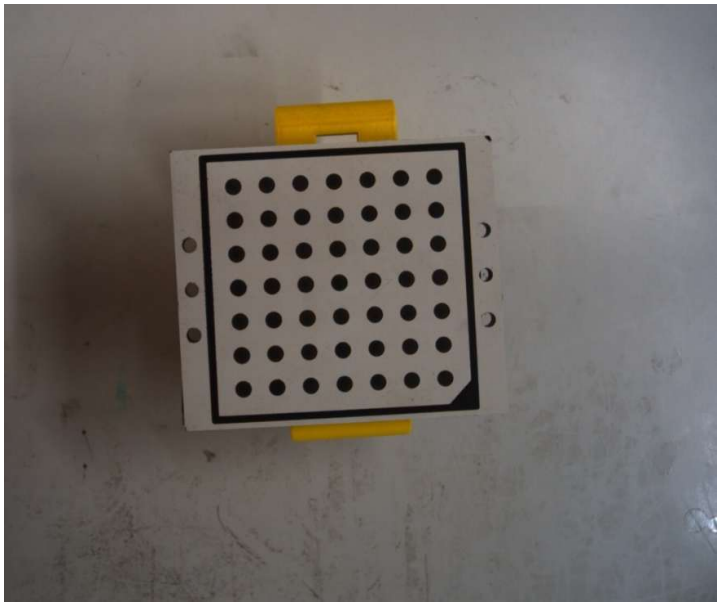


Ilustración 27. Placa de calibración

La estrategia de calibración cambia dependiendo de, qué tipo de cámara tenemos, cuantas tenemos y dónde están posicionados la cámara y el objeto referencia respecto al robot. Hay muchas combinaciones diferentes, por lo que, saber cuál es el caso por tratar, y abordarlo de manera correcta, es lo primero que se debe estudiar. Todas las calibraciones se dividen en dos pasos, una calibración intrínseca y una calibración extrínseca.

MVTec HALCON ofrece todo tipo de soluciones para calibrar cámaras 2D, 3D y estéreo, además de la posibilidad de crear códigos para calibrar cámaras tanto estáticas como en movimiento. Ofrece una amplia oferta de ejemplos que aparecen en el programa, que se puede tomar de referencia a la hora de crear un código para su situación particular [11].

3.2.1 CALIBRACIÓN INTRÍNSECA

La calibración intrínseca consiste en obtener como la cámara forma la imagen a partir de la luz que entra por el objetivo, nos permitirá poder pasar de píxeles a coordenadas reales, todo ello respecto la cámara.

Esta calibración ha sido posible gracias a una de las opciones que posee MVTec HALCON, el asistente de calibración.

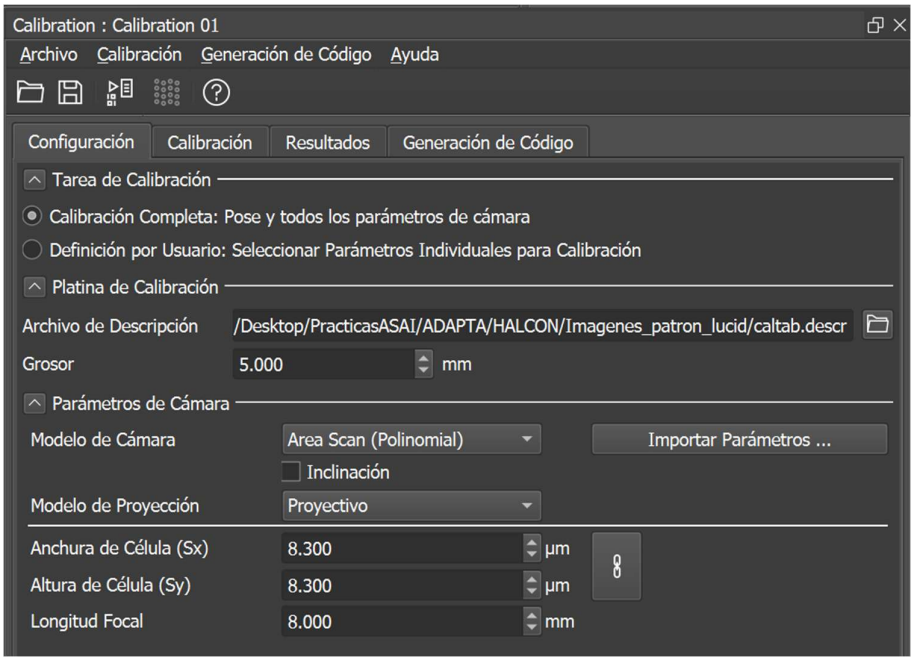


Ilustración 28. Asistente de calibración de HALCON parámetros iniciales

Esta ayuda del programa es excelente para la calibración, se puede introducir, en una interfaz sencilla, un archivo de descripción de la placa a detectar, indicar el grosor, el modelo de cámara, y unos parámetros iniciales desde los que se parte.

Como modelo de cámara se refiere al modelo matemático, que describirá la relación entre el espacio real y la imagen. Hay 4 opciones*, area scan division*, *area scan polynomial*, *line scan division* y *line scan polynomial*. Las únicas dos opciones válidas para la cámara Lucid son las dos primeras, debido a que no se está trabajando con un perfilómetro [9].

El modelo de división radial (*area scan division*) es ideal cuando se utiliza para una aplicación sencilla, tiene una velocidad de cálculo alta y no se consigue gran precisión. Para mejorar la precisión y reducir el error es recomendable utilizar el modelo polinomial, un modelo matemático más complejo que, por tanto, ofrece mayor precisión en los resultados. A continuación, se debe especificar el modelo de proyección, proyectivo o telecéntrico. En este caso se trata de una lente proyectiva.

Una vez se inicializa el asistente, se introducen imágenes tomadas con la placa estática y el robot cambiando de posición y orientación, con buen enfoque e iluminación para conseguir una correcta calibración. Además, es importante que el *grid* aparezca en todo el campo de visión de la cámara al final de la calibración.

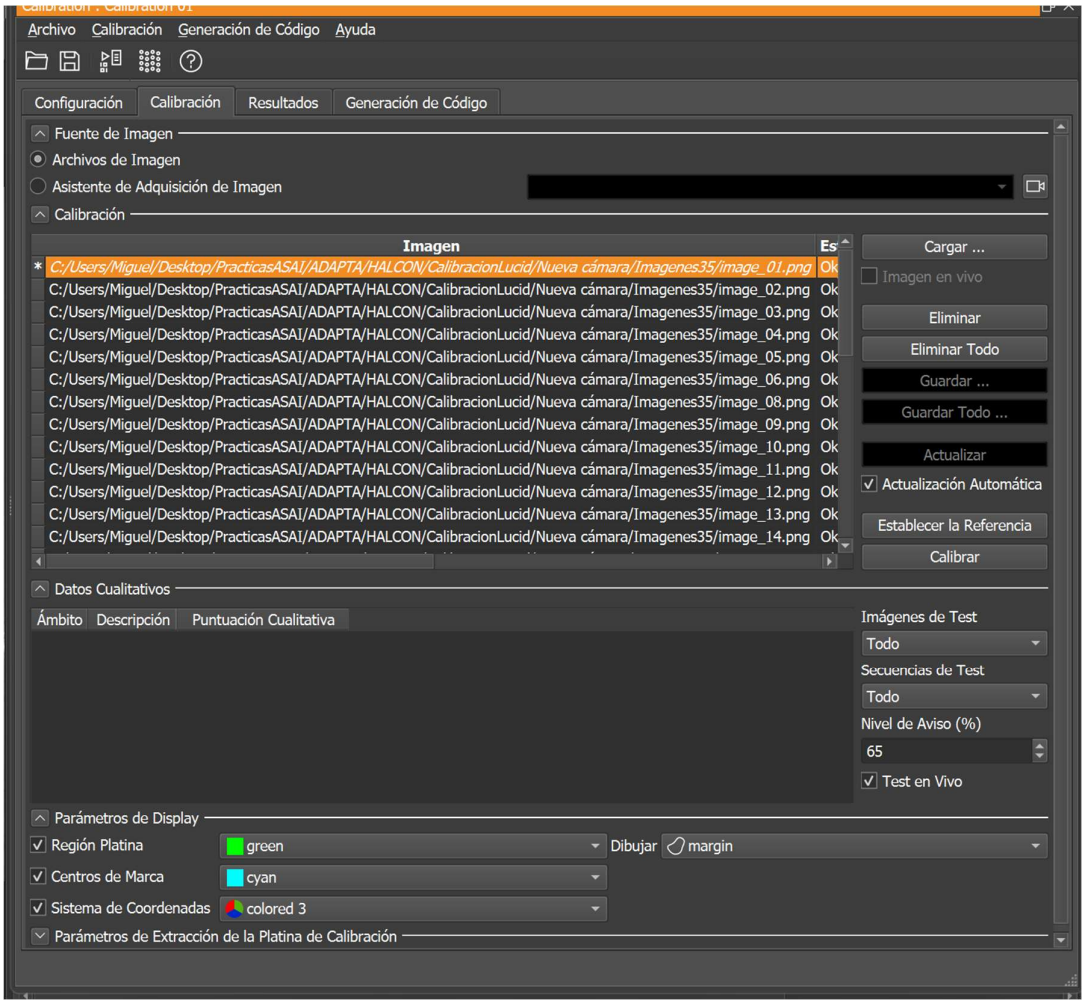


Ilustración 29. Asistente de calibración detección placa

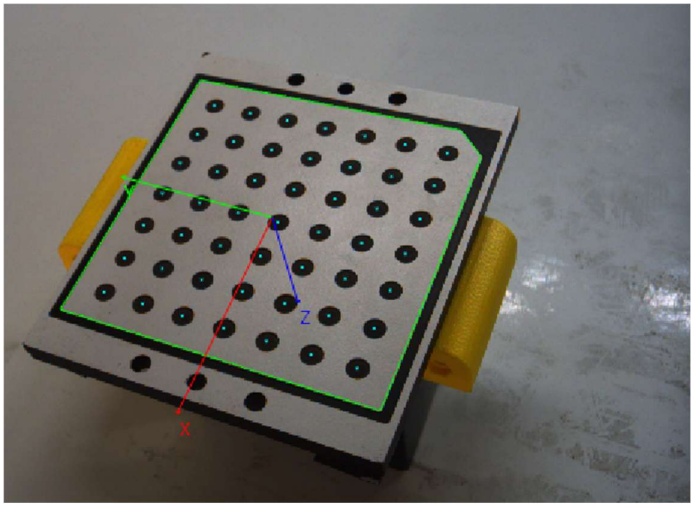


Ilustración 30. Placa detectada

Una vez detectada la placa en todas las imágenes, se establece una imagen de referencia (por defecto aparece la primera) y se pulsa calibrar. Los resultados que se obtienen indican, el estado de la calibración, parámetros finales de cámara y la pose de la placa respecto de la lente. Estos dos últimos se utilizarán en la calibración extrínseca.

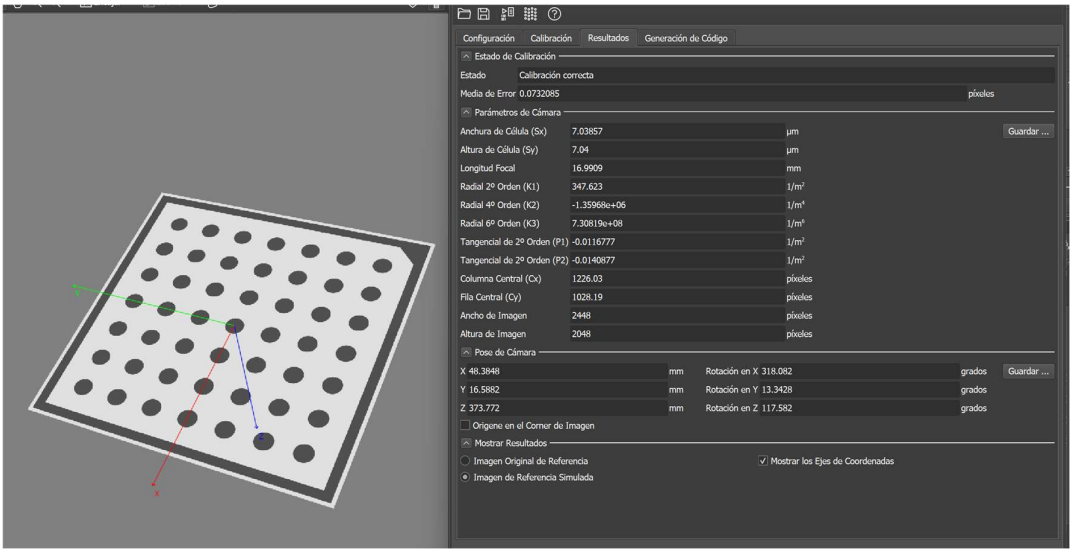


Ilustración 31. Resultados calibración intrínseca

3.2.2 CALIBRACIÓN EXTRÍNSECA

El objetivo de la calibración extrínseca es el de calcular la transformación geométrica entre el sistema de coordenadas de la cámara, y el sistema de coordenadas del robot. La relación de movimiento entre la cámara y el robot es la idea fundamental que subyace al procedimiento de calibración *eye in hand* de la cámara monocular [10].

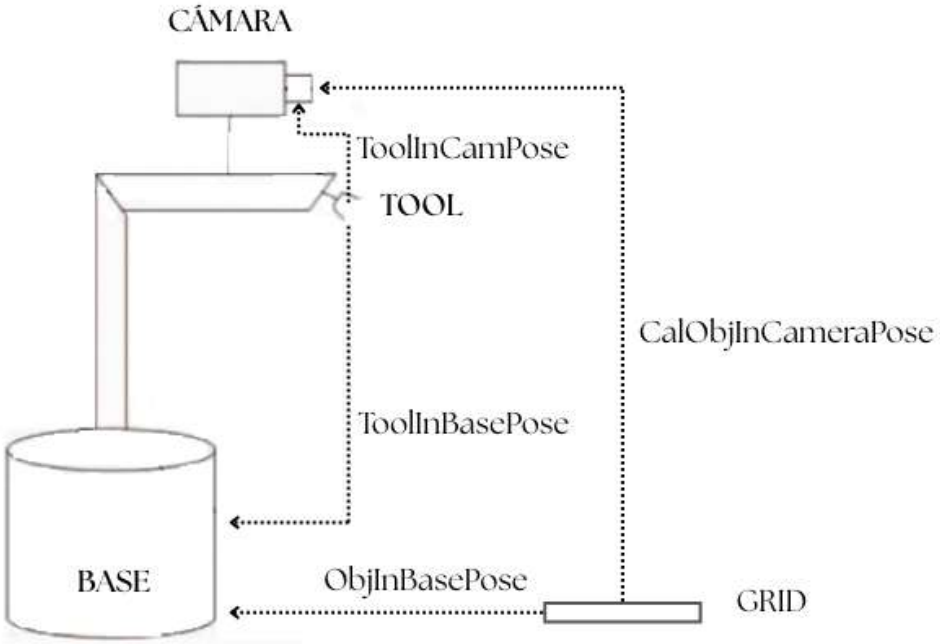


Ilustración 32. Representación calibración extrínseca

La preparación antes de calibrar comienza cargando las posturas del robot en las que se han tomado las fotos utilizadas en la calibración intrínseca. Se han utilizado un total de 35 imágenes, número suficiente para lograr una calibración precisa. Se crea un procedimiento propio para leer las posiciones directamente del programa extraído del robot para que no se pierda ningún dato y las coordenadas sean las correctas.

```
read_tool_in_base_pose
('C:/Users/migue/Desktop/PracticasASAI/ADAPTA/HALCON/CalibracionLucid/Nue
va cámara/polynomial/Poses.dat', 'Rp+T', 'rodriguez', 'point',
DictHandle)
```

Esto permite automatizar el paso de las coordenadas del robot a HALCON y asegurarnos que las coordenadas son las correctas.

Cabe destacar en el procedimiento de leer las poses, la importancia de conocer el robot con el que se trabaja. Universal Robot trabaja con una representación de coordenadas diferente a la de la mayoría de robot. Es el llamado modelo vector ángulo-eje (*Axis-Angle Representation*), o como se denomina en HALCON, vector rotación Rodrigues, donde la traslación es similar a los modelos angulares de Euler, los modelos más comunes y sencillos de visualizar para el ser humano, pero cambia la manera de representar la rotación [12].

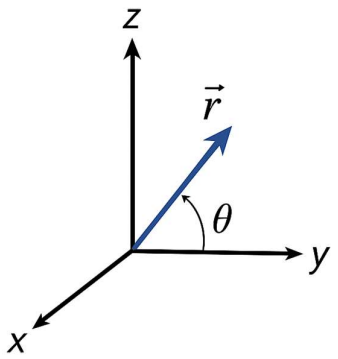


Ilustración 33. Vector rotación Rodríguez giro en eje X

$$R1(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\,\theta & -sen\theta \\ 0 & sen\theta & cos\,\theta \end{bmatrix} \qquad \text{Ec(1)}$$

$$R2(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \qquad \text{Ec(2)}$$

$$R3(\gamma) = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \text{Ec(3)}$$

El siguiente paso es detectar la placa en todas las imágenes ofrecidas por el usuario, igual que en la calibración intrínseca. En este paso se obtienen los mismos datos que en la intrínseca, pero con los nuevos parámetros de cámara y se van introduciendo en un conjunto de datos llamado CalibDataID, que contendrá todos los resultados necesarios.

