ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

CARRERA ELECTRÓNICA Y AUTOMATIZACIÓN

"DISEÑO DE UN SISTEMA DE CONTROL PARA CLASIFICACION DE OBJETOS UTILIZANDO UN BRAZO ROBOTICO KUKA KR10 R 900 SIXX MEDIANTE VISION ARTIFICIAL"

Trabajo de Integración Curricular

Tipo: Proyecto Técnico

Presentado para optar al grado académico de:

INGENIERO EN ELECTRÓNICA Y AUTOMATIZACIÓN

AUTOR: JONATHAN FRANCISCO PILCO VILLA

DIRECTOR: ING. RAMIRO FERNANDO ISA JARA

Riobamba – Ecuador

2023

Yo, **Jonathan Francisco Pilco Villa**, declaro que el presente Trabajo de Integración Curricular es de mi autoría y los resultados de este son auténticos. Los textos en el documento que provienen de otras fuentes están debidamente citados y referenciados.

Como autor asumo la responsabilidad legal y académica de los contenidos de este Trabajo de Integración Curricular. El patrimonio intelectual pertenece a la Escuela Superior Politécnica de Chimborazo.

Riobamba, 01 de Noviembre del 2023

Jonathan Francisco Pilco Villa

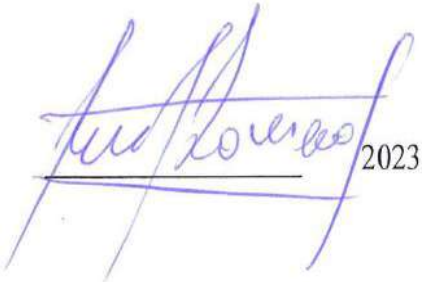
180520804-6

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

CARRERA ELECTRÓNICA Y AUTOMATIZACIÓN

El Tribunal del Trabajo de Integración Curricular certifica que: El Trabajo de Integración Curricular; Tipo: Proyecto Técnico; **"DISEÑO DE UN SISTEMA DE CONTROL PARA CLASIFICACIÓN DE OBJETOS UTILIZANDO UN BRAZO ROBÓTICO KUKA KR10 R900 SIXX MEDIANTE VISIÓN ARTIFICIAL"**, realizado por el señor **JONATHAN FRANCISCO PILCO VILLA**, ha sido minuciosamente revisado por los Miembros del Tribunal del Trabajo de Integración Curricular, el mismo que cumple con los requisitos científicos, técnicos, legales, en tal virtud el Tribunal Autoriza su presentación.

	FIRMA	**FECHA**
Ing. Mayra Alejandra Pacheco Cunduri **PRESIDENTE DEL TRIBUNAL**		2023–11-01
Ing. Ramiro Fernando Isa Jara **DIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR**		2023-11-01
Ing. Paul Patricio Romero Riera **ASESOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR**		2023-11-01

DEDICATORIA

El trabajo realizado lo dedico a mi familia, la cual día a día hace lo posible por ayudarme de una u otra manera y representan el pilar fundamental de mi vida misma, de igual manera, a todas las personas que me brindaron su apoyo durante todo y en algún momento del transcurso de mi carrera, ya que gracias a ellos pude superar todos y cada uno de los obstáculos atravesados.

Jonathan

AGRADECIMIENTO

De manera especial agradezco a Dios por brindarme todo lo necesario para mi vida, a mis padres y hermano, quienes representan mi ayuda y apoyo fundamental. Agradezco a las personas que estuvieron y compartieron conmigo momentos buenos y malos no solo en lo académico si no también en lo personal. Agradezco a los docentes que generosa y pacientemente me ayudaron durante todo el proceso académico y aún más a los ingenieros Ramiro Isa y Paul Romero, director y asesor por toda la ayuda brindada en la realización de este trabajo de titulación. Por todo su esfuerzo, sacrificio y pacencia me siento inmensamente agradecido.

Jonathan

TABLA DE CONTENIDOS

CAPÍTULO III

CAPÍTULO IV

ÍNDICE DE TABLAS

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE ANEXOS

RESUMEN

El presente trabajo de investigación tiene como objetivo el desarrollo de un sistema de control para la clasificación de objetos utilizando un robot KUKA KR10 R900Sixx mediante visión artificial. El sistema consta de la integración de dos partes, el software que permite la detección de los objetos en función de su color característico (rojo o azul) y el hardware que se encarga de la ejecución de los movimientos necesarios para trasladar los objetos. Adicionalmente, el sistema integra una plantilla que alberga los objetos para clasificar, ubicada dentro del área de trabajo del manipulador. La interconexión del hardware y software se realiza físicamente mediante una red local punto a punto entre el ordenador y el controlador del robot, mientras que, a nivel de software, la conexión se establece por medio del controlador Kukavarproxy del software RoboDK, para robots del fabricante KUKA. Dicho software, sirve como una interfaz de comunicación entre el software y hardware del sistema. Además, permite la integración de los diferentes algoritmos tanto de visón como control del movimiento utilizando Python como lenguaje de programación. Las pruebas para la detección del color para determinar un correcto funcionamiento en varios niveles de velocidad han permitido validar el sistema verificando la existencia de relación directa entre las variables de detección e iluminación del sistema, a través del análisis estadístico en base a la evaluación de hipótesis, esto ha permitido obtener un porcentaje del 95% en la precisión para la detección del color mediante matrices de confusión como métrica estadística. De esta manera, se ha determinado que el sistema funciona adecuadamente en función de los objetivos y requerimientos planteados con tiempos de ejecución entre 114,45 y 118,3 segundos.

Palabras clave: <CLASIFICACIÓN DE OBJETOS>, <DETECCIÓN DE COLOR>, <VISIÓN ARTIFICIAL>, <ROBOT KUKA>, <ROBODK>

SUMMARY

This research aimed to develop a control system for object classification using a KUKA KR10 R900Sixx robot with artificial vision. The system integrates two parts: the software allows for detecting objects based on their characteristic color (red or blue), and the hardware executes the necessary movements to transport the objects. In addition, the system integrates a template that houses the objects to be classified, located within the working area of the manipulator. The hardware and software are physically interconnected through a point-to-point local network between the computer and the robot controller, while at the software level, the connection is established through the Kukavarproxy controller within the RoboDK software, designed for KUKA-manufactured robots. This software is a communication interface between the system's software and hardware. In addition, it enables the integration of various algorithms for both vision and motion control using Python as the programming language. The tests for color detection to determine correct operation at various speed levels have allowed the system to be validated by verifying the existence of a direct relationship between the detection variables and system illumination through statistical analysis based on hypothesis evaluation. This has resulted in a 95% accuracy rate for color detection through confusion matrices as a statistical metric. This way, it has been determined that the system works effectively based on the objectives and requirements proposed, with execution times ranging between 46.32 and 47.38 seconds.

Keywords: <OBJECT CLASSIFICTION>, <COLOR DETECTION>, <ARTIFICIL VISION>, <ROBOT KUKA>, <ROBODK>

INTRODUCCIÓN

En la actualidad el desarrollo de ciertas tecnologías ha permitido incrementar el desarrollo industrial, el cual, ha otorgado una gran importancia a los robots. Esto gracias a las características de precisión, robustez y repetitividad al momento de realizar determinadas tareas. Por otra parte, los sistemas de visión artificial han tomado mucho protagonismo por la capacidad de dotar cierto grado de inteligencia a un agente integrado, permitiendo la automatización de procesos que en décadas anteriores eran muy complejas de realizar. (Rosas et al. 2017)

En gran parte de las cadenas productivas industriales, existen diferentes procesos relacionados con la inspección, clasificación, soldadura, montaje o picking de piezas y materiales. Estas tareas en su gran mayoría son realizadas manualmente o con el apoyo de algún tipo de maquinaria. Sin embargo, son procesos repetitivos que demandan mucha precisión y tiempo de quienes desarrollan estas actividades, tiempo que podría ser aprovechado de mejor manera destinándolo a otras actividades que contribuyan con el aumento de la eficiencia productiva y adecuado ambiente laboral de los trabajadores.

Ante estos hechos, algunas de las soluciones altamente utilizadas son el empleo de brazos robóticos. Estos dispositivos minimizan el trabajo de ciertos procesos de producción. Además, incrementan aspectos como la calidad y eficiencia que permiten mejorar los servicios y productos a los que está asociada la utilización de estos. Prueba de ello, en sectores productivos como el automovilístico, se han integrado sistemas robotizados que, por ejemplo, utilizan en líneas de montaje para la manipulación, clasificación, selección y transporte de piezas y componentes. De la misma manera, en el sector quirúrgico su empleo es altamente utilizado por el alto grado de precisión que contribuye a la reducción de tiempos durante una cirugía. (Flores et al. 2017)

La aplicación y uso de los robots para diferentes sectores tiene una gran proyección a futuro, ya que, la industria moderna demanda tiempos de fabricación más cortos con altos estándares de calidad mediante la aplicación de soluciones de bajo costo, flexibles y con la mayor capacidad de adaptabilidad. De esta manera, los sistemas que integren manipuladores robots con sistemas de visión artificial ofrecen una solución con las características mencionadas. Esto debido al uso de algoritmos de visión artificial para percibir características del entorno para la toma de decisiones del manipulador. Esto abre paso a nuevas aplicaciones como sistemas del control de calidad inteligentes y sistemas robóticos autónomos de manipulación de objetos. (Vélez et al. 2014)

La funcionalidad y aplicación de sistemas de manipulación y clasificación robóticos con sistemas de visión artificial es muy positiva al brindar múltiples beneficios, ya que, con sistemas

tradicionales de programación de robots se requiere la presencia y supervisión continua de los operadores durante la ejecución de sus tareas y sugieren mantener un ambiente totalmente controlado, que se soluciona mediante un sistema inteligente que trabaje de manera autónoma. Teniendo en cuenta el grado de inteligencia que pueden adoptar los robots que integren sistemas de visión, se consigue fundamentalmente entornos más seguros y eficientes gracias a la menor intervención de los operadores en los escenarios productivos. (Esneca Business School, 2022)

CAPÍTULO I

1 DIAGNOSTICO DEL PROBLEMA

En este capítulo se presenta el planteamiento del problema, la justificación del proyecto tanto teórica como aplicativa, los objetivos, general y específicos para abordar el presente trabajo de investigación.

1.1 Planteamiento del problema

¿Cómo diseñar un sistema de control para clasificación de objetos utilizando un brazo robótico KUKA KR 10 R900 sixx mediante visión artificial?

1.2 Justificación

Justificación Teórica

Una de las principales ventajas de los sistemas inteligentes radica en la funcionalidad de que un trabajo se realice de manera eficaz, segura, autónoma y aprovechando de manera óptima los recursos. Esto no sucede al emplear otras alternativas tanto manuales como programadas, las cuales requieren más tiempo y carecen de la capacidad de adaptarse a situaciones variables. Por tal razón, empresas e industrias ven la necesidad de transformar sus procesos productivos en sistemas inteligentes que se adapten a las nuevas infraestructuras y demandas que se desarrollan actualmente a nivel mundial.

El desarrollo de un sistema para la clasificar objetos mediante un brazo robótico utilizando técnicas de visión artificial permite lograr lo antes mencionado, ya que permite dotar a los dispositivos y maquinas, en este caso un brazo robótico, cierto grado de inteligencia cuya complejidad está en dependencia del sistema que lo gobierne. De esta manera, un brazo robótico puede trabajar de manera autónoma y adaptativa gracias a la integración de un sistema de visión que le permite interactuar con el entorno en el que se encuentra. Esta nueva característica garantiza la realización de tareas de manera más eficiente al no depender del control de un operador directamente o la programación de trayectorias en condiciones de trabajo no modificables.

Los sistemas de visión artificial permiten determinar características propias de un entorno que mediante el análisis de una escena se pueden tomar decisiones. Con el apoyo de las diferentes técnicas de visión artificial para captar imágenes del entorno y la capacidad de procesar dichas imágenes e incluso videos, se pueden detectar características de un medio como formas, colores o patrones que generalmente son parámetros de análisis en aplicaciones de clasificación, selección o trasporte, tareas que pueden ser desarrollados con la ayuda de un brazo robótico.(Montalvo, 2009)

El sistema propuesto pretende realizar procesos de clasificación de objetos de manera más eficiente mediante la integración de una cámara que permita determinar las características del objeto a manipular para posteriormente movilizarlo mediante el brazo robótico. De manera general, un brazo robótico trabaja gracias a la resolución de sus problemas cinemáticos directos o inversos para determinar la posición y orientación final del extremo del robot. A partir de esos datos, se fijan puntos conocidos como nodos hasta llegar aún destino. Mientras que, mediante la cámara el sistema conocería en cada instante la posición en la que se encuentra el objeto para ser movilizado en donde el proceso de clasificación lo indique.(Iñigo Madrigal, Vidal Idiarte, 2002)

Justificación Aplicativa

El trabajo de integración curricular con el tema: "Diseño de un sistema de control para clasificación de objetos utilizando un brazo robótico KUKA KR 10 R900 sixx mediante visión artificial", está centrado en la detección de las características de objetos, específicamente los bordes para determinar los colores de un objeto mediante un algoritmo de detección de los parámetros antes mencionados, así como la determinación del espacio en el que trabajará el manipulador gracias al empleo de una cámara. A su vez, el algoritmo de identificación de color en los objetos permite enviar las instrucciones de movimiento correspondientes al dispositivo controlador del brazo robótico para realizar la clasificación en grupos de acuerdo con los parámetros que definen los objetos.

El sistema de control trabajara en tres etapas, la primera consiste en la adquisición de imágenes del escenario para su posterior análisis. La siguiente etapa consiste en el procesamiento de las imágenes utilizando librerías como Open CV. Finalmente se realiza el envió de datos de trayectorias a ejecutar al robot utilizado los protocolos y medios de comunicación necesarios. (García et al. 2010)

Los objetos para clasificar presentan características definidas a fin de que el reconocimiento de estos sea lo más preciso. Los objetos serán construidos en madera por la facilidad para trabajarla de manera sencilla y rápida. En específico, los objetos son cubos en color rojo y azul ya que por

su contraste son fácilmente perceptibles y distinguibles. El tamaño puede variar dependiendo de las características del elemento terminal del robot para manipularlo.

Con esta orientación del sistema de visión para el manipulador, se obtendría un incremento importante en su eficiencia, productividad y funcionalidad porque su operación se tornaría inteligente. De esta manera se evitaría la utilización de los controladores manuales, en este caso el Smart PAD, para trabajar en uno de los modos de operación automáticos del robot KUKA a mayores velocidades. O soluciones vía software que consisten en el desarrollo de modos de operación definidos o utilización de una interfaz con botones y sliders digitales que se encargan de mover las articulaciones del brazo robótico que nuevamente implica la intervención humana y trabajan sobre espacios de trabajo con condiciones invariables necesariamente. (Brito, 2019)

El cumplimiento de este trabajo busca desarrollar entornos y procesos automatizados en las líneas productivas donde se emplee brazos robóticos para tareas de picking y clasificación, los cuales, son lentos e implican muchas horas de trabajo que en el campo industrial representa perdidas de dinero por lotes productivos incompletos. Por esta razón, la aplicación y desarrollo del sistema de control para clasificar objetos por visión artificial, contribuye al mejoramiento de estos procesos ya que se desarrollarán de manera autónoma e inteligente.

1.3 Objetivos

Objetivo General

Diseñar un sistema de control para clasificación de objetos utilizando un brazo robótico KUKA KR10 R900 sixx mediante visión artificial.

Objetivos Específicos

- Analizar el estado del arte respecto a visión artificial y controladores y lenguajes de programación de robots manipuladores KUKA.
- Definir los requerimientos que debe cumplir el sistema de control, así como las características de los elementos que lo integran.
- Verificar la correcta detección de los objetos definidos y el correcto posicionamiento del brazo.
- Identificar por medio de visión artificial las características de color y posición de los objetos a clasificar.
- Ejecutar el sistema de control para el brazo robótico KUKA de manera exitosa para su validación.

1.4 Metodología

Entre los diferentes métodos y técnicas que se utilizaran para el desarrollo de este trabajo de integración curricular se contemplan:

1.4.1 Métodos teóricos

Revisión documental. – Obtener información mediante la investigación e indagación de documentos, artículos, publicaciones, libros, etc. Relacionados con sistemas de control para la clasificación de objetos utilizando brazos robóticos y visión artificial.

Sistematización. – Empleo de la documentación revisada para determinar requerimientos de diseño y funcionamiento del sistema.

Análisis y síntesis. – Evaluar e interpretar los resultados de las pruebas realizadas para establecer conclusiones y recomendaciones de la investigación.

1.4.2 Métodos empíricos

Experimentación: realizar diferentes pruebas y simulaciones de los componentes que integran el sistema.
Observación: Validar el sistema desarrollado.

CAPÍTULO II

2 MARCO TEÓRICO

En el presente capítulo se desarrolla la investigación teórica relacionada con la visión artificial, su proceso, elementos y las diferentes etapas de ejecución. Además, se analizan los diferentes sistemas de control visual y sus aplicaciones, los brazos robóticos del fabricante KUKA y las diferentes formas de comunicación que ofrecen para el uso industrial.

2.1 Visión artificial

La visión artificial es la ciencia y tecnología que permiten que las "máquinas" puedan "ver" y extraer información, resolver alguna tarea o entender la escena que están observando mediante imágenes digitales. (Maduell i García sin fecha). Para el desarrollo de estas tareas, los sistemas de visión artificial se componen de los siguientes elementos: Dispositivo de iluminación, cámara, capturador y un computador encargado del procesamiento de las imágenes (Javier et al. 2016)

Las ventajas que ofrece la visión humana se relacionan con la capacidad de interpretación cualitativa de un escenario complejo, mientras que la visión artificial tiene mejor capacidad en la medida cuantitativa de un escenario debido a su velocidad, precisión y repetibilidad. Esto conduce a la posibilidad de inspeccionar cientos o miles de objetos por minuto con la única restricción a considerar por la resolución de la cámara y a la óptica que le permiten describir detalles específicos.(Ramírez, 2014)

Las tareas que comprenden la utilización de la visión por computador van desde la detección, tarea principal que se utiliza para el presente trabajo de investigación, así como la segmentación y reconocimiento de objetos. El mapeo y seguimiento de objetos son tareas que se están desarrollando en gran manera gracias al desarrollo tecnológico en campos de inteligencia artificial, ya que, permiten desarrollar múltiples aplicaciones muy funcionales que van de la mano de la visión por computador. (Acuña, 2020)

El desarrollo de un sistema de visión artificial implica la ejecución de ciertas etapas que acondicionan la escena percibida por una cámara como el elemento central del sistema, para poder desarrollar la aplicación deseada. Estas etapas pueden variar según la aplicación a la que este destinada el sistema, sin embargo, en la Ilustración 1 – 2 se observan las etapas comunes que se

deben seguir para el desarrollo de la aplicación de visión artificial para la clasificación de objetos con forma definida, por característica de color, para su posterior análisis.

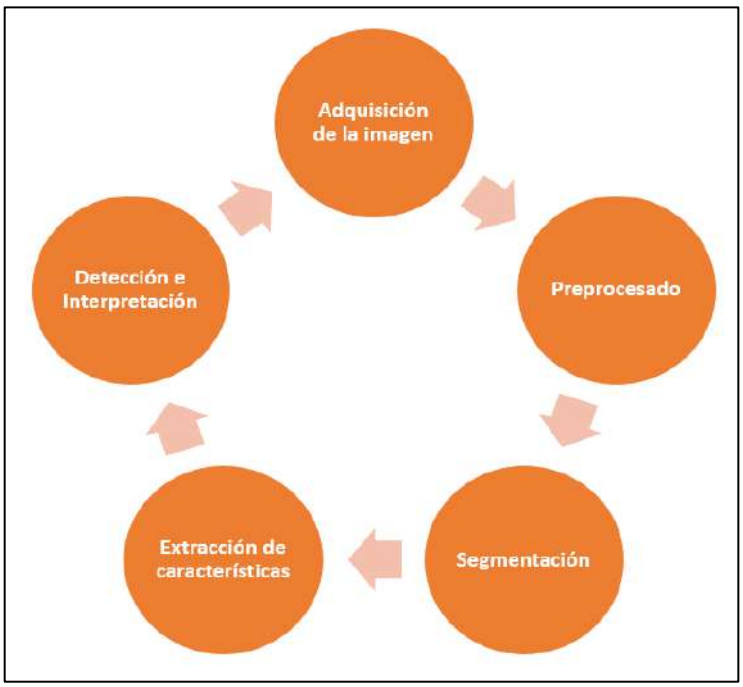


Ilustración 1 – 2: Etapas de un sistema de visión artificial.

Realizado por: Pilco, 2023.

2.1.1 Adquisición de la imagen

La primera etapa para desarrollar en un sistema de visión es digitalizar la escena de interés que se obtiene a través de una cámara digital, webcam, videocámara, escáner, entre otros. El objetivo de esta etapa es destacar las características visuales de los objetos presentes en una escena. Para ello, se emplean distintas técnicas de acondicionamiento de la escena tales como la iluminación, filtros y texturas que se analizarán en el desarrollo de este capítulo.(Vargas, 2010). Algunos de los dispositivos más utilizados para la adquisición de la imagen se analizan a continuación.

2.1.1.1 Iluminación

La iluminación es un aspecto fundamental que considerar en un sistema de visión artificial. Su variación puede generar alteraciones en las imágenes durante su proceso de adquisición. Por ello, su correcto control y aplicación puede hacer que el sistema de visión trabaje de manera adecuada.(Maduell i García, sin fecha). Existen varios sistemas de iluminación que permiten adecuar la escena para obtener mejores resultados durante su análisis. Sin embargo, escoger un modelo de

iluminación depende de la aplicación. Para el presente trabajo de investigación, de acuerdo con la aplicación del sistema de visión artificial se consideran los siguientes:

Iluminación de fondo. - Para este modelo, la fuente de iluminación se proyecta desde el fondo de la escena por detrás del objetivo. De esta manera se destacan las características del objeto sometido a la iluminación tales como orificios y contornos, pero se reducen los detalles de la superficie. Además, este modelo permite reducir sombras y minimizar la presencia de polvo, rayas e imperfecciones del objeto.(COGNEX, 2022) En la Ilustración 2 – 2 se observa un esquema de este modelo de iluminación.

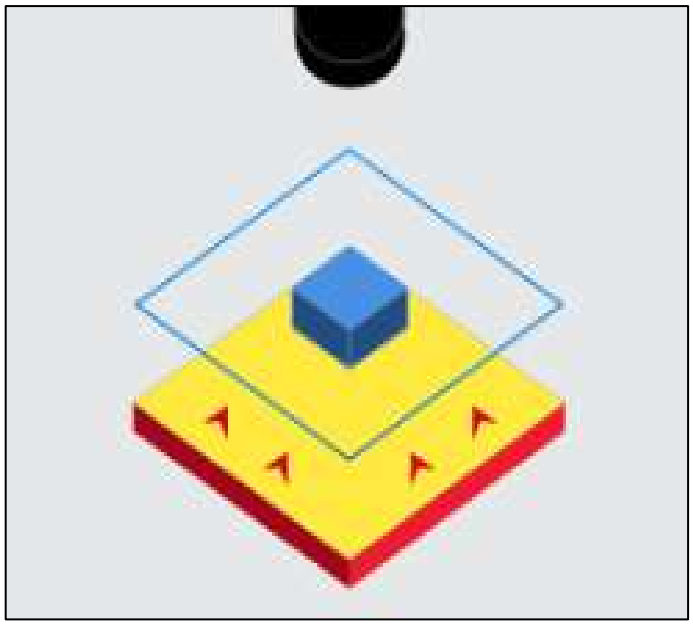


Ilustración 2 – 2: Modelo de iluminación en el fondo.
Fuente: COGNEX, 2022.

Iluminación de barras. - Se la conoce también como luz lateral. Consiste en la proyección de una línea de luz sobre el objetivo sometido a lo largo de su borde para lograr una iluminación uniforme. Esta modalidad de iluminación utiliza un barrido de la fuente de iluminación. En la figura 3 – 2 se puede apreciar el modelo. Esta técnica de iluminación puede combinarse con otros modelos para cubrir de manera adecuada el objetivo por completo desde todas las perspectivas de visualización.(COGNEX, 2022)

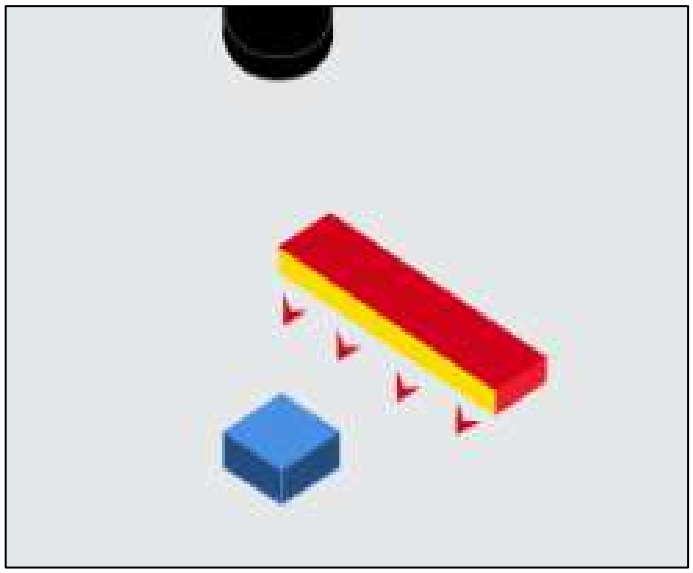


Ilustración 3 – 2: Modelo de iluminación lateral o de barras.
Fuente: COGNEX, 2022.

2.1.1.2 Dispositivos de adquisición de imágenes

Son los dispositivos encargados de capturar una imagen del escenario u objeto de interés. Estas imágenes son las que el sistema de visión artificial procesa aplicando todas las técnicas y métodos para conseguir el objetivo deseado. Los dispositivos más comunes para realizar esta actividad se detallan a continuación:

Webcams. –Son dispositivos de uso educativo y doméstico, ya que, se caracterizan por su bajo coste de fabricación y flexibilidad. Con el desarrollo en simultáneo de velocidades de interfaz, protocolos de comunicación y ópticas avanzadas, la resolución de captura varía según la necesidad en resoluciones tales como: 240p, 480p, 720p, 1080p con una cantidad considerable de FPS. (Min. de Educación, 2012)

Para tareas y aplicaciones de visión artificial, trabajan conectadas a un computador mediante el cual se pueden capturar imágenes o videos de una escena para extraer las características de los elementos que integran la imagen.(Bach. La Madrid Távara, 2019) En la Ilustración 4 – 2 se puede apreciar una webcam común.

Ilustración 4 – 2: Webcam común.

Fuente: Logitech, 2023.

Cámaras Industriales. – Las cámaras de uso industrial han evolucionado considerablemente gracias los avances tecnológicos, por esta razón, es posible clasificarlas según su campo de aplicación. De esta manera, se tienen cámaras industriales para aplicaciones de visión artificial y para fines convencionales.

Para el presente trabajo de investigación, se consideran únicamente las relacionadas a la visión artificial. Una cámara industrial cuenta con el mismo principio de funcionamiento que una webcam común. Sin embargo, pueden tener características muy puntuales que las diferencian de las mencionadas, como la tecnología que utilizan, el área de aplicación y las prestaciones generales ofrecen. Generalmente se utilizan en líneas de montaje de grandes plantas de producción con enfoques para la industria 4.0. (EDSRobotics, 2020)

Entre algunos tipos dirigidos al sector sector industrial se tiene las térmicas, que son capaces de captar longitudes de onda de entre los 900 y 14000 nanómetros. Las cámaras de alta velocidad obturación son capaces de capturar de entre 100 a 1 000 000 de frames por segundo. También pueden ser cámaras que presentan una gran cobertura sin perder resolución para aplicaciones de integración con drones.(EDSRobotics, 2020) En la Ilustración 5 – 2 se pueden apreciar algunas cámaras industriales que han sido mencionadas.

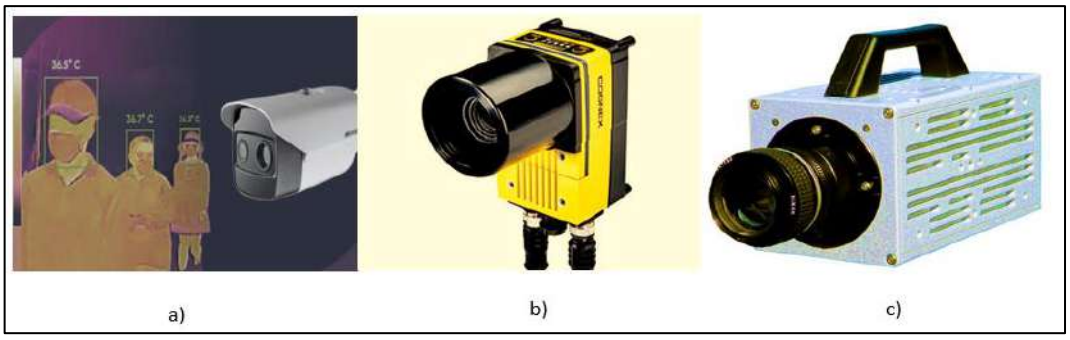


Ilustración 5 – 2: a) Cámara térmica. b) Cámara industrial. c) Cámara de alta velocidad.
Realizado por: Pilco, 2023.

Para el desarrollo de un sistema de visión artificial empleando cámaras es necesario realizar un proceso denominado calibración. Este es un proceso que consiste en determinar los parámetros intrínsecos y extrínsecos de una cámara. Los parámetros extrínsecos son aquellos que contemplan rotación y traslación para representar la localización y orientación de la cámara en un sistema de coordenadas. Mientras que, los parámetros intrínsecos corresponden a los datos propios de la cámara como el tamaño de los pixeles, coordenadas de proyección o el largo focal. (Pizarro et al. 2005)

Existen algunos modelos de calibración de cámara, tales como el Tsai o Zhang. Tsai es un modelo clásico que se basa en la medida de las coordenadas de una plantilla en 3D con relación a un punto fijo. Sin embargo, para obtener buenos resultados, se requiere una gran precisión en los datos de entrada. Mientras que, el modelo Zhang corresponde a la evolución del anterior modelo, ya que emplea los puntos situados en una plantilla 2D. En aplicaciones de visión artificial, es altamente utilizado gracias a la flexibilidad que ofrece, ya que, la precisión se mantiene a pesar de mover la cámara o la plantilla. (Viala et al. 2008)

Sistemas Integrados de Visión. – Son sistemas embebidos que se han desarrollado a la par de las tecnologías de inteligencia artificial. Ante el inminente crecimiento de las nuevas tecnologías, los sistemas embebidos tienen mucha importancia sobre otras. Los sistemas de visión integrados se caracterizan por su bajo consumo de recursos, específicamente, los energéticos. Esto es gracias a los elementos que integran el sistema, desde sus procesadores hasta las arquitecturas bajo las cuales han sido desarrolladas. Su único inconveniente es que tienden a ser sistemas costosos. (Interempresas, 2018)

Ilustración 6 – 2: Sistema integrado de visión COGNEX.

Fuente: COGNEX, 2023.

Este tipo de sistemas son altamente utilizados y pilares fundamentales en la aplicación de soluciones de Industria 4.0. Se puede apreciar un sistema de visión en la figura 6 – 2. Su contribución y aplicación permite el desarrollo y la diversificación en sectores de robótica industrial, fabricación de autos autónomos, etc. De esta manera los conceptos de fabricación y producción se orientan hacia el empleo total de sistemas inteligentes en todas sus líneas de fabricación. (Interempresas, 2018)

Análisis comparativo de los dispositivos de captura de imágenes. – En la Tabla 1 – 2 se realiza una descripción de las características más significativas de los dispositivos de captura de imagen mencionados.

Tabla 1 – 2: Características significativas de cámaras utilizadas en sistemas de visión artificial.

Características	Webcams	Cámaras Industriales	Sistemas integrados de visión
Resolución	1280 x 720	1920 x 1080	2592 x 1944
Costo	Bajo	Alto	Alto
FPS	30 – 60 FPS	55 – 1 000 000 FPS	55 – 1 000 000 FPS
Driver	PC Driver Free	Cognex VisionView PC Software	Cognex VisionView PC Software
Comunicaciones	USB	TCP/IP, UDP, FTP, RS-232C	TCP/IP, RS-232C
Peso	$\approx 80g$	$\approx 200g$	$\approx 380g$
Alimentación	5V DC	24V DC	24V DC

Realizado por: Pilco, 2023.

Fuente: CoolShark, 2023. COGNEX, 2023.

2.1.1.3 Modelos de color

Estos modelos son representaciones de color que se clasifican en función a la naturaleza de su aplicación. En este sentido, se tienen modelos sensoriales, que están destinados a dispositivos como pantallas, smartphones o cámaras. Por otra parte, existen modelos que están orientados a la interpretación biológica, es decir, a la interpretación humana, los cuales se denominan modelos perceptuales. (Sucar, Gómez sin fecha)

Modelo RGB. – Es un modelo de color sensorial basado en los colores primarios rojo, verde y azul. Está representado por un sistema de coordenadas cartesiano dentro de un cubo de dimensiones iguales a uno. De esta manera, los colores se definen por vectores de puntos que van desde el origen, ausencia total del color, hasta el punto opuesto, color blanco. (Gonzalez, Woods 2002) En la figura 7 – 2 se muestra el esquema del modelo de color RGB.