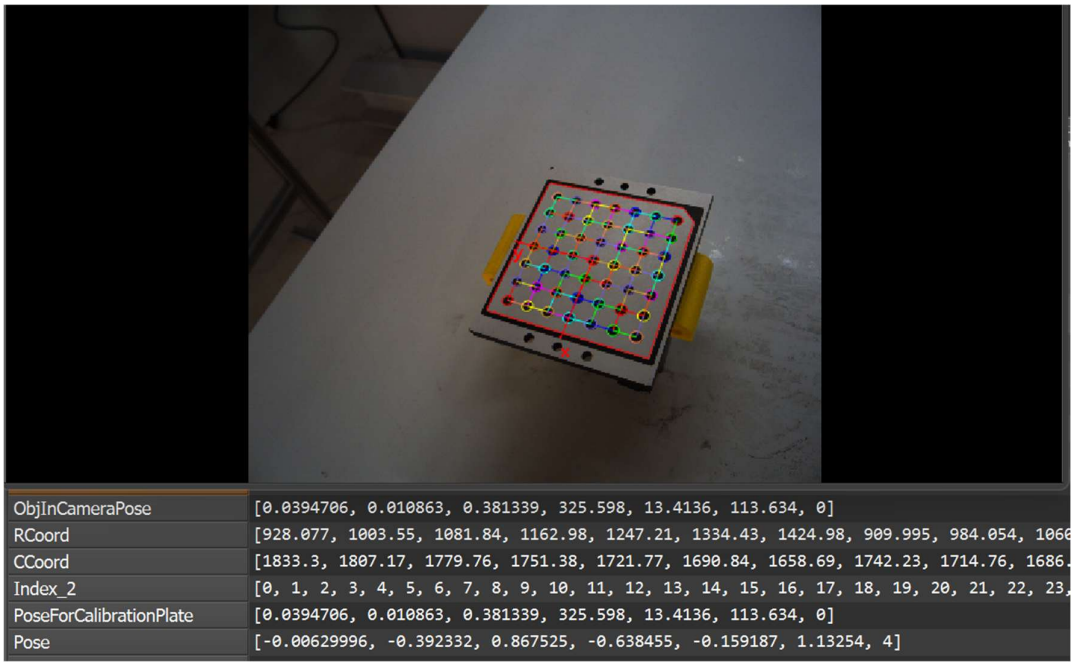


Ilustración 34. Detección placa en calibración extrínseca

Una vez detectado el *grid* en todas las imágenes, se introduce el conjunto de datos CalibDataID en el siguiente operador para determinar la posición 3D de la herramienta del robot respecto a la lente. Esto sirve para tener la cámara como un *tool* más del robot.

```
check_hand_eye_calibration_input_poses (CalibDataID, 0.5, 0.5, Warnings1)
```

Para dar por finalizada la calibración, se representa en un entorno 3D los objetos interesantes en cada una de las imágenes. De esta manera se puede comprobar si falla alguna foto, eliminar las que se consideren, y repetir la calibración. Los resultados del CalibDataID, serán los utilizados posteriormente en el guiado de trayectorias, una vez reconocida la posición de los tornillos en la imagen.

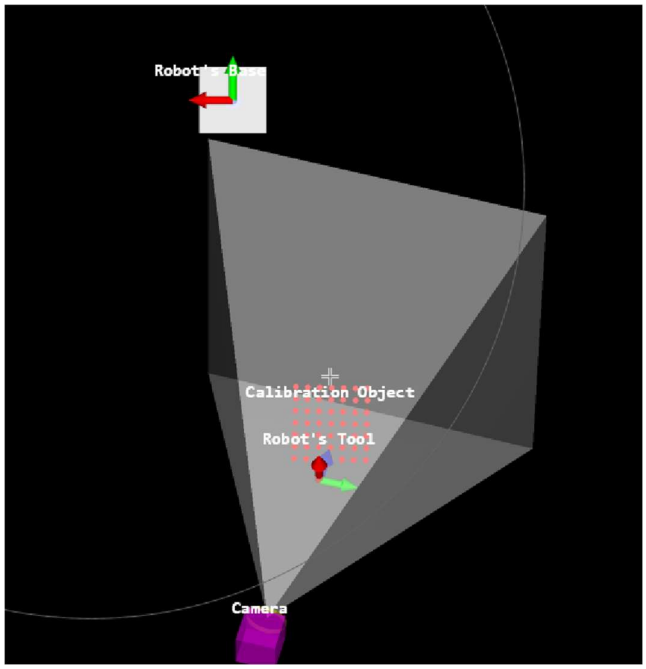


Ilustración 35. Representación pose calibración 1

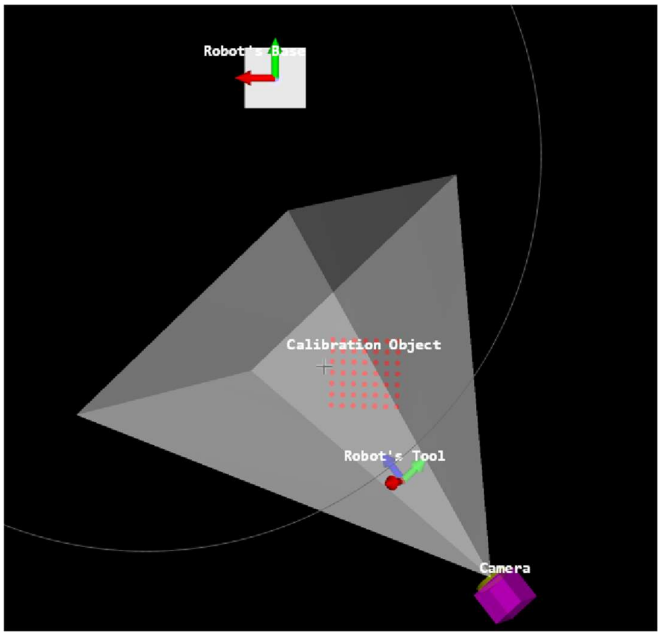


Ilustración 36. Representación pose calibración 2

4. RECONOCIMIENTO POR VISIÓN ARTIFICIAL

4.1 VISIÓN ARTIFICIAL

La visión artificial es un componente clave en la creación de sistemas inteligentes en el marco de la Industria 4.0. Este avance de la tecnología permite que los sistemas automatizados sean capaces de visualizar un entorno, identificar y clasificar lo que aparece en él, imitando e incluso excediendo en ciertos casos la habilidad de percepción humana. Este tipo de tecnología integra técnicas avanzadas de procesamiento de imágenes con inteligencia artificial [13].

En el campo de la robótica colaborativa, la visión artificial es fundamental al proporcionar a los robots la habilidad para interpretar su entorno cambiante y responder adecuadamente. Gracias a esta tecnología, los cobots pueden llevar a cabo tareas como el reconocimiento de componentes, la identificación de instrumentales, la localización exacta de objetos o la interpretación de gestos humanos, todo ello en tiempo real y en entornos compartidos con operadores.

En este estudio, la visión artificial se utiliza como sistema de guía principal en un UR10e para una tarea de atornillado la cual depende de ella para actuar como aplicación colaborativa y trabajar junto a los humanos.

El sistema de visión ha sido concebido para ser flexible frente a variaciones comunes en entornos industriales, como alteraciones de iluminación, oclusiones parciales o discrepancias en la colocación del operario. Además de tener la capacidad de detectar las manos de los operarios en todo momento, seguir su trayectoria y parar si fuera necesario el robot y el destornillador en un momento de peligro.

Las métricas clave que suelen utilizarse en este tipo de sistemas son la Precisión Media Promedio (mAP), la Intersección sobre la Unión (IoU) y la puntuación F1. Estas métricas ayudan a evaluar la exactitud y la precisión de las tareas de detección de objetos.

Al evaluar el rendimiento de los algoritmos de segmentación de anomalías a nivel de píxel, el resultado de la clasificación de cada píxel se considera igualmente importante. Un píxel puede clasificarse como verdadero positivo (VP), falso positivo (FP), verdadero negativo (VN) o falso negativo (FN). Sumando todos ellos y con la siguiente ecuación se obtiene la Intersección sobre la Unión [14] [15].

$$IoU = \frac{\sum VP}{\sum VP + \sum FP + \sum FN} \qquad \text{Ec(4)}$$

El valor de la Precisión media Promedio depende del rango de IoU que se marque. Ofrece el porcentaje de objetos detectados con un rango mayor que el IoU indicado.

El F1 es una métrica que combina precisión y recuperación(recall), que se utiliza para ver el rendimiento del modelo. Es útil para encontrar el mejor intervalo de confianza al momento de validar una predicción.

$$Precision = \frac{\sum VP}{\sum VP + \sum FP} \qquad \text{Ec(5)}$$

La precisión es la proporción de predicciones que fueron positivas.

$$Recall = \frac{\sum VP}{\sum VP + \sum FN} \qquad \text{Ec(6)}$$

La recuperación es la proporción de objetos que fueron detectados.

$$F1 = 2\,\frac{Precision \times Recall}{Precision + Reca} \qquad \text{Ec(7)}$$

4.2 YOLO

4.2.1 ROBOFLOW

Roboflow es una plataforma de visión artificial diseñada para desarrolladores y empresas. Proporciona las herramientas necesarias para seleccionar el tipo de detección que más se adapta a cada proyecto, etiquetar e incluso entrenar los modelos. Roboflow es compatible con muchos de los modelos más importantes del sector como son COCO, YOLO, Tensorflow o KERAS, además de intuitivo y fácil de utilizar. En este trabajo se utilizará únicamente para etiquetar las imágenes y obtener un conjunto de datos, que posteriormente se entrenará con un modelo de YOLO, fuera de este software.

Roboflow ofrece la posibilidad de elegir entre distintos modelos de detección como son:

- *Object Detection.* Encuentra la localización de los objetos en la imagen.
- *Classification.* Asigna una o varias etiquetas a una imagen.
- *Instance Segmentation.* Encuentra la localización del objeto a nivel de píxel.
- *Key point Detection.* Busca la localización de los objetos y sus puntos clave en la imagen, suele ser utilizada para obtener la orientación de los objetos.
- *Semantic Segmentation.* Busca el objeto a nivel de píxel y crea datos únicos de ese objeto [16].

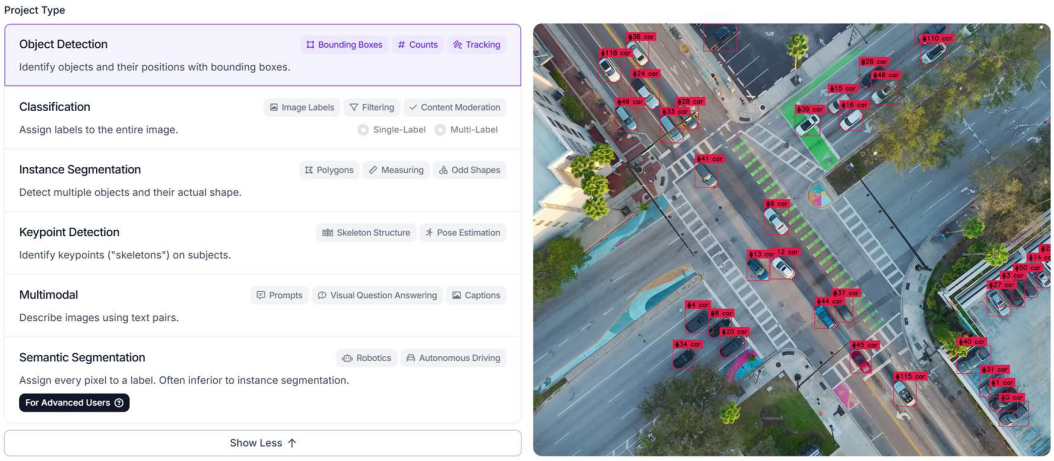


Ilustración 37. Diferentes tipos de modelos Roboflow

Una vez seleccionada la opción más favorable, el siguiente paso consiste en subir las fotos y etiquetarlas. Roboflow tiene distintas opciones de etiquetas dependiendo del tipo de modelo elegido, desde remarcar con una caja el elemento requerido hasta segmentar a la perfección un objeto.

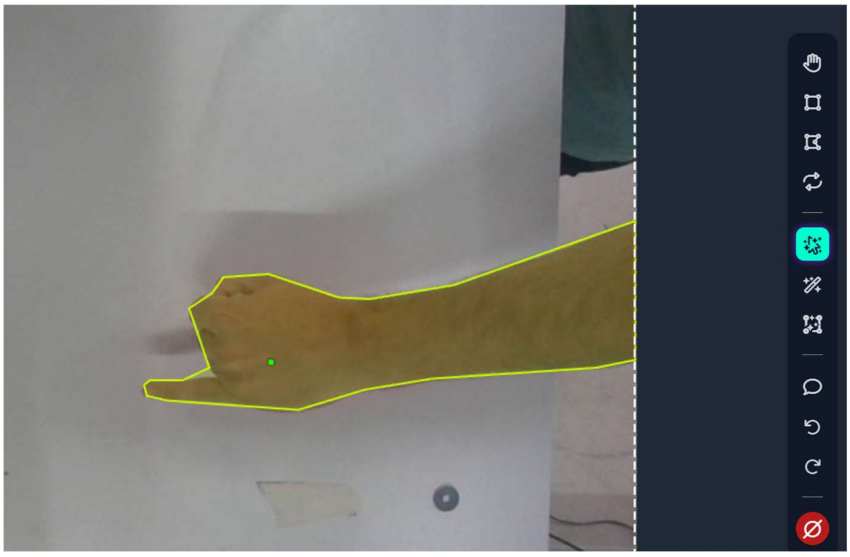


Ilustración 38. Ejemplo de segmentación automática

Por último, una vez etiquetadas todas las imágenes con sus respectivos nombres, se obtiene el dataset que posteriormente se entrenará. A la hora de su extracción, se puede elegir entre distintos modelos de archivos entrenados previamente, en este caso únicamente se utiliza YOLOv8 y YOLO11.

4.2.2 INTRODUCCIÓN A YOLO

You Only Look Once, más conocido como YOLO, es un algoritmo de detección de objetos en tiempo real basado en redes convolucionales. El rendimiento en la detección de objetos se mide tanto por precisión como por velocidad de detección. A diferencia de modelos pasados, que analizaban las imágenes dividiéndolas por partes, YOLO procesa toda la imagen en una sola pasada, dividiéndola por celdas y prediciendo simultáneamente, las coordenadas de las cajas delimitadoras, el nombre del objeto y la confianza de detección [17] [18]. Esto nos ofrece una mayor velocidad que los demás modelos, pero con una muy buena precisión. Todo esto hace que este modelo de detección de objetos se adapte fácilmente a los requerimientos de muchas aplicaciones.

Los algoritmos con mayor avance en los últimos años son los basados en redes convolucionales profundas, CNN. Estos pasos hacia adelante dejan atrás modelos de detección de objetos como Viola Jones, HOG, DPM. Técnicas que se quedan atrás, no tanto por su precisión, sino por su velocidad de detección [19] [20] [21].

La herramienta Viola Jones fue creada en 2001 y sigue siendo en la actualidad uno de los algoritmos más eficientes para el reconocimiento de rostros frontales en imágenes y vídeos. Muchos de las operaciones que se realizan en él son las bases de los actuales modelos. La técnica HOG o *Histogram of Oriented Gradients* es una propuesta con la funcionalidad de detectar personas en imágenes, su uso ha expandiéndose hacia la detección de objetos gracias a su sencillez. También ha sido utilizado para la detección y clasificación de programas malignos en Android. El modelo DPM o *Deformable Part Models* es un algoritmo reconocido cuya función es mejorar la detección de objetos deformables. Su idea clave es diferenciar partes que pueden desplazarse desde un punto central [19] [20] [21].

YOLO ha ofrecido varias versiones hasta el día de hoy, comenzando en el año 2016 con YOLOv1, hasta la versión de YOLOv13, lanzada en junio de este mismo año. Además de Yolo, Ultralytics, la empresa encargada del *software* tiene diferentes versiones dependiendo de la aplicación que le conviene al cliente.

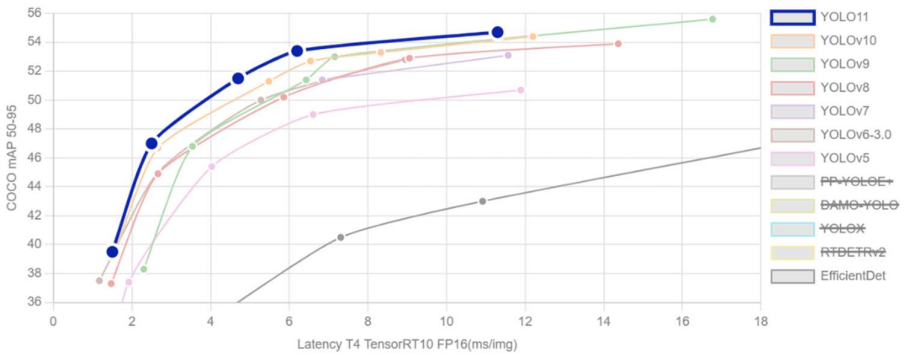


Ilustración 39. Tabla comparación versiones Yolo

Una de las alternativas que nos ofrece Ultralytics es SAM, modelo de segmentación de imagen a la vanguardia, que permite trabajar según las indicaciones del consumidor. Capaz de adaptarse a nuevas situaciones sin ningún conocimiento previo. Únicamente es capaza de realizar la inferencia de resultados, no podemos hacer una validación y un entrenamiento como sí que se puede con YOLO [18]. Tiene la posibilidad de implementarse en los teléfonos móviles y aunque no tenga mucha utilidad hasta el momento, este mercado avanza de vertiginosamente, por lo que pronto le veremos su funcionalidad.