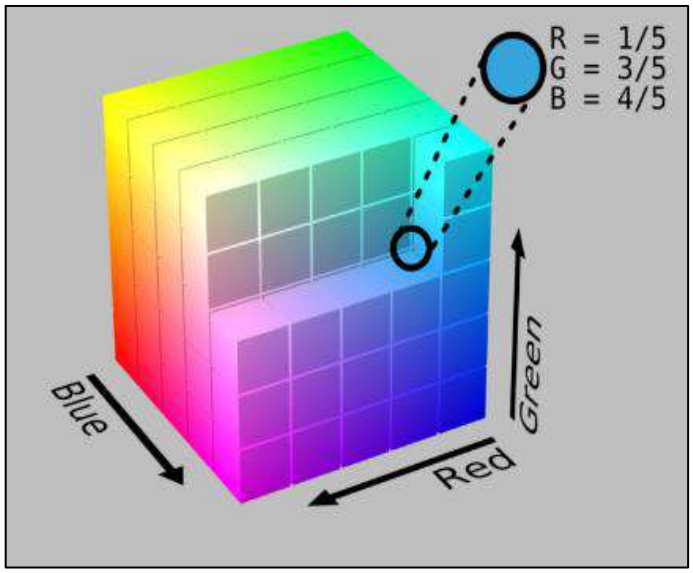


Ilustración 7 – 2: Modelo de color RGB.

Fuente: Grupo TICC, 2021.

Modelo HSV. – Este modelo es perceptual, es decir, adecuado para la percepción humana y se representa por un hexacono en tres dimensiones. Los parámetros que componen este modelo son el *tono*, *saturación* e *intensidad.* El tono denotado como 'H' se define como un ángulo entre 0 y 2π. La saturación definida como 'S' describe el nivel de pureza o profundidad en valores entre 0 y 1. Y finalmente la intensidad denotada como 'V' representa el brillo o cantidad de luz que alumbra un color en porcentajes de 0 a 100%. (Bora, Kumar Gupta, Khan 2008) En la Ilustración 8 – 2 se presenta el esquema que representa el modelo de color HSV.

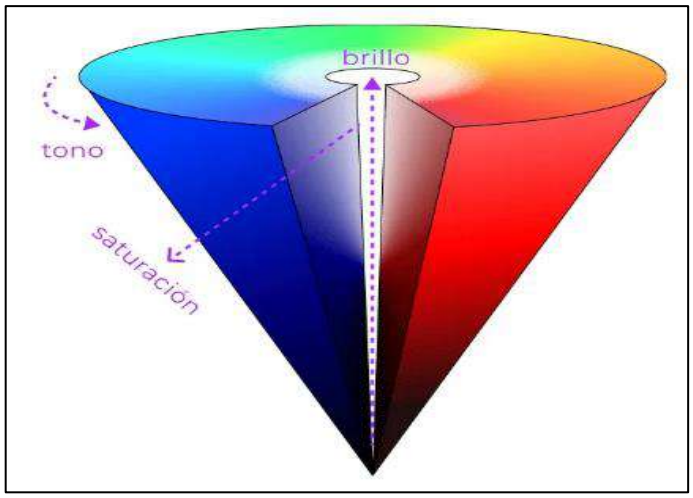


Ilustración 8 – 2: Modelo de color HSV.

Fuente: Aprenderuxui, 2023.

2.1.2 Preprocesado

Constituye la utilización de diferentes técnicas computacionales para interpretar una escena capturada. A través de este proceso, se puede solucionar alguna problemática asociada. Para el presente proyecto se han utilizado técnicas basadas en software libre mediante la utilización de la librería OpenCV. En esta se realizan tareas de eliminación de ruido, iluminación variable y realce de características en una imagen. Tras este proceso se obtiene una nueva imagen mejorada para su posterior utilización.(Bach. La Madrid Távara 2019) Algunas técnicas para el preprocesamiento de imágenes son:

- Conversión a escala de grises
- Filtrados
- Binarización

Para el desarrollo, las técnicas utilizadas para el preprocesamiento de las imágenes son el control de la iluminación y aplicación de filtrado de la imagen que se detallan en los siguientes apartados.

2.1.2.1 Filtros

Filtro de suavizado. – El funcionamiento de este filtro implica reemplazar cada píxel con la media aritmética de los puntos que lo rodean. Para su aplicación, se toma una ventana de datos N x N, donde el área se reemplaza por el área media. A continuación, se encuentra el valor medio de todos los valores de píxeles en la ventana seleccionada y el píxel en la nueva imagen se reemplaza por el valor obtenido. El efecto que proporciona equivale a un filtrado de imagen que

traza gradientes a lo largo de la imagen. Este filtro es altamente utilizado para eliminar el ruido causado por malas muestras. (Gonzáles et al. 2006)

Filtro de mediana. – Este filtro implica encontrar una lista de valores de píxeles en la ventana, ordenarlos y tomar el valor medio, es decir, un valor tiene un número igual por encima y por debajo de él. Este filtro es utilizado principalmente para eliminar el ruido de una imagen. Particularmente, el filtro de mediana elimina correctamente el ruido, sin embargo, en estructuras como puntas afiladas, no se consigue tener un correcto difuminado. (Gonzáles et al. 2006)

Filtro Gaussiano. – Con relación a filtros como el de mediana, el cual, consiste en eliminar el ruido de una imagen de entidades como bordes, el filtro gaussiano presenta un mejor resultado.(Gonzáles et al. 2006) Es mayormente utilizado para eliminar el ruido de las imágenes mediante una técnica denominada emborronar o difuminar. La expresión que lo gobierna se observa en la ecuación 1.

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{\frac{x^2+y^2}{2\sigma^2}} \qquad (1)$$

donde σ, es la desviación estándar de la distribución. Este filtro asume una mediana igual a cero para poder aplicarse sobre una imagen y conseguir el efecto de difuminado y reducción de ruido sobre una imagen. (Vives et al. 2014) En la Ilustración 9 – 2 se puede apreciar la aplicación de los diferentes filtros mencionados.

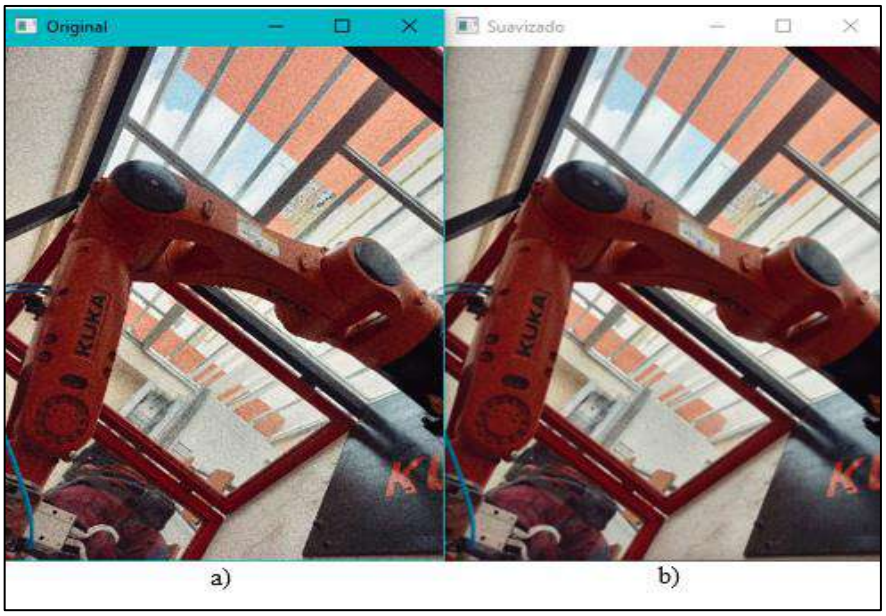


Ilustración 9 – 2: a) Imagen original. b) Aplicación de un filtro de suavizado.

Fuente: Pilco, 2023.

2.1.2.2 Binarización

Es un proceso que realiza un barrido en la matriz de la imagen de manera repetitiva mediante un bucle iterativo con el objetivo de reducir la imagen a una escala de grises de dos únicos valores, 0 y 255. En términos de color el cero representa al negro y el 255 al blanco. Esta técnica trabaja en base a un valor de umbral de sensibilidad determinado denominado threshold. A partir de este valor de umbral los valores que representan al color que sean mayores al umbral tomaran un valor de 255 y los valores menores serán negros o tomaran el valor de cero.(Magro 2013)

Una imagen binarizada se representa por medio de una matriz que contiene la información de la imagen en ceros y unos. En la Ilustración 10 – 2. se puede observar en la parte de la izquierda la matriz compuesta de ceros y unos, mientras en la derecha es representada mediante los colores correspondientes a dichos valores.

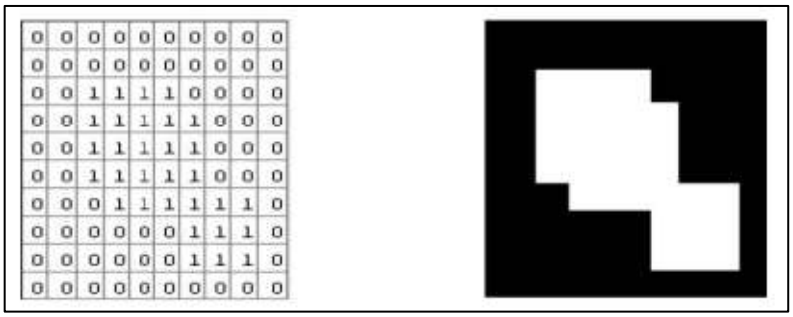


Ilustración 10 – 2: Representación binaria de una imagen.

Fuente: Cáceres, 2006.

2.1.3 Segmentación

Permite la identificación de partes significativas descompuestas en regiones correspondientes a un objeto. Estas regiones deben ser homogéneas, diferentes a las regiones adyacentes y relacionarse con los elementos del mundo real. Algunas propiedades de este proceso son la *similitud*, la cual está estrechamente ligada a la iluminación del sistema. La *conectividad*, en donde se debe evitar las oclusiones que impidan la conexión entre puntos, y la *discontinuidad* que se debe tener en cuenta en la detección de bordes. (Bach. La Madrid Távara 2019)

Idealmente trabaja a partir de la generación de una imagen binaria en la que se representa con 1 un píxel del objetivo y con un 0 los que no pertenezcan a él. Este proceso también puede basar su análisis en una característica de la imagen capturada como, los niveles de gris o texturas. A partir de estos parámetros se pueden nombrar algunos métodos de segmentación. (Bach. La Madrid Távara 2019)

2.1.3.1 Segmentación basada en transiciones

Busca aislar el objeto de interés del resto de la imagen identificando con antelación los pixeles de sus bordes, de esta manera se define el objeto y los pixeles que lo integran.(Huérfano et al. 2015)

2.1.3.2 Segmentación por crecimiento de regiones

Es un procedimiento iterativo que permite extraer regiones que son conectadas de acuerdo con un criterio predefinido como el nivel de brillo, niveles de grises, texturas, etc. Básicamente, el método se interpreta como el establecimiento de un punto semilla de forma de extraer todos los píxeles conectados a ella. (Huérfano et al. 2015)

2.1.3.3 Segmentación por umbralización

Consiste en comparar la intensidad de cada píxel de una imagen con un nivel de gris de referencia denominado como umbral. Esto permite resaltar características similares mediante la clasificación de pixeles para binarizar la imagen para separar el fondo del resto de los objetos. Para el desarrollo del proyecto se utilizó la segmentación por umbralización dentro del espacio de color HSV. La aplicación de transformaciones morfológicas, permiten realzar ciertas características y fundamentalmente bordes de la imagen para facilitar los procesos posteriores. Estas transformaciones se detallan en el siguiente apartado. (Huérfano et al. 2015) En la ilustración 11 – 2 se muestra el espacio de color HSV para la segmentación por umbralización.

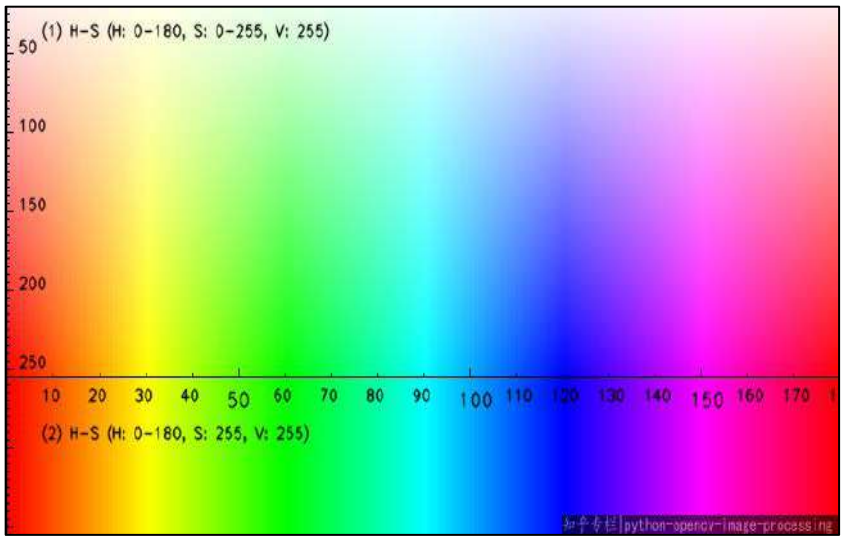


Ilustración 11 – 2: Espacio de color HSV para segmentación por umbralización.

Fuente: StackOverflow, 2023.

2.1.4 Operaciones morfológicas

Son muy utilizadas en la matemática como método para el procesamiento de imágenes. Permiten la simplificación de los datos presentes en una imagen, además, preservan las características esenciales, así como también eliminan aspectos con poca relevancia. La morfología se utiliza para tareas fundamentales en el procesamiento de imágenes tales como, supresión de ruido y simplificación de formas, destacar estructuras de objetos y la descripción cualitativa.(Cáceres, 2006)

La morfología en imágenes se aplica a partir de una imagen binarizada o discretizada a dos valores, 0 y 1 o negro y blanco respectivamente. La mayor parte de algoritmos morfológicos utilizan las operaciones de erosión y dilatación, las mismas que se encuentran en la librería OpenCV, así como con otras operaciones complementarias que tienen estas operaciones como base.(Gonzáles et al. 2006)

Dilatación. – Esta operación se representa como I+H, donde I representa el objeto y H los pixeles vecindarios. Permite aumentar el tamaño de los objetos para poder cerrar pequeños agujeros mediante la asignación de un valor de 1. Este se asigna a las partes de la imagen que contengan el objeto de análisis, a todos los píxeles vecinos del fondo que contengan al menos un píxel del objeto.(Ruiz, 2020) En la Ilustración 12 – 2 se puede observar la operación de dilatación para incrementar el tamaño de curva punteada.

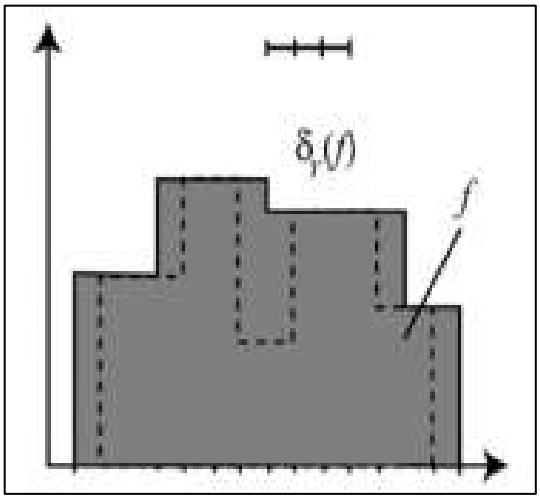


Ilustración 12 – 2: Operación de dilatación.

Fuente: Ballén. M, 2015.

Erosión. – Se representa como I–H, donde I representa el objeto y H los pixeles vecindarios. Permite eliminar todos los píxeles del objeto en cuyo vecindario haya al menos un píxel que

pertenezca al fondo. Esta técnica permite reducir el tamaño o eliminar los objetos relativamente pequeños.(Ruiz, 2020) En la Ilustración 13 – 2 se aprecia la operación de erosión para disminuir la curva punteada.

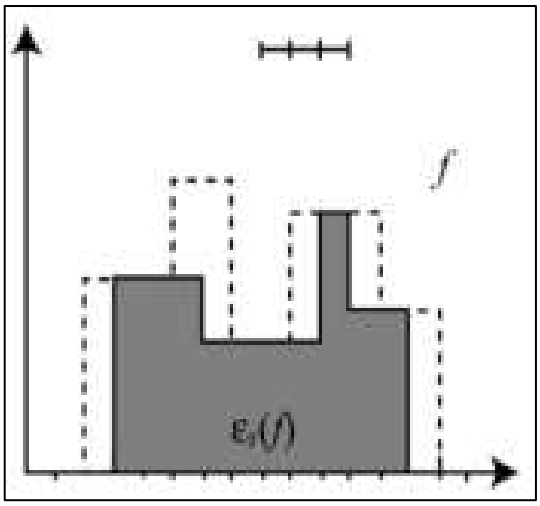


Ilustración 13 – 2: Operación de erosión.

Fuente: Ballén. M, 2015.

2.1.5 Extracción de características

El objetivo de esta etapa en el procesamiento de imágenes es poder reconocer las características fundamentales del objeto de interés tales como líneas, círculos, bordes, colores determinados o patrones. Básicamente trabaja con los pixeles de la imagen, por ello, según la aplicación a la que el sistema de visión este destinado, se pueden aplicar ciertos algoritmos que permitan reconocer los más luminosos, hacer un seguimiento del movimiento, realizar un seguimiento más sofisticado basado en un área de pixeles de un color determinado. Para el desarrollo del sistema de clasificación por detección de color se trabaja con la detección de contornos y momentos. (Maduell i García, sin fecha)

Extracción de contornos. – Para la extracción de contornos, los bordes entre los objetos de interés y el fondo se establecen mediante una discontinuidad en los valores de los píxeles vecinos. En otras palabras, los contornos se establecen como un conjunto uniforme de contornos que se pueden hallar en una porción de la imagen binarizada. Los contornos también se pueden detectar mediante la identificación de las componentes conexas del objeto. Sin embargo, para utilizar este método, se requiere un escenario en donde los objetos presenten un color uniforme y en total contraste con el fondo. (Vélez et al. 2003). En la Ilustración 14 – 2 se muestra la extracción de contornos de varios objetos en color verde.

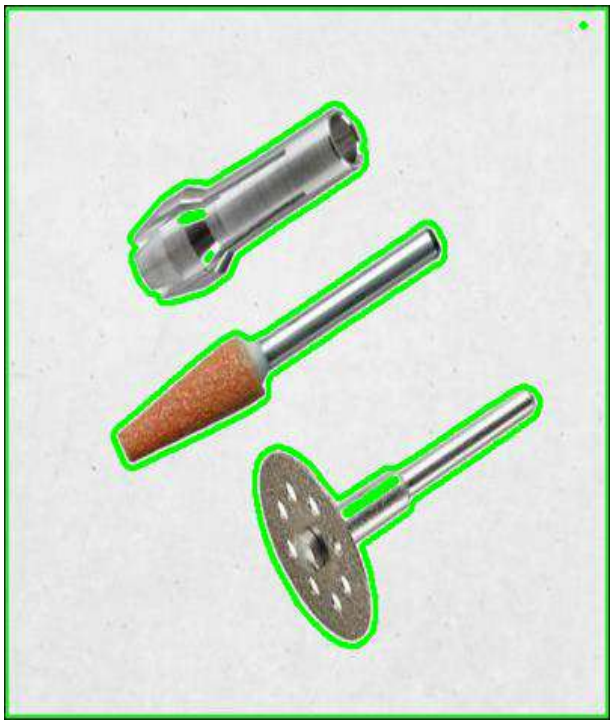


Ilustración 14 – 2: Extracción de contornos de objetos.

Fuente: C.M.A, 2018.

Momentos. – El momento de una imagen se define como una característica bruta del contorno calculado mediante la suma integral de todos los píxeles del contorno. (Bradski et al. 2008)De manera genérica, un momento (p, q) de un contorno se denota como se aprecia en la ecuación 2:

$$m_{p,q} = \sum_{i=1}^{n} I(x,y) x^p y^q \qquad (2)$$

donde, p y q corresponden al orden x, y respectivamente, por lo tanto, el orden es la potencia del componente de la suma de los n pixeles.

2.1.6 Detección e interpretación

Consiste en la clasificación del objeto de interés de acuerdo con las características obtenidas en los procesos de segmentación y extracción de características. Estas tareas se pueden realizar mediante la utilización de la librería OpenCV, ya que, permite el desarrollo de todas las tareas que componen el sistema de visión artificial. Además, es de código abierto lo cual brinda mucha facilidad tanto en código como en el acceso a todas a sus prestaciones. (Bach. La Madrid Távara, 2019)

2.1.7 Lenguajes de programación

Para el desarrollo de aplicaciones de visión artificial se necesita desarrollar su respectiva programación en un lenguaje específico. Este lenguaje provee distintas librerías y funciones para desarrollar los algoritmos, estructuras y declaraciones necesarias. Se mencionan en los siguientes apartados algunos de los lenguajes más populares y apropiados para desarrollar proyectos de visión artificial.

2.1.7.1 Python

Es un lenguaje de programación de alto nivel cuyas instrucciones están muy cercanas al lenguaje natural en inglés y se hace hincapié en la legibilidad del código. Es un lenguaje multiparadigma porque permite el desarrollo para aplicaciones en orientación a objetos, programación imperativa y programación funcional, por lo que, también se caracteriza por ser un lenguaje de propósito general. (Delgado, 2023)

2.1.7.2 C++

Es un lenguaje de programación de alto nivel que se orienta a objetos. Surge por la necesidad de simplificar el desarrollo de programas que se realizaban en lenguaje ensamblador. Las instrucciones que se utilizan para desarrollar códigos resultan más claras y potentes. En términos generales, este lenguaje se ha convertido en un estándar, ya que, de esta manera los desarrolladores pueden compartir y utilizar sus códigos en cualquier computador sin importar que tengan diferentes características o sistemas operativos.(Garrido, 2006)

Esencialmente para el desarrollo de aplicaciones y proyectos de visión artificial se requieren ciertas librerías que permitan realizar todas las etapas detalladas en los primeros apartados. Las librerías utilizadas dependen en gran manera del lenguaje de programación seleccionado para realizar los algoritmos correspondientes. Por ello, la librería OpenCV permite el desarrollo de cada actividad necesaria para realizar un sistema de visión artificial tanto en lenguaje Python como en C.

OpenCV. – The Open Computer Vision Library es una librería de código abierto que contiene estructuras y funciones en lenguaje C destinado al desarrollo de aplicaciones de visión por computador en tiempo real. Entre las aplicaciones fundamentales de la librería se pueden encontrar la interacción hombre-maquina, la segmentación, robots móviles, seguimiento de movimiento y el reconocimiento de objetos y gestos. (Arévalo et al. 2002). Las funciones de la librería están distribuidas por bloques con la finalidad de tener una orientación clara con respecto a la

aplicación a la que este destinada su utilización. En el caso del presente trabajo, la aplicación destinada es la detección, así como el seguimiento. Las funciones que ofrece la librería OpenCV son las que presentan los siguientes bloques:

- Estructuras y operaciones básicas, matrices, grafos, arboles.
- Procesamiento de imágenes, filtros, momentos, histogramas.
- Análisis estructural, geometría, procesamiento de contornos.
- Análisis y seguimiento de movimiento, plantillas de movimiento, seguidores.
- Reconocimiento de objetos, objetos propios, modelos HMM.
- Calibración de cámaras, geometría epipolar, estimación de la posición, morphing.
- Reconstrucción, detección de objetos, seguimiento de objetos tridimensionales.
- Adquisición de video e interfaces graficas de usuarios.

Algunas de las características representativas de la librería está su compatibilidad con IPL (Intel Processing Library) para mejorar el rendimiento. Además, su uso libre para uso comercial o no comercial y disponibilidad de interfaces para otro lenguajes y entornos, le dan una versatilidad importante para el desarrollo de aplicaciones de visión por computador. (Guamán, 2015)

Análisis comparativo de los lenguajes de programación para visión artificial. – Ambos lenguajes de programación presentan capacidades y propiedades únicas según los requerimientos del programa que se quiera desarrollar. En la Tabla 2 – 2 se realiza una descripción de sus características más significativas.

Tabla 2 – 2: Características representativas del lenguaje C++ y Python.

Características	Python	C++
Sintaxis	Espacios en blanco para marcar el alcance.	Requiere el empleo de llaves para marcar el alcance.
Tipificación	Se escribe de manera dinámica, conforme el tiempo de ejecución.	Es estática ya que se determinan al momento de la compilación.
Gestión de la memoria	Gestión automática mediante la recolección de elementos no utilizados.	Gestión manual mediante el uso de punteros.
Rendimiento	Sintaxis más simple de leer y escribir lo que acelera el desarrollo de los códigos.	Desarrollo de códigos más rápida por la tipificación estática.
Bibliotecas	Orientadas a la programación científica, desarrollo web y aprendizaje automático.	Orientadas a la programación a nivel de sistema.
Curva de Aprendizaje	Baja por su sintaxis simple.	Elevada por la sintaxis compleja

Realizado por: Pilco, 2023.

Fuente: Anderson J, 2019.

2.2 Sistemas de control visual

Son aquellos sistemas en los que la realimentación del sistema de control está constituida por un sistema de visión artificial que se encarga de otorgar información del sistema a controlar y su entorno. Su campo de aplicación es muy variado, ya que, puede estar orientado a tareas que comprenden desde el guiado y control de vehículos, tareas de monitoreo hasta el picking robótico. ((Pomares et al. 2009). Para esta última aplicación, los manipuladores industriales trabajan a partir del denominado visual servoing, el cual, consiste en el control de la posición del efector final de un robot relativo a un objeto utilizando la información proporcionada por un sistema de visión artificial. En esta línea, los sistemas de control visual pueden clasificar en algunos tipos que se muestran a continuación. (Galbis et al. sin fecha)

2.2.1 Tipos de control visual

Dentro del área de la robótica existen algunos modelos de control para sistemas que integren las diferentes clases de robots existentes. Para brazos robóticos que integran sistemas de visión se utilizan los siguientes modelos de control.

2.2.1.1 Control visual en lazo abierto y cerrado

Para un modelo de control en lazo abierto, las tareas de extracción de características de la imagen como el control de un robot se realizan de manera independiente, es decir, primero se procesa la imagen para realizar la detección del objetivo y luego generar la secuencia de control para determinar su posición y orientación. (Prusiel, 2018) Para poder desarrollar este proceso es necesario conocer los modelos cinemáticos directo e inverso que el manipulador utilice. El modelo de control se puede observar en la Ilustración 15 – 2.

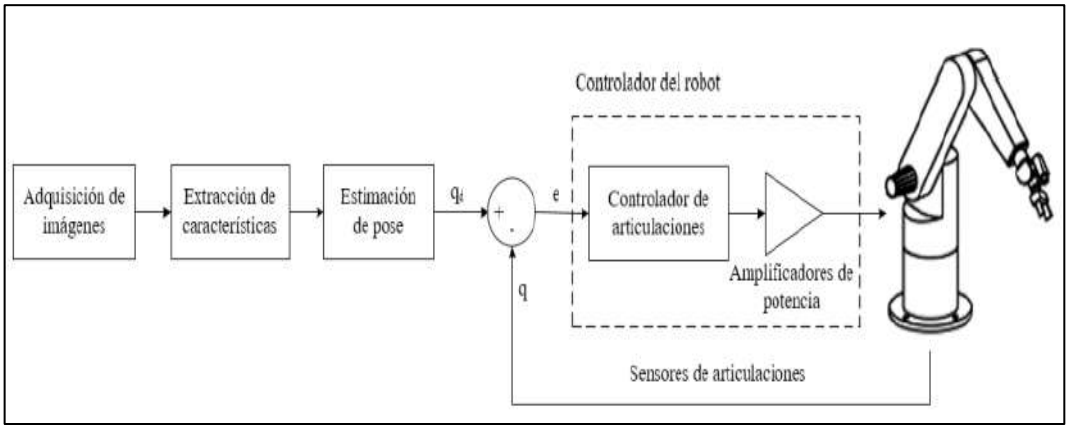


Ilustración 15 – 2: Modelo de control en lazo abierto.

Fuente: Prusiel, 2018, 10.

Mientras que un modelo de control en lazo cerrado es más utilizado actualmente, ya que, consiste en un bucle de control externo que corresponde a un sistema de visión artificial que se encarga de generar las referencias al controlador de las articulaciones de un manipulador. (Prusiel, 2018) La implantación de un esquema de control en esta modalidad permite corregir errores en la posición calculada del objetivo y modificar la trayectoria del manipulador en el caso de que existan variaciones en el espacio de trabajo. En la Ilustración 16 – 2. se puede observar el esquema de este modelo de control.

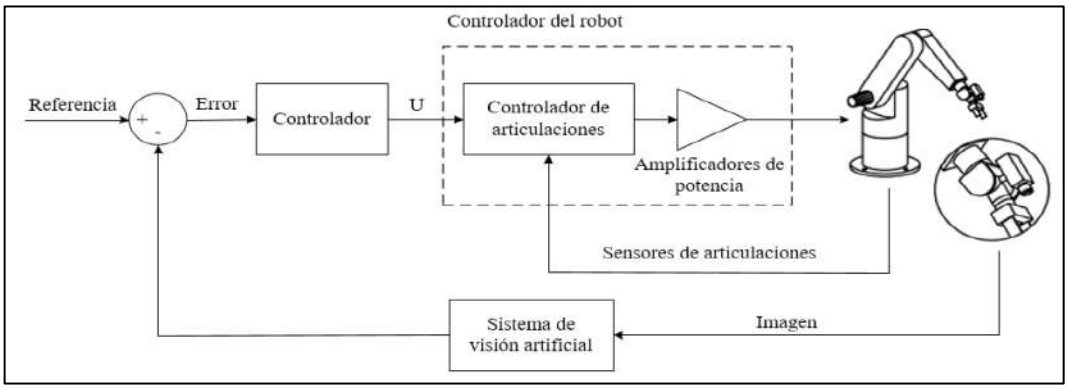


Ilustración 16 – 2: Modelo de control en lazo cerrado.

Fuente: Prusiel, 2018.

2.2.1.2 Control visual basado en imagen o posición

En un sistema de control visual la información visual se expresa mediante coordenadas en una imagen 2D o en un plano de tres dimensiones. Para el primer caso se tiene un modelo de control visual basado en imagen (IBVS), el cual, utiliza la información de la imagen en 2D para generar los movimientos hacia el objetivo deseado. Además, tiene la característica de robustez ante errores de calibración del robot.(Prusiel, 2018) El esquema de este modelo se lo puede ver en la Ilustración 17 – 2.

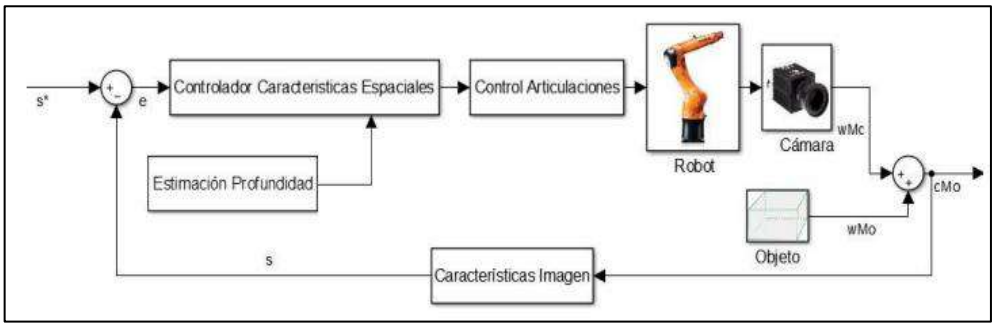


Ilustración 17 – 2: Esquema de sistema de control visual basado en imagen

Fuente: Prusiel, 2018.

Para un sistema de control visual basado en la posición (PBVS) se utiliza la información obtenida del entorno en 3D para determinar la posición y orientación del objeto con respecto al sistema de coordenadas de la cámara como se puede observar en la Ilustración 18 – 2.

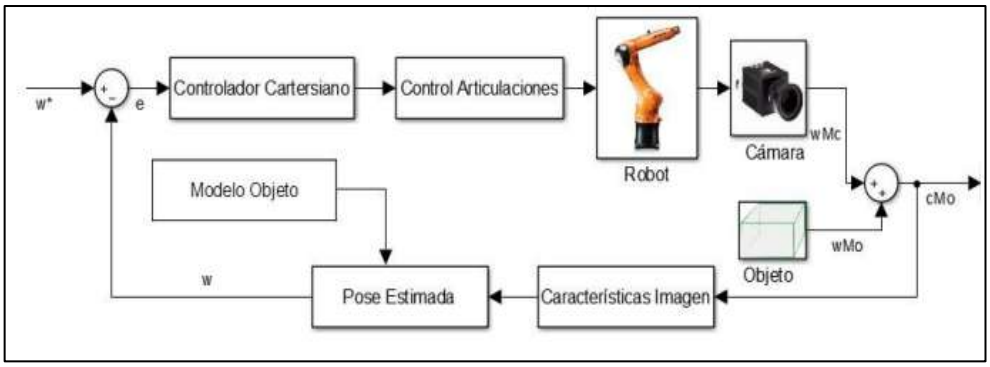


Ilustración 18 – 2: Esquema de sistema de control visual basado en la posición.