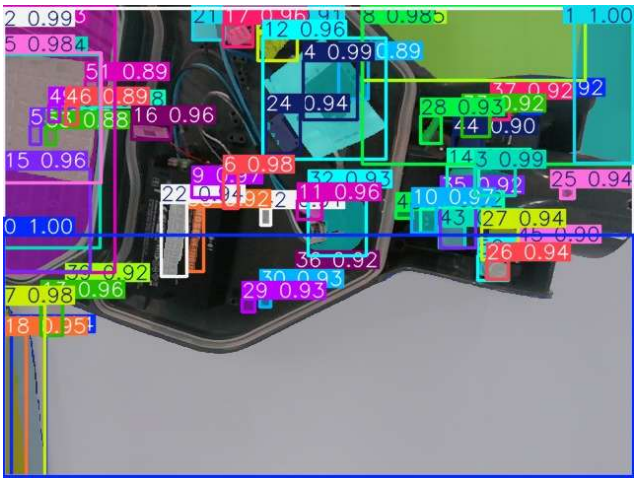


Ilustración 40. Ejemplo de segmentación SAM

Un ejemplo del tiempo de procesamiento de esta sola imagen con el modelo de SAM es de un minuto y 40 segundos aproximadamente. Un dato mayor a lo que obtenemos con YOLO o MVTec Deep Learning.

```
Speed: 9.2ms preprocess, 100026.1ms inference, 29.4ms postprocess per
image at shape (1, 3, 1024, 1024)
```

Tabla 2. Tabla comparación modelos Ultralytics

Comparación FastSAM vs YOLO

Aquí comparamos los modelos SAM 2 de Meta, incluida la variante más pequeña SAM2, con el modelo de segmentación más pequeño de Ultralytics , YOLO11n-seg:

Modelo	Tamaño (MB)	Parámetros (M)	Velocidad (CPU) (ms/im)
Meta SAM-b	375	93.7	49401
Meta SAM2-b	162	80.8	31901
Meta SAM2-t	78.1	38.9	25997
MobileSAM	40.7	10.1	25381
FastSAM conred troncal YOLOv8	23.7	11.8	55.9
Ultralytics YOLOv8n-seg	**6,7** (11,7 veces menor)	**3,4** (11,4 veces menos)	**24,5** (1061 veces más rápido)
Ultralytics YOLO11n-seg	**5,9** (13,2x menor)	**2,9** (13,4 veces menos)	**30,1** (864 veces más rápido)

Se puede observar en la *Tabla 2,* la comparativa de parámetros entre YOLO y SAM. La función que va a cumplir en el proyecto es la de detectar las manos en tiempo real, por tanto, necesitamos precisión y seguridad. Esto va a provocar que la herramienta elegida sea YOLO ya que sus modelos son 1000 veces más rápidos que los de SAM.

Los modelos de detección de objetos pasan por distintas fases antes de iniciar en funcionamiento. El primero de esos pasos es el de etiquetar e indicar al conjunto de datos qué zona de las imágenes es interesante y queremos señalar y cuál no. Este paso se realiza con la herramienta de Roboflow, explicado en el apartado anterior. Cuando todas las imágenes están etiquetadas debemos seleccionar entre distintas opciones para mejorar nuestro modelo.

Tenemos la opción de ajustar a nuestro gusto el porcentaje de imágenes que irán destinadas a entrenamiento, evaluación y prueba. Cada una de estas fases ejerce una parte importante del entrenamiento, es importante entenderlas y ajustarlas a las necesidades del proyecto.

A la hora de seleccionar un porcentaje óptimo de entrenamiento se suele utilizar el 70/20/10. Esto es, debe haber suficientes imágenes en entrenamiento, validación y test. Pero estos números pueden cambiar para cada dataset, dependen de su naturaleza, tamaño de las imágenes, tamaño de la muestra, imágenes capturadas en vídeo.

Un modelo de visión artificial se entrena ajustando sus parámetros internos para minimizar los errores. Inicialmente, el modelo se alimenta con un gran conjunto de imágenes etiquetadas. Hace predicciones sobre lo que hay en estas imágenes, y las predicciones se comparan con las etiquetas o el contenido real para calcular los errores. Estos errores muestran la discrepancia entre las predicciones del modelo y los valores verdaderos.

Durante el entrenamiento, el modelo realiza predicciones iterativamente, calcula errores y actualiza sus parámetros a través de un proceso llamado retro propagación. En este proceso, el modelo ajusta sus parámetros internos (pesos y sesgos) para reducir los errores. Al repetir este ciclo muchas veces, el modelo mejora gradualmente su precisión. Con el tiempo, aprende a reconocer patrones complejos como formas, colores y texturas.

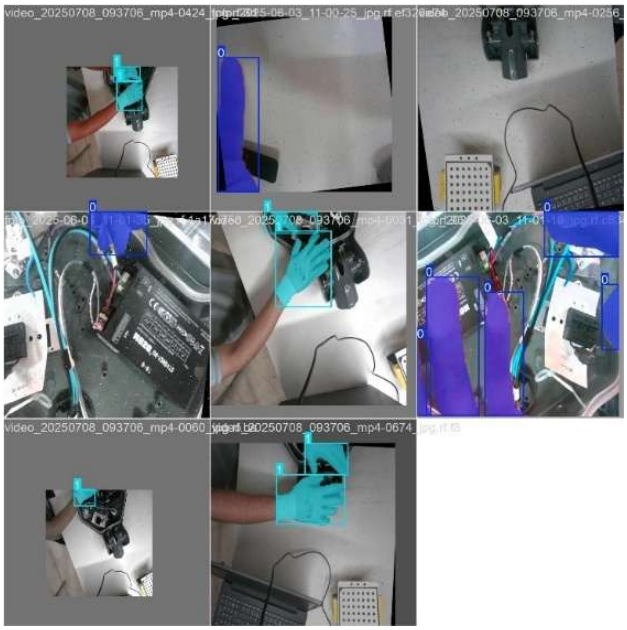


Ilustración 41. Imágenes salida entrenamiento

La validación no actualiza el entrenamiento, el modelo no aprende de esta parte del proceso. Se utiliza para comprobar los resultados obtenidos del entrenamiento y obtener una serie de métricas que servirán para comprobar si el modelo es útil o no.

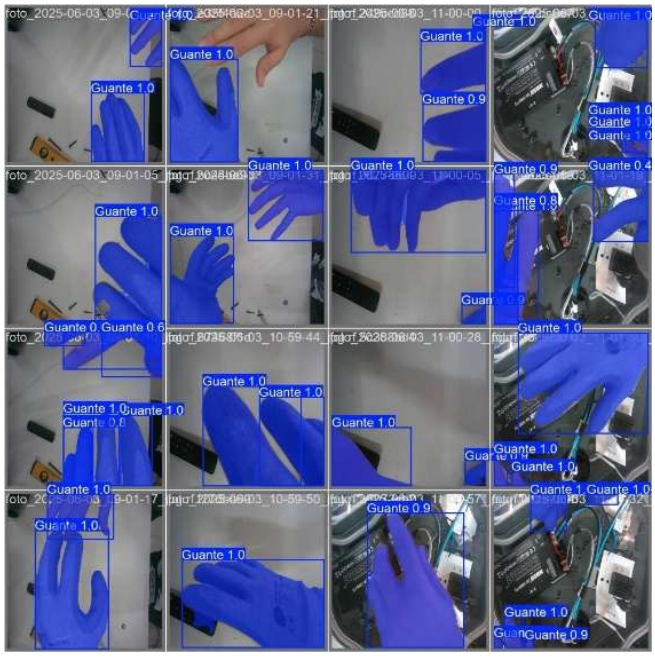


Ilustración 42. Representación de las imágenes en validación

Por último, la prueba, es una evaluación final e imparcial tras ajustar el entrenamiento y la validación. Idealmente se utiliza solamente una vez, al final del entrenamiento, obteniendo unos resultados con diferentes gráficas.

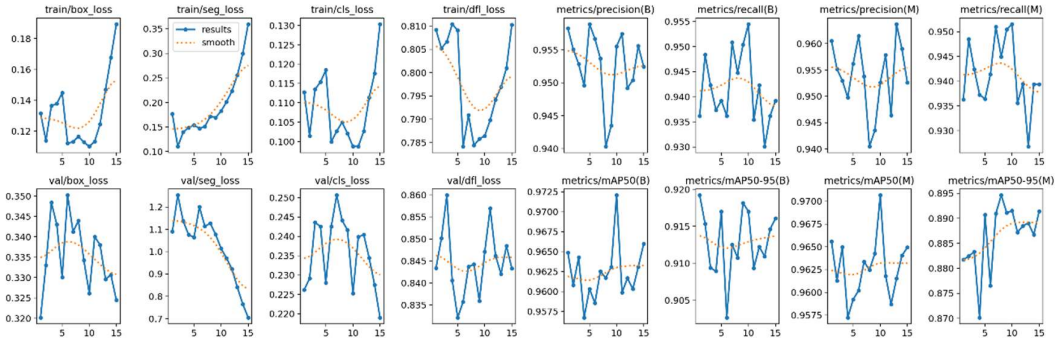


Ilustración 43. Resultados del entrenamiento

4.2.3 DETECCIÓN DE MANOS

La detección de manos es una de las partes más importantes del proceso, de ello depende la seguridad del operario y la viabilidad del proyecto. Para ello se ha elegido mejor opción utilizar un modelo de segmentación con 1202 imágenes etiquetado con Roboflow.

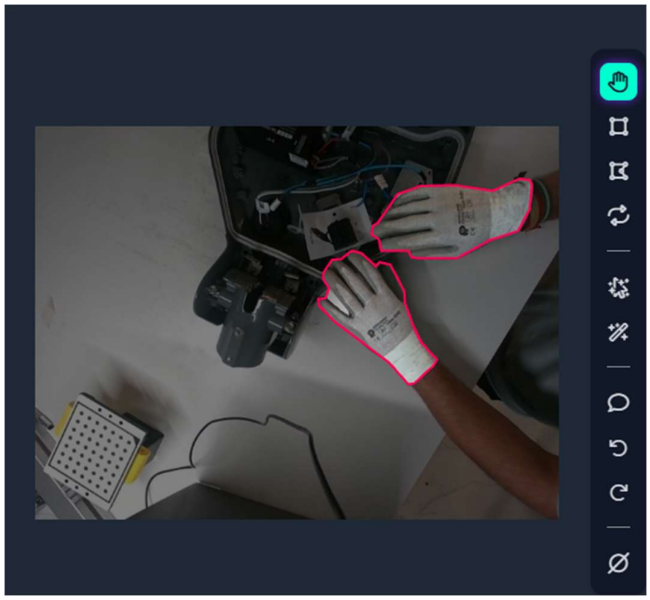


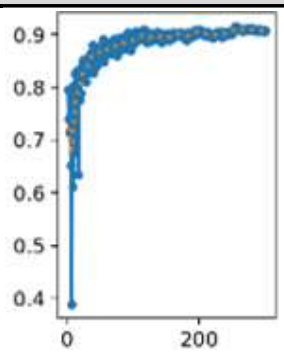
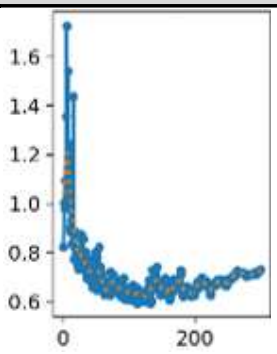
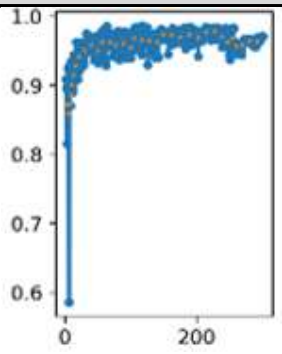
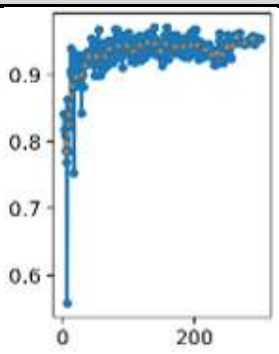
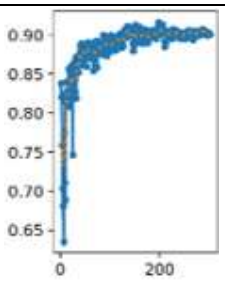
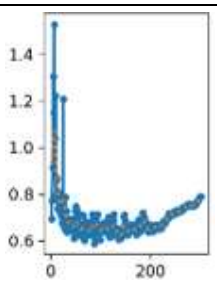
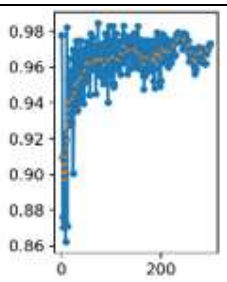
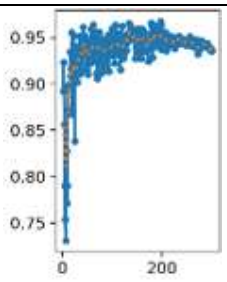
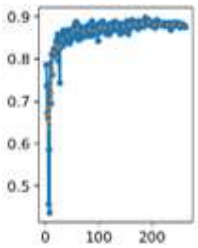
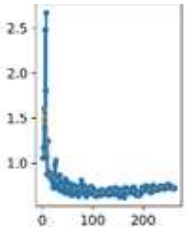
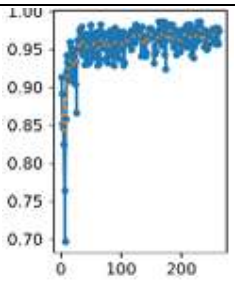
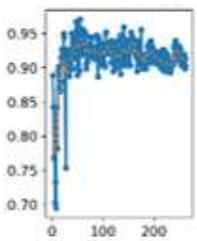
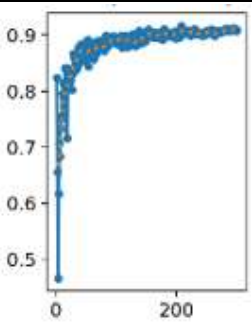
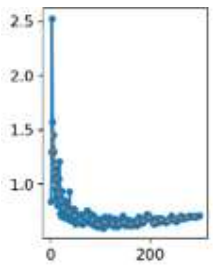
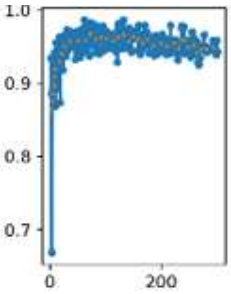
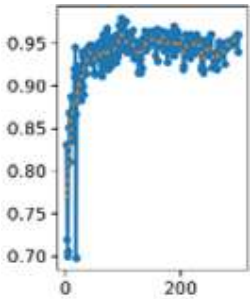
Ilustración 44. Segmentación de manos con Roboflow

Una vez etiquetado todo se aplica un preproceso anterior a la creación del dataset final, con el fin de reducir el tiempo de entrenamiento y aumentar el rendimiento aplicando transformaciones a todas las imágenes de este conjunto de datos. Se eligen las opciones de auto orientación y redimensionar a 640x640 píxeles [15].

El siguiente y último paso antes de obtener el modelo será el de aumentar el número de imágenes final x3, incluyendo imágenes ficticias a partir de las originales con ruido y con desenfoque. Se han elegido estas opciones ya que en el proceso vamos a ver manos en movimiento y no se va a trabajar con objetos estáticos apenas [15].

Se crea un dataset y, en el momento de su descarga, Roboflow ofrece gran variedad de formatos. Se han seleccionado para compararlos, tanto YOLO8, como YOLO11. Estos a su vez, se entrenan en tamaño *nano* y *small*, para comprobar cuál tiene mejores resultados, y elegir el que más se adapte a las necesidades del proyecto.

Tabla 3. Comparación modelos YOLO

MODELO	metrics /mAP50-95(M)	val/seg_loss	metrics/precision(M)	metrics/recall(M)
Yolov8 nano				
Yolov8 small				
Yolo11 nano				
Yolo11 small				

La parte más importante para analizar en un modelo de segmentación con la función de seguridad es la precisión. En este caso los cuatro modelos son bastante parecidos en este parámetro, por lo que se analiza el resto de la tabla.

La característica val/seg_loss, nos ayuda a conocer si un modelo se ajusta bien evitándolo hacer en exceso. Los mejores resultados los obtenemos en los modelos de YOLO8.

Por último, los parámetros, *metrics/precision(M)* y *metrics/recall(M)*, son útiles si quieres entender si un modelo es mejor evitando falsos positivos o detectando objetos, respectivamente. En ambas métricas la diferencia entre ellos en prácticamente nula.

La decisión final para este proyecto, conociendo que la velocidad de detección de YOLOv8 es mayor que la de YOLO11 por la información de la *Tabla 2*, es el modelo YOLOv8 nano.

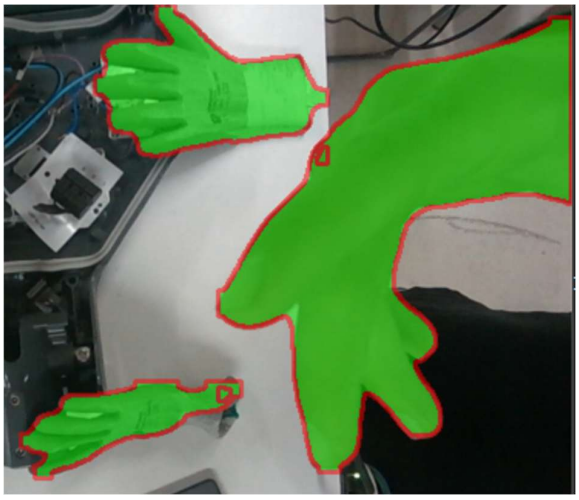


Ilustración 45. Detección manos y guantes YOLOv8 nano

4.2.4 DETECCIÓN DE TORNILLOS

La detección de tornillos será siguiendo el método de *object detection* o detección de objetos, diferente que las manos, ya que este tipo nos muestra una caja con el modelo encontrado en la imagen.