Fuente: Prusiel, 2018, 13.

2.2.1.3 Configuraciones en función del sistema de visión

Esta clasificación hace referencia a la posición en la que se ubique la cámara. Existen dos modelos, *eye-in-hand* y e*ye-to-hand* que se describen a continuación:

- **Eye in Hand. –** En esta configuración la cámara se encuentra fijada al efector final del manipulador. Por lo tanto, existe una relación permanente entre el manipulador y la posición que adopte la cámara. En otras palabras, la posición relativa entre la cámara y el objeto a manipular varia. Este modelo opera bajo la condición de que el objeto se encuentre todo el tiempo en el campo de visión de la cámara. (García, 2017) El modelo se lo puede observar en la Ilustración 19 – 2.

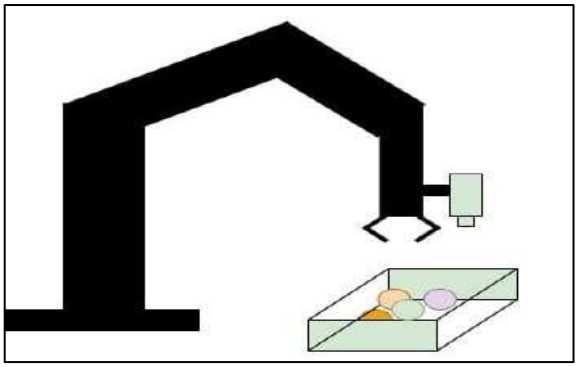


Ilustración 19 – 2: Configuración de cámara fijada al efector final.

Fuente: Control Automation, 2022.

- **Eye to Hand. –** Para esta configuración la cámara es colocada de tal forma que su campo visual permite contemplar tanto el objeto a manipular como el brazo robot. En este modelo se conoce la relación de la cámara con el sistema de coordenadas de la base del manipulador a pesar de que el movimiento del manipulador no tiene nada que ver con la cámara. (García, 2017) El modelo se lo puede observar en la Ilustración 20 – 2.

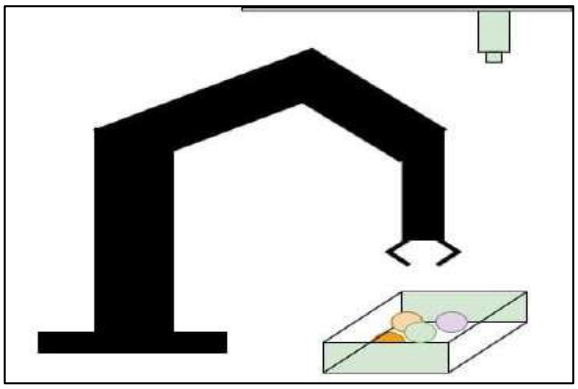


Ilustración 20 – 2: Cámara fijada en el espacio de trabajo.
Fuente: Control Automation, 2022.

2.3 Manipuladores Industriales

Según la norma ISO 8373 un robot manipulador industrial se define como: " Manipulador de 3 o más ejes, con control automático, reprogramable, multiplicación, móvil o no, destinado a ser utilizado en aplicaciones de automatización industrial. Incluye al manipulador (sistema mecánico y accionadores) y al sistema de control (software y hardware de control y potencia)."(ISO, 2022)

Esta clase de robots son muy utilizados actualmente en la industria gracias a sus características de re-programabilidad, independencia operativa y el gran aporte con la optimización de recursos al momento de realizar tares repetitivas. Sin embargo, contrario a lo que dice la definición por parte de la normativa ISO, un robot manipulador no necesariamente debe contar con al menos tres grados de libertad, ya que, de esa manera se estaría excluyendo a robots de construcciones más sencillas que siguen cumpliendo las funciones de un manipulador, pero con características más limitadas.(Barrietos et al. 2007)

2.3.1 Componentes de un robot manipulador

Los robots manipuladores industriales cuentan con una estructura básica común en todas sus variaciones independientemente de su marca y las prestaciones que puedan otorgar. Los

elementos básicos que componen un manipulador industrial son el brazo, el controlador y la unidad conversora de potencia. (Iñigo et al. 2002)

En la Ilustración 21 – 2 se muestran los componentes mencionados, además se describen a continuación:

- **Brazo. –** Es el sistema de articulaciones mecánicas, sensores y actuadores que permiten la movilidad del manipulador.
- **Controlador. –** Básicamente es un computador que recibe las señales dadas por los sensores para ejercer las respectivas acciones de control sobre el manipulador.
- **Unidad Conversora de potencia**. – Es la encargada de alimentar los motores de las diferentes articulaciones que componen el robot.

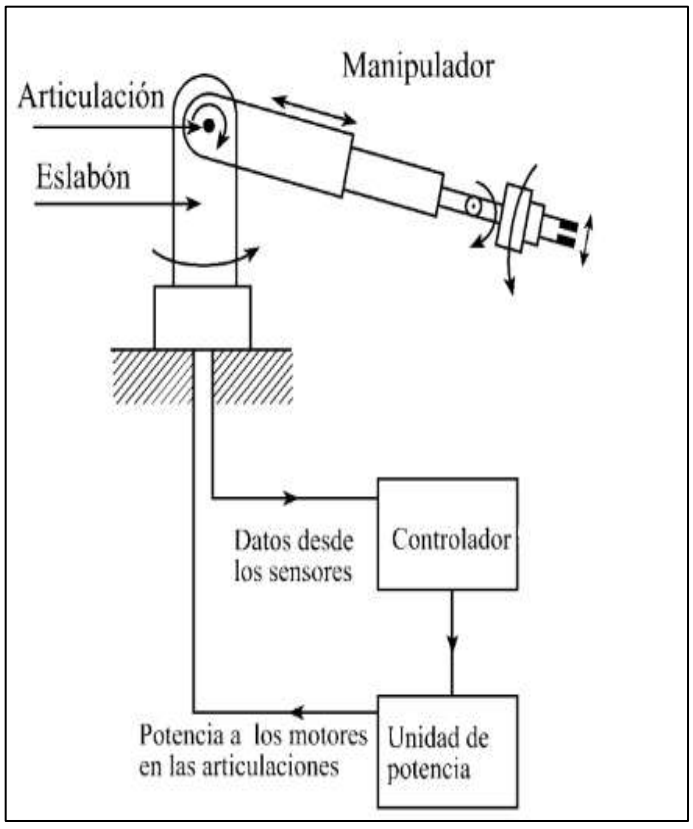


Ilustración 21 – 2: Componentes de un manipulador industrial.

Fuente: Iñigo, 2002, 2.

2.3.2 Cinemática del robot

Es la encargada del estudio del movimiento del robot con relación a un sistema de referencia sin tomar en cuenta las fuerzas que actúan sobre él. Además, se fija en la descripción analítica del

movimiento del robot en su entorno, y en específico por las distintas relaciones entre la orientación y posición de su extremo terminal. Dentro de la cinemática del robot existen dos problemas para su resolución. Estos se denominan, problema cinemático directo e inverso.(Barrietos et al. 2007)

2.3.2.1 Cinemática Directa

Consiste en determinar la orientación y posición del elemento terminal del robot, con relación a un sistema de coordenadas de referencia, a partir de conocer los valores de las articulaciones y los parámetros geométricos del robot.(Barrietos et al. 2007)

El problema cinemático directo para encontrar los valores de traslación y rotación de las articulaciones se puede resolver mediante varios métodos, dependiendo de los grados de libertad del manipulador que se esté analizando. El modelo geométrico es un modelo muy utilizado cuando los grados de libertad del robot no superan las 3 articulaciones. Cuando los grados de libertad superan este valor, el método más apropiado es el de Denavit-Hartenberg.(Ramírez et al. 2012)

2.3.2.2 Cinemática Inversa

En este método se resuelve la configuración que debe adoptar el robot para una posición y orientación del elemento terminal conocidos.(Miranda, 2016) Mediante este proceso se pueden obtener los valores que los ángulos deben adoptar para posicionarse en un punto en el espacio. Al igual que en el problema cinemático directo, existen métodos de resolución de este problema. El método geométrico es igualmente utilizado para este caso, sin embargo, es posible plantear un conjunto de ecuaciones que, mediante métodos matemáticos, se puede obtener una solución al problema para determinar trayectorias. (Ramírez et al. 2012)

2.3.2.3 Algoritmo Denavit Hartenberg

Es un método de análisis para cinemática de robots que emplea la transformación de sistemas de coordenadas mediante productos matriciales. El método consiste en la aplicación de trece pasos que componen el algoritmo de manera sistemática sobre las articulaciones del robot analizado. En los primeros tres pasos se asignan sistemas de referencia generales, mientras que, en los pasos restantes se establecen sistemas de referencia para cada articulación. La finalidad del algoritmo es obtener el valor de las variables para construir las matrices que permitirán resolver el problema cinemático directo o inverso según corresponda.(Garatejo et al. 2021) Estas variables son θ, d, a y α,

las cuales se obtienen mediante cuatro trasformaciones de rotación y traslación en los ejes X y Z que se presentan a continuación:

- θ, Rotación entorno al eje Z.
- d, Traslación en el eje Z.
- a, Traslación en el eje X.
- α, Rotación entrono al eje X.

2.3.3 Programación de robots

Los robots industriales en su gran mayoría son programados de manera guiada, es decir, el operador o programador ejecuta las instrucciones para almacenar las trayectorias punto a punto a la vez en el sistema de control del robot. Desde sus inicios, la programación de robots se ha ido desarrollando y evolucionando continuamente a la par de la construcción de estos sistemas. Desde los modelos de programación tipo CNC que estaba más orientado a maquinas o herramientas, no robots, hasta los denominados actualmente frameworks. Estos últimos ofrecen una gran variedad de posibilidades gracias a sus librerías, componentes y entornos de desarrollo. (Comíns, 2018) La programación de robots industriales se puede clasificar en dos grandes grupos: programación gestual y textual.

2.3.3.1 Programación gestual

Conocida como programación punto a punto, es un modelo que se basa en que el programador es el encargado de guiar al robot manipulador en toda la trayectoria que debe desarrollar. Su principio de funcionamiento es análogo a lo que realiza una grabadora, ya que, incluye instrucciones como de reproducir, detener, pausar, adelantar y grabar. Todas estas instrucciones permiten crear trayectorias que son almacenadas en la memoria del controlador del robot. En este sentido, todos los programas desarrollados pueden ser ejecutados en cualquier momento sin necesidad de volver a realizar la programación de los puntos. Este modelo es online, es decir, es en tiempo real y requiere estrictamente del robot físico para poder desarrollarlo. (ESNECA, 2022). Como se puede observar en la Ilustración 22 – 2 el programador es el encargado de enseñarle al robot los movimientos que debe reproducir.

Ilustración 22 – 2: Modelo de programación de gestual.

Fuente: ESNECA, 2022.

2.3.3.2 Programación textual

Es una modalidad de programación offline, es decir, el manipulador no necesita intervenir durante el proceso. El elemento estrella de este modelo es el controlador del robot. Este es el encargado de calcular las posiciones a partir de las instrucciones que el operador o programador le vaya asignando en el proceso de desarrollo.(ESNECA, 2022) En contraste con la programación gestual, las instrucciones textuales ofrecen mayor precisión para ciertas aplicaciones específicas, ya que, el procesamiento en cálculo es más robusto en los controladores del robot que se esté programando.(Comíns, 2018) En la Ilustración 23 – 2 se puede observar un ejemplo de programación textual.

```
1  DEF High_Hopes( )
2  EXT BAS (BAS_COMMAND :IN,REAL :IN )
3  ; GLOBAL INTERRUPT DECL 3 WHEN $STOPMESS==TRUE DO IR_STOPM ( )
4  ; INTERRUPT ON 3
5  Initialise and set default speed
6  ;;FOLD STARTPOS
7  $ADVANCE = 5
8  ---- Quickly skip BCO ----
9  ---- GO HOME ----
10
11 $BASE = {FRAME: X 0.000,Y 0.000,Z 0.000,A 0.000,B 0.000,C 0.000}
12 ; BASE_DATA[1] = {FRAME: X 0.000,Y 0.000,Z 0.000,A 0.000,B 0.000,C 0.000}
13 ; $BASE = BASE_DATA[1]
14 ; --------------------------
15 PTP {A1 0.00000,A2 -90.00000,A3 90.00000,A4 0.00000,A5 0.00000,A6 0.00000}
16 PTP {A1 26.51030,A2 -48.21040,A3 89.34230,A4 -0.48766,A5 48.14090,A6 26.83340}
17 LIN {X 600.500,Y -297.979,Z 269.908,A -180.000,B 0.814,C 180.000}
18 LIN {X 598.519,Y -297.979,Z 409.343,A -180.000,B 0.814,C 180.000}
19 PTP {A1 -29.94450,A2 -45.94760,A3 85.04000,A4 0.52865,A5 50.20370,A6 -30.28040}
20 LIN {X 600.054,Y 344.128,Z 301.321,A 180.000,B 0.814,C 180.000}
21 LIN {X 598.519,Y 344.128,Z 409.343,A -180.000,B 0.814,C 180.000}
22 PTP {A1 2.69807,A2 -63.48000,A3 97.33730,A4 -0.04657,A5 55.32990,A6 2.72429}
23 PTP {A1 39.86080,A2 -35.58410,A3 64.94540,A4 -0.60212,A5 60.01540,A6 40.15890}
24 LIN {X 600.433,Y -498.796,Z 274.634,A -180.000,B 0.814,C 180.000}
25 LIN {X 598.519,Y -498.796,Z 409.343,A -180.000,B 0.814,C 180.000}
26 END
```

Ilustración 23 – 2: Modelo de programación textual**.**

Realizado por: Pilco, 2023.

2.3.4 Brazo robot KUKA KR10 R900 sixx

Es un robot manipulador de 6 grados de libertad capaz de manejar una carga útil de 10 Kg. Es un robot muy flexible y versátil con una amplia variedad de aplicaciones como: ensamblaje, paletización, picking, manejo de materiales, clasificación de objetos o preparación de pedidos. (KUKA, 2022) En la Ilustración 24 – 2. se pueden observar los componentes del manipulador descrito.

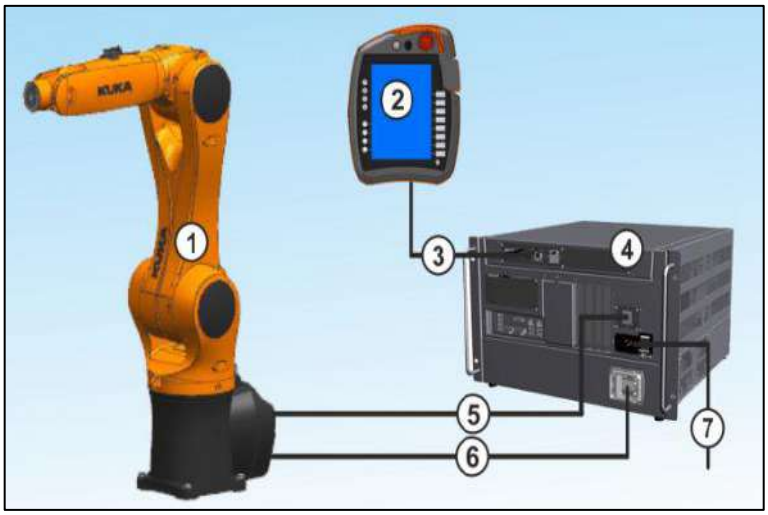


Ilustración 24 – 2: Componentes del sistema manipulador KUKA.
Realizado por: Pilco, 2023.

2.3.4.1 Componentes

El sistema de operación del robot KUKA en su modelo KR10 R900 sixx consta de múltiples componentes que en su integración adecuada permite que el manipulador pueda trabajar de manera adecuada. Esencialmente consta de los siguientes 5 componentes: manipulador de 6 ejes, SmartPad, controlador, efector final, los cuales se detallan a continuación.

Manipulador. – Es el elemento físico que cuenta con 6 ejes de movilidad que permiten la interconexión de sus eslabones formando una cadena cinemática con múltiples posibilidades. (Roboter GmbH, 2013) En la Ilustración 25 – 2 se puede apreciar el manipulador KUKA KR10 R900 Sixx con sus elementos que se enlistan a continuación:

- Base
- Columna giratoria
- Bastidor
- Antebrazo

- Brazo
- Muñeca

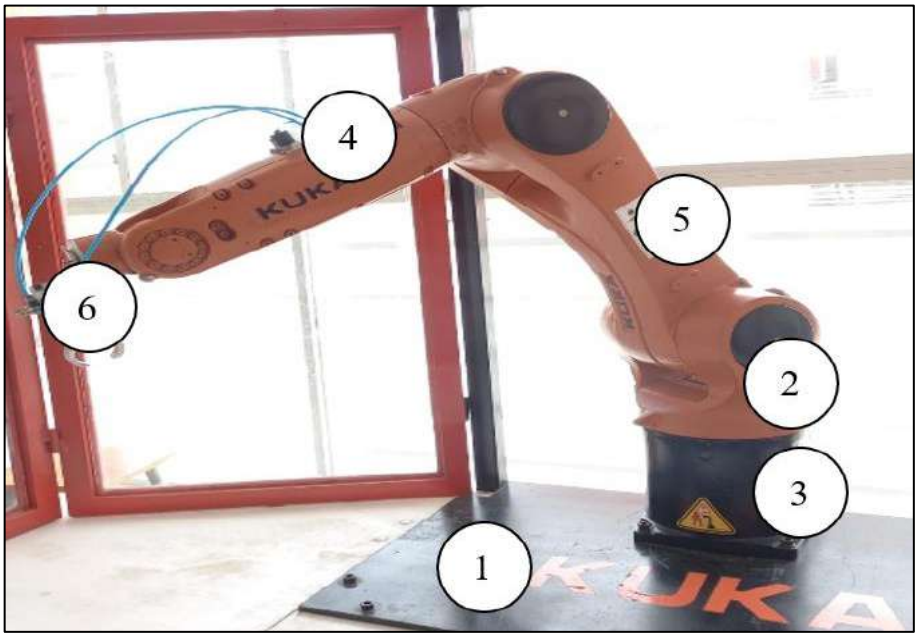


Ilustración 25 – 2: Brazo robot KUKA KR10 R900 Sixx.

Realizado por: Pilco, 2023.

SmartPAD. – Es un dispositivo físico que permite la programación manual del manipulador KUKA. Se puede apreciar el dispositivo en la Ilustración 26 – 2.

Ilustración 26 – 2: SmartPAD para programación de robot KUKA.

Fuente: Roboter Gmbh, 2015.

Este dispositivo permite realizar la programación punto a punto del manipulador. Cuenta con los siguientes elementos que han sido agrupados en 5 secciones a detalle cómo se puede ver en la Ilustración 27 – 2. Cada sección se enlista a continuación:

1. Paro de emergencia, selector de modo de operación, botón de desconexión del SmartPAD.
2. Control de cada articulación y control de velocidad.
3. Control de ejecución y edición de programas.
4. Control de apertura y cierre de la herramienta.
5. Pantalla de visualización.

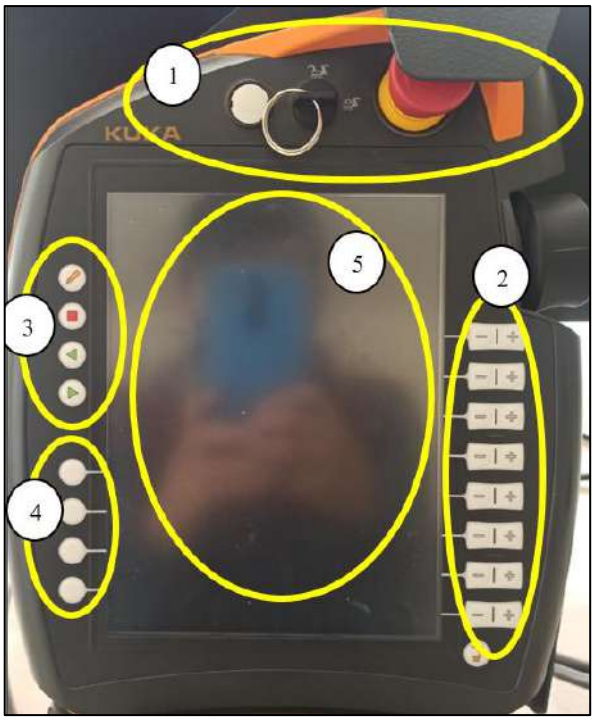


Ilustración 27 – 2: Componentes del SmartPAD.

Realizado por: Pilco, 2023.

Controlador KR C4 Compact. – Es el dispositivo encargado de realizar el procesamiento para resolver los problemas cinemáticos directos e inversos para poder realizar las trayectorias a partir de las instrucciones que el programador asigne en el SmartPAD. Se puede apreciar el dispositivo en la Ilustración 28 – 2. Este dispositivo ejecuta el sistema operativo Windows 7 que prevé una interfaz de control HMI ejecutada por el SmartPAD para el control del movimiento del manipulador. El controlador cuenta con varios puertos para la conexión del SmartPAD, conexión al brazo manipulador, alimentación, entradas y salidas para múltiples dispositivos. También incluye los diferentes puertos para establecer comunicación con sistemas externos, como ordenadores o más aplicado al campo industrial con PLCs. (KUKA Deutschland GmbH, 2018)

Ilustración 28 – 2: Controlador KUKA KR C4 Compact.

Realizado por: Pilco, 2023.

Los puertos de conexión más significativos se presentan en la tabla 3 – 2:

Tabla 3 – 2: Puertos de conexiones del controlar KR C4 Compact.

Puerto	Descripción
K1	Conector de alimentación de energía.
X11	Interfaz de seguridad.
X19	Puerto de conexión para el smartPAD.
X21	Interfaz del manipulador.
X65 EtherCAT	Puerto de conexión para esclavos EtherCAT fuera del controlador.
X66	Interfaz de línea KLI para programación a través de ordenadores externos.
X69	Interfaz de servicio KSI para diagnóstico a través de ordenadores externos.
USB	Puerto de conexión para periféricos de entrada.
DVI - I	Puerto de salida de video.

Realizado por: Pilco, 2023.

Fuente: KUKA GmbH, 2018

Efector final. – También denominados como gripper, cuya traducción es pinza, son las herramientas que el manipulador acopla como una extensión en el extremo final de su estructura. Se utilizan básicamente para realizar las distintas tareas a las que pueda estar asociada el robot manipulador. Estas tareas comprenden: sujeción, pintura, soldadura, transporte de materiales, etc. La selección de un tipo de herramienta está relacionado al tipo de material, objetos o estructuras con las que va a interactuar el gripper. (Aguilar et al. 2008) Para el presente trabajo se consideran

herramientas para sujeción y manipulación de objetos. En específico se utilizará una pinza de sujeción, la cual, puede ser de dos tipos:

- **Pinzas de dos dedos**

  Son las más comunes ya que se componen en base a un mecanismo relativamente sencillo y más utilizado con robots manipuladores industriales. Son fácilmente aplicables para tareas de manipulación y sujeción de objetos. Su accionamiento depende del escenario en donde se encuentre trabajando el robot, de esta manera, pueden ser eléctricos o neumáticos. (Revista de Robots, 2022). Existen pinzas que añaden un extra que permite manipular objeto un tanto delicados con seguridad y firmeza. Sin embargo, por el hecho de integrar un dedo adicional, incrementa su precio considerablemente con relación a una de dos dedos. En la Ilustración 29 – 2 se puede observar una pinza de dos dedos de accionamiento neumático.

Ilustración 29 – 2: Pinza neumática para sujeción de objetos.
Realizado por: Pilco, 2023.

- **Pinzas de vacío**

  Su principio de funcionamiento es muy sencillo. Cuenta con ventosas de materiales como goma o poliuretano. Este tipo de pinza generalmente es de accionamiento neumático para succionar el objeto a manipular a partir de generar vacío. Es ideal para manejar, sujetar o

movilizar desde materiales frágiles como cristales hasta planchas o superficies de características similares a las anteriores. (Revista de Robots, 2022). Dependiendo de las características del escenario en la que se utilice, puede integrar 1, 2 o más ventosas. En la Ilustración 30 – 2 se observa una pinza de vacío simple.

Ilustración 30 – 2: Pinza de vacío para brazo manipulador.
Fuente: Robotiq, 2023.

Análisis comparativo de efectores finales. – En el apartado anterior se han detallado algunos de los efectores finales más utilizados para aplicaciones de picking. Con este antecedente en la Tabla 4 – 2 se presentan las características más relevantes de cada herramienta revisada.

Tabla 4 – 2: Tabla de características significativas de las pinzas con dedos y de ventosa.

Característica	**Pinza de dos dedos**	**Ventosa simple**
Accionamiento	Eléctrico / Neumático	Neumático
Aplicación	Objetos pequeños	Objetos pequeños y de gran tamaño
Funcionamiento	Mecánica simple	Puede requerir módulos generadores de vacío
Carga	Ligera	Ligera / Pesada
Tiempo de respuesta	Rápidos	Rápido

Realizado por: Pilco, 2023.
Fuente: Revista de Robots, 2022.

2.3.4.2 Características del manipulador

Es un brazo manipulador de la familia KR10 R900 sixx. En la Ilustración 31 – 2. se puede observar las dimensiones de cada elemento que conforma el robot manipulador. (Roboter GmbH, 2015)

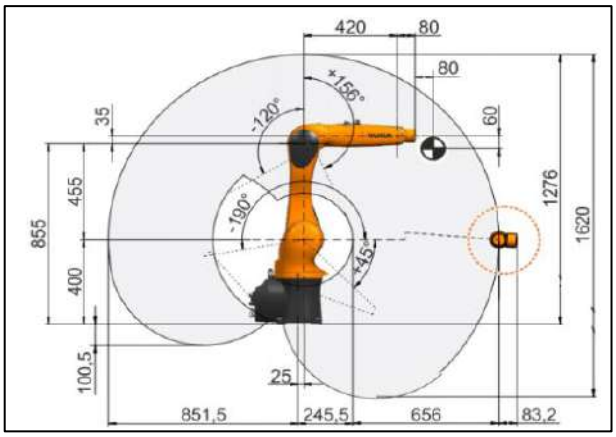


Ilustración 31 – 2: Dimensiones del Brazo KUKA KR10 R900 sixx

Fuente: Roboter Gmbh, 2015.

Los desplazamientos angulares que cada articulación puede adoptar para la ejecución de trayectorias programadas se especifican a detalle en la Tabla 5 – 2.

Tabla 5 – 2: Especificaciones de los ejes del manipulador.

Eje	Rango de desplazamiento angular	Velocidad a carga nominal
1	+/- 170º	300 º/s
2	+45º a –190º	225 º/s
3	+156º a –120º	225 º/s
4	+/-185º	381 º/s
5	+/-120º	311 º/s
6	+/-350º	492 º/s

Realizado por: Pilco, 2023.

Fuente: KUKA GmbH, 2013.

Conocidas las dimensiones y ángulos de desplazamiento del manipulador, se puede tener en cuenta el espacio en el que puede trabajar. En la Ilustración 32 – 2. se puede observar un esquema del espacio de trabajo determinado para este modelo de robot KUKA.(Roboter GmbH, 2015)

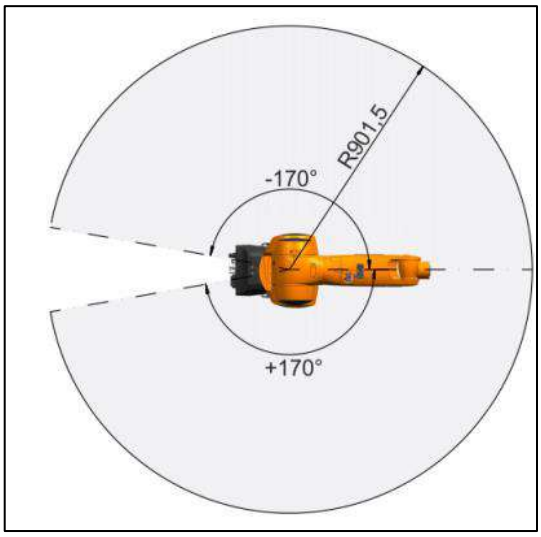


Ilustración 32 – 2: Espacio de trabajo del KUKA KR10

Fuente: Roboter Gmbh, 2015.

Algunas características adicionales son la gran precisión, ya que cuenta con una repetibilidad de 0.03 mm. Gracias a su robustez, puede ejecutar movimientos en alta velocidad en el modo de operación automático. Casi no requiere mantenimiento, ya que, sus motores mantienen una lubricación permanente para un uso continuo. (Roboter GmbH, 2015)

2.3.4.3 Descripción de Trayectorias

Las trayectorias de un robot son realizadas a partir de sus modelos cinemáticos y dinámicos. La definición del movimiento de un robot implica controlarlo de manera que siga una trayectoria preestablecida. Por lo tanto, la definición de trayectorias tiene como finalidad establecer los movimientos que debe seguir cada articulación en un periodo de tiempo tomando en cuenta ciertas restricciones físicas impuestas por los actuadores ya sean de suavidad, precisión, etc.(Miranda, 2016) Las trayectorias básicas que el manipulador KUKA KR10 R900 sixx puede desarrollar son las de punto a punto (PTP), Lineales (LIN) y circulares (CIRC). Estas instrucciones se detallan a continuación:

Movimiento Point To Point. – En este tipo de movimiento el controlador resuelve la trayectoria más rápida para llegar desde un punto inicial al punto de destino. Para ello, se accionan todos los mecanismos del robot a fin de alcanzar el objetivo deseado. En la Ilustración 33 – 2 se puede apreciar este movimiento.(Roboter GmbH, 2013)

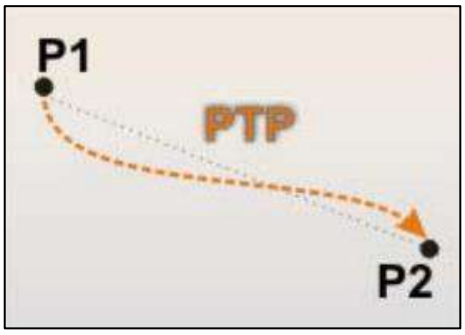


Ilustración 33 – 2: Movimiento PTP.

Fuente: Roboter GmbH, 2013.

Movimiento Lineal. – Este movimiento se ejecuta de manera rectilínea desde un punto inicial a uno final. Se caracteriza básicamente porque se desplaza a velocidad constante y la herramienta no varía su posición en ningún momento al realizar la trayectoria.(Roboter GmbH, 2013) Se puede apreciar el movimiento descrito en la Ilustración 34 – 2.

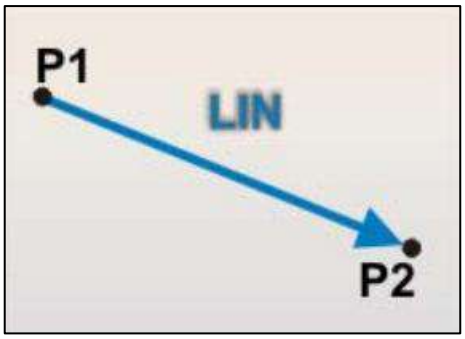


Ilustración 34 – 2: Movimiento lineal.

Fuente: Roboter GmbH, 2013.

Movimiento Circular. – Este movimiento se realiza en apoyo con un punto auxiliar intermedio entre los puntos de inicio y destino. Al igual que en el movimiento lineal, se realiza a velocidad constante y con la herramienta en una posición definida a lo largo de todo el desplazamiento.(Roboter GmbH, 2013) El movimiento se aprecia en la Ilustración 35 – 2.

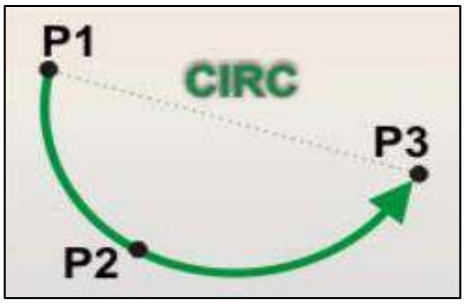


Ilustración 35 – 2: Movimiento circular.

Fuente: Roboter GmbH, 2013.

2.3.4.4 Programación Robot KUKA

Los robots del fabricante KUKA se programan en general mediante Kuka Robot Language (KRL). Este es un lenguaje de programación similar a Pascal. Permite realizar sentencias y bucles a partir de variables declaradas para ejecutar movimientos e interactuar con herramientas. KRL cuenta con diversas estructuras para controlar el flujo de programas tales como: condicionales IF y SWITCH, de bucle, FOR y WHILE y de saltos, GO TO. Los datos que maneja KRL pueden ser de tipo INT, REAL, BOOL y CHAR. Además, se pueden definir estructuras propias para robótica como AXIS y POS, variables globales para el control de velocidad o salidas como $VEL_AXIS o $OUT respectivamente. (Mühe et al. 2010) En la Ilustración 36 – 2 se aprecia un ejemplo de programación en KRL.

```
DEF example()
  DECL INT i
  DECL POS cpos
  DECL AXIS jpos

  FOR i=1 TO 6
    $VEL_AXIS[i]=60
  ENDFOR

  jpos = {AXIS: A1 0,A2 -90,A3 90,A4 0,A5 0,A6 0}
  PTP jpos

  IF $IN[1] == TRUE THEN
    cpos = {POS: X 300,Y -100,Z 1500,A 0,B 90,C 0}
  ELSE
    cpos = {POS: X 250,Y -200,Z 1300,A 0,B 90,C 0}
  ENDIF

  INTERRUPT DECL 3 WHEN $IN[2]==FALSE DO BRAKE
  INTERRUPT ON 3

  TRIGGER WHEN DISTANCE=0 DELAY=20 DO $OUT[2]=TRUE
  LIN cpos
  LIN {POS: X 250,Y -100,Z 1400, A 0,B 90,C 0} C_DIS
  PTP jpos

  INTERRUPT OFF 3
END
```

Ilustración 36 – 2: Ejemplo de programación de robot en lenguaje KRL.