Se ha elegido este tipo de detección porque es muy fácil obtener el centro del tornillo, ya que el cuadrado se representa gracias a dos esquinas opuestas y con las siguientes ecuaciones se obtiene el centro en píxeles.

$$RowCentro = \frac{Row1+Row}{2} \qquad \text{Ec(8)}$$

$$ColCentro = \frac{Col1+Col2}{2} \qquad \text{Ec(9)}$$

La detección de tornillos no necesita ser ejecutada con tanta velocidad como la detección de manos, pero sí que necesita precisión. Por ello se ha decidido que el modelo para este proceso sea el *Yolov8 medium*, que ofrece grandes resultados, incluso en directo.

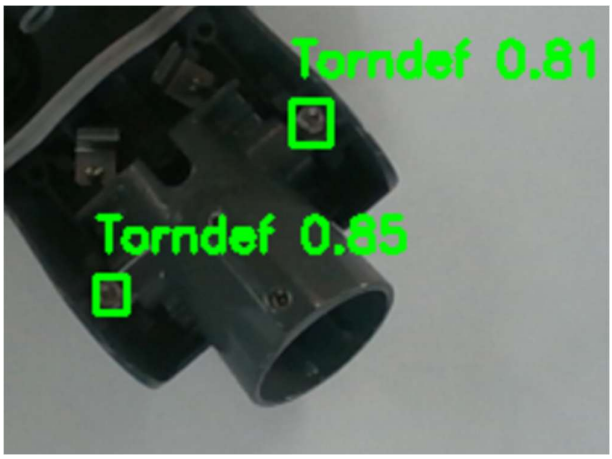


Ilustración 46. Detección de tornillos con YOLOv8 medium

4.3 DEEP LEARNING

4.3.1 INTRODUCCIÓN DEEP LEARNING

Deep Learning Tool, forma parte del conjunto de aplicaciones pertenecientes a MVTec, por lo que la transferencia de datos al código en HALCON es prácticamente automático. Es una herramienta para etiquetar datos de entrenamiento para la detección de objetos, clasificación y segmentación semántica basadas en aprendizaje profundo.

La función que ejerce es similar a la de Roboflow, un software con interfaz sencilla para etiquetar las imágenes. Posteriormente, se entrenan en el propio programa, y se obtienen unos archivos, con los que luego se podrá realizar la inferencia con imágenes nuevas.

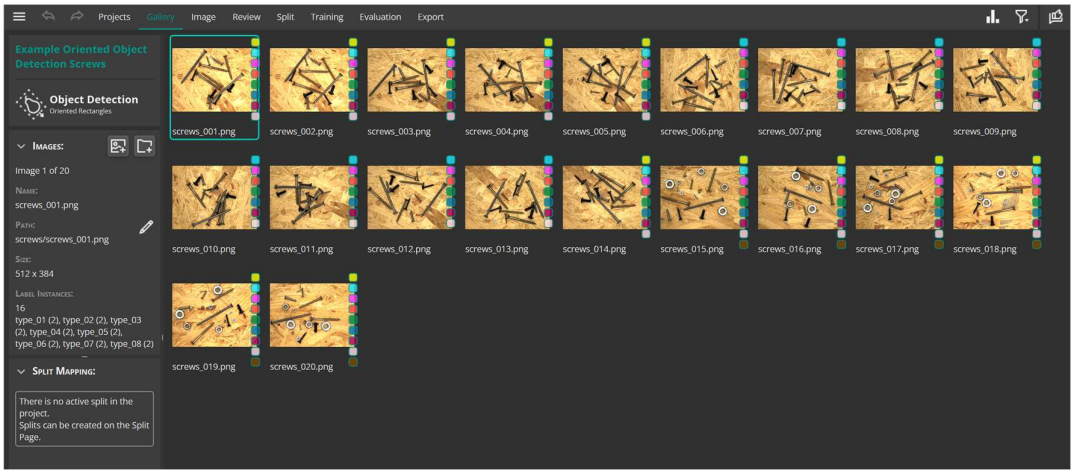


Ilustración 47. Deep Learning tool de MVTec

4.3.2 DETECCIÓN DE TORNILLOS

Para conseguir una válida inferencia de tornillos se ha utilizado un dataset con 300 imágenes, tomadas desde distintos ángulos y con diferente luminosidad. La detección de tornillos ha sido con cajas, como se puede ver en la imagen y los resultados son idóneos.

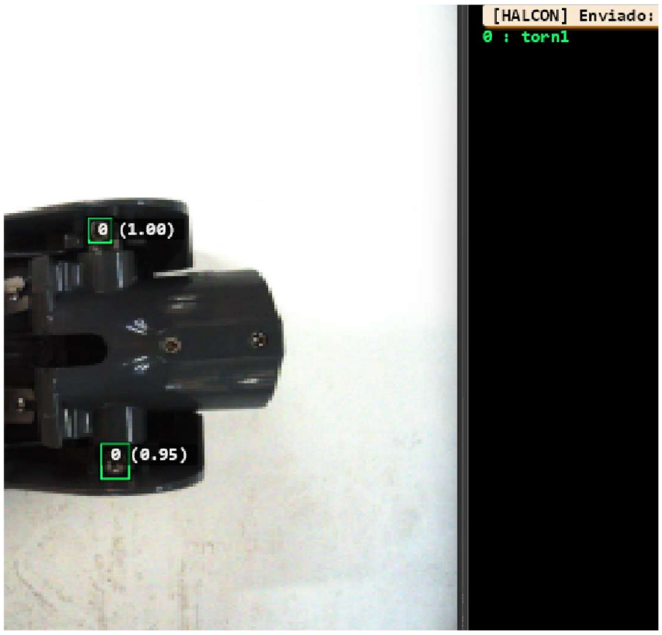


Ilustración 48. Ejemplo detección tornillos en HALCON

4.3.3 DETECCIÓN DE MANOS

La detección de manos con DeepLearning no es tan sencilla como con YOLO, se ha decidido utilizar un modelo diferente llamado *anomaly detection*.

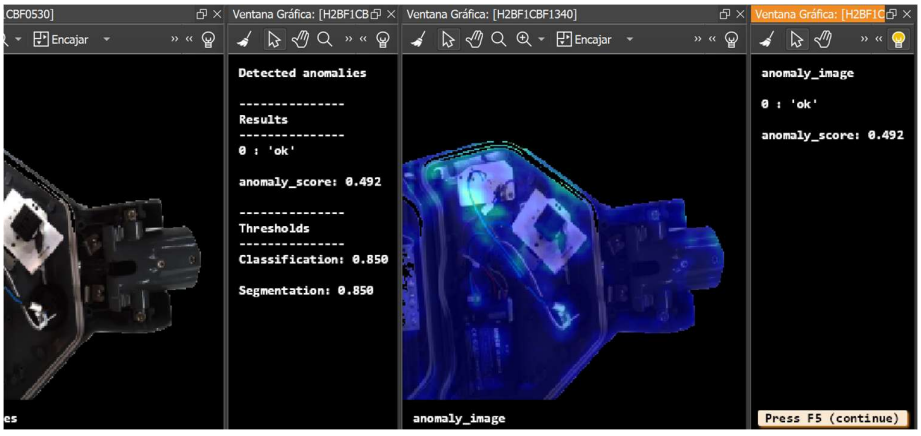


Ilustración 49. Resultados 1 de aplicación de anomaly detection en HALCON

Anomaly detection es entrenado con multitud de imágenes distintas, en las que se le indica qué imágenes son correctas y cuáles no. Por ejemplo, si hay una imagen en la que aparece la mesa únicamente, es imagen buena, si aparece la luminaria, es imagen buena. Pero si en la ilustración aparece una mano, se etiqueta como imagen no okey, y así aprende a detectar manos, pero sin saber que son manos. Es otro estilo de detección con resultados peores que los de segmentación de YOLO.

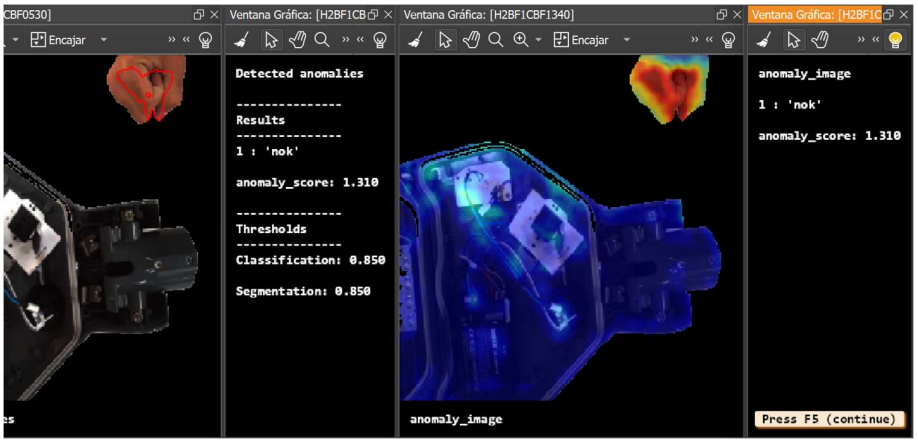


Ilustración 50. Resultados 2 de aplicación de anomaly detection en HALCON

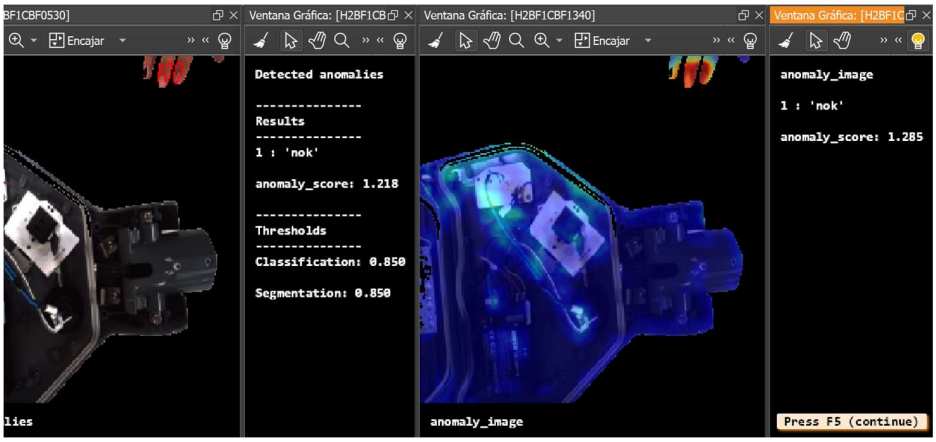


Ilustración 51. Resultados 3 de aplicación de anomaly detection en HALCON

4.3.4 CONTROL DE CALIDAD

Aprovechando que el proyecto se basa en la visión artificial y que se va a tener una cámara funcionando en tiempo real en el robot, se podría realizar un control de calidad final.

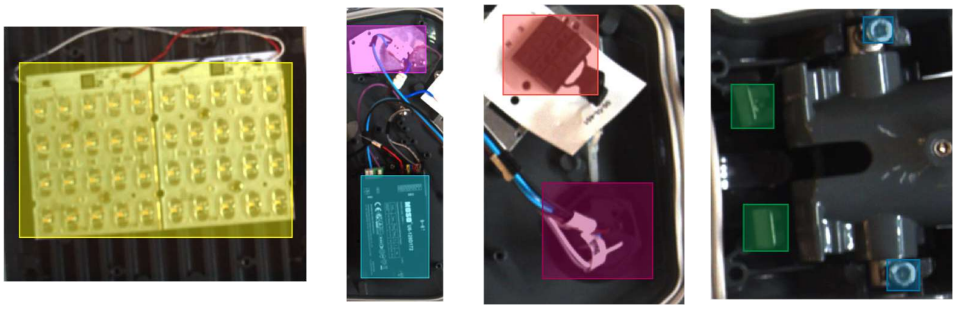


Ilustración 52. Objetos detectados

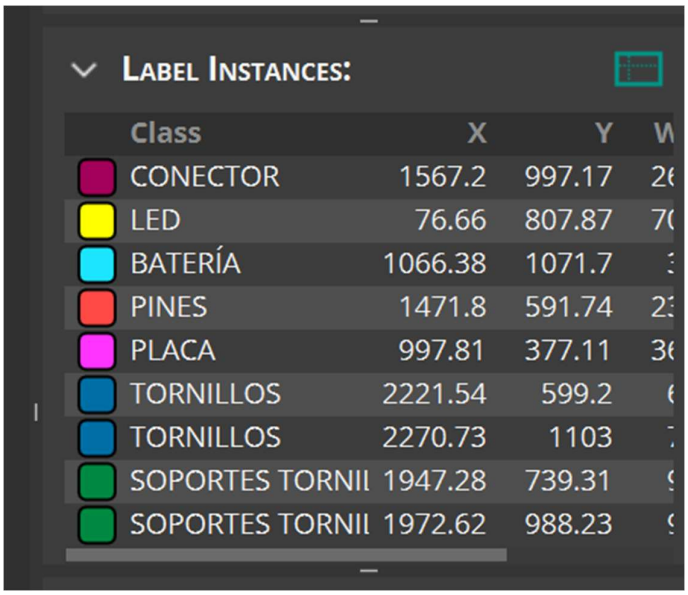


Ilustración 53. Ejemplo de detección de objetos en DEEP LEARNING

Este control de calidad detectará la presencia, o no, de diferentes objetos que deben estar en la luminaria, y ofrecerá por pantalla si falta alguno de ellos.

Se trata de un modelo de detección de objetos similar al de los tornillos, pero en cuenta de marcar solo una clase, cada objeto será una distinta. Se le ha entrenado 120 épocas, con uno resultados aceptables para tener tan solo 130 tomas. El dato de precisión tras la evaluación, con un IoU de 75%, es de 91.8%. La clase en la que más datos se detectan es la de tornillos, se solucionaría tomando más capturas de ellos únicamente.

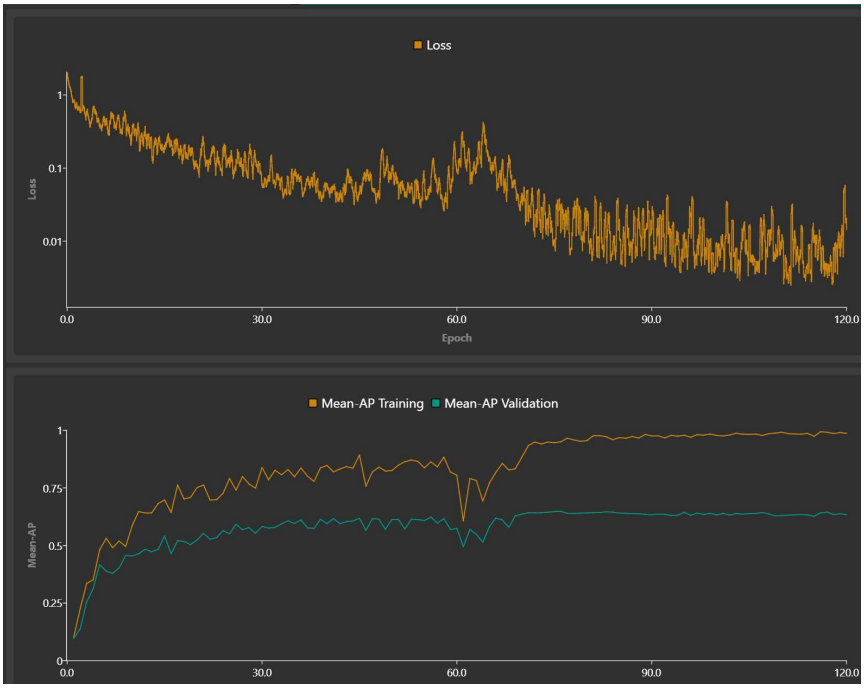


Ilustración 54. Resultados entrenamiento control de calidad

Ilustración 55. Matriz de confusión control de calidad

En la siguiente ilustración se puede comprobar que funciona a la perfección, con tomas en las que se ve la luminaria completa.

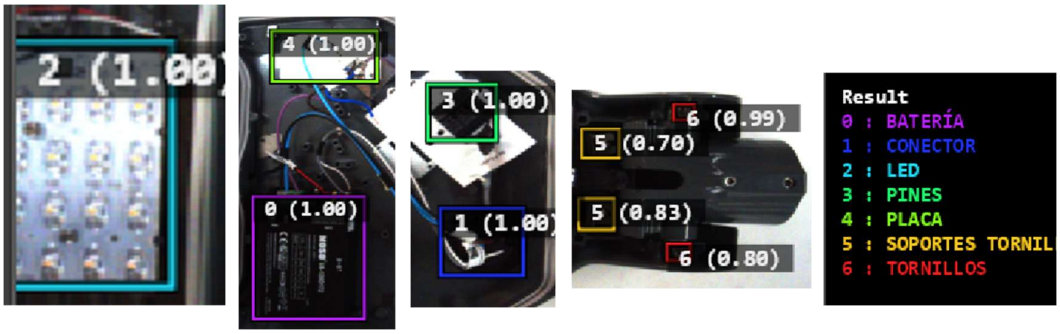


Ilustración 56. Control de calidad en HALCON

5. COMUNICACIÓN VÍA SOCKETS

La comunicación por socket es esencialmente un mecanismo de comunicación bidireccional entre procesos al que se le puede vincular un nombre, aunque también puede considerarse como un intercambio de archivos. Cada socket en uso tiene un tipo y uno o más procesos asociados.

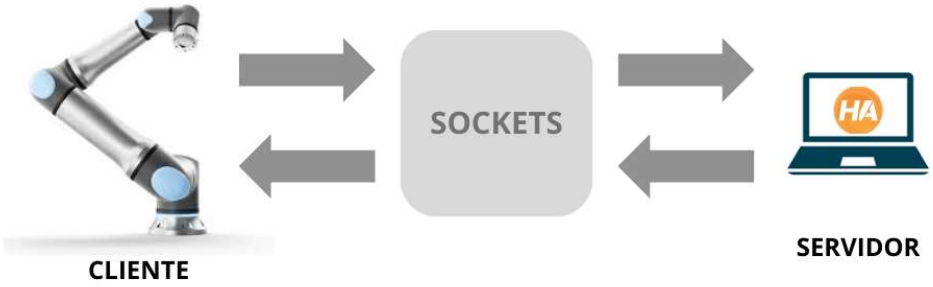


Ilustración 57. Comunicación vía sockets

Los sockets se clasifican según las propiedades de comunicación visibles para el usuario. Se supone que los procesos solo se comunican entre sockets del mismo tipo. La comunicación en este proyecto entre computadora y robot está diseñada para realizar intercambio de archivos de texto, tanto en una dirección como en la otra.

La conexión entre las dos partes será mediante un cable Ethernet, cuya conexión se tendrá que adaptar para actuar en un rango de IP 192.168.1. en su protocolo TCP4. El UR10e debe estar en el mismo rango, en este caso tendrá la IP 192.168.1.250.

Para comenzar, en primera instancia, se debe seleccionar quién debe ser el servidor y quién debe ser el cliente de la operación. El servidor, la computadora en este proyecto, será quién comience el proceso, abriendo un puerto y una IP, quedando a la escucha de alguna conexión. Una vez queda abierta la entrada al servidor el cliente, el UR10e, debe conectarse a esa entrada de comunicaciones abierta. Este proceso se realiza al inicio del programa tanto del robot, como en HALCON.

Desde el ordenador comenzaremos con los siguientes operadores,

```
LocalPort:= 30004
Protocol:= 'TCP4'
open_socket_accept (LocalPort,['protocol'],[Protocol],ServSock)
socket_accept_connect (ServSock, 'true', ConnSock)
```

Hasta que no se ejecuta la parte del código correspondiente en el URscript, el servidor se queda a la espera. El primer valor del cliente es la IP del ordenador, el segundo el puerto de entrada que ha abierto el servidor y el tercero, el nombre que toma la variable.

```urscript
socket_open("192.168.1.80",30004,"Socket")
```

Una vez establecida la conexión, se puede comenzar una *conversación* entre las dos partes.

```
CÓDIGO HALCON
send_data (ConnSock, 'z', < +'Mandado' + >, [])
receive_data(ConnSock, 'z', Estado, From)
CÓDIGO UR
socket_read_string("Socket","<",">",timeout=0)
socket_send_string("Recibido", "Socket")
```

6. GUIADO DE TRAYECTORIAS

6.1 PREPARACIÓN DEL CÓDIGO

Todo lo estudiado hasta ahora se unifica para conseguir ir al lugar correcto de los tornillos de manera precisa, con un rápido procesamiento y teniendo en cuenta la presencia del operario.

A esto se le une la idea de guiar al robot por comandos de voz, una actividad interesante que le da un salto al proyecto. Ofrece la posibilidad de un nuevo tipo de comunicación entre robot persona. Este operario es capaz de trabajar con ambas manos ocupadas e indicar al robot una acción exacta que ejecutar. Es ideal en este tipo de operaciones en los que conviven ambas partes.

A continuación, se explicará paso por paso el código que se ha utilizado para conseguir llevar al robot a dónde nosotros queremos.

Para empezar, hay que cargar todos los archivos se han ido obteniendo hasta este momento. Estos archivos son poses, datas de la calibración y modelos de entrenamiento de detección de objetos.

Como segunda tarea está la apertura de la grabación de la cámara con `open_framegrabber`, comando que sale al abrir un asistente de adquisición, conectarnos a la Lucid e insertar el código generado. El siguiente paso con la cámara es ajustar unos parámetros iniciales con los que vamos a realizar el proyecto, como son la ganancia y la exposición, necesarios para obtener una buena resolución y facilitar la tarea de detección.

Posteriormente debemos conectarnos con el robot, para que pueda haber una "conversación", entre robot y computadora, como se ha explicado en el capítulo anterior. Abriremos comunicación con el ordenador en modo escucha, por lo que será el "maestro" de la operación. El programa quedará esperando una entrada en el puerto y la IP que se le ha asignado previamente hasta que haya un cliente que se haya conectado a él. Para que esto ocurra este es el momento en el que se lanza el programa desde el robot, que directamente se conecta al ordenador y ya están preparados para comunicarse entre ellos.

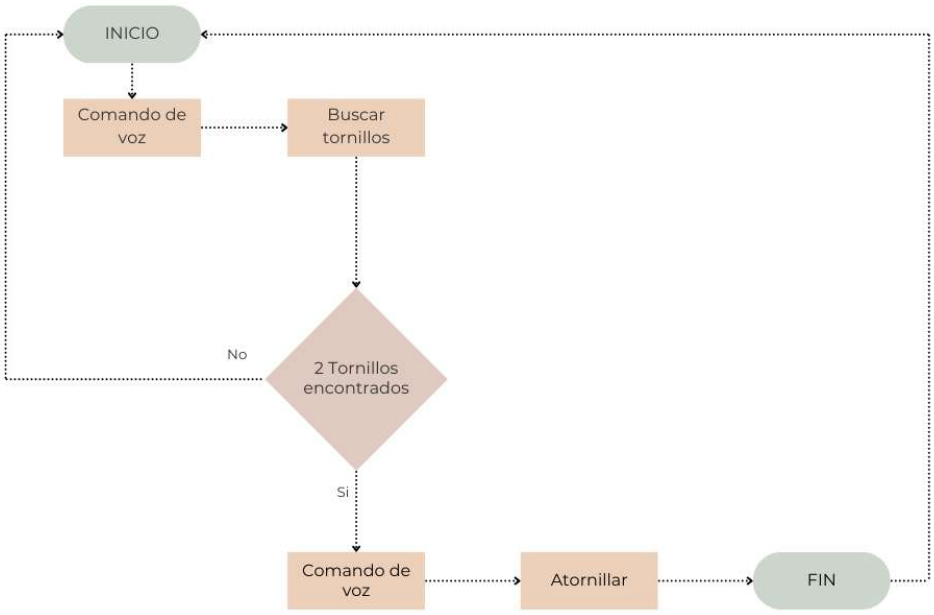


Ilustración 58. Diagrama de flujo del código en HALCON

6.2 DETECTAR TORNILLOS

En este punto comienza un bucle, cuya duración depende de la conexión con el robot, en el momento que algún procedimiento falle, y salte la excepción, acaba el programa.

Espera al comando de voz "comienza a hacer fotos", para empezar a tomar las diferentes instantáneas en busca de ambos tornillos.

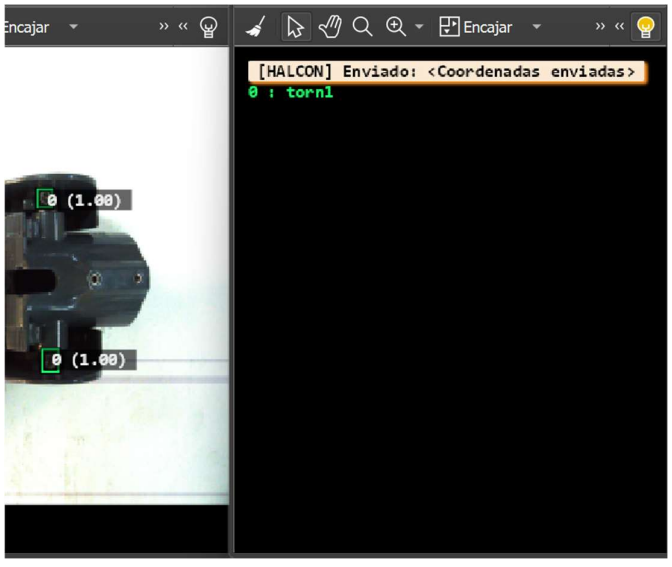


Ilustración 59. Detección de dos tornillos

Una vez recibe el comando comienza a tomar fotos en la posición de inicio, pide permiso al robot para ver si está en la posición indicada y cuando llega comienza a tomar fotos. Existe la restricción, la cual, en el momento que haga 20 fotos en una posición, si no encuentra los dos tornillos, cambia de posición. Podría suceder que el robot se moviera a todas las posiciones programadas, acabara el proceso sin resultados claros, entonces, el ciclo se terminaría, y se podría pedir al operario que moviera la luminaria. Si en algún momento detecta dos tornillos, pasa por una restricción de distancia entre ellos, si esa distancia entre ellos no es la esperada seguirá tomando fotos. Es una forma de evitar detecciones erróneas. Si está todo bien se le avisa al robot de que no tiene que cambiar más de posición, teniendo que esperar ahí el comando de voz para bajar a por tornillos. Una vez se recibe este comando, se tiene en cuenta la posibilidad de que el operario haya movido la luminaria durante el período intermedio, por lo que se vuelven a tomar fotos, esta vez sin mover al robot de posición, para una detección final. Una vez detectados ya está listo para continuar con el siguiente paso.

6.3 MANDAR POSICIONES AL ROBOT

Este momento es el más importante y complejo, ya que necesitaremos realizar los cambios de coordenadas, es clave tener claro los datos que tenemos, qué queremos conseguir, cómo lo vamos a conseguir y en qué orden.

Llegados a esta parte del código tenemos las posiciones de los dos tornillos detectados en píxeles. Tendremos cuatro vectores, como los que aparecen en la Ilustración.

BboxRow1s1	[89.9749, 179.787]
BboxCol1s1	[134.428, 138.566]
BboxRow2s1	[99.5162, 193.538]
BboxCol2s1	[141.792, 147.62]

Ilustración 60. Caja de detección de tornillos

Esto nos da las coordenadas en píxeles de las cuatro puntas del rectángulo dónde ha detectado los tornillos.

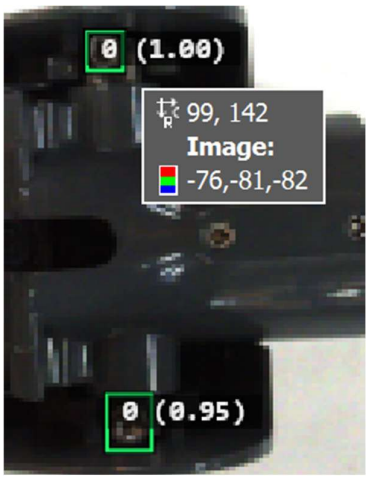


Ilustración 61. Detección de tornillo 1 con coordenadas en píxeles de una esquina

Ilustración 62. Detección de tornillo 2 con coordenadas en píxeles de una esquina

Para realizar el cambio de coordenadas es importante saber que se han obtenido ya y cuáles faltan. En primer lugar, se tiene ya la posición de la herramienta respecto de la cámara, la posición de la herramienta respecto de la base del robot en la posición de toma de imagen, ya que esta nos la ofrece el robot y está definida por el usuario. Se tiene también la pose de la placa de calibración respecto de la cámara. Con todos estos datos, se trabajará en HALCON para obtener la pose de los tornillos respecto de la base del robot.

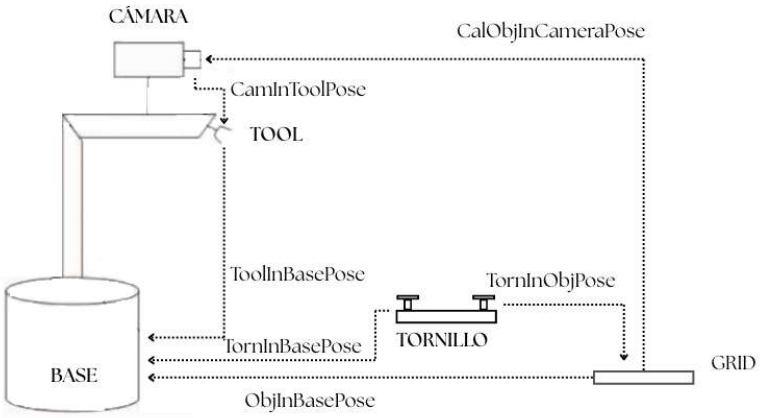


Ilustración 63. Representación poses utilizadas en el cambio de coordenadas

En primer lugar, con el siguiente operador de HALCON, se consigue la posición en X y en Y en metros entre el *grid* y los tornillos.

```
image_points_to_world_plane(CameraParam, CalObjInCameraPose,
RowC[cont_torn], ColC[cont_torn], 'm', WorldX, WorldY)
```

Posteriormente se crea una pose con esos datos, y el resto de ellos se igualan a 0, ya que la distancia en Z entre ellos es 0, y el ángulo de rotación no es interesante para la aplicación.

Ahora solo falta tener el objeto de calibración en función de la base del robot, que esto se obtiene tras una serie de operaciones entre las poses que ya teníamos desde el principio. Una vez lo obtenemos, haciendo una pose con pose se tiene la pose final de los tornillos en la base del robot, preparada para mandar por sockets al UR10e.

Con esto el guiado de trayectorias se daría por acabado, solo faltaría exportarlo a código CSharp. Esto se hace siempre a la hora de implementar los códigos en el puesto de trabajo final, y es mucho mejor para conectar con otros programas, además de poder hacer una interfaz personalizada.

7. CONCLUSIONES

Se ha llegado a la conclusión de que el proyecto hasta este punto es viable. Ya que era una de las dudas que se podía tener, ante la dificultad de encontrar el software adecuado, para convertir la operación de atornillado en colaborativa.

Además, los modelos más fiables hasta este punto son el modelo de detección de tornillos de DeepLearning de MVTec, y el modelo de segmentación de detección de manos de YOLO. Lo cual los hace los elegidos para ser implementados en el proyecto final.

Se ha podido comprobar el funcionamiento del programa y es capaz de funcionar de manera fluida, parar en el caso de reconocer manos y reaccionar ante las órdenes de voz o manuales.

Los siguientes pasos en este proyecto serían, conseguir unos mejores modelos de detección tanto de manos o de guantes, ya sean entrenados por mí mismo con un software propio o incluyendo muchas más imágenes en sus respectivos dataset. Se podría mejorar la comunicación con el robot. Conseguir una herramienta colaborativa mucho más compacta, sin perfiles, y aún más redondeada con más personalización hacia el proyecto final. Además de programar el atornillado, paso importante del proyecto.

Otro de los puntos para tener en cuenta es la dificultad de realizar este proyecto con una sola cámara y 2D, debido a que hay momentos del programa que la imagen es utilizada en varios modelos de detección, esto no permite ver el proceso tan fluido como se debería. Con dos cámaras 2D el proceso mejoraría bastante, con la opción de que la cámara que detecte manos esté embarcada en la mano del robot y la que detecte tornillos esté estática, haciendo su calibración también, mucho más sencilla y consiguiendo mayor precisión en el atornillado.

El trabajo se ha centrado en conocer y aplicar tecnologías nuevas e innovadoras que pueden resolver la problemática del reconocimiento de objetos y manos desde el punto de vista de interacción y análisis del entorno. Con esto se permitiría la flexibilidad de operación controlada, pero quedaría un interesante camino por recorrer hasta la potencial verificabilidad de estas tecnologías como “seguras”, sin duda será así en el futuro.

8. BIBLIOGRAFÍA

[1] «OnRobot Productos,» OnRobot, [En línea]. Available: https://onrobot.com/es/productos/atornillador-onrobot-screwdriver. [Último acceso: 5 Septiembre 2025].

[2] weberusa, «WEBER Screwdriving Systems Products,» 7 Septiembre 2025. [En línea]. Available: https://www.weberusa.com/product-fixtured-screwdrivers-fixtured-screwdriver-sev-c-for-human-robot-collaboration.

[3] «Reprobots Productos,» [En línea]. Available: https://www.reprobots.com/es/product-page/jaka-vps. [Último acceso: 5 Septiembre 2025].

[4] E. Fournier, C. Jeoffrion, B. Hmedan, D. Pellier, H. Fiorino, and A. Landry, «Human-cobot collaboration's impact on success, time completion, errors, workload, gestures and acceptability during an assembly task,» *Applied Ergonomics,* vol. 119, nº 104306, p. 8, 2024.

[5] U. Robots, «Universal Robots,» [En línea]. Available: https://www.universal-robots.com/es/. [Último acceso: Agosto 28 2025].

[6] I. Díaz, D. Borro, O. Iparraguirre, M. Eizaguirre, F. A. Ricardo, N. Muñoz, and J. J. Gil, «Robotic system for automated disassembly of electronic waste: Unscrewing,» *Robotics and Computer-Integrated Manufacturing,* vol. 95, nº 103032, p. 9, 2025.

[7] «Mech-Eye Industrial 3D Camera User Manual,» [En línea]. Available: https://docs.mech-mind.net/en/eye-3d-camera/2.3.1/hardware/specifications-nano.html. [Último acceso: 2 Septiembre 2025].