Fuente: Mühe et al, 2010.

Otra forma de programación de los robots KUKA es mediante softwares que permiten la simulación de movimientos y escenarios antes de ser probados en una implementación física. Entre los softwares más comunes para la programación de robots KUKA se tienen:

Gazebo. – Es un software de código abierto, que permite el modelado de robots y entornos generados mediante una caja de herramientas con varias bibliotecas de desarrollo y servicios en la nube que ofrecen una infinidad de posibilidades. Además, permite el diseño de entornos realistas de alta fidelidad, los cuales se pueden programar mediante los diferentes plugin que integra. (Gazebo, 2023) En la Ilustración 37 – 2 se puede observar el logotipo del software Gazebo.

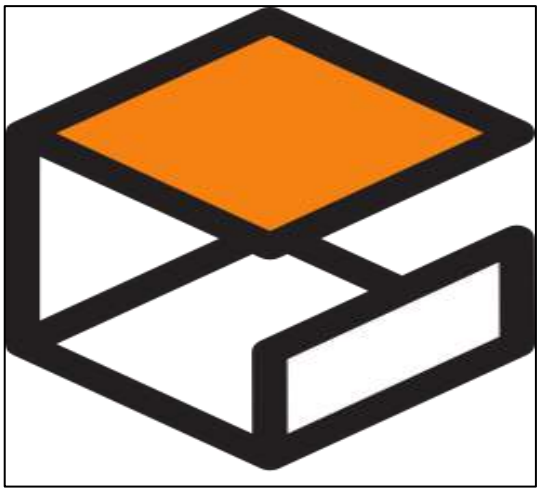


Ilustración 37 – 2: Software de simulación Gazebo.

Fuente: Gazebo, 2023.

KUKA.SIM. – Es un software propietario de KUKA que permite realizar la programación y simulación offline de robots KUKA y escenarios que los integren. Permite reproducir un gemelo digital de un manipulador, así como, la ejecución de sus movimientos. Además, permite la ejecución de programas de robot en tiempo real en sincronía con el robot físico.

Entre las ventajas que ofrece el software está el ahorro de tiempo al realizar una implementación, debido a que se pueden simular los entornos a reproducir y corregir errores antes de hacerlo. La seguridad en la planificación es un rasgo importante ya que se optimiza recursos al máximo en el momento de realizar los diseños antes de la implementación. (KUKA, 2023) En la Ilustración 38 – 2 se puede apreciar el entorno del software KUKA.Sim.

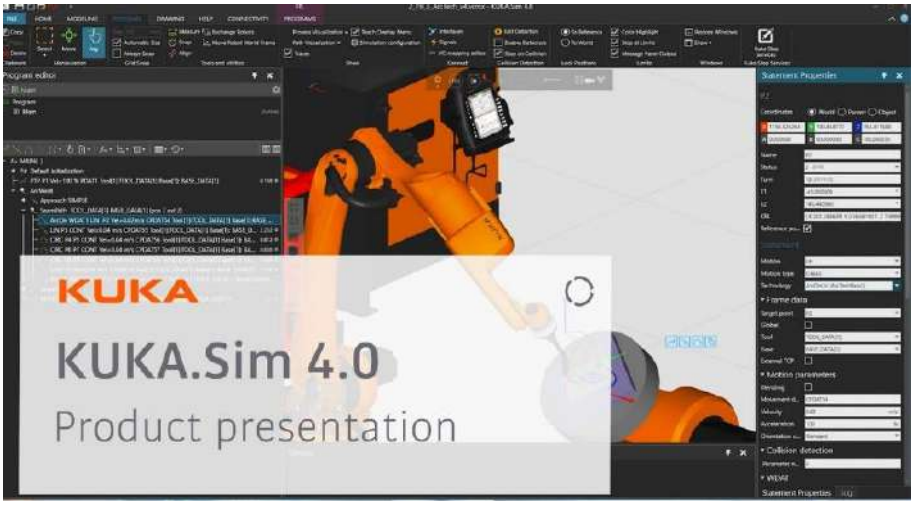


Ilustración 38 – 2: Software de Programación y simulación KUKA.Sim.

Fuente: KUKA, 2023.

RoboDK. – Es un software de desarrollo que permite programar y simular robots de las marcas más reconocidas en el mundo de la robótica. Cuenta con una basta librería con la mayoría de los modelos de robots de las marcas KUKA, ABB, FANUC, Mitsubishi, etc. Los programas de simulación de un robot se pueden desarrollar desde un ordenador sin necesidad de que el robot físico esté presente. A este tipo de programación se lo conoce como offline. El software también permite la programación online, es decir, las simulaciones de programas de robot se pueden ver en sincronía con el robot físico. (aRoboDK, 2023) En la Ilustración 39 – 2 se puede observar el entorno de programación del software.

Ilustración 39 – 2: Entorno de programación de RoboDK.

Fuente: RoboDK, 2023.

Una particularidad que ofrece el software es la posibilidad de desarrollar programas de robot en lenguaje Python tanto para la programación offline como online. En la función de programación en línea los programas de movimiento de robot se pueden desarrollar en lenguaje Python. Esta característica permite desarrollar la aplicación a la que está orientada este trabajo, gracias a la API de RoboDK para Python, ya que se pueden integrar todas las ventajas que ofrece ese lenguaje para el desarrollo de proyectos con visión artificial.(RoboDK, 2023b)

Análisis comparativo de softwares de simulación de robots. – En el mercado se encuentran múltiples opciones en software para la simulación de robots manipuladores y de múltiple propósito. Cada uno presenta funcionalidades y características únicas que permiten el desarrollo de proyectos de distinta naturaleza. En la Tabla 6 – 2 se muestran las características más importantes de cada software mencionado.

Tabla 6 – 2: Características de los softwares de simulación Gazebo, KUKA Sim y RoboDK.

Características	Gazebo	KUKA.Sim	RoboDK
Programación	Textual	Textual	Textual / Gestual
Licencia	Código abierto	De pago	De pago / Gratuito
Compatibilidad	Múltiples marcas	KUKA	Múltiples Marcas
Portabilidad	Linux / MAC / Windows	Windows	Windows

Realizado por: Pilco, 2023.

2.3.4.5 Comunicaciones

Para la programación de cada marca de manipuladores industriales, cada una maneja sus propios protocolos de comunicación y lenguajes de programación. Sin embargo, también cuentan con la integración de estándares en cuanto a comunicaciones se refiere. Todos los manipuladores industriales integran protocolos de comunicaciones como TCP/IP y protocolos propietarios. En la mayoría de los casos, para establecer una comunicación entre un manipulador industrial y un ordenador u otro controlador externo, se requiere de una comunicación cliente - servidor mediante la aplicación de sockets, con el controlador del robot mediante el protocolo TCP/IP.

Sockets. – Los sockets permiten crear comunicaciones entre un cliente y un servidor a través de una red. Se utilizan con sistemas de peticiones o de llamadas denominados como APIs. Los sockets trabajan bajo dos componentes fundamentales, una dirección IP y un puerto de comunicación. Es una arquitectura que consiste en la aplicación distribuida de los servicios que ofrecen los proveedores conocidos como servidores y los clientes que son los encargados de realizar las peticiones de un servicio. Generalmente es una arquitectura ampliamente utilizada en sistemas operativos multiusuario distribuido por medio de computadoras. (Fúquene, 2011)

Controladores de Robot. – En el presente trabajo se utiliza un manipulador del fabricante KUKA. Por ello, entornos de simulación analizados en este documento como RoboDK, ofrecen un controlador de robot basada en una API para la programación de este tipo de brazos manipuladores. Esta API permite la programación en lenguaje C# o en Python. Además, se utiliza Python con la librería robolink, que permite la comunicación entre Python y RoboDK. Estos controladores trabajan mediante una interfaz de software para controlar el movimiento del robot.(RoboDK, 2023b). Los controladores que se pueden utilizar en el entorno de simulación de RoboDK para modelos de robot del fabricante KUKA son:

- KUKAVARProxy
- C3 Bridge

Protocolo TCP/IP. – Es un protocolo originado en los años 80 adoptado por organizaciones, universidades y centros de investigación militar generalmente para la utilización de correo electrónico, pero con la característica de que podía ser utilizado en cualquier computadora conectada a internet. Este protocolo define todas las reglas que permiten la comunicación entre computadoras de diferentes marcas y tecnologías diferentes. El protocolo TCP/IP está formado por cinco capas estructurales: Aplicación, trasporte, internet, físico y red. Cada nivel se encarga de realizar una tarea para poder establecer comunicaciones mediante la aplicación de este protocolo. (Estrada, 2004)

2.4 Aplicaciones

El robot industrial KUKA KR10 R900 sixx está destinado a la manipulación de objetos, piezas, herramientas y accesorios, procesar y transferir componentes o productos. Gracias a estas funcionalidades el KUKA KR10 se lo puede aplicar en tareas de Picking, paletización, carga y descarga de piezas, mecanizados, manipulación de piezas, medición, ensamblaje de partes, etc.(KUKA, 2022)

2.4.1 Picking

Es el proceso por el cual un sistema combinado de robótica y visión artificial realiza tareas de recolección, clasificación y transporte de objetos. Para desarrollar estos procesos el sistema necesita captar imágenes que permitan determinar la posición de un objeto para posteriormente tomarlo según la delicadeza que se requiera para el trabajo y reubicarlo. (Gutiérrez et al. 2015)

Si bien es cierto el hecho de tomar un objeto de un lugar y recolocarlo, no parece ser una tare muy complicada, sin embargo, desde el punto de vista tecnológico representa una combinación de sistemas complejos tales como, reconocer las piezas a clasificar sin importar su posición y reconocer la distancia para adaptarse a diferentes ambientes de trabajo.(Gutiérrez et al. 2015) Por lo tanto, el proceso de picking orientado a la clasificación consta de las siguientes tareas:

- Trazar rutas que optimicen el tiempo de ejecución.
- Mover el brazo robótico.
- Tomar el objeto sin dañarlo.
- Reubicar el objeto en el lugar que corresponda.

El paletizado de cajas es una de las aplicaciones de picking robótico. En la Ilustración 40 – 2 se puede observar la tarea mencionada.

Ilustración 40 – 2: Paletizado de cajas con un manipulador robot.

Fuente: Mecalux, 2022.

CAPÍTULO III

3 MARCO METODOLÓGICO

En este capítulo se describen las especificaciones correspondientes a los bloques necesarios para el desarrollo del sistema de clasificación de objetos utilizando el brazo robótico KUKA KR10 R900 Sixx mediante un sistema de visión artificial, tanto en hardware como en software. Se detallan las distintas etapas de cada bloque, componentes utilizados, características, esquemas y especificaciones generales del diseño del sistema.

3.1 Descripción general del sistema

El sistema se compone de un brazo manipulador industrial KUKA KR10 R900 Sixx encargado de trasladar los objetos, y de una aplicación de visión artificial encargada de procesar la escena capturada por la webcam para detectar el color y la posición de los objetos para clasificar. El espacio de trabajo se describe como una forma irregular delimitada por una estructura de protección integrada con el manipulador. Se puede apreciar un esquema del espacio de trabajo disponible en la Ilustración 1 – 3.

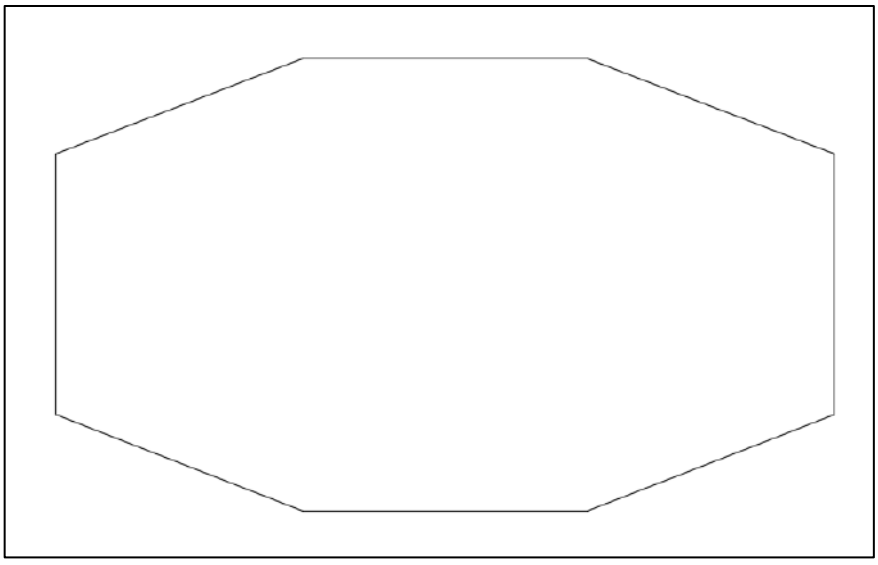


Ilustración 1 – 3: Espacio de trabajo delimitado por la jaula de seguridad.
Realizado por: Pilco, 2023.

El funcionamiento del sistema se realiza a partir de una plantilla dividida en tres secciones. La primera sección en el lado izquierdo, tomando en cuenta el frente del robot, es el espacio en donde

se colocan los objetos detectados como rojos. De manera análoga con la misma referencia del robot, la sección al lado derecho es donde se colocan los objetos detectados como azules. Finalmente, la sección central cuenta con una matriz de 3 x 3 con espacios definidos para la colocación de 9 objetos que se posicionan de manera aleatoria para el proceso de clasificación. En la Ilustración 2 – 3 se puede apreciar el esquema de la plantilla descrita.

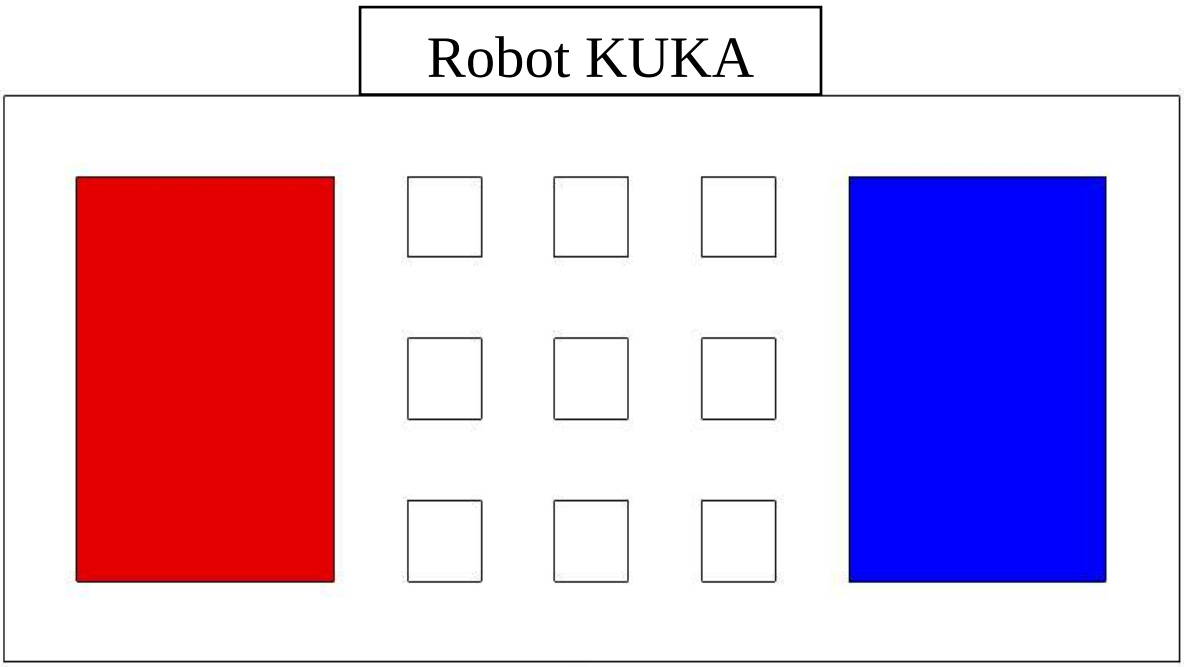


Ilustración 2 – 3: Plantilla de trabajo para el sistema de clasificación.
Realizado por: Pilco, 2023.

Para que el sistema pueda integrar los distintos bloques tanto de software con el sistema de visión para la detección del color, como de hardware para el control del movimiento del robot, se requiere una conexión física y vía software. La conexión física es punto a punto entre el controlador del robot y ordenador, mientras que la conexión a nivel de software emplea RoboDK. En este sentido, RoboDK es utilizado como una interfaz de comunicación para establecer la conexión, a su vez, permite desarrollar el programa de control de movimiento y visión artificial para ejecutar el sistema propuesto para la investigación. En la Ilustración 3 – 3 se aprecia un esquema general del sistema.

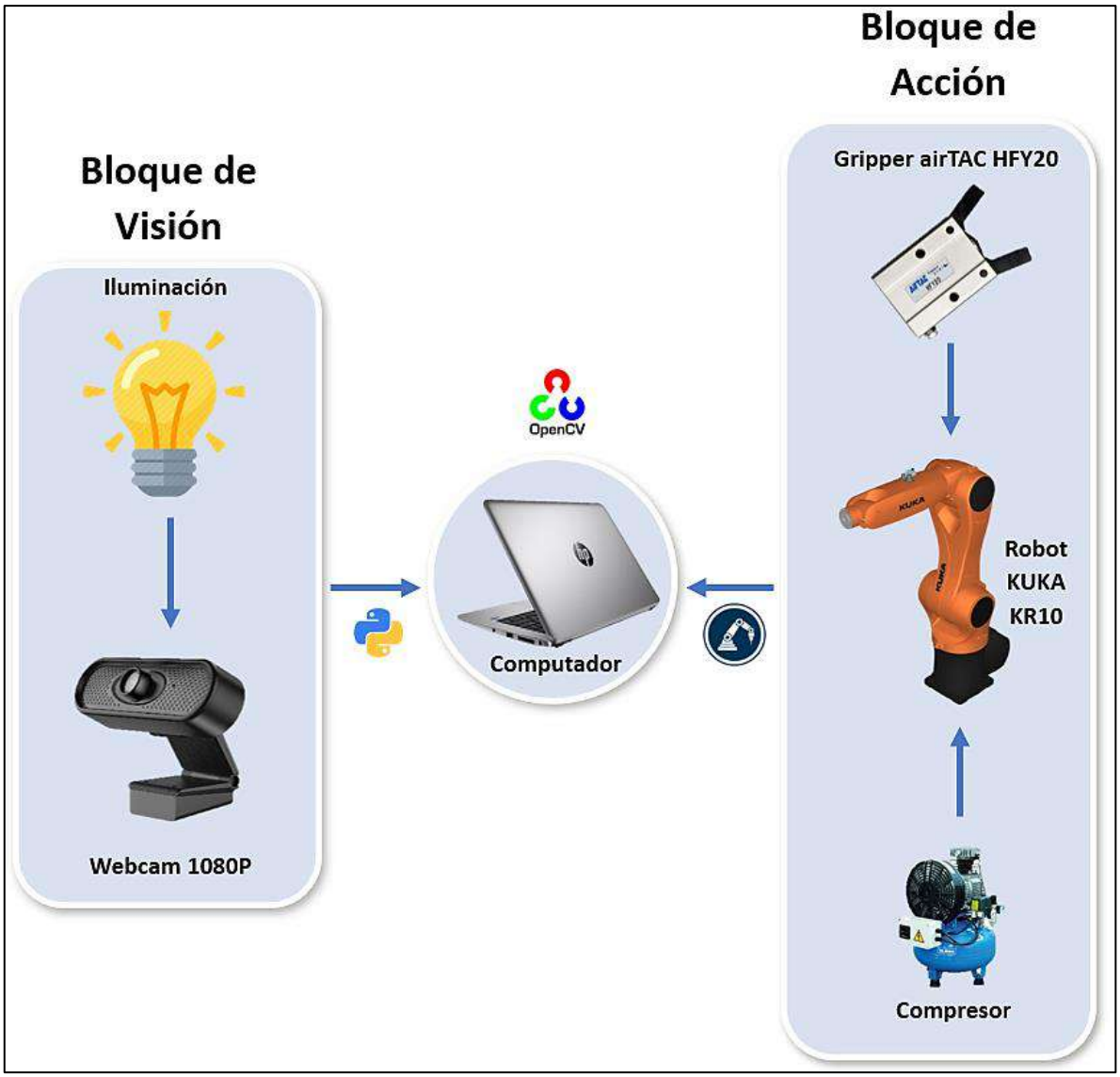


Ilustración 3 – 3: Esquema general del sistema de clasificación.

Realizado por: Pilco, 2023.

3.2 Parámetros y requerimientos del sistema

En este apartado se detallan los requerimientos que el sistema de control para la clasificación de objetos debe cumplir de acuerdo con el planteamiento del problema. Con base en la bibliografía revisada se plantean los siguientes parámetros y requerimientos en software y hardware para el diseño del sistema.

- El sistema está diseñado para la clasificación de objetos de características específicas utilizando un brazo robótico KUKA de la familia KR10 R900 Sixx mediante visión por computador. Se pueden ver las características de los objetos en la Tabla 1 – 3.

Tabla 1 – 3: Tabla de características de los objetos a manipular.

Características	Definiciones
Material	Madera
Color	Rojo / Azul
Figura	Cubo Regular
Dimensiones	5 cm
Peso	40 g

Realizado por: Pilco, 2023.

- El sistema se conforma a nivel de hardware por el robot manipulador, el ordenador y la cámara. El último componente mencionado se ubica de tal forma que se tenga un campo de visión de la sección central de la plantilla de trabajo.
- A nivel de software, el sistema establece la comunicación entre el sistema de visión y el robot manipulador empleando el servidor kukavarproxy.
- Los elementos utilizados para la fijación de la cámara se colocarán en la estructura de protección que encierra al robot.
- Los recursos energéticos para el robot son provistos por la red eléctrica del laboratorio mientras que la cámara es alimentada vía USB por el ordenador.
- El sistema de visión monitorea el área de trabajo todo el tiempo, sin embargo, la detección del color y posición se realiza cada vez que se completa el transporte de un objeto.
- Funcionamiento inteligente del sistema de clasificación, es decir, sin importar el color o la ubicación de un objeto sobre la plantilla, el sistema clasificara los objetos de acuerdo con su color característico en las secciones descritas.
- El sistema captura 10 frames cada vez que se completa el traslado de un objeto.
- El sistema de visión procesa la escena capturada para extraer características de color y posición mediante la aplicación de diferentes técnicas de visión artificial.
- Determinación de la posición de los objetos con respecto a la cámara en la plantilla de trabajo.
- Agrupación de los datos de posición e identificadores de color para realizar la clasificación de los objetos.
- Control del gripper para poder manipular los objetos colocados en la plantilla de trabajo.
- Realizar la operación de pick and place para clasificar los objetos adecuadamente.
- Repetir el proceso exitosamente para los 9 objetos presentes en la plantilla de trabajo.

3.3 Arquitectura general del sistema

La concepción de la arquitectura del sistema de control se compone de dos bloques interconectados. El primero es el de visión artificial, el cual, esta subdivido en 4 etapas. El segundo bloque es el de control y creación de targets empleando el principio de cinemática inversa para que el robot asigne los valores angulares correspondientes para cada articulación en función de los objetivos definidos por la plantilla. En la Ilustración 4 – 3 se muestra el diagrama de la arquitectura general del sistema.

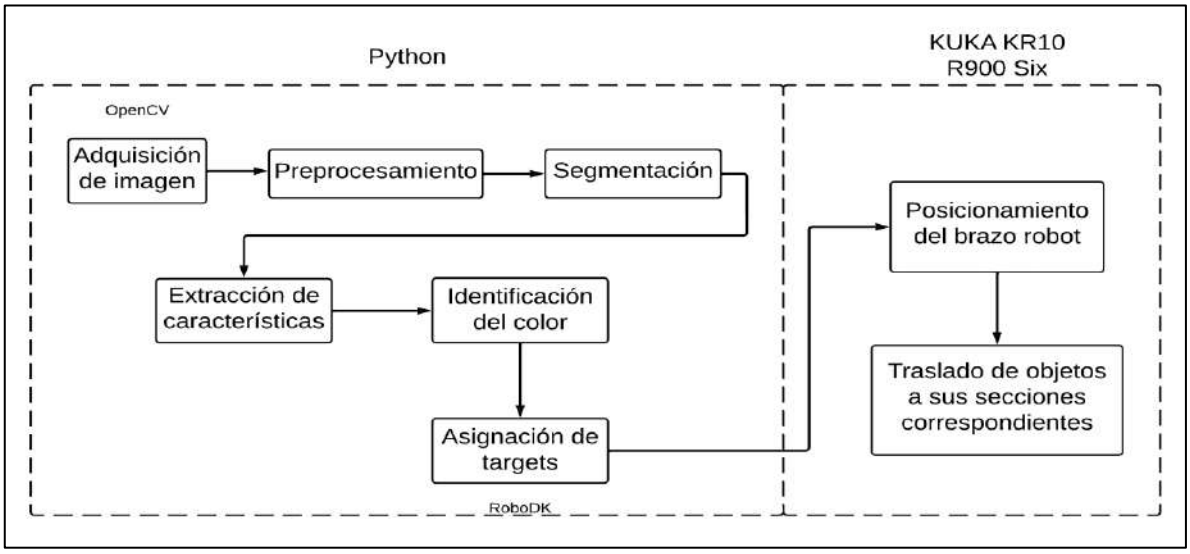


Ilustración 4 – 3: Diagrama de la arquitectura del sistema.

Realizado por: Pilco, 2023.

Bloque de visión. – Tiene como objetivo procesar la escena capturada del espacio de trabajo para determinar las características de los objetos que se clasificaran empleando distintas técnicas de visión artificial. Este bloque realiza las siguientes tareas:

- Adquisición de la imagen.
- Preprocesado
- Segmentación
- Extracción de características

Bloque de control. – En este bloque se crean los objetivos denominados targets para cada posición de la plantilla que posteriormente permiten la ejecución de las trayectorias utilizando el principio de cinemática inversa. Estos targets se asocian a las posiciones registradas en la última etapa del bloque de visión. Además, se ejerce el control del gripper para la manipulación de los objetos.

Comunicaciones. – Es el medio a través del cual se interconecta el sistema de visión (software) con el robot físico (hardware) utilizando el software RoboDK, mediante una conexión punto a punto entre el controlador del manipulador y el ordenador bajo el protocolo TCP/IP. Cuando se ha realizado la conexión se ejecuta el programa de clasificación que permite mover el robot gracias al controlador de RoboDK para robots KUKA.

3.4 Descripción y selección de dispositivos del sistema

En este apartado se presentan y definen los elementos hardware y software seleccionados para ejecutar la implementación del sistema de control para la clasificación de objetos utilizando un manipulador robótico mediante visión artificial. Además, se detallan las principales características y parámetros de operación de los dispositivos y componentes que fueron considerados para el diseño del sistema.

3.4.1 Componentes Hardware

3.4.1.1 Cámara

Tomando en cuenta la revisión bibliográfica realizada en el capítulo II y en base con las características de la Tabla 1 – 2 se determinó que el dispositivo ideal para la captura de imágenes es una webcam. El dispositivo seleccionado es del fabricante CoolShark ya que cuenta con las características óptimas para la aplicación de visión desarrollada. En la Ilustración 5 – 3 se puede observar el dispositivo utilizado para desarrollar el sistema de detección del color.

Ilustración 5 – 3: Webcam FHD empleada.

Fuente: CoolShark, 2023.

Las especificaciones técnicas de la webcam seleccionada se aprecian en la Tabla 2 – 3.

Tabla 2 – 3: Especificaciones técnicas de la webcam.

Webcam FHD	
Resolución máxima	1080p
Frames Per Second	30fps
Tipo de enfoque	Foco variable
Tipo de lente	Angulo ancho
Modo de Disparo	Low Light
Micrófono	Mono
Alcance de micrófono	20 ft
Campo visual	55º
Interfaz	USB

Realizado por: Pilco, 2023.

Con relación a la **¡Error! No se encuentra el origen de la referencia.**Tabla 2 – 3, se establece q ue la Webcam FHD del fabricante CoolShark es la apropiada, su interfaz USB permite una integración fácil y rápida con el ordenador para el desarrollo de los algoritmos para la detección de color.

3.4.1.2 Iluminación

Es una variable que tiene como propósito controlar la manera en la que la webcam interpreta el objetivo. Para ello, se debe tener cuenta algunas consideraciones para la elección del mejor tipo de iluminación. (bcnvision, 2017) En la Tabla 3 – 3 se aprecian las consideraciones a tomar en cuenta para la selección del sistema de iluminación.

Tabla 3 – 3: Consideraciones para la selección del sistema de iluminación.

Consideraciones	**Descripción**
Modo de trabajo	A color
Velocidad	Nula, ya que los objetivos son estáticos
Campo de visión	Estático y fácilmente distinguible

Tipo de objetivos para iluminar	Forma definida y color en tonalidad mate
Fondo	Blanco
Característica para resaltar	Contornos y color
Duración	Constante durante todo el proceso de clasificación de los 9 cubos
Condiciones ambientales	Incidencia de luz natural muy intensa

Realizado por: Pilco, 2023.

Fuente: bcnvision, 2017.

En función de las consideraciones contempladas en la Tabla 3 – 3 el color de los objetos es fácilmente reconocible y en los colores empleados en tonalidad mate para evitar fenómenos de reflexión de la luz. Además, el lugar en donde se encuentra ubicado el sistema cuenta con una incidencia de la luz ambiental muy grande, constante e invariable. Además, se complementa con la iluminación provista por las luminarias del laboratorio. Por lo tanto, no se requiere ningún sistema de iluminación adicional al provisto naturalmente por luz ambiente y complementado por la del laboratorio. En la Ilustración 6 – 3 se puede observar el espacio de trabajo expuesto a la iluminación presente en la ubicación del sistema de clasificación.

Ilustración 6 – 3: Iluminación ambiente presente en el espacio de trabajo.

Realizado por: Pilco, 2023.

3.4.1.3 Gripper

El elemento terminal del manipulador es uno de los componentes más importantes en el sistema ya que por medio de él se pueden manipular los objetos para trasladarlos y clasificarlos. En

función de las características presentadas en la Tabla 4 – 2 y la bibliografía revisada, se ha definido una pinza de dos dedos para la manipulación de los objetos. La selección de este gripper se apoya en las características de los objetos a manipular ya que son ligeros, presentan una forma definida y son relativamente pequeños. Concretamente, se ha seleccionado una pinza de dos dedos de la marca AirTAC en su modelo HFY20. Esta pinza se puede apreciar en la Ilustración 7 – 3.

Ilustración 7 – 3: Pinza neumática para sujeción de objetos.

Realizado por: Pilco, 2023.

Como se observa en la Ilustración 7 – 3, la pinza seleccionada integra un par de extensiones en forma curvilínea con el fin de extender el alcance al momento de la sujeción. Las especificaciones del gripper elegido se presentan en la Tabla 4 – 3.

Tabla 4 – 3: Especificaciones técnicas del Gripper AirTAC HFY20.

Parámetro	**Especificación**
Tipo de accionamiento	Neumático de simple y doble efecto
Fluido de accionamiento	Aire comprimido
Temperatura de operación	-20 ~ 70 ºC
Lubricación	No requiere
Presión de operación	0.1 ~ 0.7 MPa (22 ~ 100 psi)
Torque teórico	Cerrada: 152 x P
	Abierta: 252 x P
Material	Aluminio

Realizado por: Pilco, 2023.

Fuente: AirTAC, 2023.

3.4.2 Componentes Software

3.4.2.1 Lenguaje de programación

De acuerdo con los distintos requerimientos del sistema, se ha determinado que el lenguaje de programación Python es el más apropiado para el desarrollo de los algoritmos de visión y para la creación del programa principal que integra los movimientos con la detección del color. La compatibilidad sencilla y directa con la biblioteca OpenCV permite desarrollar todas las etapas del sistema de visión gracias a las diferentes funciones que integra. Las razones para la selección de este lenguaje se sustentan en todos los beneficios que se pueden ver en la Tabla 5 – 3.

Tabla 5 – 3: Características del lenguaje de programación Python.