[8] M. S. GmbH, «About,» LinkedIn, [En línea]. Available: https://www.linkedin.com/company/mvtec-software-gmbh/about/. [Último acceso: 9 Agosto 2025].

[9] M. S. GmbH, «HALCON,» [En línea]. Available: https://www.mvtec.com/products/halcon. [Último acceso: 9 Agosto 2025].

[10] Z. Liang, T. Li, M. Huang, H. Hu, and W. Wang, «Research on hand-eye calibration method based on monocular vision and accuracy chain analysis,» de *Proc. 2023 IEEE 16th Int. Conf. Electron. Meas. Instrum. (ICEMI)*, Harbin, 2023.

[11] M. Ulrich, C. Steger, F. Butsch, and M. Liebe, «Vision-guided robot calibration using photogrammetric methods,» *ISPRSJournalofPhotogrammetryandRemoteSensing,* vol. 218, nº Part A, pp. 645-662, 2024.

[12] H. A. Hashim, «Special orthogonal group SO(3), Euler angles, angle-axis, Rodriguez vector and unit-quaternion: Overview, mapping and challenges,» *arXiv preprint,* vol. arXiv:1909.06669, p. 52, 2019.

[13] B. Malobický, M. Hruboš, J. Kafková, J. Krško, M. Michálik, R. Pirník, and P. Kuchár, «Towards seamless human–robot interaction: Integrating computer vision for tool handover and gesture-based control,» *Applied Sciences,* vol. 15, nº 3575, p. 29, 2025.

[14] P. Bergmann, K. Batzner, M. Fauser, D. Sattlegger, and C. Steger, «The MVTec anomaly detection dataset: A comprehensive real-world dataset for unsupervised anomaly detection,» *International Journal of Computer Vision,* vol. 129, pp. 1038-1059, 2021.

[15] S. Chomean, N. Khemtonglang, E. Mukda, and C. Kaset, «AI-powered body fluid cell classification: Development and validation using Roboflow and YOLOv11n framework,» *Telematics and Informatics Reports,* vol. 19, nº 100243, p. 15, 2025.

[16] «Roboflow Documentation,» [En línea]. Available: https://docs.roboflow.com/datasets/create-a-project. [Último acceso: 29 Agosto 2025].

[17] A. Vijayakumar and S. Vairavasundaram, «YOLO-based Object Detection Models: A Review and its Applications,» *Multimedia Tools and Applications,* vol. 83, nº 35, p. 83535–83574, 2024.

[18] Ultralytics, «Documentación de Ultralytics YOLO,» [En línea]. Available: https://docs.ultralytics.com/es/. [Último acceso: 23 Agosto 2025].

[19] P. Biswas, J. Islam, S. Baral, and T. Howlader, «Real-time face detection system for security surveillance using the Viola–Jones algorithm,» de *Proc. Int. Conf. Recent Progress Sci. Eng. Technol. (ICRPSET)*, Rajshahi, Bangladesh, 2024.

[20] R. Thomas, S. Yerima, and K. Shaalan, «Botnet detection using network traffic visualization and histogram of oriented gradients,» de *Proc. Int. Conf. Comput. Inf. Commun. Netw. (CICN)*, Indore, India, 2024.

[21] M. Hirabayashi, S. Kato, M. Edahiro, K. Takeda, and S. Mita, «Accelerated deformable part models on GPUs,» *IEEE Transactions on Parallel and Distributed Systems,* vol. 27, nº 6, pp. 1589-1602, 2016.

9. ANEXOS

ANEXO A. TECNOLOGÍAS UTILIZADAS

Este anexo contiene diferentes tecnologías o softwares utilizados para la realización del TFG.

A1. HALCON MVTec

HALCON es un *software* líder en visión industrial desarrollado por MVTec. Integrado en un entorno robusto como es HDevelop y se utiliza para crear aplicaciones de visión artificial con estabilidad y eficiencia. Permite una gran variabilidad de aplicaciones, está hecho a medida para la novedosa entrada de la inteligencia visual en la industria. Además de todo esto es sencillo de exportar a otros lenguajes como CSharp.

A2. Deep Learning MVTec

La herramienta Deep Learning de MVTec, es una aplicación con interfaz sencilla para etiquetar y preparar conjuntos de datos para entrenamiento por redes neuronales. Posee distintas opciones de modelos y facilidad de integración en lenguaje HDevelop.

A3. Roboflow

Roboflow es una Plataforma completa que está preparada para toda la parte de etiquetado y entrenamiento de una gran variabilidad de tipos de modelos. Con una interfaz sencilla y la capacidad de crear hasta 10 proyectos de manera totalmente gratuita. Ofrece la posibilidad de uso desde Python y acceso a una gran biblioteca de datasets y modelos preajustados. Además de facilidad de uso con otros modelos como YOLO o COCO.

A4. YOLO

YOLO es un modelo de detección de objetos en tiempo real con diferentes versiones creadas desde el 2016 hasta ahora que han ido mejorando en precisión y en velocidad de detección y procesamiento hasta la actualidad. Es pionero en detecciones de un solo disparo ideal para videovigilancia o seguridad.

A5. Microsoft Visual Studio 2022

Microsoft Visual Studio 2022 es un entorno de desarrollo integrado para Windows que reúne todas las herramientas necesarias para programar de forma eficiente. Ahora es una aplicación de 64 bits con mejoras de rendimiento y soporte creciente de herramientas de inteligencia artificial como GitHub Copilot. Ofrece la posibilidad de utilizar diferentes lenguajes dependiendo de la necesidad del cliente como Python, C#, C++, entre otros.

A6. Microsoft Visual Studio Code

Microsoft Visual Studio Code es un editor de código gratuito, de código abierto, altamente personalizable. Interfaz más sencilla que Microsoft Visual Studio 2022 y funciona en Windows, macOS y Linux.

A7. Solid Edge 2024

Solid Edge 2024, de Siemens, es un *software* para diseño y desarrollo de productos en 3D, con alto rendimiento en ensamblajes y colaboración en la nube con un equipo de trabajo.

A8. SolidWorks2022

SolidWorks 2022 es un programa de diseño 3D con mejoras centradas en el usuario y el rendimiento. Gráficos rápidos, herramientas para simulación y visualización, además de otras aplicaciones subyacentes complementarias, Simulation, Electrical, CAM, Visualize o Composer. Cada una de ellas adaptadas a opciones del diseño diferentes.

ANEXO B. FICHAS TÉCNICAS

B1. UR10e

UR10e

Technical Specification

Our most versatile robot, the UR10e delivers an impressive payload of 12.5 kg and 1300 mm reach, making it ideal for a wide range of applications.

Today, more than 90,000 UR collaborative industrial robots have been delivered to customers across industries and around the world. The e-Series of cobots brings incredible flexibility and unparalleled ease of use to your application.

Updated December 2024.

UR10e

Specification

Payload	12.5 kg (27.5 lbs)
Reach	1300 mm (51.2 in)
Degrees of freedom	6 rotating joints
Programming	12 inch touchscreen with PolyScope graphical user interface
Power consumption (average)	
Maximum power	615 W
Moderate operating settings	350 W
Operating temperature range	Ambient temperature: 0-50°C (32-122°F)
Safety functions	17 configurable safety functions
In compliance with	EN ISO 13849-1 (PLd Category 3) and EN ISO 10218-1

Performance

Force sensing, tool flange/torque sensor	Force, x-y-z	Torque, x-y-z
Range	± 100.0 N	± 10.0 Nm
Precision	± 5.0 N	± 0.2 Nm
Accuracy	± 5.5 N	± 0.5 Nm

Movement

Max TCP speed	4 m/s
Pose Repeatability per ISO 9283	± 0.05 mm

Axis movement	Working range	Maximum speed
Base	± 360°	± 120°/s
Shoulder	± 360°	± 120°/s
Elbow	± 360°	± 180°/s
Wrist 1	± 360°	± 180°/s
Wrist 2	± 360°	± 180°/s
Wrist 3	± 360°	± 180°/s

Features

IP classification	IP54
Cleanroom classification ISO 14644-1 *robotic arm payload	Class 5 at ≤ 40%* Class 6 at 80%*
Noise	< 65 dB(A)
Robot mounting	Any orientation
I/O Ports	
Digital In	2
Digital Out	2
Analog In	2
Tool I/O power supply voltage	12/24 V
Tool I/O power supply	2 A (Dual pin) 1 A (Single pin)

Physical

Footprint	Ø 190 mm
Materials	Aluminum, plastic, steel
Tool flange interface	M8 8-pin female, EN ISO 9409-1-50-4-M6
Connector type	M8 8-pin female
Cable length (robot arm)	6 m (236 in)
Weight including cable	33.5 kg (73.9 lbs)
Humidity	≤ 90% RH (non-condensing)

Contact

Universal Robots A/S
Energivej 51
5260 Odense
Denmark

+45 89 93 89 89
sales@universal-robots.com
universal-robots.com

Teach Pendant

Features

IP classification	IP54
Humidity	≤ 90% RH (non-condensing)
Display resolution	1280 x 800 pixels

Physical

Materials	Plastic
Teach Pendant size (W x H x D)	300 mm x 231 mm x 50 mm (11.8 in x 9.1 in x 1.97 in)
Weight	1.8 kg (3.96 lbs) including 1 m of teach pendant cable
Cable length	4.5 m (177.17 in)

Control Box

Features

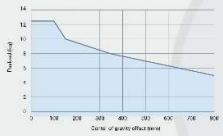
IP Classification	IP44
Operating temperature range	Ambient temperature: 0-50°C (32-122°F)
Humidity	≤ 90% RH (non-condensing)
I/O Ports	
Digital In	16
Digital Out	16
Analog In	2
Analog Out	2
Quadrature Digital Inputs	4
I/O Power Supply	24V, 2A
Industrial Protocols	Modbus-TCP (Client/Sever) EthernetIP Adapter PROFINET device/PROFIsafe ROS/ROS2
Hardware Interfaces	Ethernet 1 Gb/s USB 2.0, USB 3.0 Mini DisplayPort Injection Moulding Machine Interface (SPI AN-146 & Euromap-67)
Power Source	100-240 VAC, 47-440 Hz

Physical

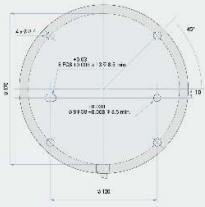
Control Box Size (W x H x D)	460 mm x 449 mm x 254 mm (18.2 in x 17.6 in x 10 in)
Weight	12 kg (26.5 lbs)
Materials	Powder coated steel

The control box is also available in an OEM version

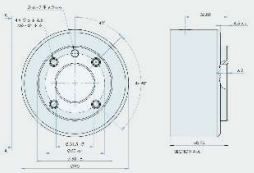
UR10e payload curve

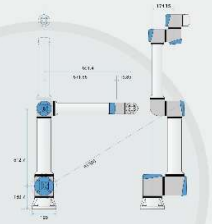


Interface for UR10e base flange



Tool flange for UR10e





B2. TRITON 5.0 MP

IP67 Protection
with IP67 lens tube and IP67 cables

Triton Models

Model	MP	Resolution	FPS	Sensor	Format	Pixel Size	Shutter	Lens Mount	Chroma	GigE Interface
TRI245S	24.5 MP	5320 x 4600 px	4.5 fps	Sony IMX540 CMOS	4/3"	2.74 µm	Global	C	M/C	M12
TRI204S	20.4 MP	4510 x 4510 px	5.5 fps	Sony IMX541 CMOS	1.1"	2.74 µm	Global	C	M/C	M12
TRI200S	20.0 MP	5472 x 3648 px	5.6 fps	Sony IMX183 CMOS	1"	2.40 µm	Rolling	C	M/C	M12
TRI162S	16.2 MP	5320 x 3040 px	6.9 fps	Sony IMX542 CMOS	1.1"	2.74 µm	Global	C	M/C	M12
TRI124S	12.3 MP	4096 x 3000 px	9.0 fps	Sony IMX545 CMOS	1/1.1"	2.74 µm	Global	C	M/C	M12
TRI120S	12.3 MP	4096 x 3000 px	9.0 fps	Sony IMX304 CMOS	1.1"	3.45 µm	Global	C	M/C	M12
TRI122S	12.2 MP	4024 x 3036 px	9.1 fps	Sony IMX226 CMOS	1/1.7"	1.85 µm	Rolling	C	M/C	M12
TRI089S	8.9 MP	4096 x 2160 px	12.5 fps	Sony IMX267 CMOS	1"	3.45 µm	Global	C	M/C	M12
TRI081S	8.1 MP	2840 x 2840 px	13.8 fps	Sony IMX546CMOS	2/3"	2.74 µm	Global	C	M/C	M12
TRI071S	7.1 MP	3208 x 2200 px	15.8 fps	Sony IMX428 CMOS	1.1"	4.5 µm	Global	C	M/C	M12
TRI064S	6.3 MP	3072 x 2048 px	17.7 fps	Sony IMX178 CMOS	1/1.8"	2.40 µm	Rolling	C	M/C	M12
TRI054S	5.4 MP	2880 x 1860 px	20.8 fps	Sony IMX490 CMOS	1/1.55"	3.0 µm	Rolling	C	Color HDR	M12
TDR054S	5.4 MP	2880 x 1860 px	13.8 fps	Sony IMX490 CMOS	1/1.55"	3.0 µm	Rolling	C	Color (AltaView On-Camera Adaptive Tone Mapping)	M12
TRI050S-P/Q	5.0 MP	2448 x 2048 px	22 fps	Sony IMX250MZR CMOS Sony IMX250MYR CMOS	2/3"	3.45 µm	Global	C	Polarized M Polarized C	M12
TRI050SI-P/Q	5.0 MP	2448 x 2048 px	22 fps	Sony IMX264MZR CMOS Sony IMX264MYR CMOS	2/3"	3.45 µm	Global	C	Polarized M Polarized C	M12
TRI051S	5.0 MP	2448 x 2048 px	22 fps	Sony IMX547 CMOS	1/1.8"	2.74 µm	Global	C	M/C	M12
TRI050S	5.0 MP	2448 x 2048 px	22 fps	Sony IMX264 CMOS	2/3"	3.45 µm	Global	C	M/C	M12
TRI032S	3.2 MP	2048 x 1536 px	35.4 fps	Sony IMX265 CMOS	1/1.8"	3.45 µm	Global	C	M/C	M12
TRI028S	2.8 MP	1936 x 1464 px	39.2 fps	Sony IMX429 CMOS	2.3"	4.5 µm	Global	C	M/C	M12
TRI023S	2.3 MP	1920 x 1200 px	48.3 fps	Sony IMX392 CMOS	1/2.3"	3.45 µm	Global	C	M/C	M12
TRI016S	1.6 MP	1440 x 1080 px	71.6 fps	Sony IMX273 CMOS	1/2.9"	3.45 µm	Global	C	M/C	M12
TRI005S	0.5 MP	812 x 620 px	166.4 fps	Sony IMX433 CMOS	1/1.7"	9.0 µm	Global	C	M/C	M12
TRI004S	0.4 MP	720 x 540 px	286 fps	Sony IMX287 CMOS	1/2.9"	6.9 µm	Global	C	M/C	M12

LUCID VISION LABS
sales@thinklucid.com
www.thinklucid.com

Triton®

29 x 29 mm Factory Tough™ Machine Vision Camera

LUCID VISION LABS

Specifications

Interface, Power, and Size Information	
Digital Interface	1000BASE-T GigE M12, PoE
GPIO Interface	8 pin M8 connector
Opto-isolated I/O ports	1 input, 1 output
Non-isolated I/O ports	2 bi-directional
Dimensions	29 x 29 x 45* mm
Lens Mount	C-mount
Weight	67 g
Power Requirement	PoE (IEEE 802.3af), or 12-24 VDC external
Power Consumption	2.5W via V_ext; ~3.1W via PoE

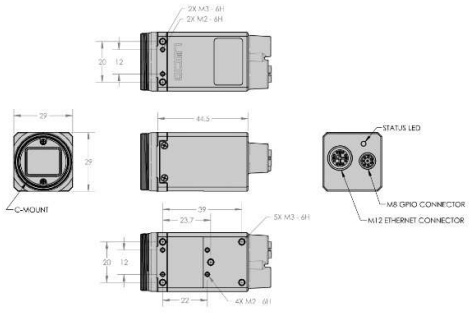
***Not including lens barrel or interface ports**

Imaging Properties	
Image Buffer	128 MB
Image Processing	Gain, gamma, black level, white balance, LUT, CCM, pixel correction, hue, saturation, color space conversion
Pixel Formats	Mono8/10/12/16, Bayer8/10/12/16, RGB8, YUV422, YUV411Mono8/12/16, PolarizeMono8/12/16 (TRI050S-P) BayerRG8/12/16, PolarizeMono8/12/16 (TRI050S-Q)
Image Modes	Horizontal and vertical binning, decimation, ROI, horizontal and vertical flip
ADC	12 bit
Gain Range	0 dB to 48 dB analog and digital
Exposure Time	47.2 µs to 10 s ****varies by model, see our website**

Standard and Certifications	
Standard	GigE Vision v2.0
Compliance	CE, FCC, RoHS, REACH, WEEE
Ingress Protection	IP67 (For IP67 protection Triton must be used with IP67 lens tube and cables)
Storage Temperature	-30 to 60°C
Operating Temperature	-20 to 55°C ambient
Shock and Vibration	DIN EN 60068-2-27, DIN EN 60068-2-64 DIN EN 60068-2-6
Humidity	Operating: 20% ~ 80%, relative, non-condensing
Warranty	3 year