Beneficios	
Sintaxis simple	Permite la lectura, escritura y comprensión sencilla de los códigos escritos para el desarrollo de proyectos.
Productividad	Gracias a la sintaxis sencilla los códigos complejos se pueden desarrollar en menos líneas de código.
Versatilidad	Presenta códigos estándar que se pueden utilizar como base para el desarrollo de proyectos.
Compatibilidad	Los códigos en Python se pueden utilizar e integrar con otros lenguajes como Java, C y C++.
Soporte	Existe una gran comunidad mundial que brinda soporte de forma inmediata en el caso de presentar problemas en el desarrollo de un proyecto.
Accesibilidad de documentación	Variedad de recursos en internet como foros, videos, tutoriales y documentación en general para el desarrollo de códigos.
Multiplataforma	Se puede programar en este lenguaje en diferentes dispositivos con diferentes sistemas operativos como Windows, macOS o Linux.

Realizado por: Pilco, 2023.
Fuente: AWS, 2023.

La biblioteca principal utilizada para realizar las diferentes partes del sistema de visión es OpenCV, ya que, se utiliza para segmentar las imágenes, aplicación de filtros para mejorar la imagen, aplicación de operaciones morfológicas, el procesado en general y la extracción de las características de los objetos para clasificar. En la Ilustración 8 – 3 se puede apreciar un segmento de código desarrollado en lenguaje Python utilizando bibliotecas como OpenCV y Numpy para el establecimiento de los umbrales de detección de los colores rojo y azul.

```python
import cv2
import numpy as np
# --------- Parametros Color Rojo
rojo_bajo1 = np.array([0, 100, 20], np.uint8)
rojo_alto1 = np.array([10, 255, 255], np.uint8)
rojo_bajo2 = np.array([175, 100, 20], np.uint8)
rojo_alto2 = np.array([180, 255, 255], np.uint8)
# --------- Parametros Color Azul
azul_bajo = np.array([90, 130, 30], np.uint8)
azul_alto = np.array([140, 255, 255], np.uint8)
```

Ilustración 8 – 3: Código en Python para la segmentación de color.

Realizado por: Pilco, 2023.

3.4.2.2 Software de simulación y conexión

Para establecer la conexión entre el ordenador y el robot manipulador es necesario utilizar una conexión punto a punto bajo el protocolo TCP/IP. Esta conexión se utiliza para asociar el sistema de detección del color y el manipulador con el fin de hacer posible el proceso de clasificación propuesto. Teniendo en cuenta los requerimientos planteados, el software RoboDK presenta diferentes funcionalidades para integrar todos los elementos que intervienen en la ejecución del sistema.

El entorno de simulación de RoboDK permite crear el escenario con el que trabaja el sistema gracias a su amplia biblioteca de robots y objetos. Significativamente, el software permite la creación de programas para el movimiento de robots en lenguaje Python, lo cual facilita en gran manera el desarrollo del proyecto. Además, integra el controlador de robot Kukavarproxy, el cual, permite establecer una conexión directa con el controlador del robot físico. En la Ilustración 9 – 3 se observa la pantalla de configuraciones para establecer la conexión Software – Manipulador.

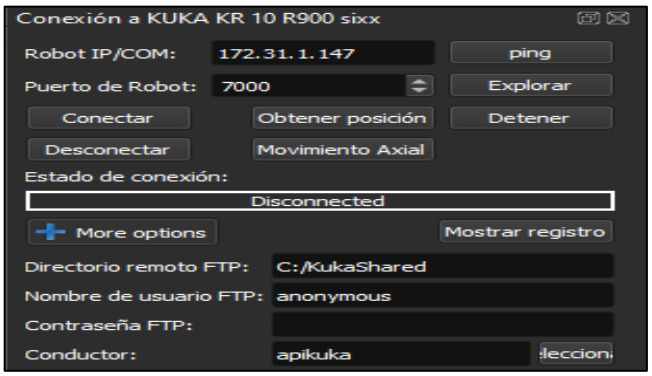


Ilustración 9 – 3: Sección de configuración de conexión con el manipulador.

Realizado por: Pilco, 2023.

En los siguientes apartados se describe a detalle el proceso y cada una de las configuraciones correspondientes para establecer la correcta comunicación entre el software y el robot. Con estos antecedentes se determina que el software RoboDK es el más apropiado para el desarrollo del proyecto.

3.5 Diseño del espacio de trabajo

En esta etapa se desarrollan múltiples tareas antes de empezar con la ejecución del sistema de clasificación para poder extraer las características de los objetos. El acondicionamiento del espacio de trabajo es un factor diferencial ya que, ante la mínima variación de las características de este, el sistema de visión podría trabajar de una u otra manera.

Como se vio en el apartado de las características del manipulador, el espacio de trabajo ideal es una esfera de 90.15 cm de radio. Sin embargo, la ubicación actual del robot está sobre una estructura de seguridad la cual limita el espacio de trabajo impidiendo la utilización de toda la esfera de movilidad posible. En la Ilustración 10 – 3 se puede observar el escenario actual del robot manipulador.

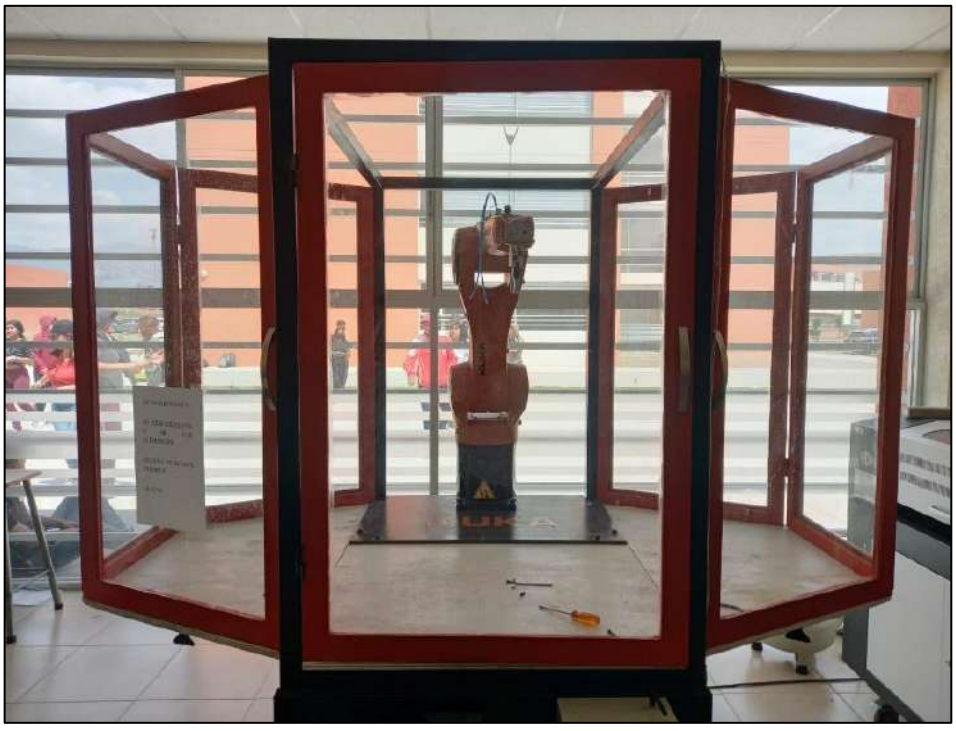


Ilustración 10 – 3: Espacio de trabajo del manipulador.

Realizado por: Pilco, 2023.

Las dimensiones de la jaula de protección son de 184 cm de altura total, 74 cm de altura de la mesa de trabajo, 102 cm de profundidad y 204 cm de ancho. Finalmente, el espacio de trabajo útil está dado en forma de un polígono irregular. Su forma, así como las especificaciones métricas se pueden apreciar en la Ilustración 11 – 3.

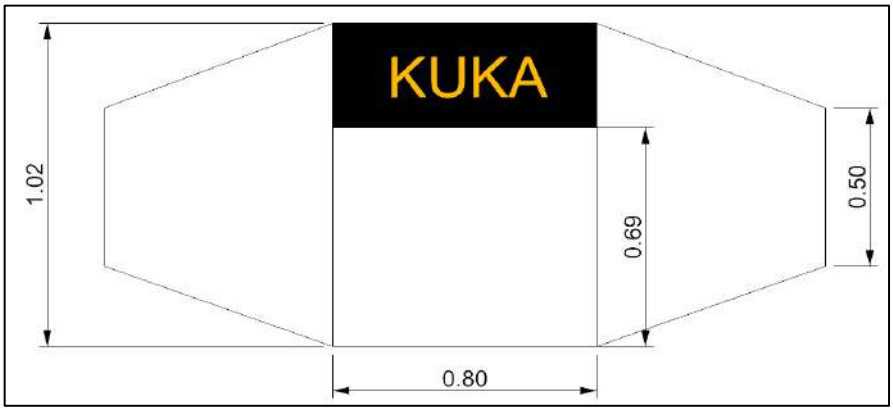


Ilustración 11 – 3: Área de trabajo delimitada por la Jaula.

Realizado por: Pilco. 2023.

3.5.1 Objetos para clasificar

Como se ha de definido en la Tabla 1 – 3, los objetos están construidos en madera por su facilidad al momento de trabajarlos para su construcción. Además, son relativamente de bajo costo y cumplen con las características de peso y forma adecuados para la manipulación con el gripper seleccionado como efector final del brazo robótico. Los objetos concretamente son cubos regulares de 5 cm de longitud tanto en su alto, ancho y profundidad. En la Ilustración 12 – 3 se puede apreciar el detalle métrico de los cubos de los objetos descritos.

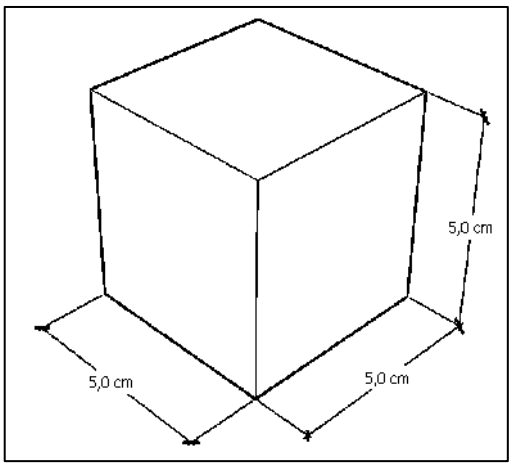


Ilustración 12 – 3: Dimensiones de los cubos**.**

Realizado por: Pilco, 2023.

Bajo esas características se han construido 12 cubos de los cuales 9 son utilizados para el sistema de clasificación. Seis de ellos se han pintado de color rojo y los otros seis restantes de color azul para colocar nueve de ellos de manera aleatoria sobre la plantilla de trabajo antes de iniciar el sistema. La construcción de doce objetos se realizó con el fin de tener mayor flexibilidad en la preparación de las diferentes configuraciones empleadas para la realización de pruebas del sistema de visión. En la Ilustración 13 – 3 se puede apreciar los cubos terminados.

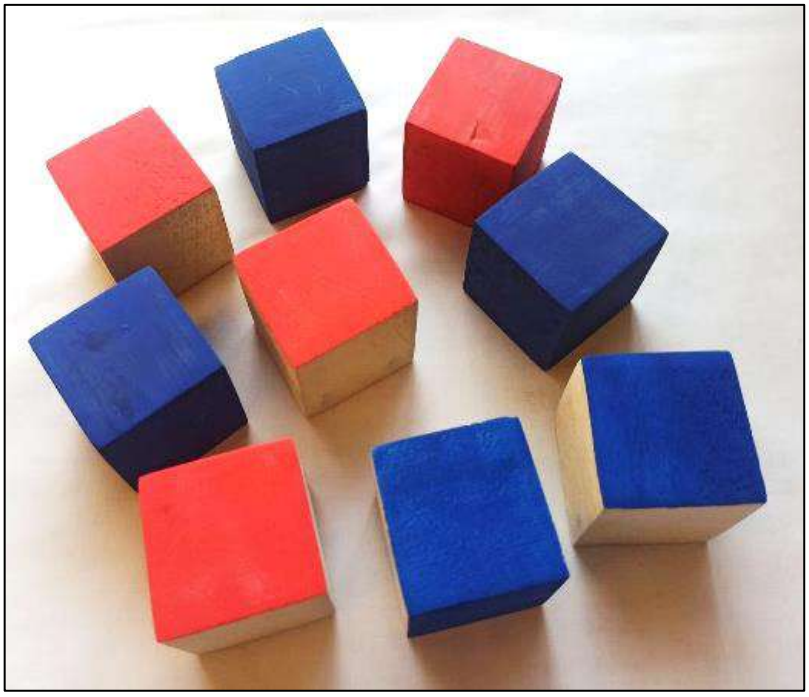


Ilustración 13 – 3: Cubos terminados para clasificación.

Realizado por: Pilco, 2023.

3.5.2 Plantilla de trabajo

A partir de la definición del espacio trabajo y cubos para clasificar, es posible continuar con el diseño de la plantilla de trabajo en la que se desarrolla el sistema de clasificación. Para este elemento se ha seleccionado tríplex de 12 mm de espesor como materia prima por su gran utilización en grabados y cortes en laser o maquinas CNC. Como se vio en la descripción general del proyecto, la plantilla está dividida en tres secciones. Las dimensiones de cada una de ellas se han establecido en función del tamaño de los cubos. Se puede apreciar las dimensiones de la plantilla en la Ilustración 14 – 3.

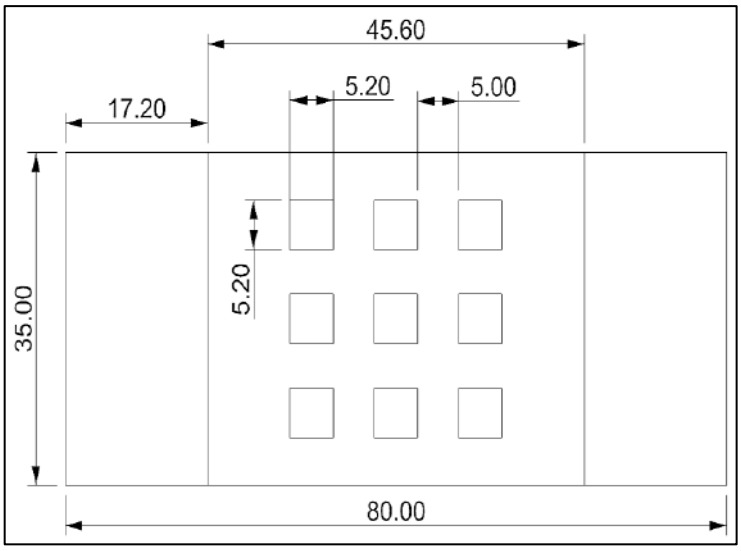


Ilustración 14 – 3: Dimensiones en metros de la plantilla de trabajo.

Realizado por: Pilco, 2023.

Las dimensiones de la plantilla son 80 cm de ancho por 40 cm de largo. Tomando como referencia al manipulador, al lado izquierdo se encuentra la sección destinada a albergar los objetos que sean clasificados como rojos. Esta sección es de 23 cm de ancho por 25 cm de largo en color rojo. De manera análoga, al lado derecho se colocarán los cubos clasificados como azules. Las dimensiones de este compartimento son exactamente iguales al anterior. Finalmente, la sección central es una matriz de 3x3. Cada espacio tiene una dimensión de 5 cm de ancho, 5 cm de largo y una profundidad de 5mm, de esta manera es posible colocar un cubo sobre el espacio y evitar que su posición se modifique. La separación de cada espacio es de 5cm para que el gripper no colisione con otro cubo durante el traslado. En la Ilustración 15 – 3 se puede apreciar el diseño en tercera dimensión.

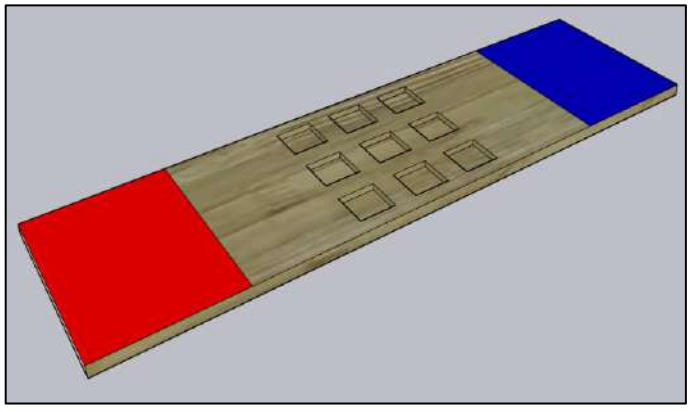


Ilustración 15 – 3: Plantilla de trabajo del sistema de clasificación.

Realizado por: Pilco, 2023.

Finalmente, la plantilla es colocada de manera fija en la posición contigua a la base del manipulador. Esta ubicación permite evitar colisiones con la cámara. En la Ilustración 16 – 3 se aprecia la ubicación final de la plantilla de trabajo.

Ilustración 16 – 3: Ubicación final de la plantilla.
Realizado por: Pilco, 2023.

3.5.3 Ubicación de la cámara

Utilizando la jaula de seguridad, se ha colocado un listón de madera en la puerta de la estructura a 55 cm de la superficie en donde se encuentra el robot. Se considera esta medida para la colocación del listón porque de tal forma, la cámara empleada para el sistema de visión puede tener la plantilla de trabajo en su área de cobertura. Además, se encuentra a 15 cm de la plantilla por lo que no existe peligro de que el brazo colisione con ella durante su movimiento.

La etapa de acondicionamiento del espacio de trabajo es muy importante ya que, de parámetros como la ubicación de la cámara, la iluminación y el seccionamiento de la plantilla, depende el correcto funcionamiento del sistema. En primera instancia, la correcta ubicación de cámara permite tener un panorama claro de los elementos presentes en el escenario de trabajo. De manera similar, la correcta iluminación facilita el desarrollo del algoritmo de visión artificial y el seccionamiento de la plantilla simplifica de cierto modo la creación de trayectorias que el robot manipulador KUKA va a desarrollar. En la Ilustración 17 – 3 se puede ver el escenario final.

Ilustración 17 – 3: Escenario final de trabajo.

Realizado por: Pilco, 2023.

3.6 Conexiones del sistema

En este apartado se detallan las distintas conexiones que hacen posible la ejecución del sistema de clasificación. En función de los bloques mencionados en la Ilustración 4 – 3 se definen las conexiones que integran el sistema. Por lo tanto, se analizan las conexiones de la cámara correspondiente al bloque de visión, las conexiones que permiten la comunicación entre el software y hardware del sistema y por último las conexiones correspondientes al bloque de acción y control.

3.6.1 Conexión de la cámara

Es el componente más sencillo de instalar en cuanto a las conexiones se refiere. Este dispositivo tiene una modalidad de conexión de *plug and play*, por lo que la alimentación y trasferencia de

datos se realiza al conectarlo únicamente mediante su cable USB hacia el computador con el que se desarrolla el sistema. En la Ilustración 18 – 3 se puede el esquema de conexión.

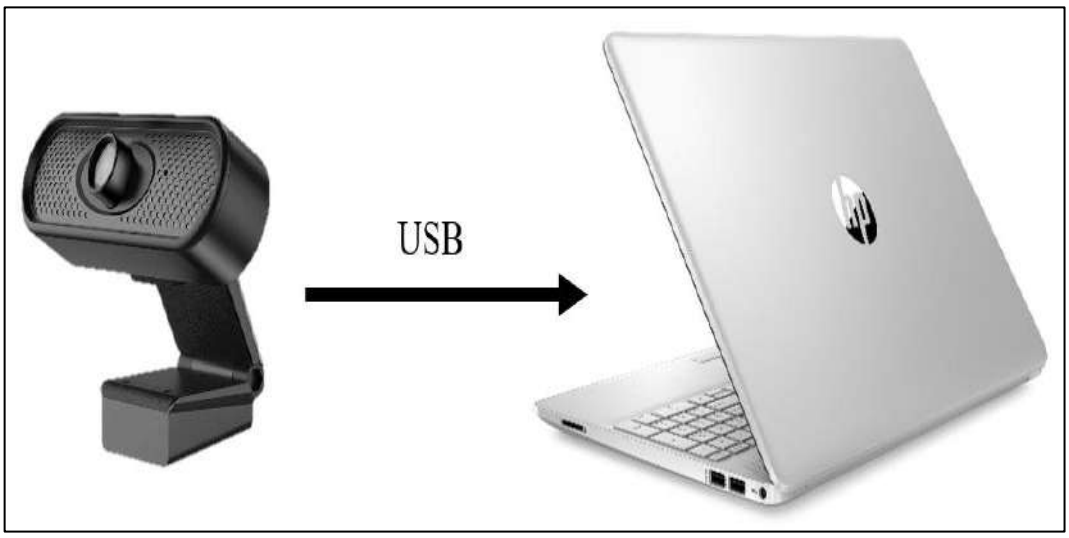


Ilustración 18 – 3: Esquema de conexión de la webcam con el ordenador.
Realizado por: Pilco, 2023.

Por la ubicación definida para la cámara con relación a la estructura de seguridad, se ha instalado una extensión de su cable de conexión para que no interfiera en el momento de la ejecución del sistema y resulte cómoda la conexión hacia el computador.

3.6.2 Conexión PC – Controlador KR C4

Esta conexión permite la comunicación entre el sistema de visión (software) con el manipulador físico (hardware). La conexión física es de tipo punto a punto entre el controlador del robot y el ordenador. Para ello, el controlador cuenta con una serie de puertos de comunicación para diferentes protocolos. Como se muestra en la Tabla 3 – 2, los puertos de comunicación para la programación del manipulador se los puede hacer mediante el X21 por medio del smartPAD, el X65 para la comunicación y programación mediante EtherCAT y finalmente el puerto X66 para la programación mediante ordenadores externos.

La conexión se realiza mediante un patchcord de conexión cruzada ya que, tanto el controlador como el ordenador operan bajo el sistema operativo Windows y la conexión es similar a la de dos computadores. El puerto utilizado para la conexión es el X66 por su capacidad de conectar una computadora externa con fines de instalación, programación, depuración y diagnóstico. En la Ilustración 19 – 3 se puede apreciar el esquema de conexión de la etapa de comunicaciones.

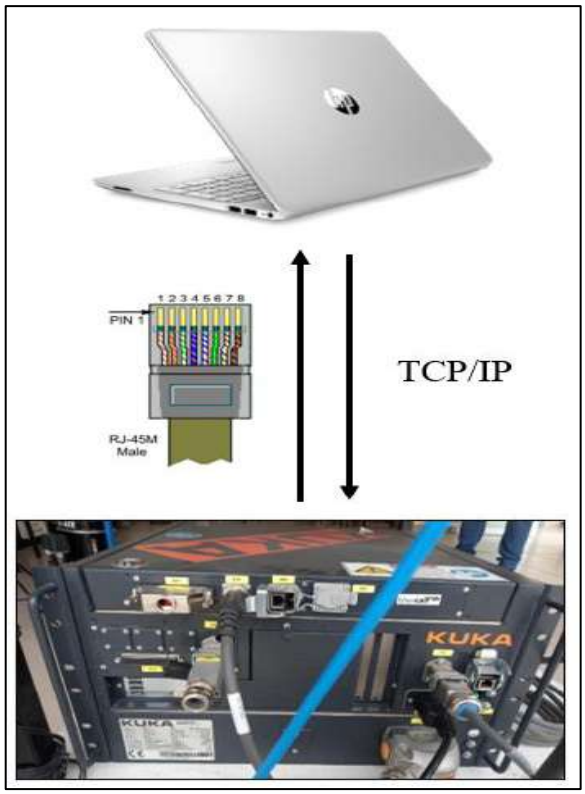


Ilustración 19 – 3: Conexión entre el controlador y el ordenador.

Realizado por: Pilco, 2023.

3.6.3 Conexión del Gripper

Esta herramienta es de accionamiento neumático. Por lo tanto, trabaja en conjunto con un compresor que suministra el aire en la presión de trabajo requerida por la pinza. En la Ilustración 20 – 3 se puede apreciar un esquema de la conexión realizada.

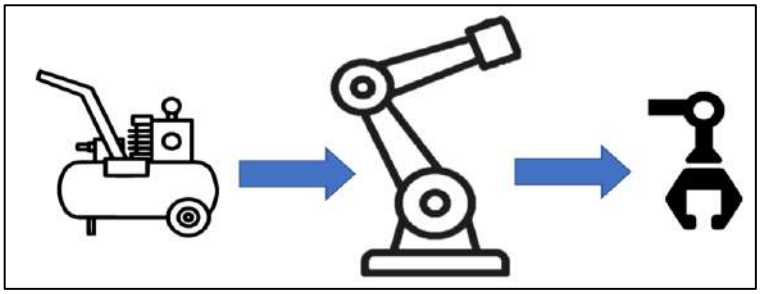


Ilustración 20 – 3: Esquema de la conexión del Gripper.

Realizado por: Pilco, 2023.

Para conectar este elemento primeramente se lo fija en la brida de montaje al extremo final del robot. Luego se conecta el suministro de aire como se aprecia en la Ilustración 21 – 3a desde el

compresor hacia el bastidor del robot. Finalmente, se conectan las dos salidas de aire presentes en el eslabón 4 hacia el gripper como se ve en la Ilustración 22 – 3b.

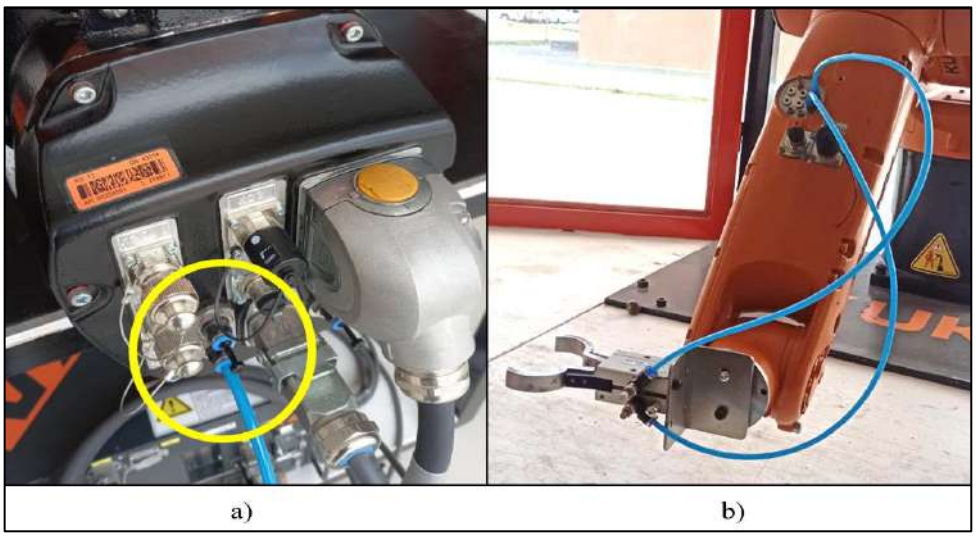


Ilustración 21 – 3: a) Conexión desde el compresor hacia el bastidor del manipulador. b) Conexión desde las tomas del eslabón 4 hacia la pinza.

Realizado por: Pilco, 2023.

3.7 Diseño del software

En esta etapa se detalla el diseño de todos los componentes a nivel de software tales como el sistema de visión para la detección de colores, la construcción del escenario, la programación de los movimientos en el software RoboDK y la comunicación.

3.7.1 Sistema de Visión artificial

3.7.1.1 Captura de la imagen

En esta etapa se capturan las imágenes que el sistema utiliza para reconocimiento de la escena. En la Ilustración 22 – 3 se puede observar la imagen que presenta características como la interpretación de color, brillo, saturación y nitidez en función de la resolución y características propias de la webcam. El número de frames utilizado para el procesamiento son 10, sin embargo, su adquisición es continua con el fin de seguir monitoreando la escena.

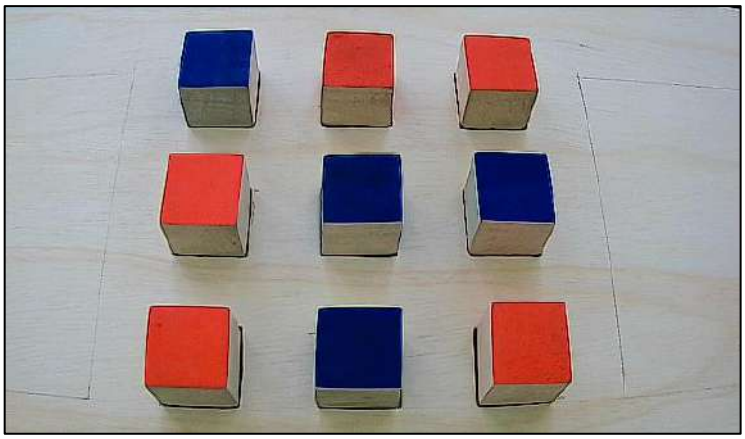


Ilustración 22 – 3: Captura de la imagen del espacio de trabajo.

Realizado por: Pilco, 2023.

3.7.1.2 Pre-procesamiento

El objetivo de esta etapa es mejorar la imagen mediante la aplicación de un filtro de tipo gaussiano, el cual recibe la imagen inicialmente capturada como parámetro de entrada y de acuerdo con el tamaño del kernel, para esta aplicación es una matriz cuadrada de 3, se puede disminuir el ruido que presenta la imagen, así como un ligero efecto de desenfoque para minimizar ciertas imperfecciones. En la Ilustración 24 – 3 se aprecia la imagen después de ser filtrada.

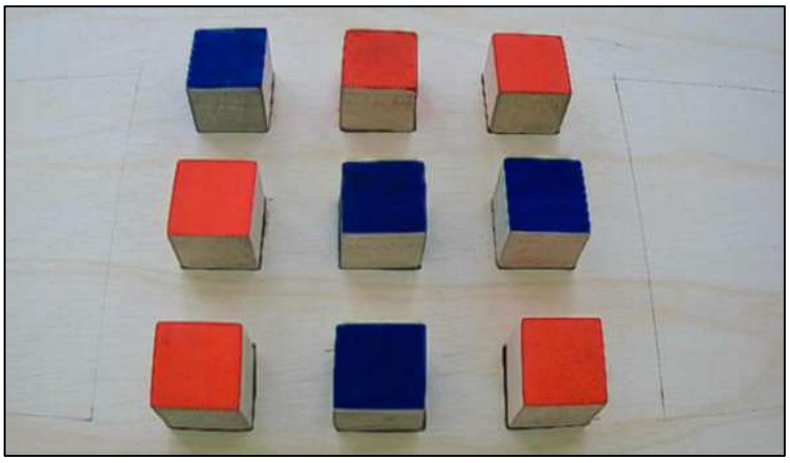


Ilustración 23 – 3: Imagen preprocesada mediante un filtro Gaussiano.

Realizado por: Pilco, 2023.

3.7.1.3 Segmentación

La técnica empleada para esta etapa es la umbralización. El empleo de esta técnica implica destacar en un rango de pixeles las partes de color y en otro rango diferente el fondo. Para ello se

realiza una conversión de la imagen hacia el espacio de color HSV. De esta forma, los objetos para clasificar se muestran con mayor intensidad con respecto del fondo como se aprecia en la Ilustración 24 – 3.

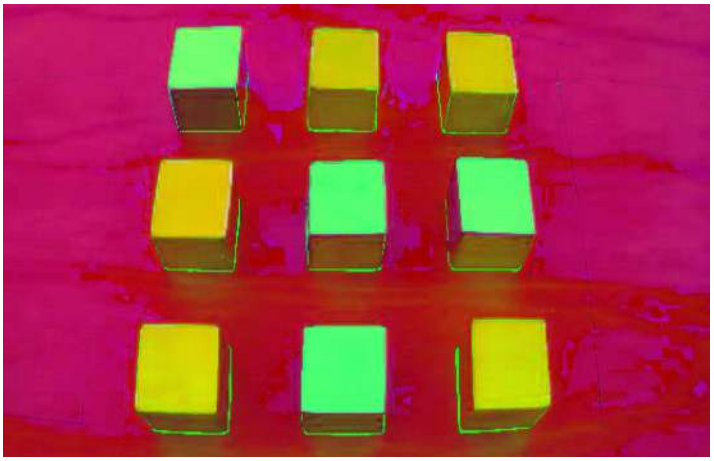


Ilustración 24 – 3: Imagen convertida al espacio de color HSV.

Realizado por: Pilco, 2023.

Para determinar los umbrales se realizó un análisis de la imagen basado en binarización. Para ello se separaron los 3 canales del espacio de color HSV como se aprecia en la Ilustración 26 – 3.

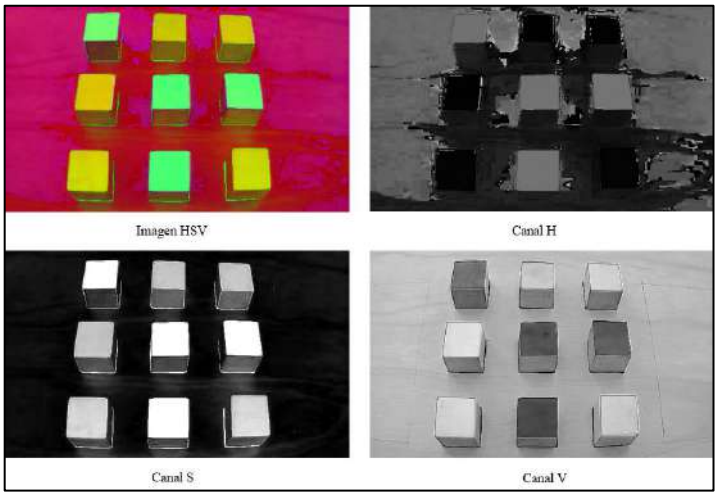


Ilustración 25 – 3: Canales del espacio de color HSV.