Camera Features	
User Sets	1 default and 2 custom user set
File system size	16 MB
Chunk Data	Frame counter, offset X/Y, width/height, exposure time, gain, black level, line status, sequencer set
Event Data	Acquisition start/end, exposure start/end, line rise/fall, error
Counter & Timer	2 counters and 2 timers
Sequencer	Exposure time, gain
Synchronization	Software trigger, hardware trigger, PTP (IEEE 1588)

C-Mount Model



GiGE VISION GEN<i>CAM

LUCID VISION LABS
sales@thinklucid.com
www.thinklucid.com

B3. FANUC CRX 10iA

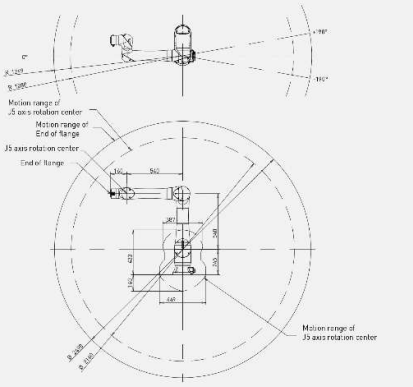
CRX-10*i*A

Max. load capacity at wrist: **10 kg**

Max. reach: **1249 mm**

Controlled axes	Repeatability (mm)	Mechanical weight (kg)	Motion range (°)						Maximum speed (°/s) *2						Maximum linear speed (mm/s)	J4 Moment/Inertia (Nm/kgm²)	J5 Moment/Inertia (Nm/kgm²)	J6 Moment/Inertia (Nm/kgm²)
			J1	J2	J3	J4	J5	J6	J1	J2	J3	J4	J5	J6				
6	± 0.04*	40	380	360	570	380	360	450	120	120	180	180	180	180	1000*1	34.8/1.28	26.0/0.90	11.0/0.30

Working range



*1) 2000mm/s in high speed mode *2) During short distance motions, the speed may not reach the maximum value stated *3) RS485 interface included *4) No dew nor frost allowed *5) Within one month *6) Not with food grade option *7) Food grade grease for all axes

Robot	**CRX-10*i*A**
Robot footprint (mm)	190
Mounting position Floor	●
Mounting position Upside down	●
Mounting position Angle	●
Controller	**R-30*i*B Plus**
Mini Plus	●
Dimensions (mm)	410 x 277 x 370
Tablet TP	●
Electrical connections	
Voltage 50/60Hz 3phase [V]	-
Voltage 50/60Hz 1phase [V]	200-240
Average power consumption [kW]	0.3
Integrated services	
Integrated signals on upper arm In/Out	2/2 *3
Integrated air supply	-
Environment	
Acoustic noise level [dB]	<70
Ambient temperature [° C]	0-45
Ambient humidity [%]	≤ 75 *4
Ambient humidity - Short term [%RH]	≤ 95 *5
Vibration acceleration [m/s²]	≤ 4.9 (0.5G)
Protection	
Body standard/optional	IP67
Wrist & J3 arm standard/optional	IP67
Cleanroom class (ISO Class 2)	o *6
Food grade / mechanical options	
Food grade grease	o *7
White epoxy paint	o

● standard o on request - not available ( ) with hardware and/or software option *based on ISO9283

B4. ABB GoFa CRB 15000

Specification			
	GoFa 5	**GoFa 12**	**GoFa 10**
Reach (mm)	950 (wrist) 1050 (flange)	1270 (wrist) 1370 (flange)	1520 (wrist) 1620 (flange)
Payload (kg)	5	up to 14kg*	up to 12kg*
Arm load (kg)	1 (mounted on axis 4)	1 (mounted on axis 2, 3, or 4)	1 (mounted on axis 2, 3 or 4)
Number of axes	6	6	6
Protection	IP54	IP67	IP67
Cleanroom class	ISO cleanroom 4	ISO cleanroom 4	ISO cleanroom 4
Mounting	Any angle, including table mounting, wall mounting, and ceiling mounting		
Controller	OmniCore C30		
Controller power needed to run the GoFa arm	100 – 230 VAC 50 – 60 Hz		
Customer power supply	M8, 3 pins connector, 24V/3A		
Customer signals	M8, 4 pins connector, CAT 5 cable, up to 4 digital signals availble for digital IOs,Analog IOs, Fieldbuses and Ethernet based protocols		
Tool flange	Standard ISO 9409-1-50		
Ambient Temperature	5°C to 40°C	5°C to 40°C	5°C to 45°C
Humidity	<75% relative humidity For limited period (<1 month): <95% relative humidity	<75% relative humidity For limited period (<1 month): <95% relative humidity	<75% relative humidity For limited period (<1 month): <95% relative humidity
Average Power Consumption	250W	250W	250W
Additional Options	Force control, integrated vision, conveyor tracking	Force control, integrated vision, conveyor tracking	Force control, integrated vision, conveyor tracking
Functional safety	SafeMove Collaborative included all safety functions certified to Category 3, PL d. It includes upto 16 safety zones		
Certifications			
SGS Functional safety	ISO 10218-1:2011 and the applicable portions of ISO/TS 15066, ISO 13849-1:2015, IEC 60204-1:2016 The product is compliant with ISO 10218-1:2011		
Additional Certifications (Standards)	UL1740 (for USA), CSA Z434 (for Canada)		

* wrist down configuration

OmniCore ™			
	GoFa 5	**GoFa 12**	**GoFa 10**
Physical Controller Dimensions	C30 I 449x443x170 (WxDxH) mm I Weight 24kg		
Available Connections	Expandable Digital I/Os, Analog I/Os, Safety I/Os		
Power requirements	100 - 230 VAC 50 - 60 Hz single phase		
Available Communication Protocol	TCP/UDP IP, RTU, Fieldbuses: PROFINET Controller, PROFINET Device, PROFIenergy, PROFIsafe Controller PROFIsafe Device, EtherNet/IP Scanner, EtherNet/IP Adapter, CC-Link IE FB Master, CC-Link IE FB Device DeviceNet c/d		
Mounting Options	Standing, Rack mount, Vertical mount, Desktop Design		
IP Level	IP20		

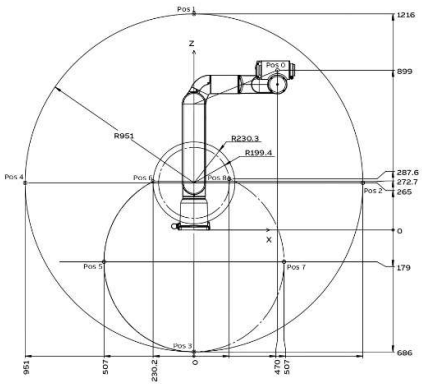
FlexPendant			
	GoFa 5	**GoFa 12**	**GoFa 10**
Screen	8'' color touch screen		
Resolution	1024x768		
Protection	IP54		
Available Cable Length	3m, 10m + extensions		
Additional Features	3D Joystick Adaptable for left-handed users		

Performance			
	GoFa 5	**GoFa 12**	**GoFa 10**
Max TCP Velocity	2,2 m/s	2 m/s	2 m/s
Max TCP acceleration (Controlled motion for nominal load)	36,9 m/s²	27 m/s²	28 m/s²
Max TCP acceleration (e-stop for nominal load)	61,6 m/s²	79 m/s²	94 m/s²
Pose repeatability	0.02 mm	0.02 mm	0.02 mm

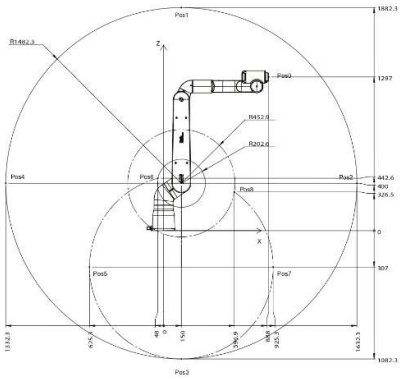
Ultra Accuracy (Feature)			
	GoFa 5	**GoFa 12**	**GoFa 10**
Absolute Position Accuracy	0.1mm	0.1mm	0.1mm
Path Accuracy	Down to 0.03mm	Down to 0.03mm	Down to 0.03mm

Physical			
	GoFa 5	**GoFa 12**	**GoFa 10**
Dimensions robot base	165 x 165 mm	200 x 200 mm	200 x 200 mm
Weight	28 kg	48 kg	51 kg

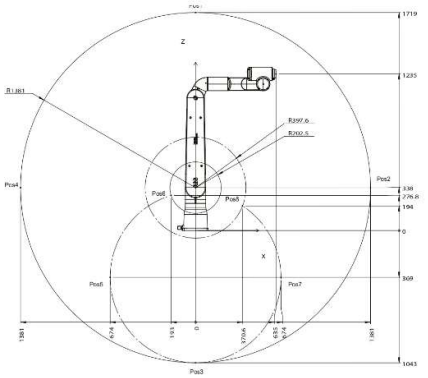
Movement						
	GoFa 5		**GoFa 12**		**GoFa 10**	
Axis movement	Working range	Axis max. speed	Working range	Axis max. speed	Working range	Axis max. speed
Axis 1 rotation	-180° to 180°	125 °/s	-270° to 270°	120 °/s	-270° to 270°	120 °/s
Axis 2 arm	-180° to 180°	125 °/s	-180° to 180°	120 °/s	-180° to 180°	120 °/s
Axis 3 arm	-225° to 85°	140 °/s	-225° to 85°	125 °/s	-225° to 85°	125 °/s
Axis 4 wrist	-180° to 180°	200 °/s	-180° to 180°	200 °/s	-180° to 180°	200 °/s
Axis 5 bend	-180° to 180°	200 °/s	-180° to 180°	200 °/s	-180° to 180°	200 °/s
Axis 6 turn	-270° to 270°	200 °/s	-270° to 270°	200 °/s	-270° to 270°	200 °/s



01 GoFa 5 Working range, Side view



02 GoFa 10 Working range, Side view

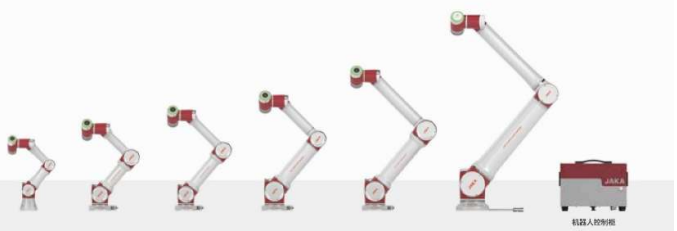


03 GoFa 12 Working range, Side view

abb.com/robotics

We reserve the right to make technical changes or modify the contents of this document without prior notice. With regard to purchase orders, the agreed particulars shall prevail. ABB AG does not accept any responsibility whatsoever for potential errors or possible lack of information in this document.

JAKA Zu 系列协作机器人



标准化
即插即用 便捷部署
柔性生产

智能化
灵活智能 操作简单
高效协作

专业化
满足高精度协作场景
可靠作业需求

	产品型号	JAKA Zu 3		JAKA Zu 5		JAKA Zu 7		JAKA Zu 12		JAKA Zu 18		JAKA Zu 20	
产品特性	额定负载	3 kg (6.6 lb)		5 kg (11 lb)		7 kg (15.4 lb)		12 kg (26.4 lb)		18 kg (39.6 lb)		20 kg (44.1 lb)	
	重量(含电缆)	12 kg (26.46 lb)		23 kg (50.71 lb)		22 kg (48.50 lb)		41 kg (90.39 lb)		35 kg (77.16 lb)		68 kg (149.9 lb)	
	臂展	626 mm (24.64 in)		954 mm (37.5 in)		819 mm (32.2 in)		1327 mm (52.2 in)		1073 mm (42.24 in)		1780 mm (70.1 in)	
	重复定位精度	±0.02 mm (±0.00079 in)		±0.02 mm (±0.00079 in)		±0.02 mm (±0.00079 in)		±0.03 mm (±0.00118 in)		±0.03 mm (±0.00118 in)		±0.05 mm (±0.00120 in)	
	自由度	6		6		6		6		6		6	
	编程	图形化编程、拖拽编程		图形化编程、拖拽编程		图形化编程、拖拽编程		图形化编程、拖拽编程		图形化编程、拖拽编程		图形化编程、拖拽编程	
	示教器类型	移动终端(电脑/平板/手机)		移动终端(电脑/平板/手机)		移动终端(电脑/平板/手机)		移动终端(电脑/平板/手机)		移动终端(电脑/平板/手机)		移动终端(电脑/平板/手机)	
动作范围及速度	机械臂	动作范围	速度	动作范围	速度	动作范围	速度	动作范围	速度	动作范围	速度	动作范围	速度
	关节1	±360°	180°/s	±360°	180°/s	±360°	180°/s	±360°	120°/s	±360°	120°/s	±360°	120°/s
	关节2	-85°~+265°	180°/s	-85°~+265°	180°/s	-85°~+265°	180°/s	-85°~+265°	120°/s	-85°~+265°	120°/s	-85°~+265°	120°/s
	关节3	±175°	180°/s	±175°	180°/s	±175°	180°/s	±175°	120°/s	±175°	180°/s	±175°	120°/s
	关节4	-85°~+265°	220°/s	-85°~+265°	180°/s	-85°~+265°	180°/s	-85°~+265°	180°/s	-85°~+265°	180°/s	-85°~+265°	220°/s
	关节5	±360°	220°/s	±360°	180°/s	±360°	180°/s	±360°	180°/s	±360°	180°/s	±360°	220°/s
	关节6	±360°	220°/s	±360°	180°/s	±360°	180°/s	±360°	180°/s	±360°	180°/s	±360°	220°/s
	工具端最大速度	/	1.5 m/s (4.922 ft/s)	/	3 m/s (9.843 ft/s)	/	2.5 m/s (8.203 ft/s)	/	3 m/s (9.843 ft/s)	/	3.5 m/s (11.484 ft/s)	/	5 m/s (16.404 ft/s)
物理性能及其他	平均功率	150W		350W		350W		500W		500W		750W	
	IP等级	IP54		IP54		IP54		IP54		IP54		IP65	
	工具I/O端口	数字输入 2		数字输入 2		数字输入 2		数字输入 2		数字输入2		数字输入2	
		数字输出 2		数字输出 2		数字输出 2		数字输出 2		数字输出2		数字输出2	
		模拟输入 2		模拟输入 2		模拟输入 2		模拟输入 2		模拟输入2		模拟输入2	
	底座直径	129 mm (5.079 in)		158 mm (6.220 in)		158 mm (6.220 in)		188 mm (7.402 in)		188 mm (7.402 in)		246 mm (9.685 in)	
机器人控制柜	IP等级	IP44		IP44		IP44		IP44		IP44		IP44	
	电控柜I/O端口	16个数字输入，16个数字输出，2个模拟输入或输出		16个数字输入，16个数字输出，2个模拟输入或输出		16个数字输入，16个数字输出，2个模拟输入或输出		16个数字输入，16个数字输出，2个模拟输入或输出		16个数字输入，16个数字输出，2个模拟输入或输出		16个数字输入，16个数字输出，2个模拟输入或输出	
	通信标配	TCP/IP, Modbus TCP, Modbus RTU, Profinet, Ethernet/IP		TCP/IP, Modbus TCP, Modbus RTU, Profinet, Ethernet/IP		TCP/IP, Modbus TCP, Modbus RTU, Profinet, Ethernet/IP		TCP/IP, Modbus TCP, Modbus RTU, Profinet, Ethernet/IP		TCP/IP, Modbus TCP, Modbus RTU, Profinet, Ethernet/IP		TCP/IP, Modbus TCP, Modbus RTU, Profinet, Ethernet/IP	
	电源	100-240VAC, 50-60Hz		100-240VAC, 50-60Hz		100-240VAC, 50-60Hz		100-240VAC, 50-60Hz		100-240VAC, 50-60Hz		100-240VAC, 50-60Hz	
	电控柜尺寸	410×307×235 (mm) (W×H×D)		410×307×235 (mm) (W×H×D)		410×307×235 (mm) (W×H×D)		410×307×235 (mm) (W×H×D)		410×307×235 (mm) (W×H×D)		410×307×235 (mm) (W×H×D)	
	重量	13.5 kg (29.762 lb)		15.4 kg (33.95 lb)		15.4 kg (33.95 lb)		18 kg (39.68 lb)		18 kg (39.68 lb)		18 kg (39.68 lb)	

B6. PHOENIX CONTACT 1452615

SACB-4/ 8-L-10,0PUR SCO P - Caja distribuidora

1452615

https://www.phoenixcontact.com/es/productos/1452615

Tenga en cuenta que los datos mostrados en este documento PDF se generaron a partir de nuestro catálogo online. Por favor, encontrará todos los datos en la documentación del usuario. Prevalecen nuestras condiciones generales de uso para descargas.

Caja distribuidora, aplicación: Estándar, tipo de conexión: Conector hembra M12 Plástico, número de puestos enchufables: 4, número de polos: 5, codificación: A, ocupación de los puestos enchufables: Doble, indicación de estado: sí, PNP; conexión de cable principal: Conexión fija 180°, PUR/PVC, longitud del cable: 10 m, apantallamiento: no

Sus ventajas

- Seguro en el campo gracias a las carcasas selladas y a los elevados grados de protección
- Cómodo: disponibilidad de máquina aumentada mediante diagnóstico rápido y sencillo
- Ahorrar espacio: caja de distribución con ocupación doble para dos sensores en un puesto enchufable
- Ahorra tiempo gracias a la instalación con bloqueo rápido SPEEDCON

Datos comerciales

Código de artículo	1452615
Unidad de embalaje	1 Unidades
Cantidad mínima de pedido	1 Unidades
Clave de venta	AF3CBA
Clave de producto	AF3CBA
GTIN	4046356553933
Peso por unidad (incluido el embalaje)	1.489 g
Peso por unidad (sin incluir el embalaje)	1.387,5 g
Número de tarifa arancelaria	85444290
País de origen	DE

Datos técnicos

Notas

Indicaciónes de montaje	IMPORTANTE: Al tender cables, respete los radios de curvatura admisibles, ya que las fuerzas de curvatura excesivas pueden menoscabar el índice de protección. Alivie las cargas mecánicas antes del conector, p. ej., mediante bridas.
Nota sobre la aplicación	Las ranuras no ocupadas deben cerrarse antes de la puesta en servicio. Encontrará los elementos de cierre adecuados en "Accesorios".

Propiedades del artículo

Tipo de producto	Caja distribuidora
Aplicación	Estándar
Número de polos	5
Número de puestos enchufables	4
Codificación	A
LED	sí

Propiedades de sistema

Diagnóstico local

Función vigilada	Tensión de alimentación
Representación óptica	LED verde

Propiedades eléctricas

Tensión de servicio máxima $U_{máx.}$	30 V DC
Diagnóstico local	Tensión de alimentación LED verde
	Indicación de estado de E/S LED amarillo
Tensión nominal U_N	24 V DC
Corriente asignada total	12 A
Capacidad de corriente por pista máx.	2 A
Capacidad de corriente por puesto enchufable	4 A

Datos de conexión

Conexión de conductores

Tipo de conexión sensor/actuador	Conector hembra M12
Par de apriete Puesto enchufable Cable de sensores/actuadores	0,4 Nm
Par de apriete del tornillo de montaje para la fijación de la carcasa	0,5 Nm

Asignación de conexiones

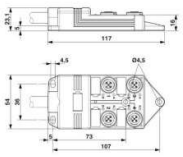
Puesto enchufable/polo = Color del conductor o conexión	1 / 4 (A) = WH
	1 / 2 (B) = GYPK
	2 / 4 (A) = GN
	2 / 2 (B) = RDBU

	3 / 4 (A) = YE
	3 / 2 (B) = WHGN
	4 / 4 (A) = GY
	4 / 2 (B) = BNGN
	1-4 / 1 (+ 24 V) = BN
	1-4 / 3 (0 V) = BU
	1-4 / 5 (PE) = GNYE

Señalización

Indicación de estado disponible	sí

Dimensiones

Esquema de dimensiones	
Anchura	54 mm
Altura	23 mm
Longitud	117 mm

Datos del material

Material Manguito roscado	PBT
Material Carcasa	PBT
Material Material de sellado	PUR
Clase de inflamabilidad según UL 94	V0
Material junta Junta tórica	NBR
Material contacto	Aleación de Cu
Material superficie del contacto	dorado
Material soporte de contactos	PA

Cable/línea

Longitud del cable	10 m

PUR/PVC negro [PUR]

Esquema de dimensiones	
Peso del cable	122 kg/km
UL AWM Style	20549
Número de polos	11
Apantallado	no
Tipo de cable	PUR/PVC negro [PUR]
Construcción del conductor cable de señales	28x 0,15 mm
Línea de señales AWG	20
Construcción del conductor alimentación de tensión	56x 0,15 mm
Alimentación de tensión AWG	17
Sección de línea	8x 0,5 mm² (conductor de señales)
	3x 1 mm² (Cable de energía)
Diámetro de conductor incl. aislamiento	1,5 mm ±0,1 mm (conductor de señales)
	2,1 mm ±0,1 mm (Cable de energía)
Diámetro exterior del cable	8,70 mm ±0,2 mm
Envoltura exterior, material	PUR
Envoltura exterior, color	negro RAL 9005
Material Conductor	Conductor Cu desnudo
Material Aislamiento de conductor	PVC
Conductor individual, color	marrón, azul, verde/amarillo, blanco, verde, amarillo, gris, gris/rosa, rojo/azul, blanco/verde, marrón/verde
Grosor de pared envoltura interior	≥ 0,15 mm
Grosor de pared Envoltura exterior	≥ 0,38 mm
Cableado total	conductores cableados en capas
Tensión nominal Cable	300 V
Tensión de prueba	2000 V
Radio de curvatura mínimo, colocado de forma fija	7,5 x D
Radio de curvatura mínimo, colocado de forma flexible	10 x D
Menor radio de flexión, montaje fijo	66 mm
Menor radio de flexión, montaje móvil	87 mm
Capacidad de carga dinámica (flexión)	Ciclos de flexión máx.: 1500000, Radio de flexión: 87 mm, Trayecto de avance: 2 m, Velocidad de avance: 2 m/s
Resistencia a las llamas	DIN EN 50265
Resistencia al aceite	según VDE 0472 parte 803
Otras resistencias	buena resistencia a ácidos, álcalis y disolventes
Características especiales	sin silicona
	-40 °C ... 90 °C (cable, disposición fija)

ANEXO C. CÓDIGO RECONOCIMIENTO DE VOZ

Poder hablar con el robot, que el robot entienda lo que se le dice e interprete realizar una acción u otra es lo ideal para una tarea colaborativa. Puede tener interés en un gran número de aplicaciones en las que el operario tenga ambas manos ocupadas y de esta manera reducir tiempos de ciclo.

Esta tarea se ha desarrollado en Visual Studio Code en lenguaje Python, gracias a la librería *vosk*.

Previamente, es importante tener descargado el idioma español, y en este caso el francés, en Windows. Para que se puede reconocer el idioma perfectamente, y el robot conteste también en el idioma indicado.

Se decidió tener la capacidad de poder cambiar de idioma para mostrar la viabilidad y el alcance de esta herramienta. Se tiene la posibilidad de cambiar en todo momento de un idioma a otro con unos comandos de voz, sin necesidad de tocar el teclado.

La comunicación no es directamente con el robot, el operario hablará, lo detectará un micrófono conectado al ordenador para obtener mejor audio y la idea es que se comunique con el robot mediante comandos, palabras simples de manera rápida y eficaz.

```
import sounddevice as sd
import queue
import vosk
import sys
import json
import time
import os
import subprocess
import winsound

#RUTA_ARCHIVO_COMANDO = r"C:\Users\Miguel\Desktop\TFG\ARCHIVOS\GUIADO\Voz\comandosrobot.json"

RUTAS_MODELOS = {
    "fr": r"C:\Users\Miguel\Desktop\TFG\ARCHIVOS\GUIADO\Voz\vosk-model-small-fr-0.22",
    "es": r"C:\Users\Miguel\Desktop\TFG\ARCHIVOS\GUIADO\Voz\vosk-model-small-es-0.42"
}
corriendo = False
existe = False

```

En la primera parte del código se cargan las librerías necesarias y los modelos de *vosk* previamente descargados al ordenador para poder trabajar offline. Se podría trabajar con los modelos grandes pero el funcionamiento del pequeño es correcto ya, no sería necesario utilizar el grande.

```
FRASES_CLAVE = {
    "fr": {
        "en bas": "TORNILLOS",
        "va en bas" : "TORNILLOS",
        "descend en bas": "TORNILLOS",
        "descends en bas": "TORNILLOS",
        "descends en": "TORNILLOS",
        "on va en bas": "TORNILLOS",
        "arrête robot": "STOP_ROBOT",
        "stop le robot": "STOP_ROBOT",
        "arrête toi": "STOP_ROBOT",
        "arrête renault" : "STOP_ROBOT",
        "retourne à la position initiale": "INICIAL",
        "prends en photo": "FOTOS",
        "prends une photo": "FOTOS",
        "Prend la photo": "FOTOS",
        "photo": "FOTOS",
        "photos": "FOTOS",
        "fin du programme": "END_PROGRAM",
        "arrête programme": "END_PROGRAM",
        "fin": "END_PROGRAM",
        "stop programme": "END_PROGRAM",
        "stop le programme": "END_PROGRAM",
    },
```

```
    "es": {
        "baja por tornillos": "TORNILLOS",
        "baja a por tornillos": "TORNILLOS",
        "ve a por tornillos": "TORNILLOS",
        "ve por tornillos": "TORNILLOS",
        "vea por tornillos": "TORNILLOS",
        "tornillos": "TORNILLOS",
        "escucha baja por tornillos": "TORNILLOS",
        "escucha baja a por tornillos": "TORNILLOS",
        "escucha ve a por tornillos": "TORNILLOS",
        "escucha ve por tornillos": "TORNILLOS",
        "escucha tornillos": "TORNILLOS",
        "para robot": "STOP_ROBOT",
        "para el robot": "STOP_ROBOT",
        "paro robot": "STOP_ROBOT",
        "para": "STOP_ROBOT",
        "vuelve posición inicial": "INICIAL",
        "vuelve a posición inicial": "INICIAL",
        "ve a la posición inicial": "INICIAL",
        "posicion inicial": "INICIAL",
        "comienza a hacer fotos": "FOTOS",
        "empieza a hacer fotos": "FOTOS",
        "fotos": "FOTOS",
        "escucha comienza a hacer fotos": "FOTOS",
        "escucha empieza a hacer fotos": "FOTOS",
        "escucha fotos": "FOTOS",
        "fin del programa": "END_PROGRAM",
        "termina": "END_PROGRAM",
        "termina programa": "END_PROGRAM",
        "finaliza programa": "END_PROGRAM",
    },
}
```

Con esta parte de código se muestran las frases a detectar, si alguna de estas frases es dicha por el operario se le mandará al robot el comando que aparece a la derecha suya. Para evitar problemas, en el programa principal, primero hay que decir *ok robot*, es una manera de asegurar el comando de voz y reforzar el código.

```
78  FRASES_RESPUESTA_HABLADA = {
79      "es": {
80          "ASK_LANG": "Hola. Dime qué idioma quieres usar: español o francés.",
81          "LANG_SELECTED": "Idioma español seleccionado.",
82          "WAKE_ACK": "Te escucho.",
83          "NO_CMD": "No he oído un comando.",
84          "TORNILLOS": "Voy a buscar tornillos.",
85          "STOP_ROBOT": "Me detengo.",
86          "INICIAL": "Vuelvo a la posición inicial.",
87          "FOTOS": "Empiezo a hacer fotos.",
88          "END_PROGRAM": "Orden fin de programa enviada.",
89      },
90      "fr": {
91          "ASK_LANG": "Bonjour. Dis-moi la langue à utiliser : espagnol ou français.",
92          "LANG_SELECTED": "Langue française sélectionnée.",
93          "WAKE_ACK": "Je t'écoute.",
94          "NO_CMD": "Je n'ai pas entendu de commande.",
95          "TORNILLOS": "Je vais chercher les boulons.",
96          "STOP_ROBOT": "Je m'arrête.",
97          "INICIAL": "Je reviens à la position initiale.",
98          "FOTOS": "Je commence à prendre des photos.",
99          "END_PROGRAM": "Ordre fin de programme envoyé.",
100     },
101 }
```

```
103 FRASES_DESPERTAR = {
104     "fr": ["ok robot", "salut robot", "salut roman"],
105     "es": ["okay robot", "okay y robot", "ok robot", "okay robots", "okay y robots"],
106 (constant) VENTANA_COMANDO_SEG: float
107 VENTANA_COMANDO_SEG = 8.0
108
109 FRASES_CAMBIO_IDIOMA = {
110     "fr": ["change de langue", "espagnol", "changer de langue", "langue espagnole"],
111     "es": ["cambiar idioma", "pon español", "cambia de idioma", "francés"],
112 }
113
114 PALABRAS_CLAVE_IDIOMA_INICIAL = {
115     "fr": ["français", "je veux parler français", "parler français", "langue français"],
116     "es": ["español", "quiero hablar español", "hablar español", "idioma español"],
117 }
118
119 NOMBRES_IDIOMAS = {"fr": "Francés", "es": "Español"}
120
```

La variable FRASES_RESPUESTA_HABLADA corresponde a las frases que tiene predeterminadas el código para contestar de manera automática cuando se le ordena.

El resto de las variables son comandos tanto para empezar a hablar y frases para cambiar de idioma. Todas ellas en español y francés como se puede observar.

```python
def detectar_idioma_inicial(texto):
    t = normalizar(texto)
    for idioma, palabras in PALABRAS_CLAVE_IDIOMA_INICIAL.items():
        for p in palabras:
            if p in t:
                return idioma
    if any(x in t for x in ["español", "hola", "idioma", "quiero", "hablar"]):
        return "es"
    if any(x in t for x in ["français", "bonjour", "langue", "parler"]):
        return "fr"
    return None

def elegir_idioma_por_voz():

    tts(FRASES_RESPUESTA_HABLADA["es"]["ASK_LANG"], "es")
    time.sleep(5)
    tts(FRASES_RESPUESTA_HABLADA["fr"]["ASK_LANG"], "fr")

    print("===> Di el idioma (15 s) <===")
    modelo_temp = vosk.Model(RUTAS_MODELOS["es"])
    with sd.RawInputStream(samplerate=16000, blocksize=8000, dtype='int16',
                           channels=1, callback=callback):
        rec = vosk.KaldiRecognizer(modelo_temp, 16000)
        inicio = time.time()
        while time.time() - inicio < 15:
            data = cola_audio.get()
            if rec.AcceptWaveform(data):
                texto = json.loads(rec.Result())["text"]
                if texto:
                    print("[Idioma detectado?] →", texto)
                    idioma = detectar_idioma_inicial(texto)
                    if idioma:
                        tts(FRASES_RESPUESTA_HABLADA[idioma]["LANG_SELECTED"], idioma)
                        return idioma

        tts("Idioma no detectado, voy a hablar español.", "es")
        return "es"
```

```python
def cargar_modelo(idioma):
    ruta = RUTAS_MODELOS[idioma]
    print(f"Cargando modelo {NOMBRES_IDIOMAS[idioma]}…")
    return vosk.Model(ruta)

def detectar_accion(texto, idioma):
    t = normalizar(texto)
    for frase, accion in FRASES_CLAVE[idioma].items():
        if t == normalizar(frase):
            return accion
    return None

def detectar_cambio_idioma(texto, idioma):
    t = normalizar(texto)
    for frase in FRASES_CAMBIO_IDIOMA[idioma]:
        if t == normalizar(frase):
            return "es" if idioma == "fr" else "fr"
    return None

def escribir_comando(corriendo,existe,accion):
    datos = {

        "corriendo": corriendo,
        "existe": existe,
        "accion": accion
        }
    print (json.dumps(datos),flush=True)
    print(f"Acción mandada: {accion}")
    try:
        winsound.Beep(1500, 120)
    except Exception:
        pass

def confirmar_por_voz(accion, idioma):
    frase = FRASES_RESPUESTA_HABLADA[idioma].get(accion)
    if frase:
        tts(frase, idioma)
```

Estas partes de código son diferentes funciones que actúan en algún momento, ya sea para detectar el idioma inicial, cargar modelos, cambiar idioma, responder al usuario. Todo ello va dentro de un programa principal.

```python
def main():
    idioma = elegir_idioma_por_voz()
    modelo = cargar_modelo(idioma)

    print(f"Modelo cargado. Di 'ok robot'. Idioma: {NOMBRES_IDIOMAS[idioma]}")
    corriendo=True
    estado = "wake"
    limite = 0.0

    with sd.RawInputStream(samplerate=16000, blocksize=8000, dtype='int16',
                           channels=1, callback=callback):
        rec = vosk.KaldiRecognizer(modelo, 16000)
        try:
            while True:
                data = cola_audio.get()
                if not rec.AcceptWaveform(data):
                    continue

                texto = json.loads(rec.Result())["text"]
                if not texto:
                    continue
                print(f"[{estado.upper()} - {NOMBRES_IDIOMAS[idioma]}] {texto}")

                nuevo_idioma = detectar_cambio_idioma(texto, idioma)
                if nuevo_idioma:
                    idioma = nuevo_idioma
                    modelo = cargar_modelo(idioma)
                    rec = vosk.KaldiRecognizer(modelo, 16000)
                    tts(FRASES_RESPUESTA_HABLADA[idioma]["LANG_SELECTED"], idioma)
                    estado = "wake"

                    try:
                        while True:
                            cola_audio.get_nowait()
                    except queue.Empty:
                        pass
                    continue

                if estado == "wake":
                    if contiene(texto, FRASES_DESPERTAR[idioma]):
                        tts(FRASES_RESPUESTA_HABLADA[idioma]["WAKE_ACK"], idioma)
                        limite = time.time() + VENTANA_COMANDO_SEG
                        estado = "cmd"
                    continue
```

```python
                if estado == "wake":
                    if contiene(texto, FRASES_DESPERTAR[idioma]):
                        tts(FRASES_RESPUESTA_HABLADA[idioma]["WAKE_ACK"], idioma)
                        limite = time.time() + VENTANA_COMANDO_SEG
                        estado = "cmd"
                    continue

                if estado == "cmd":
                    accion = detectar_accion(texto, idioma)
                    if accion:
                        existe=True
                        escribir_comando(corriendo,existe,accion)
                        confirmar_por_voz(accion, idioma)

                        time.sleep(1)
                        escribir_comando(corriendo,False,"No hay ningún comando")
                        estado = "wake"
                        continue

                    if time.time() > limite:
                        tts(FRASES_RESPUESTA_HABLADA[idioma]["NO_CMD"], idioma)
                        estado = "wake"
                        continue

        except KeyboardInterrupt:
            print("\nDetenido por el usuario.")
            corriendo = False
            existe = False
            accion = "No hay comandos"
            escribir_comando(corriendo,existe,accion)
            sys.exit(0)
```