Realizado por: Pilco, 2023

Con esta premisa, se desarrolló un algoritmo a través del cual, con la ayuda de sliders se pueden variar los valores de tono, saturación y valor de la imagen. En la Ilustración 26 – 3 se puede observar la interfaz del algoritmo mencionado con los elementos descritos para obtener los parámetros correspondientes a los colores en análisis.

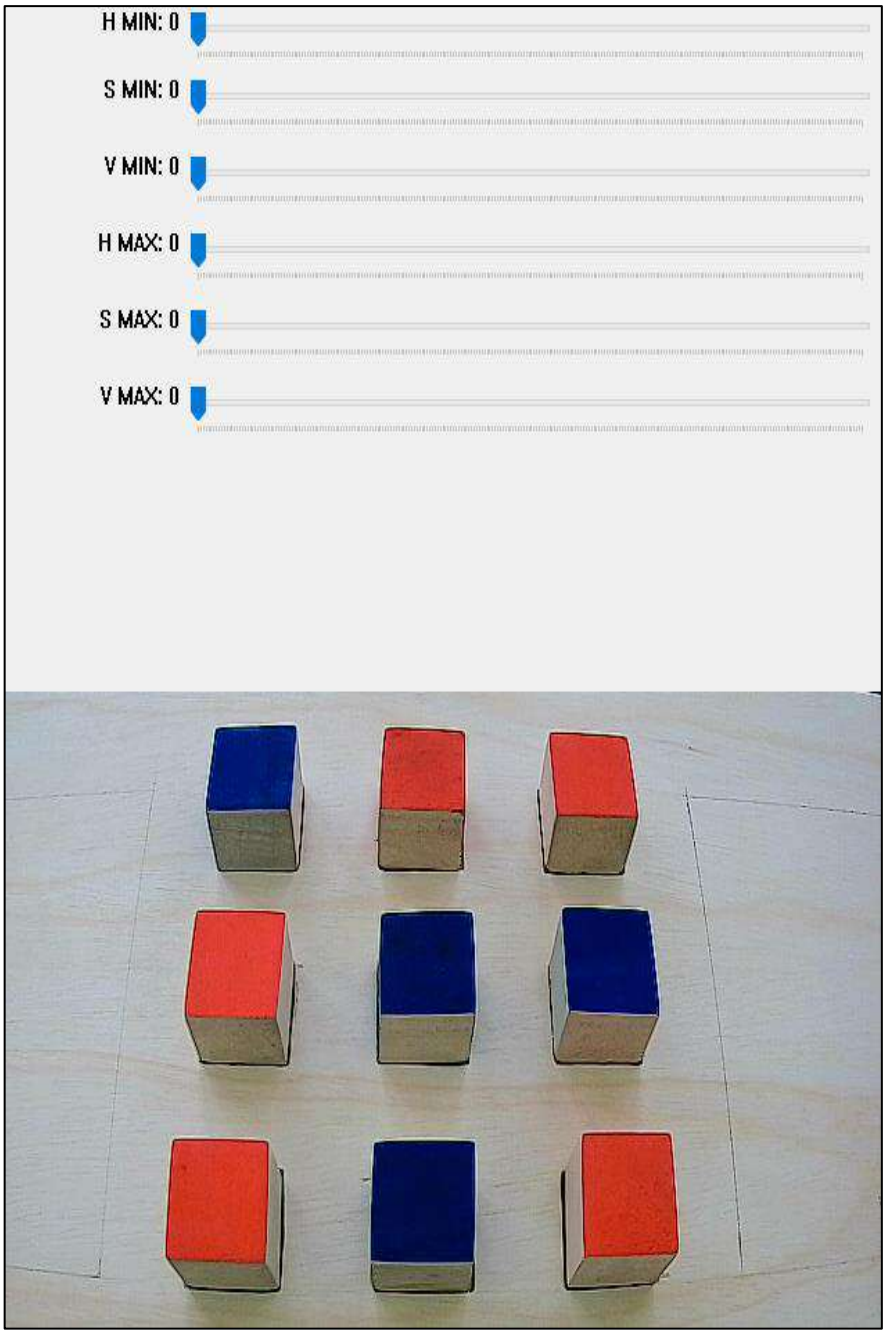


Ilustración 26 – 3: Interfaz para obtener los parámetros de umbral de los colores en análisis.

Realizado por: Pilco, 2023.

En la Ilustración 27 – 3 se puede apreciar la imagen binaria resultante con los parámetros HSV obtenidos para los dos colores, mientras que en la Tabla 6 – 3 se muestran los valores de umbral determinados.

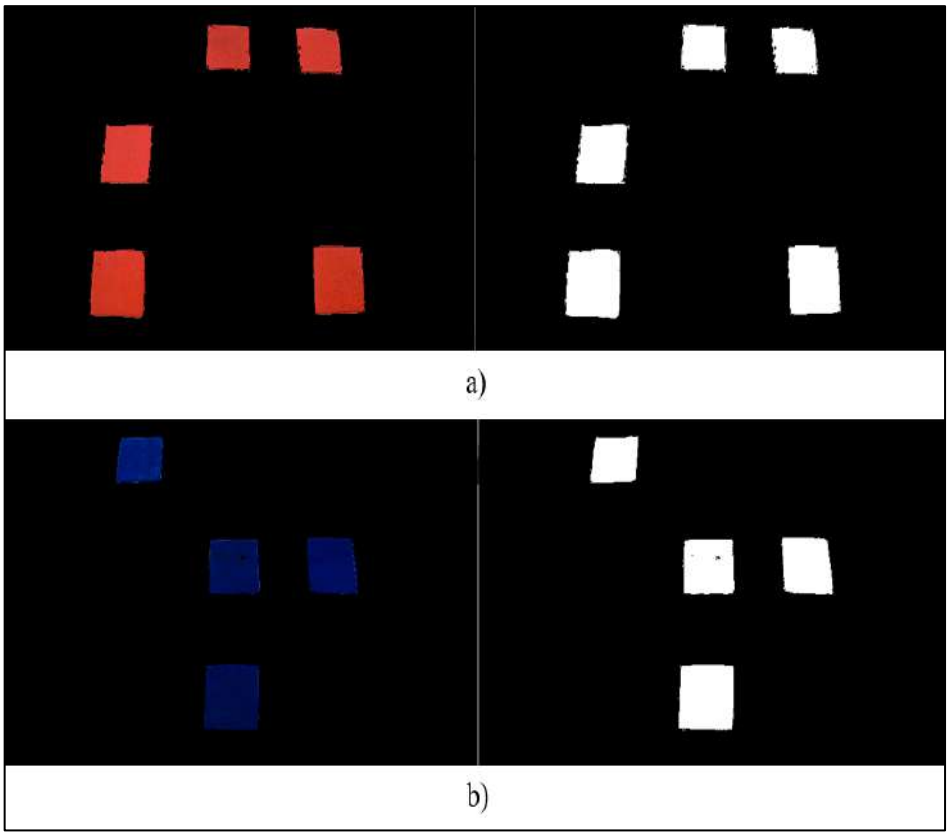


Ilustración 27 – 3: a) Imagen binarizada para la detección del color azul. b) Imagen binarizada para la detección del color rojo.

Realizado por: Pilco, 2023.

Tabla 6 – 3: Parámetros para la segmentación por color.

Canal / Objeto	H		S		V	
	Bajo	**Alto**	**Bajo**	**Alto**	**Bajo**	**Alto**
Cubo Azul	30	140	160	255	50	255
Cubo Rojo	0	10	190	255	200	255

Realizado por: Pilco, 2023.

3.7.1.4 Operaciones morfológicas

La aplicación de operaciones morfológicas de erosión y dilatación sobre la imagen binarizada permite eliminar imperfecciones, regiones o manchas que no correspondan a los objetos de interés, rellenar espacios del objeto que no se definan adecuadamente y diferenciar correctamente objetos que se encuentren muy próximos para el proceso de extracción de características. Se puede ver la imagen con las trasformaciones morfológicas descritas en la Ilustración 28 – 3.

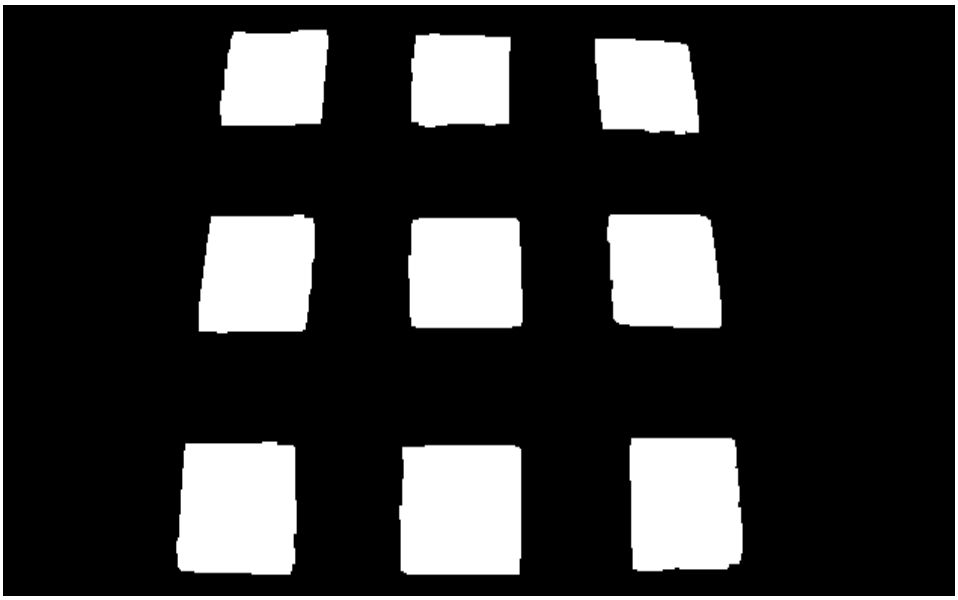


Ilustración 28 – 3: Imagen con aplicación de operaciones morfológicas.

Realizado por: Pilco, 2023.

3.7.1.5 Extracción de características

En esta etapa se obtienen los contornos y momentos de los objetos detectados con base a las imágenes resultantes obtenidas en las etapas anteriores. En función de este hecho, en la Ilustración 29 – 3 se muestra la imagen con la representación de la extracción de los contornos de los objetos, los cuales presentan una buena definición de los cubos y sin discontinuidades.

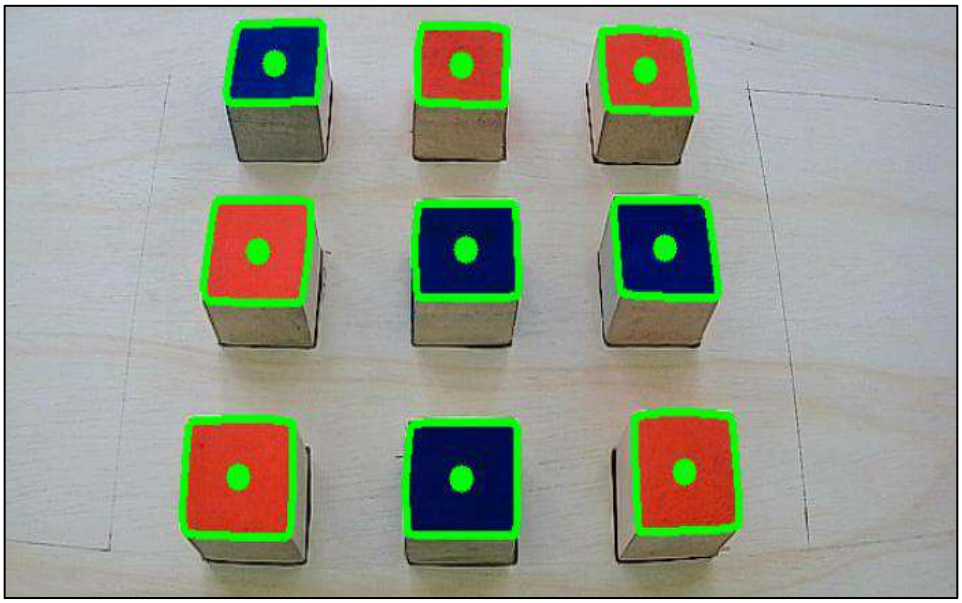


Ilustración 29 – 3: Detección de los contornos.

Realizado por: Pilco, 2023.

Una vez definidos los contornos de los cubos para clasificar, se identifica el centro de los contornos marcados mediante un par de coordenadas dado en pixeles con relación a la cámara y una leyenda con el color respectivo como se muestra en la Ilustración 30 – 3.

Ilustración 30 – 3: Determinación de los centros e identificador por color de los objetos.

Realizado por: Pilco, 2023.

3.7.2 Entorno de simulación y monitoreo de movimiento

En esta sección del diseño del software se construye el escenario de simulación y monitoreo del sistema, así como el establecimiento de la comunicación entre el ordenador y el controlador del manipulador y se analizan las configuraciones respectivas para llevar a cabo la interconexión de los componentes hardware y software.

3.7.2.1 Construcción del escenario

La construcción del escenario inicia a partir de la selección del manipulador KUKA de la familia KR10 en su modelo R900 Sixx de la biblioteca online del software. El gripper de dos dedos seleccionado simula la pinza instalada en el robot físico y todos los elementos necesarios como la mesa de trabajo, y los objetos para clasificar se han colocado desde las bibliotecas nativas del software. Al colocar el robot con el que se va a trabajar, se inserta automáticamente un sistema de referencia base que corresponde al origen a partir del cual se realizan las trayectorias. De igual manera al insertar el gripper en el manipulador, se añade un sistema de referencia para la herramienta. Y finalmente se tiene un sistema de referencia en el espacio central de las posiciones

en las que se colocan los objetos. Estos sistemas de referencia se pueden apreciar en la Ilustración 31 – 3.

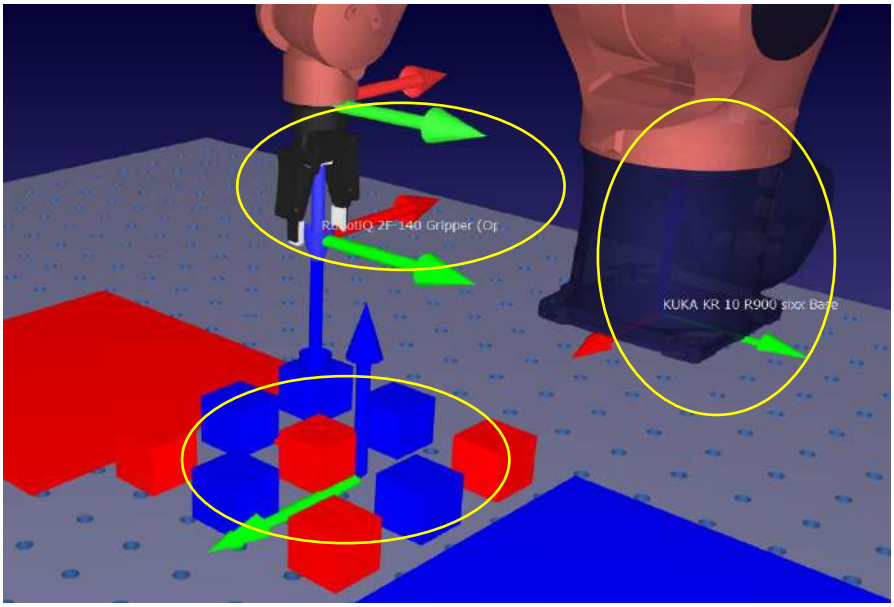


Ilustración 31 – 3: Sistemas de referencia principales.

Realizado por: Pilco, 2023.

El escenario diseñado es muy parecido al sistema real, sin embargo, la plantilla de trabajo y la cámara son componentes que no se pueden representar en el software por lo que han sido obviados. En la Ilustración 32 – 3 se puede ver el escenario construido.

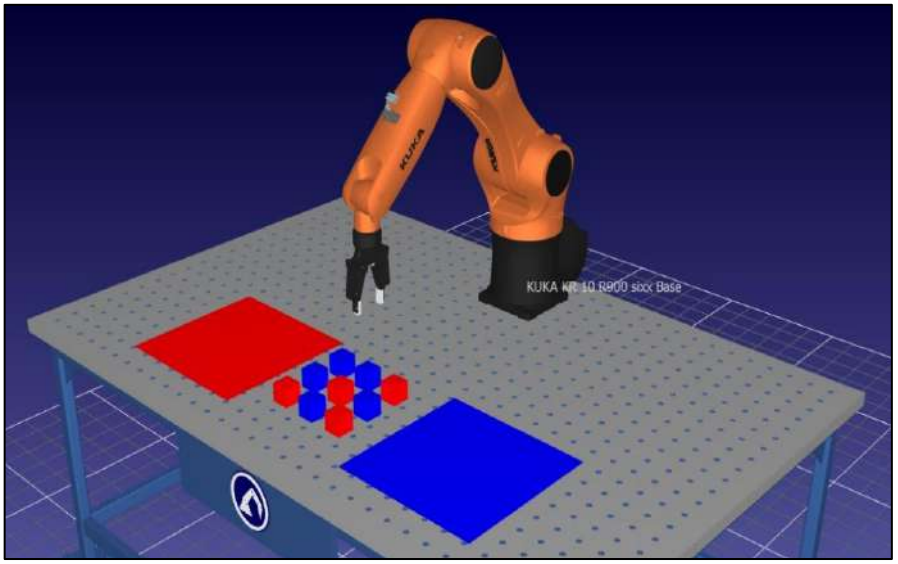


Ilustración 32 – 3: Escenario de trabajo desarrollado en RoboDK.

Realizado por: Pilco, 2023.

3.7.2.2 Programación de targets

De acuerdo con la plantilla de trabajo se establecen los targets u objetivos que el robot alcanza conforme se desarrolla el proceso. Estos targets son creados en función de cada posición en la plantilla y la colocación de los cubos es aleatoria en cada espacio. Se han definido un total de 24 targets cuya nomenclatura se detalla en la Tabla 7 – 3.

Tabla 7 – 3: Nomenclatura de los targets de posición definidos.

Targets	Descripción
Pick	Objetivos de las posiciones en la matriz cuando la pinza esta sobre el cubo listo para tomarlo.
Aprox	Objetivos cuando la pinza está a cierta distancia del objetivo para posicionarse adecuadamente.
Rojo	Objetivo cuando la pinza ha soltado el objeto de color rojo.
Azul	Objetivo cuando la pinza ha soltado el objeto de color azul.
Wait	Objetivo en la posición de espera antes de acercarse a cada posición de la matriz.
Home	Posición del manipulador antes y al finalizar la ejecución del sistema de clasificación.

Realizado por: Pilco, 2023.

Los targets con el prefijo **Pick** se cuentan del 1 al 9 y se distribuyen en cada posición de la plantilla, mientras que los de prefijo **Aprox** corresponden a las mismas posiciones en la matriz, pero con una altura mayor para posicionar la herramienta. Los targets **Rojo** y **Azul** son los objetivos de la sección en donde se colocan los objetos clasificados respectivamente. Finalmente, los targets **Wait** y **Home** son posiciones de espera al iniciar y cada vez que se traslada un objeto. En la Ilustración 33 – 3 se parecía los targets creados en el entorno de simulación.

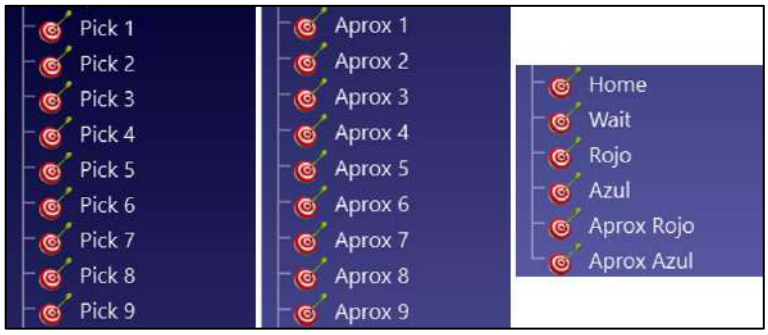


Ilustración 33 – 3: Targets de movimiento del robot.

Realizado por: Pilco, 2023.

En la Ilustración 34 – 3, se pueden observar las trayectorias que desarrolla el robot en color amarillo, de acuerdo con cada target asociado a las posiciones en la plantilla.

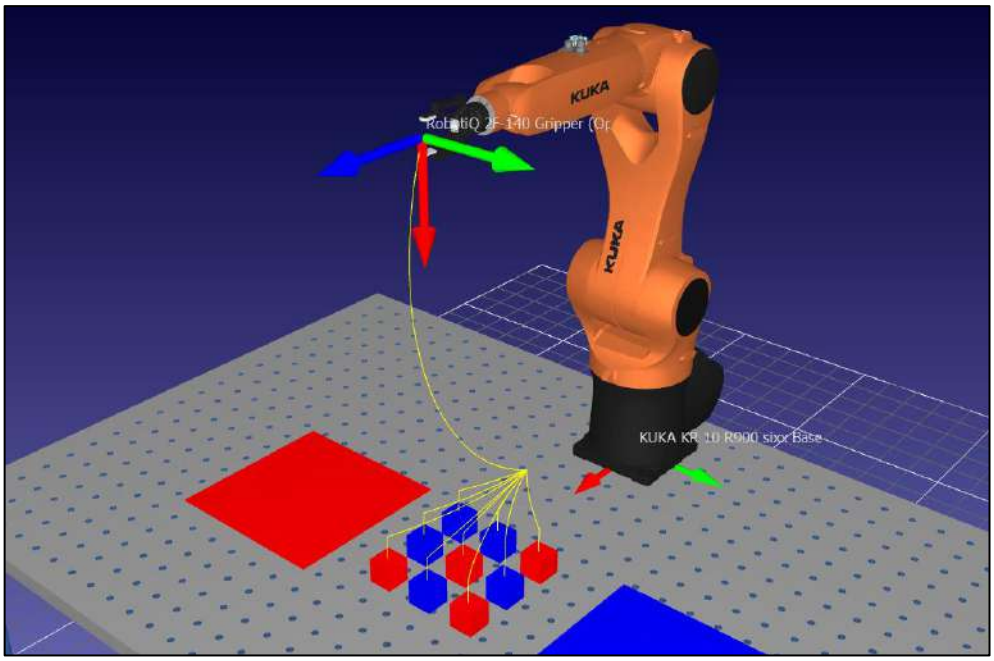


Ilustración 34 – 3: Descripción de trayectorias de movimiento.

Realizado por: Pilco, 2023.

3.7.2.3 Programación del gripper

La programación del gripper se realiza mediante la instrucción IO en RoboDK. Esta herramienta permite la gestión de las entradas y salidas del robot tanto digitales como analógicas. El gripper empleado utiliza dos salidas digitales, concretamente la 21 para el cierre de la pinza y la 22 para la apertura. Estas salidas tienen que ser activadas y desactivadas para que ejecuten la instrucción de apertura y cierre de manera adecuada. Por ello, se han creado dos programas adicionales para el control de la pinza. Se aprecian los programas con las instrucciones IO en la ilustración 35 – 3.

Ilustración 35 – 3: Programas de control del gripper.

Realizado por: **Pilco, 2023.**

3.7.3 Conexión software - controlador

La conexión entre el sistema de software y el manipulador físico es posible gracias al driver **apikuka** que trabaja con el servidor Kukavarproxy que ofrece el software RoboDK para robots

del fabricante KUKA. Este controlador de robot es una interfaz que trabaja con el modelo servidor-cliente empelando el protocolo de comunicación TCP/IP. El driver, permite hacer la lectura y escritura de variables de robot dentro de una red local. En este caso es una red LAN entre el ordenador, el robot y su controlador. En los siguientes apartados se describe el proceso de instalación del driver, así como la puesta en línea del sistema ya conectado. Como se mostró en el análisis del controlador KR C4 compact, el dispositivo ejecuta Windows 7 embebido. Con este antecedente, para establecer la comunicación entre los dos dispositivos de la red se sigue el procedimiento que se aprecia en las siguientes secciones:

3.7.3.1 Instalación del controlador KUKAVARPROXY

La instalación del controlador se describe en los siguientes pasos:

1. Ingresar al perfil de usuario como **Administrador** para tener los permisos necesarios para la instalación del driver en el controlador. Para ello, nos dirigimos hacia el icono de configuración en la esquina superior izquierda. En esta ventana de configuraciones nos dirigimos al apartado de **Configuración**, luego a la sección de **Grupo de usuario** y seleccionar el perfil de usuario **Administrador**. Se desplegará una ventana que solicita la contraseña de acceso a este perfil, la contraseña es: **kuka**

2. Nos dirigimos nuevamente al icono de configuración en la esquina superior izquierda. En esta ventana de configuración nos dirigimos al apartado de **Puesta en marcha**, luego en la sección de **Servicio** escogemos la opción **Minimizar HMI** (nos aparecerá la interfaz de Windows).

3. Ahora conectamos una unidad de almacenamiento USB al controlador para copiar la carpeta **KUKAVARPROXY** que contiene los archivos de instalación del driver en el escritorio del controlador.

4. Ejecutamos el archivo **KUKAVARPROXY.exe** y el driver quedara instalado correctamente.

5. Finalmente, es recomendable que el driver se inicie automáticamente al encender el controlador. Por lo tanto, se crea un acceso directo del driver que presenta un icono del personaje Bender. Este acceso se tiene que pegar en la carpeta de inicio del sistema a la cual se accede de la siguiente forma: **Inicio**, clic derecho sobre la carpeta **Todos los programas** y **Abrir**.

De esta manera el driver se iniciará automáticamente al encender el sistema y estará listo para escuchar e intercambiar las variables globales con RoboDK en un ordenador remoto.

3.7.3.2 Habilitación del puerto

Para que se puedan intercambiar las variables para el control del movimiento del robot se tiene que habilitar un puerto de comunicaciones en el controlador. Para ello se realizan las siguientes operaciones en el smartPAD:

1. Abrir la ventana de configuraciones en el icono de la esquina superior izquierda. En el apartado **Puesta en marcha**, seleccionar la sección de **Configuración de red** y seleccionar la opción **Avanzado.**

2. En la nueva ventada desplegada seleccionamos **NAT** y se escoge la opción **Añadir puerto**. Aquí se asigna el valor **7000** en el apartado del **Número de puerto** y en el apartado de **Protocolos permitidos** seleccionamos la opción **tcp/udp**. Una vez asignados todos los parámetros seleccionamos la opción **Guardar** y se acepta en **OK** en la ventana de confirmación desplegada.

3. Para que se guarden los cambios realizados se tiene que reiniciar el controlador. En la ventana de configuración nos dirigimos al apartado **APAGAR**, marcamos la opción **Refrescar archivos** y presionamos el botón **Reiniciar Controlador**. Se confirma la acción en el botón **YES** de la ventana que aparece.

3.7.3.3 Configuración para sincronización

Para ejecutar los programas desarrollados en RoboDK directamente en el robot se tiene que ejecutar un archivo de sincronización en el smartPAD en el modo de operación automático. Este archivo es de extensión **.src** y contiene las instrucciones, declaraciones y estructuras que permiten a las variables globales controlar el movimiento del robot. Sin embargo, este archivo de sincronización no se puede ejecutar de manera directa, por lo que se tiene que modificar el archivo $CONFIG en la raíz del controlador. Para realizar este procedimiento se siguen los siguientes pasos.

1. Ingresar al perfil de usuario **Administrador**, para ello, se sigue el paso 1 visto en la sección de **Instalación del driver**.

2. Por medio de una unidad de almacenamiento USB se carga el archivo de sincronización **RoboDKsync35.src** y el archivo **$CONFIG.dat** al sistema.

3. En el panel lateral de navegación identificamos nuestra unidad de almacenamiento, se copia los archivos mencionados en la carpeta **R1** que se despliega al presionar la unidad raíz de

control en el panel lateral que se presenta con un icono de robot. Se notará que el archivo de sincronización está marcado con una X indicando que no se puede ejecutar.

4. Para poder ejecutar el archivo de sincronización se tiene que modificar el archivo **$CONFIG** ubicado en la carpeta **STEU** de la unidad raíz del sistema. Para esto, se debe dirigir a la carpeta **R1** y abrir el archivo **$CONFIG**, se selecciona todo el contenido de este archivo (variables globales) y se selecciona la opción **Editar** y marcar **Copiar**. Se debe dirigir nuevamente a la carpeta **STEU**, marcar el archivo **$CONFIG** y seleccionar la opción **Abrir**, dentro de este archivo, se selecciona la opción **Editar** y se copia las variables provenientes del archivo modificado seleccionando **Insertar**. Cerrar en el icono en forma de X a la izquierda y se confirma en **Yes** para guardar los cambios realizados. Se pueden ver las variables mencionadas en la Ilustración 36 – 3.

```
DEFDAT  $CONFIG
INT COM_ACTION=0
INT COM_ACTCNT=0
REAL COM_ROUNDM=0
REAL COM_VALUE1=0
REAL COM_VALUE2=0
REAL COM_VALUE3=0
REAL COM_VALUE4=0
DECL E6AXIS COM_E6AXIS
DECL POS COM_POS
DECL FRAME COM_FRAME
DECL E6POS COM_E6POS
ENDDAT
```

Ilustración 36 – 3: Variables globales para sincronización del RoboDK con el manipulador.

Realizado por: Pilco, 2023.

En la carpeta R1 se observa que el archivo de sincronización **RoboDKsync35.src** ya no está marcado con una X y se lo puede ejecutar sin ningún inconveniente.

3.7.3.4 Configuraciones de red

La comunicación entre el software y el controlador se realiza por medio de una conexión ethernet estándar bajo el protocolo TCP/IP. Para ello, se crea una red local y se emplea una conexión punto a punto mediante un patchcord con conectores **RJ45** en el puerto del controlador identificado como **X66,** que permite la programación desde controladores externos y en el puerto de conexión del ordenador remoto.

Direcciones de red en el controlador KR C4 Compact. – La red local se crea en función de las configuraciones de red predeterminadas del controlador. Para identificar las direcciones predefinidas accedemos a la interfaz de Windows desde el smartPAD siguiendo los pasos 1 y 2 vistos en el apartado de **Instalación del Driver.** En la interfaz de Windows, se debe dirigir al icono de red presente en la barra de tareas y accedemos a la configuración de red e internet, y se selecciona **Cambiar opciones del adaptador.** Se desplegará una ventana con todos los adaptadores de red, se selecciona el que corresponda a la conexión con el patchcord, y dar clic derecho y seleccionar **Propiedades.** En la ventana que aparece, se marca la opción **Protocolo de internet versión 4(TCP/IPv4)** y seleccionar **Propiedades**, aquí se despliega una nueva ventana con las direcciones estáticas de red predefinidas en el controlador que se pueden ver en la Tabla 8 – 3.

Tabla 8 – 3: Configuraciones de red del controlador.

Parámetros	Detalle
Dirección IP	192.168.0.1
Mascara de Subred	255.255.255.0
Gateway por defecto	192.168.0.2
Servidor DNS preferido	172.31.0.1

Realizado por: Pilco, 2023.

Direcciones de red en el Ordenador. – Al igual que en el controlador, en el ordenador remoto se debe dirigir al icono de red en la barra de tareas y marcar **Configuración de red e Internet**. Se abrirá una ventana y marcar la opción **Cambiar opciones del adaptador, se** identifica la conexión de red realizada por el patchcord, y debe dar clic derecho sobre el adaptador de red y escoger **Propiedades**. En la ventana desplegada seleccionar la opción **Protocolo de internet versión 4(TCP/IPv4)** y acceder a sus propiedades. En la ventana que se despliega se marca la opción **Usar la siguiente dirección IP** y **Usar las siguientes direcciones de servidor DNS** para asignar manualmente la dirección IP, la máscara de subred y el servidor DNS preferido. Se pueden ver las direcciones estáticas asignadas al ordenador en la Tabla 9 – 3.

Tabla 9 – 3: Configuración de red del ordenador.

Parámetros	Detalle
Dirección IP	172.31.1.45
Mascara de Subred	255.255.0.0
Gateway por defecto	– – – –

Servidor DNS preferido	172.31.0.1

Realizado por: Pilco, 2023.

Direcciones de red en el smartPAD. – Las direcciones de red que se configuran en el SmartPAD son las correspondientes al brazo manipulador. Al igual que en el controlador, estas direcciones son predeterminadas por el fabricante. Sin embargo, estas direcciones se pueden modificar ingresando al perfil de usuario **Administrador** en la ventana que aparece tras realizar el primer paso de la sección **Habilitación del puerto de comunicaciones.** Las direcciones se ven en la Tabla 10 – 3.

Tabla 10 – 3: Configuración de red del manipulador.

Parámetros	Detalle
Dirección IP	172.31.1.47
Mascara de Subred	255.255.0.0
Gateway por defecto	192.168.0.2
Servidor DNS preferido	172.31.0.1

Realizado por: Pilco, 2023.

3.7.3.5 Establecimiento de la comunicación y ejecución del programa

El procedimiento final para conectar RoboDK con el robot y poder programarlo en línea se muestra a continuación.

1. Conectar el patchcord en el puerto **X66** del controlador KR C4 Compact y en el ordenador remoto.

2. Asignar el direccionamiento estático en las configuraciones de red del ordenador remoto para tener los tres elementos que intervienen en la conexión (controlador, ordenador y robot) dentro de la misma red local.

3. En el smartPAD girar la llave de selección de modo de operación ubicado en la parte superior, hacia la derecha, marcar **Automático** y girar la llave de selección hacia la posición original a la izquierda. Existen 4 cuadros de notificación de estados del smartPAD en la parte superior. El cuarto cuadro corresponde al modo de operación del robot, por lo que seleccionado el modo Automático se tendrá que evidenciar el cuadro en color verde con una letra **A**.

4. En el panel de navegación, se debe dirigir a la unidad raíz para acceder a la carpeta **R1** que aloja el archivo de sincronización **RoboDKsync35.src**. Se marca el archivo y se presiona en la opción **Seleccionar.** El primer cuadro de notificación con una **S** estará de color verde y el tercer cuadro con una **R** estará en blanco.

5. El segundo cuadro de notificación estará con un **O**, esto indica que los seguros de las articulaciones están activados y el robot no se podrá mover. Para deshabilitarlos, se presiona el cuadro de notificación y seleccionar la opción **I** que libera los seguros. Entonces el icono cambiara a verde y el tercer icono con **R** cambiara a rojo.

6. Presionar el botón **Play** que se encuentra en los botones físicos al lado izquierdo del smartPAD para cambiar el estado de **Paro** al de **Ejecución**. De esta manera los cuatro cuadros de notificación estarán en color verde. Esto indica que el archivo de sincronización está operativo a la espera de recibir instrucciones.

7. En RoboDK con la estación diseñada para el sistema de clasificación, se debe dirigir a la pestaña Conectar y seleccionar la opción Conectar Robot como se muestra en la Ilustración 37 – 3.

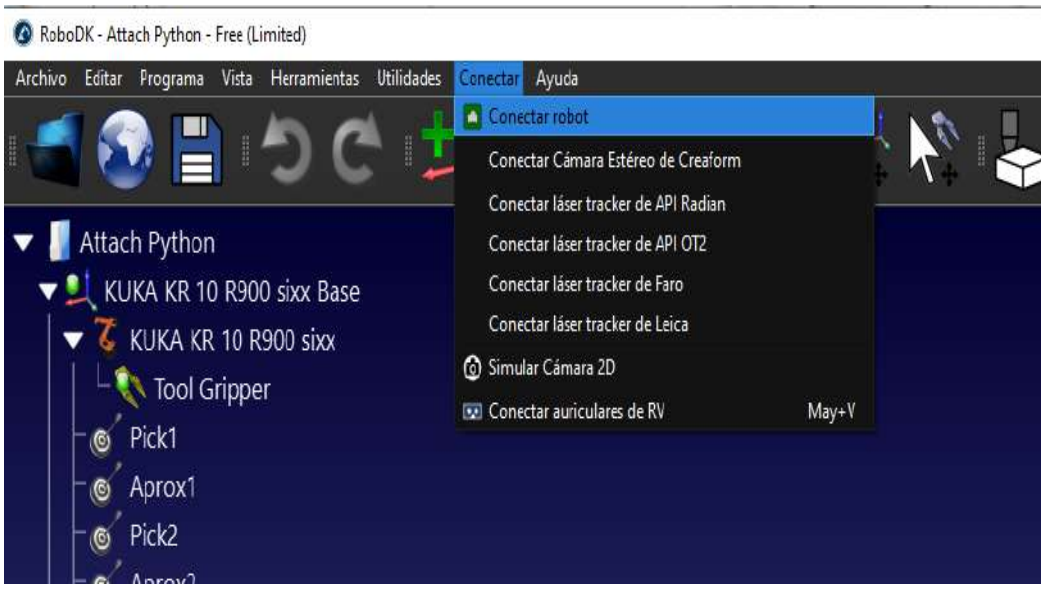


Ilustración 37 – 3: Función Conectar robot.

Realizado por: Pilco, 2023.

8. Aparecerá una nueva ventana en la parte izquierda en donde tendrá que asignar la dirección IP del robot, el puerto de comunicación y en la parte inferior seleccionar el driver. Estas direcciones se presentan en la Tabla 11 – 3.

Tabla 11 – 3: Configuraciones en el software para la sincronización.

Parámetros	Detalle
Dirección IP del robot	172.31.1.47
Puerto	7000
Conductor	Apikuka

Realizado por: Pilco, 2023.

9. Cuando se han completado todos los campos, seleccionar la opción conectar. La conexión estará lista cuando se observe que el estado de la conexión cambia a color verde mostrando el mensaje **READY.**

10. Para poder ejecutar el programa en el robot, se da clic derecho sobre el programa y marcamos la casilla **Ejecutar en el robot.**

11. Se puede ejecutar el programa dando clic derecho sobre el programa y seleccionar opción Ejecutar o directamente dando doble clic sobre el programa.

12. Finalmente, antes de ejecutar un programa se debe bajar la velocidad de operación del robot al 10% ya que, al seleccionar el modo de operación automático, la velocidad se establece al 100% por lo que se podría generar un accidente.

CAPÍTULO IV

4 RESUTADOS

En este capítulo se evaluó el desempeño del sistema de clasificación de objetos para verificar que la detección del color mediante los algoritmos desarrollados sea la apropiada en los colores definidos para los objetos. Además, se evaluó la correcta manipulación de los objetos con el brazo robótico hacia los lugares de destino correspondientes, así como sus tiempos de ejecución.

4.1 Consideraciones generales

Teniendo en cuenta la naturaleza del sistema desarrollado, los datos obtenidos por el sistema provienen de sus variables cualitativas fundamentales, es decir, la luminosidad incidente en el escenario de trabajo y la detección que predice el sistema para cada objeto procesado. En este sentido como menciona (Cienfuegos et al. 2016), este tipo de variables se analizan estadísticamente mediante técnicas no paramétricas, es decir, no se pueden calcular parámetros como la media, desviación estándar y otros.

La escala de medición empleada para este sistema de clasificación es nominal, ya que se trabaja en función de categorías o clases. De manera puntual, en el caso de la variable luminosidad, se definen tres clases, alta, media y baja, mientras que, para la variable detección se tiene dos clases, correcta y errónea. En este sentido, no se pueden realizar operaciones matemáticas o establecer relaciones de orden como se efectúa en el análisis de datos con estadística paramétrica. Sin embargo, se pueden obtener conteos, frecuencias y porcentajes con la ayuda de tablas de contingencia. (Cienfuegos et al. 2016)

De acuerdo con lo mencionado en (Ochoa, 2013) se requieren al menos 30 muestras para validar las pruebas mediante el análisis del error. Por lo tanto, entre mayor sea el número de muestras, menor será el error obtenido de manera que las conclusiones estadísticas sean válidas. Con este antecedente, para determinar la validez del sistema se considera en primer lugar el error absoluto que sostiene que la estimación adquirida es más exacta cuando más próxima sea a la real. Este error se define en la expresión vista en la ecuación 3.

$$\Delta X = |X_r - X_m| \qquad (3)$$

Donde X_m representa el valor medido y X_r el valor real.

Adicionalmente, un indicador de calidad con relación a las estimaciones adquiridas se realiza mediante la obtención del error relativo que presenta de manera porcentual dicha característica. Este error se expresa mediante la ecuación 4.

$$e_r = \frac{\Delta X}{X_r} * 100\% \quad (4)$$

Donde, ΔX representa el error absoluto y X_r el valor relativo.

La validez del sistema mediante el análisis de los errores se considera buena si el error relativo no supera el 1%, mientras que, el sistema tendrá una validez aceptable si el error relativo porcentual se encuentra en un intervalo entre el 5 y 10%. (Santo et al. 2018)

Otro aspecto estadístico por considerar es la relación existente entre las variables fundamentales del sistema. Para ello, el sistema se considera relativamente bueno ya que reconoce apropiadamente los colores definidos, sin embargo, la variable luminosidad es muy importante para determinarlo como tal. De esta manera, como menciona (COGNEX, 2022) la iluminación del espacio de trabajo es un aspecto crucial para que el sistema trabaje de manera adecuada o no.

Ante este hecho, se realiza un análisis de la relación existente entre las variables iluminación y detección mediante la prueba de chi – cuadrado. Esta prueba utiliza tablas de contingencia que sirven para comprobar la dependencia o no de las frecuencias entre dos variables. (Tinoco, 2008) El método que emplea para determinar esa relación son la aceptación o rechazo de hipótesis.

Adicionalmente, el análisis del sistema mediante matrices de confusión permite evaluar la precisión del sistema de detección para el color rojo y azul de manera independiente con el fin de obtener mejores resultados. Este método emplea el análisis de errores y aciertos del sistema. Los errores presentes pueden ser de omisión cuando un elemento de una clase no se define como tal, y errores de comisión cuando un elemento aparece en una clase que no pertenece.(Sánchez, 2016)

4.2 Correlación de variables

Para realizar el análisis de correlación de variables se recolectaron 30 imágenes. Cada una de ellas contiene 9 objetos ubicados dentro de la plantilla de trabajo con algunos en colores diferentes a los definidos. En esta configuración, el análisis de la relación de las variables se realizó con 270 muestras recolectadas de manera equitativa en las tres condiciones mencionadas para la variable

iluminación. Cabe mencionar que cada grupo de 90 muestras fue el mismo en cuanto a la disposición en la plantilla de trabajo, sin embargo, diferían en la situación de iluminación. En la Ilustración 1 – 4 se aprecian 4 imágenes que indican la manera en la que se adquirieron las muestras.

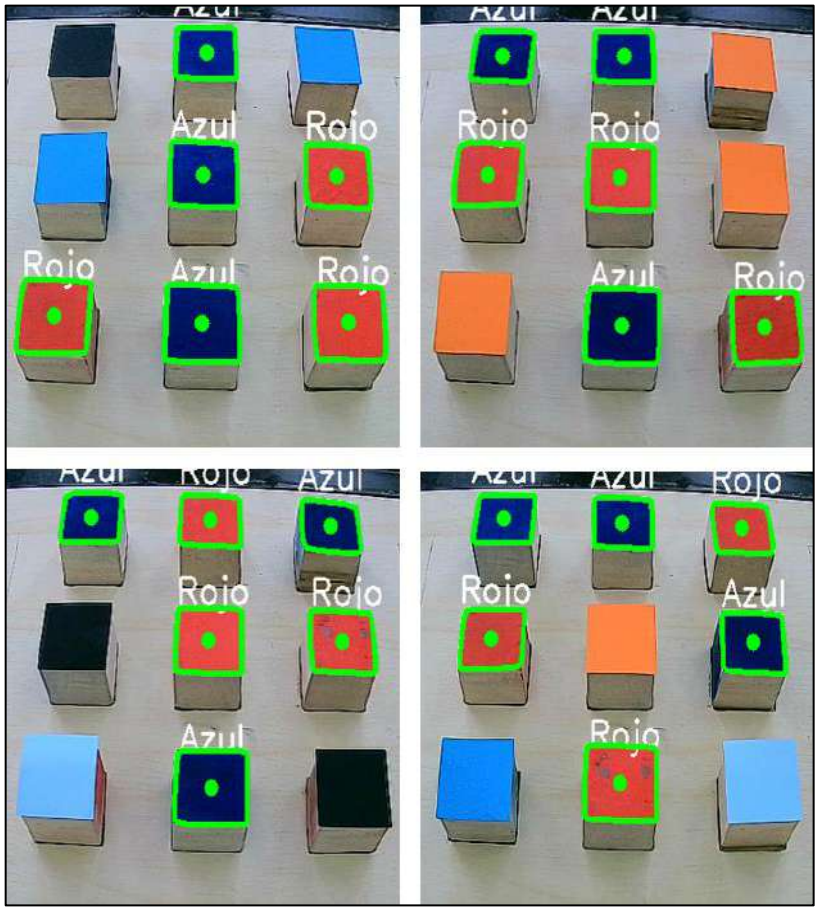


Ilustración 1 – 4: Detección de múltiples objetos en color rojo.

Realizado por: Pilco, 2023.

Los datos de las muestras analizadas se pueden apreciar en la Tabla 1 – 4 donde se puede ver que se han clasificado en función de la clase de iluminación en las 10 imágenes tomados para cada situación.

Tabla 1 – 4: Datos adquiridos para análisis de relación de variables.

Frame	**Iluminación Baja**				**Iluminación Media**				**Iluminación Alta**			
	Rojo / Azul		Otros Colores		Rojo / Azul		Otros colores		Rojo / Azul		Otros colores	
	Real	Software	Real	Software	Real	Software	Real	Software	Real	Software	Real	Software
1	6	7	3	2	6	5	3	4	6	6	3	3
2	6	6	3	3	6	6	3	3	6	6	3	3

3	6	7	3	2	6	6	3	3	6	6	3	3
4	6	6	3	3	6	5	3	4	6	6	3	3
5	6	6	3	3	6	6	3	3	6	6	3	3
6	6	9	3	0	6	7	3	2	6	6	3	3
7	6	5	3	4	6	6	3	3	6	6	3	3
8	6	5	3	4	6	6	3	3	6	6	3	3
9	6	7	3	2	6	6	3	3	6	6	3	3
10	6	7	3	2	6	6	3	3	6	6	3	3

Realizado por: Pilco, 2023.

De los datos observados en la Tabla 1 – 4 se realizó un desglose en cada muestra, en donde se representa en la clase CORRECTA cuando una muestra de color rojo o azul es identificada como tal. Mientras que, para cuando un objeto de otro color es identificado como uno de los definidos o un color rojo o azul no es identificado como tal, se determina el elemento en la clase ERRONEA.

La prueba de chi cuadrado para el análisis de la correlación de las variables se realizó en el software SPSS. Con esta aclaración, se obtenido la tabla de contingencia vista en la Tabla 2 – 4.

Tabla 2 – 4: Tabla de contingencia de frecuencias observadas.

		DETECCIÓN		**TOTAL**
		CORRECTA	ERRONEA	
ILUMINACIÓN	ALTA	90	0	90
	BAJA	81	9	90
	MEDIA	87	3	90
TOTAL		258	12	270

Realizado por: Pilco, 2023.

Esta tabla de contingencia permitió realizar el contraste con la hipótesis para determinar la relación de las variables. En este sentido, las hipótesis se definen de la siguiente manera:

Hipótesis nula H0.- Las variables analizadas son independientes.
Hipótesis alternativa H1.- Las variables analizadas son dependientes.

Cada valor de la tabla de contingencia representa una frecuencia real, para la prueba chi cuadrado es necesario calcular las frecuencias esperadas cuya expresión se define como se aprecia en la ecuación 5.

$$fe_{ij} = \frac{(Total\ fila\ i - ésima)(Total\ columna\ j - ésima)}{T} \quad (5)$$

Donde, *Total fila i-ésima* es la suma de los elementos de la fila i, *Total columna i-ésima* representa la suma de los elementos de la columna i, y *T* hace referencia al total de las muestras.

En este sentido se tiene las frecuencias esperadas en el caso de que las variables fueran independientes entre sí. Se aprecian dichas frecuencias en la Tabla 3 – 4.

Tabla 3 – 4: Tabla de contingencia de frecuencias esperadas.

		DETECCIÓN		TOTAL
		CORRECTA	ERRONEA	
ILUMINACIÓN	ALTA	86	4	90
	BAJA	86	4	90
	MEDIA	86	4	90
TOTAL		258	12	270

Realizado por: Pilco, 2023.

Para determinar la relación de las variables aceptando o rechazando la hipótesis alternativa se ha calculado el coeficiente de chi cuadrado de Pearson, el cual está definido de acuerdo con la ecuación 6.

$$\chi^2 = \sum_{ij} \frac{\left(f_{O_{ij}} - f_{e_{ij}}\right)^2}{f_{e_{ij}}} \quad (6)$$

Donde, $f_{O_{ij}}$ es la frecuencia observada para la ij-ésimo elemento de la tabla, $f_{e_{ij}}$representa la frecuencia esperada para el ij-ésimo elemento de la tabla.

Los resultados de la prueba chi cuadrado se muestran en la Tabla 4 – 4.

Tabla 4 – 4: Resultados de la prueba chi – cuadrado.

	Valor	gl	Significación asintótica
Chi – cuadrado de Pearson	10.988	2	0.004
Razón de verosimilitud	13.362	2	0.001
Número de casos válidos	270		

Realizado por: Pilco, 2023.

Para que la hipótesis alternativa sea aceptada χ^2 tiene que seguir una distribución chi cuadrado en función de los grados de libertad relacionados con el tamaño de la tabla. De esta manera, se asume un nivel crítico igual a 0.05 el cual presenta según la tabla de distribución Chi Cuadrado vista en el Anexo 1, un valor de 5.9915, mientras que la prueba refleja un valor de 10.988. Por lo tanto, para aceptar H_1 se tiene que cumplir lo visto en la ecuación 7.

$$\chi^2_{Experimental} > \chi^2_{Crítico} \tag{7}$$

Donde, $\chi^2_{Crítico}$ es el coeficiente chi cuadrado critico definido para 0.05 de significancia y $\chi^2_{Experimental}$ es el coeficiente obtenido de la prueba realizada.
Remplazando los términos de la ecuación 6 con sus valores numéricos para este caso se tiene que 10.988 es mayor 5.9915, por lo que se acepta con certeza la hipótesis alternativa.

4.3 Validación del sistema clasificación

Para determinar la validez del sistema de detección se realizaron 2 pruebas de identificación de color. Una de ellas exclusivamente para el color azul y otra para el color rojo. Cabe mencionar que los patrones de medición se encuentran en función de la cantidad de objetos reales del color en cada imagen.

4.3.1 Detección del color rojo

Para esta prueba analizaron 25 imágenes cuyos objetos son de color rojo y de colores diferentes a este. Con este antecedente se aprecian los datos adquiridos en la Tabla 5 – 4.

Tabla 5 – 4: Análisis de errores absoluto y relativo para la detección del color rojo.

Frame	Objetos Reales	Objetos detectados	Error Absoluto	Error Relativo %
1	9	9	0	0%
2	8	8	0	0%
3	7	7	0	0%
4	6	6	0	0%
5	5	6	1	20%
6	3	4	1	33,3%
7	2	2	0	0%
8	1	2	1	100%
9	4	4	0	0%
10	6	7	1	16,7%

11	3	3	0	0%
12	3	3	0	0%
13	3	3	0	0%
14	3	3	0	0%
15	3	3	0	0%
16	5	5	0	0%
17	4	5	1	25%
18	3	3	0	0%
19	5	6	1	20%
20	5	5	0	0%
21	5	5	0	0%
22	2	2	0	0%
23	2	2	0	0%
24	2	2	0	0%
25	1	1	0	0%
		Promedios:	0.4	8,6%

Realizado por: Pilco, 2023.

Con dependencia en la cantidad de objetos reales en color rojo de cada imagen se obtuvo un error absoluto de 0.4 y un error relativo del 8,6%. Este último según lo establecido por (Santo et al. 2018) es elevado, sin embargo, se encuentra dentro del rango definido por el autor, por lo que la detección del color rojo es aceptable.

4.3.2 Detección del color Azul

De igual manera que para el color rojo, se adquirieron 25 imágenes con objetos en color azul y otros colores distintos. Los datos obtenidos se aprecian en la Tabla 6 – 4.

Tabla 6 – 4: Análisis de errores absoluto y relativo para la detección del color azul.

Frame	Objetos Reales	Objetos Detectados	Error Absoluto	Error Relativo%
1	8	8	0	0%
2	7	7	0	0%
3	6	6	0	0%
4	5	5	0	0%
5	4	4	0	0%
6	3	3	0	0%
7	2	2	0	0%
8	1	1	0	0%
9	5	7	2	40%
10	3	5	2	66.67%
11	5	5	0	0%

12	6	6	0	0%
13	4	4	0	0%
14	3	3	0	0%
15	4	4	0	0%
16	4	2	2	50%
17	4	4	0	0%
18	4	2	2	50%
19	4	3	1	25%
20	4	4	0	0%
21	6	6	0	0%
22	3	3	0	0%
23	2	2	0	0%
24	1	1	0	0%
25	6	6	0	0%
		Promedios:	0.36	9.27%

Realizado por: Pilco, 2023.

La medida patrón está en función de la cantidad de objetos reales en el color azul, entonces de acuerdo con los datos vistos en la Tabla 5 – 4, el error absoluto en la detección de este color es de 0.36. Además, presenta un error relativo del 9.27%.

4.4 Precisión del sistema de sistema de clasificación

Para evaluar la precisión del sistema de control para la clasificación de objetos se considera la precisión en la detección de los colores definidos y la precisión de clasificación.

4.4.1 Precisión de la detección del color

Dado a que la variable ***detección*** analizada es cualitativa, el método más apropiado para determinar la precisión de detección es mediante la utilización de matrices de confusión, ya que, como se mencionó en la sección de consideraciones generales, es una herramienta que permite visualizar el desempeño del algoritmo de visión artificial desarrollado. En este sentido, se ha definido el análisis independiente de cada color en función de las métricas definidas para la evaluación mediante matrices de confusión.

4.4.1.1 Color azul

Se consideraron las 25 imágenes con 9 muestras empleadas para la validación en el apartado anterior para un total de 225 muestras para el análisis. De manera similar al análisis con la prueba

de chi cuadrado, cada muestra contenida en una imagen ha sido representada con el numero 1 si la detección es correcta y con un cero si la detección es errónea para su análisis en el Python. El conjunto de datos analizados se puede ver en la Tabla 7 – 4.

Tabla 7 – 4: Muestras analizadas para el color azul.

Frame	**Objetos Detectados**			
	Color Azul		Otros colores	
	Real	Software	Real	Software
1	8	8	1	1
2	7	7	2	2
3	6	6	3	3
4	5	5	4	4
5	4	4	5	5
6	3	3	6	6
7	2	2	7	7
8	1	1	8	8
9	5	9	4	0
10	3	9	6	0
11	5	5	4	4
12	6	6	3	3
13	4	4	5	5
14	3	3	6	6
15	4	4	5	5
16	4	1	5	8
17	4	4	5	5
18	4	2	5	7
19	4	2	5	7
20	4	4	5	5
21	6	6	3	3
22	3	3	6	6
23	2	2	7	7
24	0	0	9	9
25	6	6	3	3
Totales	104	103	121	122

Realizado por: Pilco, 2023.

Los totales de las muestras reales en color azul, muestras de colores diferentes y las detectadas por el software permiten interpretar de mejor manera los resultados de la matriz de confusión. Este análisis para el color azul se presenta en la Ilustración 2 – 4.

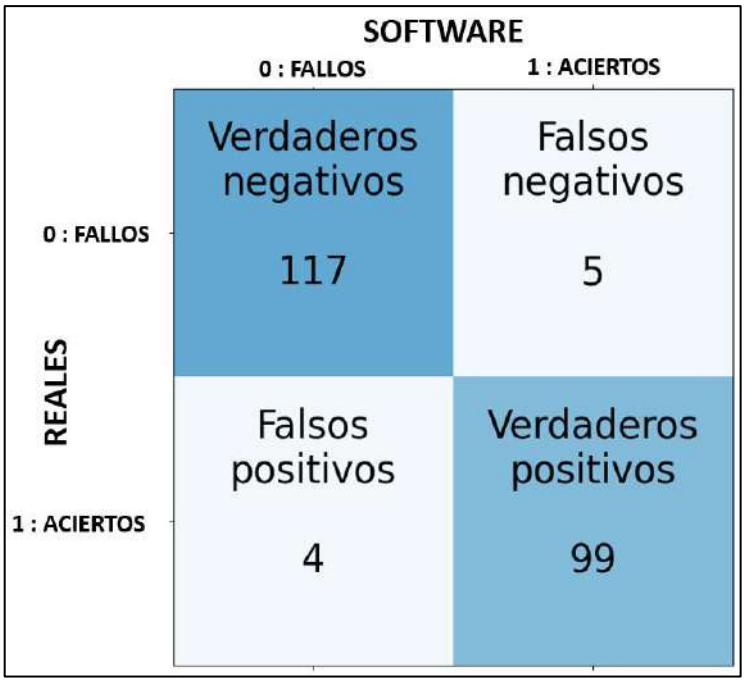


Ilustración 2 – 4: Matriz de confusión para el color azul.

Realizado por: Pilco, 2023.

En base con Tabla 7 – 4 se analizaron 104 objetos de color azul y 121 de colores diferentes. La matriz de confusión muestra 99 verdaderos positivos o 99 objetos de color azul que fueron detectados como azules, 5 falsos negativos que representan los objetos de color azul que no fueron detectados como tal, 117 verdaderos negativos que indican que 117 de los 121 fueron detectados como de colores diferentes. Finalmente se tienen 4 falsos positivos que indican los objetos de distinto color que fueron detectados como azules. Las métricas de la evaluación mediante matrices de confusión para el color azul se aprecian en la Tabla 8 – 4.

Tabla 8 – 4: Resultados de la matriz de confusión.

Métrica	Valor
Exactitud (Accuracy)	96%
Precisión (Precision)	96%
Sensibilidad (Recall)	95%
F1 Score	96%

Realizado por: Pilco, 2023.

4.4.1.2 Color Rojo

De manera análoga al análisis realizado para el color azul, el análisis de la detección para el color rojo se realizó a partir de los datos recolectados en la Tabla 9 – 4.

Tabla 9 – 4: Datos recopilados para análisis de detección del color Rojo.

Frame	**Objetos Detectados**			
	Color Rojo		Otros colores	
	Real	Software	Real	Software
1	9	9	0	0
2	8	8	1	1
3	7	7	2	2
4	6	6	3	3
5	5	6	4	3
6	3	5	6	4
7	2	3	7	6
8	1	2	8	7
9	4	5	5	4
10	6	8	3	1
11	3	3	6	6
12	3	3	6	6
13	3	3	6	6
14	3	3	6	6
15	3	3	6	6
16	5	6	4	3
17	4	7	5	2
18	3	3	6	6
19	5	8	4	1
20	5	5	4	4
21	5	7	4	2
22	2	3	7	6
23	2	2	7	7
24	2	2	7	7
25	0	1	9	8
Totales	100	106	125	119

Realizado por: Pilco, 2023.

De manera que se puedan interpretar los resultados de la matriz de confusión realizada para el color rojo se han extraído los totales de las muestras reales en color rojo y en diferentes colores. Los resultados se pueden ver en la Ilustración 3 – 4.

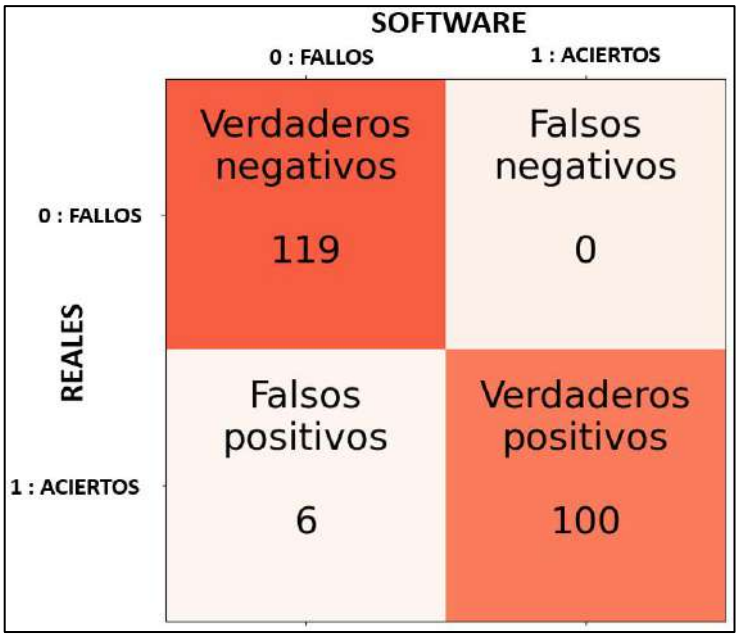


Ilustración 3 – 4: Matriz de confusión para el color rojo.

Realizado por: Pilco, 2023.

La matriz refleja 100 verdaderos positivos que señalan que 100 de los 100 objetos reales de color rojo fueron detectados como tal. Un valor de 0 falsos negativos que indican que todos los objetos de color rojo fueron detectados correctamente. Existen 119 verdaderos negativos, es decir, se detectaron 119 objetos de otro color de los 125 totales, en consecuencia, se tienen 6 falsos positivos que indican el reconocimiento de objetos de otro color como rojos. Los resultados de acuerdo con las métricas resultantes de la aplicación de matrices de confusión se aprecian en la Tabla 10 – 4.

Tabla 10 – 4: Resultados de la matriz de confusión.

Métrica	Valor
Exactitud (Accuracy)	97%
Precisión (Precision)	94%
Sensibilidad (Recall)	100%
F1 Score	97%

Realizado por: Pilco, 2023.

4.4.2 Precisión de clasificación

Para definir la precisión se evaluó la clasificación de los objetos según su color característico. Para ello, se consideró el posicionamiento, agarre, traslado y colocación de los objetos en la

sección azul o roja según corresponde desde su ubicación en la plantilla de trabajo. Para determinar que la clasificación sea exitosa, el objeto trasladado de color azul o rojo tiene que ser colocado en la sección de color azul (lado derecho) y rojo (lado izquierdo) respectivamente. Se aprecia el proceso de clasificación en la Ilustración 4 – 4 en donde se evidencian las cuatro consideraciones.

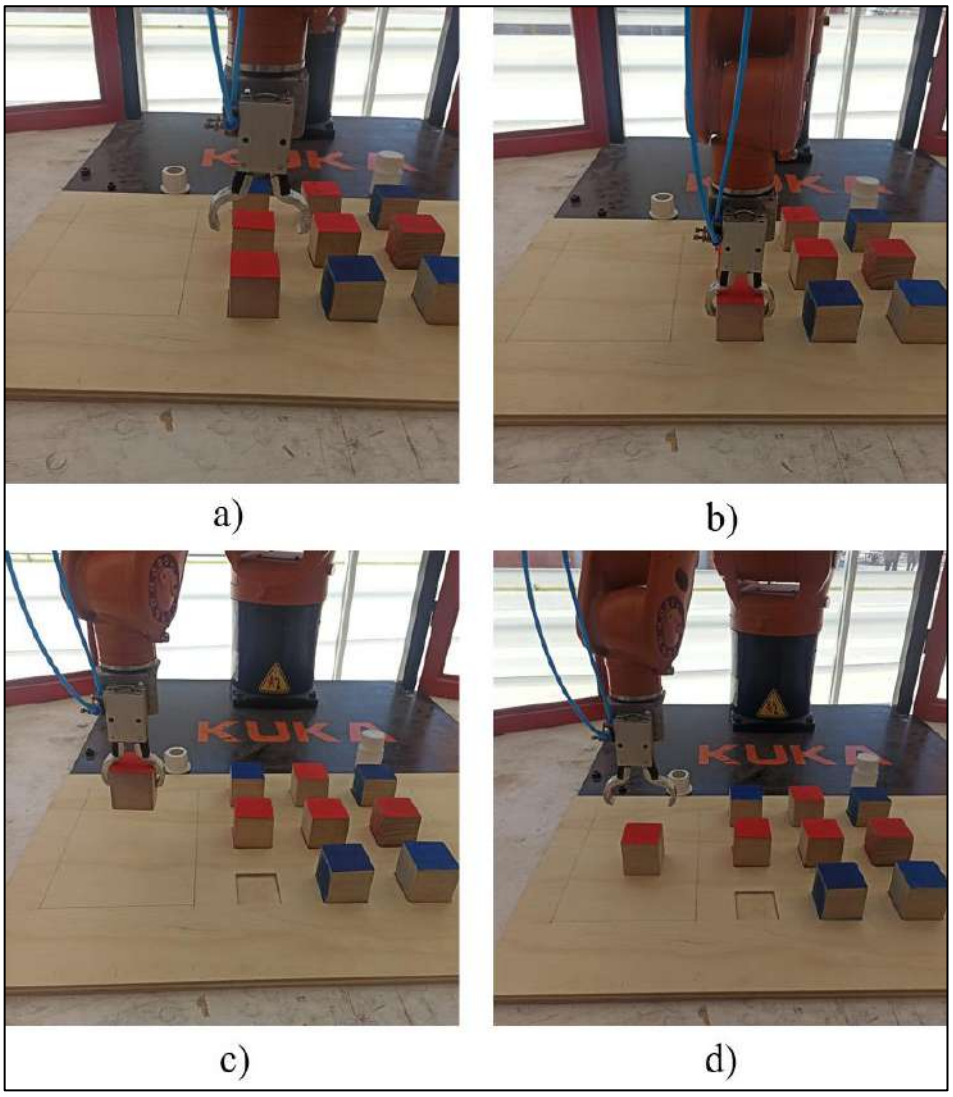


Ilustración 4 – 4: a) Posicionamiento. b) Agarre. c) Traslado. d) Colocación.

Realizado por: Pilco, 2023.

Como se muestra en la Ilustración 4 – 4 el traslado para un objeto de color rojo se realizó de manera exitosa. La prueba se realizó con los escenarios definidos para las pruebas de detección de color. En este sentido, la precisión del sistema para la clasificación de objetos es exactamente la misma que la obtenida en la sección de la detección del color, ya que la asignación de objetivos de movimiento depende de la detección del color de los objetos.

4.5 Evaluación de los tiempos de clasificación

Se consideró el tiempo que emplea el manipulador en el traslado de cada objeto desde la detección de su color hasta su colocación exitosa en la sección de color correspondiente. Para ello, se emplearon cuatro niveles de velocidad del robot en el modo de operación automático, 5, 10, 30 y 50%, para la realización de dos pruebas de clasificación con los objetos distribuidos de la siguiente manera:

Prueba 1. – Se consideran cinco objetos de color **Rojo** y cuatro de color **Azul**.
Prueba 2. – Se consideran cinco objetos de color **Azul** y cuatro de color **Rojo**.

Esta distribución ha sido considerada porque representa un equilibrio de los objetos según su color, en función de la cantidad de espacios existentes. Sin embargo, no existe variación si se consideran diferentes distribuciones del color en los objetos, es decir, no se tiene una variación significativa en el tiempo total de clasificación como se ve en los resultados mostrados en la Tabla 8 – 4.

Tabla 11 – 4: Evaluación del tiempo de clasificación.

Prueba	Velocidad de Programación	Velocidad del robot	Velocidad Final	Tiempo objetos Rojos	Tiempo objetos Azules	Tiempo total
1	100%	5%	5%	122.83 s	94.98 s	217.81 s
	100%	10%	10%	62.51 s	54.38 s	116.89 s
	100%	30%	30%	26.63 s	20.75 s	47.38 s
	100%	50%	50%	18.9 s	15.11 s	34.01 s
2	100%	5%	5%	98.20 s	120.48 s	218.68 s
	100%	10%	10%	53.75 s	60.65 s	114.4 s
	100%	30%	30%	21.61 s	24.7 s	46.32 s
	100%	50%	50%	14.24 s	18.66 s	32.9

Realizado por: Pilco, 2023.

La velocidad final se obtuvo del producto de la velocidad del robot con la velocidad de programación. Como se puede observar en la Tabla 8 – 4, los tiempos de clasificación totales al mismo nivel de velocidad, por ejemplo, al 10% tanto en la prueba 1 como en la 2, son bastantes similares 116, 89 segundos y 114.4 segundos respectivamente. Sin embargo, la operación adecuada del gripper sugiere un manejo de velocidades no superiores al 30%.

4.6 Puesta en marcha del sistema

4.6.1 Preparación del escenario

En la Ilustración 5 – 4 se puede apreciar la ubicación de los cubos en cada posición de la plantilla de trabajo. Cabe mencionar que la selección de un cubo de cualquier color es aleatoria para cada posición.

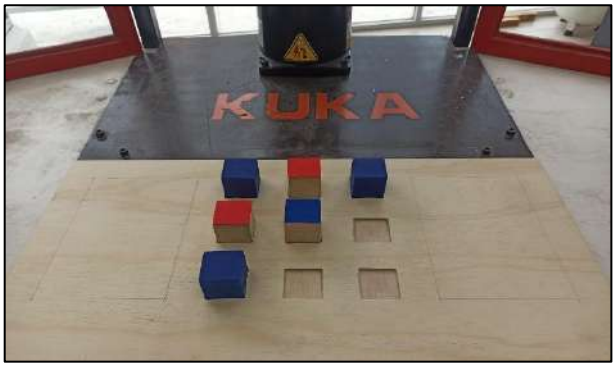


Ilustración 5 – 4: Ubicación de los objetos sobre la plantilla de trabajo.
Realizado por: Pilco, 2023.

4.6.2 Agarre y traslado del objeto

El manipulador parte de la posición ***HOME*** hacia la posición de ***WAIT*** a la espera de la detección de color del objeto ubicado en la plantilla. Una vez generada la información del color de ese objeto el manipulador se mueve al objetivo **APROX** para ubicarse correctamente sobre el objeto luego se mueve al objetivo **PICK** para activar la pinza de agarre y trasladar el objeto a los objetivos **ROJO** o **AZUL** según corresponda. En la Ilustración 6 – 4 se aprecia el proceso de clasificación del sistema.

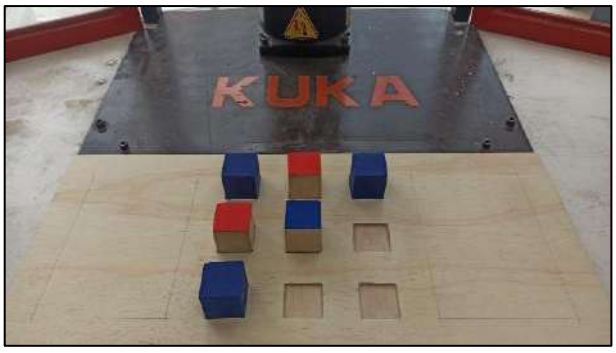


Ilustración 6 – 4: Agarre y transporte de un objeto rojo.
Realizado por: Pilco, 2023.

CAPÍTULO V

5 CONCLUSIONES Y RECOMENDACIONES

5.1 Conclusiones

- Con la prueba chi cuadrado para analizar la correlación de las variables del sistema se determinó mediante el planteo de hipótesis que las variables son dependientes, es decir, la detección CORRECTA o ERRONEA de un objeto depende directamente de la iluminación incidente sobre el espacio de trabajo, por lo tanto, conforme disminuye la iluminación, la estimación de objetos detectados de manera correcta también disminuye.

- El análisis del error relativo para la detección tanto del color rojo como del color azul se han determinado como aceptables ya que presentan un error del 8,6 y 9,27% respectivamente. Por lo tanto, en base con la bibliografía revisada se establece un rango de aceptabilidad para la detección de los colores definidos de entre el 5 y 10%.

- Para el algoritmo de detección del color, la métrica de precisión y sensibilidad vistos en las matrices de confusión, sugieren el enfoque en los verdaderos positivos (objetos en los colores definidos) y verdaderos negativos (objetos en colores distintos) respectivamente, por lo que para ambos colores se presentaron porcentajes superiores al 94% lo cual indica que el sistema maneja adecuadamente la detección de los colores azul y rojo.

- Dada la configuración del sistema, la precisión al momento de trasladar los objetos para su clasificación resultó igual que la precisión en la detección del color, es decir mayor al 94%, dado a que las instrucciones correspondientes al movimiento del robot son asignadas en función del color que el algoritmo de visión detecta para cada objeto ubicado en la plantilla.

- El sistema funciona de manera adecuada sin importar el color del objeto que se ubique en cada posición de la plantilla por lo que los tiempos de clasificación a un nivel de velocidad determinado son similares independiente de la cantidad de objetos de un color u otro. Sin embargo, hay que considerar la operabilidad del sistema ya que, a niveles del 5, 10 y 30% de velocidad, el sistema trabajó perfectamente, mientras que en velocidades superiores al 50% se presentaron problemas en el agarre de los objetos y en la estabilidad del brazo con respecto a la jaula de seguridad.

- La guía de conexión entre el ordenador y el controlador del brazo robot desarrollada, permitió establecer de manera exitosa la comunicación entre el hardware y software del sistema. En este sentido, se pudieron aplicar todas las técnicas de visión artificial y la creación de objetivos de movimiento a través del lenguaje Python en el entorno de programación RoboDK. Posteriormente tras la realización de diversas pruebas, se ejecutó el sistema de control para la clasificación de objetos adecuadamente.

- Los sistemas de visión vistos en el estado del arte sugieren soluciones sofisticadas y relativamente complejas dado a factores para considerar como el costo de los elementos que integran o la complejidad de su implementación. En este sentido, el sistema de control para la clasificación de objetos en color rojo y azul representa una solución adecuada por la flexibilidad al momento de su instalación, bajo costo, adaptabilidad y robustez demostrada con la precisión y sensibilidad del sistema de visión artificial.

- El sistema desarrollado cumple con todos los requerimientos y parámetros considerados teniendo en cuenta la capacidad del reconocimiento del color de los objetos mostrando la dependencia del sistema con la iluminación para su funcionamiento óptimo, la extracción de las características de los objetos desde su color hasta su posición y la capacidad de trasladar de manera adecuada los objetos en función de la configuración definida para la puesta en marcha del sistema.

5.2 Recomendaciones

- Utilizar el brazo robótico hasta el 30% de velocidad en el modo automático para mantener una buena operabilidad de los elementos finales de manipulación en complementación con el diseño de estructuras de seguridad que no limiten al robot, es decir, que permitan el movimiento libre en toda el área que físicamente puede cubrir teniendo en cuenta factores como la fuerza y torque que el brazo robótico puede desarrollar.

- Implementar una red neuronal que mediante su entrenamiento permita la identificación de todo el espectro de color, permitiendo obtener mayor versatilidad y flexibilidad del sistema para empelarlo en aplicaciones más complejas relacionadas con el picking robótico en industrias de cualquier naturaleza.

- Establecer una conectividad inalámbrica online que permita la programación de manera remota para tener el acceso desde cualquier lugar con el fin de maximizar la seguridad de sus programadores y la integración con otros sistemas.

- Emplear otras configuraciones de cámara para la interpretación y reconocimiento de la profundidad en el espacio y elementos de trabajo con el fin de conseguir la utilización del sistema en aplicaciones de mayor complejidad como con la integración de plataformas móviles para evadir obstáculos.

BIBLIOGRAFÍA

ACUÑA, Miguel Ángel, 2020. *Diseño de un sistema de visión artificial para la clasificación de limón utilizando Raspberry PI*.

AGUILAR CERÓN, Oscar y HERNÁNDEZ HERNÁNDEZ, José Luís, 2008. *Programa de Maple para la determinación de la cinemática directa e inversa en lazo abierto*.

ARÉVALO, V M, GONZÁLEZ, J y AMBROSIO, G, 2002. *LA LIBRERÍA DE VISIÓN ARTIFICIAL OPENCV APLICACIÓN A LA DOCENCIA E INVESTIGACIÓN 1* [en línea]. Recuperado a partir de: www.sourceforge.net

BACH. LA MADRID TÁVARA, Luis Eduardo, 2019. *Implementación de un algoritmo de control de calidad para la selección de productos agrícolas utilizando visión artificial*.

BARRIETOS, Antonio et al., 2007. *FUNDAMENTOS DE ROBÓTICA*. Segunda.

BCNVISION, 2017. *Sistemas de iluminación para aplicaciones de visión artificial* [en línea]. Recuperado a partir de: https://www.bcnvision.es/blog-vision-artificial/iluminacion-vision-artificial2/ [consultado 7 mayo 2023].

BORA, Dibya Jyoti, KUMAR GUPTA, Anil y KHAN, Fayaz Ahmad, 2008. *Comparing the Performance of L*A*B* and HSV Color Spaces with Respect to Color Image Segmentation* [en línea]. Recuperado a partir de: www.ijetae.com

BRADSKI, Gary R. y KAEHLER, Adrián, 2008. *Learning OpenCV: computer vision with the OpenCV library*. O'Reilly. ISBN 9780596516130.

BRITO, Marcelo, 2019. *Diseño y construcción de un brazo robótico con visión artificial*.

CÁCERES, Jesús, 2006. *La visión artificial y las operaciones morfológicas en imágenes binarias*. Universidad de Castilla-La Mancha. Departamento de Sistemas Informáticos. ISBN 8468995606.

CIENFUEGOS, María de los Ángeles y CIENFUEGOS, Adriana, 2016. *Lo cuantitativo y cualitativo en la investigación. Un apoyo a su enseñanza*.

COGNEX, 2022. Iluminación de visión artificial. . . 25 octubre 2022.

DELGADO, Sergio, 2023. *Aprende Python*.

EDSROBOTICS, 2020. Cámaras industriales: tipos y funcionamiento. 8 octubre 2020.

SUCAR, Enrique y GÓMEZ, Giovani, sin fecha. *Visión Computacional*.

ESNECA BUSINESS SCHOOL, 2022. ¿Cómo se hace la programación de robots industriales? [en línea]. 21 febrero 2022. Recuperado a partir de: https://www.esneca.lat/blog/programacion-robots-industriales-clasificacion/ [consultado 21 enero 2023].

ESNECA, 2022. ¿Cómo se hace la programación de robots industriales? [en línea]. 21 febrero 2022. Recuperado a partir de: https://www.esneca.lat/blog/programacion-robots-industriales-clasificacion/ [consultado 24 abril 2023].

ESTRADA, Adrián, 2004. *PROTOCOLOS TCP/IP DE INTERNET INTERNET* [en línea]. Recuperado a partir de: http://www.revista.unam.mx/vol.5/num8/art51/art51.htm

COMÍNS, Fernando, 2018. Lenguajes de programación de robots industriales. [en línea]. pp. 6–8. Recuperado a partir de: https://www.gestiopolis.com/lenguajes-de-programacion-de-robots-industriales-del-control-numerico-a-los-frameworks-roboticos/ [consultado 23 abril 2023].

FLORES, Ángel Miguel et al., 2017. *Detección de Objetos a Color en Tiempo Real con Técnicas de Visión Artifical y Arduino* [en línea]. Recuperado a partir de: www.ecorfan.org/spain

FÚQUENE, Héctor Julio, 2011. *Implementación cliente servidor mediante sockets*.

GALBIS, Solanes et al., sin fecha. *Control de robots industriales con realimentación visual: Image-Based Visual Servoing*.

GARATEJO, Juan Miguel y RAUDALES, Víctor Manuel, 2021. *Diseño de un módulo para la validación del algoritmo de Denavit - Hartenberg de un brazo antropomórfico aplicado al área de Automatización*.

GARCÍA, Alberto, 2017. Diseño de algoritmos de control visual basado en imagen y basado en posición para sistemas robotizados y validación mediante simulación y experimentación real.

GARCÍA, Rodrigo, GORDILLO, Rodolfo y MORILLO, Diego, 2010. *DISEÑO DEL SISTEMA DE CONTROL DEL BRAZO ROBOTICO CRS A255 UTILIZANDO LA PLATAFORMA KINETIX DE ALLEN BREDLEY*.

GARRIDO, Antonio, 2006. *Fundamentos de programación en C++*. 1. Madrid.

GAZEBO, 2023. Gazebo. [en línea]. 24 abril 2023. Recuperado a partir de: https://staging.gazebosim.org/about [consultado 23 abril 2023].

GONZÁLES, Ana et al., 2006. Técnicas y Algoritmos básicos de Visión Artificial. pp. 42–43.

GONZALEZ, Rafael C. y WOODS, Richard E. (Richard Eugene), 2002. *Digital image processing*. ISBN 0201180758.

GUAMÁN, Rodrigo Luis, 2015. *Utilización de métodos de visión artificial para PC como apoyo en la automoción*.

GUTIERREZ, José Rubén y CIFUENTES, Giovanny, 2015. *Pick and Place para montaje de componentes de montaje superficial SMD*. México: Instituto Politécnico Nacional.

HUÉRFANO, Yoleidy et al., 2015. *Métodos de segmentación de imágenes cardiacas: Fundamentos y alcance*.

INTEREMPRESAS, 2018. Automatización en la Industria 4.0. . . 29 noviembre 2018.

IÑIGO, Rafael y VIDAL, Enric, 2002. *Robots industriales manipuladores*. Primera. Barcelona.

ISO, 2022. ISO. *ISO 8373:2021*. 7 noviembre 2022.

JAVIER, César et al., 2016. *Diseño de un Sistema de Visión Artificial para la Clasificación de Chirimoyas basado en medidas*.

KUKA DEUTSCHLAND GMBH, 2018. *KR C4 compact Specification*.

KUKA, 2022. KUKA KR C4 | KUKA AG. [en línea]. 2022. Recuperado a partir de: https://www.kuka.com/es-es/productos-servicios/sistemas-de-robot/unidades-de-control-del-robot/kr-c4 [consultado 6 noviembre 2022].

KUKA, 2023. KUKA.Sim. [en línea]. 24 abril 2023. Recuperado a partir de: https://www.kuka.com/es-es/productos-servicios/sistemas-de-robot/software/planificaci%C3%B3n-proyecci%C3%B3n-servicio-seguridad/kuka,-d-,sim [consultado 23 abril 2023].

MADUELL I GARCÍA, Eloi, sin fecha. *Visión artificial Eloi Maduell i García PID_00184756* [en línea]. Catalunya. Recuperado a partir de: https://www.exabyteinformatica.com/uoc/Informatica/Diseno_de_interaccion/Diseno_de_interaccion_(Modulo_5).pdf [consultado 4 noviembre 2022].

MAGRO, Rafael, 2013. *BINARIZACIÓN DE IMÁGENES DIGITALES Y SU ALGORITMIA COMO HERRAMIENTA APLICADA A LA ILUSTRACIÓN ENTOMOLÓGICA*.

MIN. DE EDUCACION, 2012. Aplicación práctica de la visión artificial en el control de procesos industriales.

MIRANDA, Roger, 2016. *Cinemática y dinámica de robots manipuladores*. Primera. México: Alfaomega.

MONTALVO, Martín, 2009. *Técnicas de visión estereoscópica para determinar la estructura tridimensional de la escena*. Madrid.

MÜHE, Henrik et al., 2010. On reverse-engineering the KUKA Robot Language. [en línea]. pp. 1–2. Recuperado a partir de: http://arxiv.org/abs/1009.5004

OCHOA, Diego Alveiro, 2013. Grounded Theory as a methodology for the integration of structural and processual analysis in the investigation of Social Representations.

PIZARRO, Diego, CAMPOS, Pedro y TOZZI, Clésio, 2005. *COMPARACIÓN DE TÉCNICAS DE CALIBRACIÓN DE CÁMARAS DIGITALES*.

POMARES, Jorge, ANDRÉS, Francisco y HERÍAS, Candelas, 2009. *Control visual*.

PRUSIEL, Ignacio, 2018. Control por realimentación visual de un brazo robot para tareas de agarre.

RAMÍREZ, José Luis y RUBIANO, Astrid, 2012. Modelamiento matemático de la cinemática directa e inversa de un robot manipulador de tres grados de libertad. pp. 47–49.

RAMÍREZ, Daniel, 2014. *Introducción a la visión artificial*.

REVISTA DE ROBOTS, 2022. Gripper y pinzas para robots industriales. [en línea]. 4 septiembre 2022. Recuperado a partir de: https://revistaderobots.com/robots-y-robotica/gripper-y-pinzas-para-robots-industriales/?cn-reloaded=1 [consultado 24 abril 2023].

ROBODK, 2023a. Guía Básica. [en línea]. 24 abril 2023. Recuperado a partir de: https://robodk.com/doc/es/Basic-Guide.html#Start [consultado 23 abril 2023].

ROBODK, 2023b. Controladores de Robot. [en línea]. 24 abril 2023. Recuperado a partir de: https://robodk.com/doc/es/Robot-Drivers.html#RobotDrivers [consultado 23 abril 2023].

ROBOTER GMBH, Kuka, 2013. Programación de robots 1 KUKA System Software 8 Documentación para la formación. DOI 10.12.2013.

ROBOTER GMBH, Kuka, 2015. *KR AGILUS sixx With W and C Variants Operating Instructions*.

ROSAS, Leonel et al., 2017. *Robot clasificador de objetos de color utilizando técnicas de filtrado RGB* [en línea]. Recuperado a partir de: www.ecorfan.org/spain

RUIZ, Luis, 2020. *Aplicación de filtros morfológicos en imágenes*.

SÁNCHEZ, Manuel José, 2016. *Análisis de Calidad Cartográfica mediante el estudio de la Matriz de Confusión*.

SANTO, Marisa y LECUMBERRY, Graciela, 2018. *El proceso de medición - Análisis y comunicación de datos experimentales*. Buenos Aires: Universidad Nacional de Río Cuarto.

TINOCO, Oscar, 2008. Una aplicación de la prueba chi cuadrado con SPSS.

VARGAS, Victor, 2010. *Sistema de Visión Artificial para el control de calidad en piezas cromadas*.

VÉLEZ, C, OROZCO, F y ÁLVAREZ, M, 2014. Vista de Control de los movimientos de un brazo robótico desde un computador utilizando software libre de control y comunicación inalámbrica con módulos X-BEE. pp. 1–8.

VÉLEZ, José et al., 2003. *Visión por computador*. 2. Madrid. ISBN 8497720695.

VIALA, Carlos Ricolfe y SALMERÓN, Antonio José, 2008. Complete method for camera calibration using a two-dimensional template. *RIAI - Revista Iberoamericana de Automática e Informática Industrial*. Vol. 5, número 1, pp. 93–101. DOI 10.1016/s1697-7912(08)70126-2.

VIVES, Luis Alberto et al., 2014. VISIÓN ARTIFICIAL: APLICACIÓN DE FILTROS Y SEGMENTACIÓN EN IMÁGENES DE HOJAS DE CAFÉ. pp. 4–5.

ANEXOS

Anexo A: Tabla de Chi cuadrado.

v/p	0,001	0,0025	0,005	0,01	0,025	0,05	0,1	0,15	0,2	0,25	0,3	0,35	0,4	0,45	0,5
1	10,8274	9,1404	7,8794	6,6349	5,0239	3,8415	2,7055	2,0722	1,6424	1,3233	1,0742	0,8735	0,7083	0,5707	0,4549
2	13,8150	11,9827	10,5965	9,2104	7,3778	5,9915	4,6052	3,7942	3,2189	2,7726	2,4079	2,0996	1,8326	1,5970	1,3863
3	16,2660	14,3202	12,8381	11,3449	9,3484	7,8147	6,2514	5,3170	4,6416	4,1083	3,6649	3,2831	2,9462	2,6430	2,3660
4	18,4662	16,4238	14,8602	13,2767	11,1433	9,4877	7,7794	6,7449	5,9886	5,3853	4,8784	4,4377	4,0446	3,6871	3,3567
5	20,5147	18,3854	16,7496	15,0863	12,8325	11,0705	9,2363	8,1152	7,2893	6,6257	6,0644	5,5731	5,1319	4,7278	4,3515
6	22,4575	20,2491	18,5475	16,8119	14,4494	12,5916	10,6446	9,4461	8,5581	7,8408	7,2311	6,6948	6,2108	5,7652	5,3481
7	24,3213	22,0402	20,2777	18,4753	16,0128	14,0671	12,0170	10,7479	9,8032	9,0371	8,3834	7,8061	7,2832	6,8000	6,3458
8	26,1239	23,7742	21,9549	20,0902	17,5345	15,5073	13,3616	12,0271	11,0301	10,2189	9,5245	8,9094	8,3505	7,8325	7,3441
9	27,8767	25,4625	23,5893	21,6660	19,0228	16,9190	14,6837	13,2880	12,2421	11,3887	10,6564	10,0060	9,4136	8,8632	8,3428
10	29,5879	27,1119	25,1881	23,2093	20,4832	18,3070	15,9872	14,5339	13,4420	12,5489	11,7807	11,0971	10,4732	9,8922	9,3418
11	31,2635	28,7291	26,7569	24,7250	21,9200	19,6752	17,2750	15,7671	14,6314	13,7007	12,8987	12,1836	11,5298	10,9199	10,3410
12	32,9092	30,3182	28,2997	26,2170	23,3367	21,0261	18,5493	16,9893	15,8120	14,8454	14,0111	13,2661	12,5838	11,9463	11,3403
13	34,5274	31,8830	29,8193	27,6882	24,7356	22,3620	19,8119	18,2020	16,9848	15,9839	15,1187	14,3451	13,6356	12,9717	12,3398
14	36,1239	33,4262	31,3194	29,1412	26,1189	23,6848	21,0641	19,4062	18,1508	17,1169	16,2221	15,4209	14,6853	13,9961	13,3393
15	37,6978	34,9494	32,8015	30,5780	27,4884	24,9958	22,3071	20,6030	19,3107	18,2451	17,3217	16,4940	15,7332	15,0197	14,3389
16	39,2518	36,4555	34,2671	31,9999	28,8453	26,2962	23,5418	21,7931	20,4651	19,3689	18,4179	17,5646	16,7795	16,0425	15,3385
17	40,7911	37,9462	35,7184	33,4087	30,1910	27,5871	24,7690	22,9770	21,6146	20,4887	19,5110	18,6330	17,8244	17,0646	16,3382
18	42,3119	39,4220	37,1564	34,8052	31,5264	28,8693	25,9894	24,1555	22,7595	21,6049	20,6014	19,6993	18,8679	18,0860	17,3379
19	43,8194	40,8847	38,5821	36,1908	32,8523	30,1435	27,2036	25,3289	23,9004	22,7178	21,6891	20,7638	19,9102	19,1069	18,3376
20	45,3142	42,3358	39,9969	37,5663	34,1696	31,4104	28,4120	26,4976	25,0375	23,8277	22,7745	21,8265	20,9514	20,1272	19,3374
21	46,7963	43,7749	41,4009	38,9322	35,4789	32,6706	29,6151	27,6620	26,1711	24,9348	23,8578	22,8876	21,9915	21,1470	20,3372
22	48,2676	45,2041	42,7957	40,2894	36,7807	33,9245	30,8133	28,8224	27,3015	26,0393	24,9390	23,9473	23,0307	22,1663	21,3370
23	49,7276	46,6231	44,1814	41,6383	38,0756	35,1725	32,0069	29,9792	28,4288	27,1413	26,0184	25,0055	24,0689	23,1852	22,3369
24	51,1790	48,0336	45,5584	42,9798	39,3641	36,4150	33,1962	31,1325	29,5533	28,2412	27,0960	26,0625	25,1064	24,2037	23,3367
25	52,6187	49,4351	46,9280	44,3140	40,6465	37,6525	34,3816	32,2825	30,6752	29,3388	28,1719	27,1183	26,1430	25,2218	24,3366
26	54,0511	50,8291	48,2898	45,6416	41,9231	38,8851	35,5632	33,4295	31,7946	30,4346	29,2463	28,1730	27,1789	26,2395	25,3365
27	55,4751	52,2152	49,6450	46,9628	43,1945	40,1133	36,7412	34,5736	32,9117	31,5284	30,3193	29,2266	28,2141	27,2569	26,3363
28	56,8918	53,5939	50,9936	48,2782	44,4608	41,3372	37,9159	35,7150	34,0266	32,6205	31,3909	30,2791	29,2486	28,2740	27,3362
29	58,3006	54,9662	52,3355	49,5878	45,7223	42,5569	39,0875	36,8538	35,1394	33,7109	32,4612	31,3308	30,2825	29,2908	28,3361

Anexo B: Código del Sistema.

```
###### LIBRERIAS PARA LA CONEXIÓN

from robodk import *      # RoboDK API
from robolink import *    # Robot toolbox
RDK = Robolink()

###### DETECCIÓN DEL ROBOT PARA SIMULAR
robot = RDK.Item('', ITEM_TYPE_ROBOT)

###### SUBPROGRAMAS PARA CONTROL DEL GRIPPER
opn = RDK.Item('OPEN')
clo = RDK.Item('CLOSE')

###### DEFINICIÓN DE TARGETS DE MOVIMIENTO
##--TARGETS DE MANIPULACIÓN
pick1 = RDK.Item('Pick 1')
pick2 = RDK.Item('Pick 2')
pick3 = RDK.Item('Pick 3')
pick4 = RDK.Item('Pick 4')
pick5 = RDK.Item('Pick 5')
```

```
pick6 = RDK.Item('Pick 6')
pick7 = RDK.Item('Pick 7')
pick8 = RDK.Item('Pick 8')
pick9 = RDK.Item('Pick 9')
red1 = RDK.Item('Rojo1')
red2 = RDK.Item('Rojo2')
red3 = RDK.Item('Rojo3')
red4 = RDK.Item('Rojo4')
red5 = RDK.Item('Rojo5')
red6 = RDK.Item('Rojo6')
red7 = RDK.Item('Rojo7')
red8 = RDK.Item('Rojo8')
red9 = RDK.Item('Rojo9')
blue1 = RDK.Item('Azul1')
blue2 = RDK.Item('Azul2')
blue3 = RDK.Item('Azul3')
blue4 = RDK.Item('Azul4')
blue5 = RDK.Item('Azul5')
blue6 = RDK.Item('Azul6')
blue7 = RDK.Item('Azul7')
blue8 = RDK.Item('Azul8')
blue9 = RDK.Item('Azul9')

##-- TARGETS DE APROXIMACIÓN
home = RDK.Item('Home')
target = RDK.Item('Wait')
aprox1 = RDK.Item('Aprox 1')
aprox2 = RDK.Item('Aprox 2')
aprox3 = RDK.Item('Aprox 3')
aprox4 = RDK.Item('Aprox 4')
aprox5 = RDK.Item('Aprox 5')
aprox6 = RDK.Item('Aprox 6')
aprox7 = RDK.Item('Aprox 7')
aprox8 = RDK.Item('Aprox 8')
aprox9 = RDK.Item('Aprox 9')
aproxred1 = RDK.Item('AproxRojo1')
aproxred2 = RDK.Item('AproxRojo2')
aproxred3 = RDK.Item('AproxRojo3')
aproxred4 = RDK.Item('AproxRojo4')
aproxred5 = RDK.Item('AproxRojo4')
aproxred5 = RDK.Item('AproxRojo5')
aproxred6 = RDK.Item('AproxRojo6')
aproxred7 = RDK.Item('AproxRojo7')
aproxred8 = RDK.Item('AproxRojo8')
aproxred9 = RDK.Item('AproxRojo9')
aproxblue1 = RDK.Item('AproxAzul1')
aproxblue2 = RDK.Item('AproxAzul2')
aproxblue3 = RDK.Item('AproxAzul3')
aproxblue4 = RDK.Item('AproxAzul4')
aproxblue5 = RDK.Item('AproxAzul5')
aproxblue6 = RDK.Item('AproxAzul6')
aproxblue7 = RDK.Item('AproxAzul7')
```

```
aproxblue8 = RDK.Item('AproxAzul8')
aproxblue9 = RDK.Item('AproxAzul9')

###### REFERENCIAS DE POSICIÓN
target_pose = target.Pose()
xyz_ref = target_pose.Pos()

###### TARGETS DE INICIALIZACIÓN
robot.MoveJ(aprox8)
robot.MoveJ(aproxblue5)
robot.MoveJ(aproxred5)
opn.RunProgram()
robot.MoveJ(home)

###### SISTEMA DE VISIÓN ARTIFICIAL
import cv2
import numpy as np
from operator import itemgetter

def dist(px, py):
    return np.sqrt((px-py)**2)

def trajectory(cx1, cy1, ide1, i_):

    ##-- IDENTIFICACIÓN Y ASIGNACIÓN DE TARGETS
    i_ += 10
    ##-- POSICIÓN 1
    robot.MoveJ(target)
    if  B > cx1 > A and X > cy1 > W:
        robot.MoveJ(aprox1)
        if ide1 == 'r':
            print('el objeto es rojo')
            robot.MoveJ(pick1)
            clo.RunProgram()
            robot.MoveJ(aprox1)
            robot.MoveJ(aproxred1)
            robot.MoveJ(red1)
            opn.RunProgram()
            robot.MoveJ(aproxred1)
        else:
            print('el objeto es azul')
            robot.MoveJ(pick1)
            clo.RunProgram()
            robot.MoveJ(aprox1)
            robot.MoveJ(aproxblue1)
            robot.MoveJ(blue1)
            opn.RunProgram()
            robot.MoveJ(aproxblue1)

        ##-- POSICIÓN 2
    elif  C > cx1 > B and X > cy1 > W:
        robot.MoveJ(aprox2)
```

```
    if ide1 == 'r':
        print('el objeto es rojo')
        robot.MoveJ(pick2)
        clo.RunProgram()
        robot.MoveJ(aprox2)
        robot.MoveJ(aproxred2)
        robot.MoveJ(red2)
        opn.RunProgram()
        robot.MoveJ(aproxred2)
    else:
        print('el objeto es azul')
        robot.MoveJ(pick2)
        clo.RunProgram()
        robot.MoveJ(aprox2)
        robot.MoveJ(aproxblue2)
        robot.MoveJ(blue2)
        opn.RunProgram()
        robot.MoveJ(aproxblue2)

##-- POSICIÓN 3
elif  D > cx1 > C and X > cy1 > W:
    robot.MoveJ(aprox3)
    if ide1 == 'r':
        print('el objeto es rojo')
        robot.MoveJ(pick3)
        clo.RunProgram()
        robot.MoveJ(aprox3)
        robot.MoveJ(aproxred3)
        robot.MoveJ(red3)
        opn.RunProgram()
        robot.MoveJ(aproxred3)
    else:
        print('el objeto es azul')
        robot.MoveJ(pick3)
        clo.RunProgram()
        robot.MoveJ(aprox3)
        robot.MoveJ(aproxblue3)
        robot.MoveJ(blue3)
        opn.RunProgram()
        robot.MoveJ(aproxblue3)

##-- POSICIÓN 4
elif  B > cx1 > A and Y > cy1 > X:
    robot.MoveJ(aprox4)
    if ide1 == 'r':
        print('el objeto es rojo')
        robot.MoveJ(pick4)
        clo.RunProgram()
        robot.MoveJ(aprox4)
        robot.MoveJ(aproxred4)
        robot.MoveJ(red4)
        opn.RunProgram()
```

```
        robot.MoveJ(aproxred4)
    else:
        print('el objeto es azul')
        robot.MoveJ(pick4)
        clo.RunProgram()
        robot.MoveJ(aprox4)
        robot.MoveJ(aproxblue4)
        robot.MoveJ(blue4)
        opn.RunProgram()
        robot.MoveJ(aproxblue4)

##-- POSICIÓN 5
elif  C > cx1 > B and Y > cy1 > X:
    robot.MoveJ(aprox5)
    if ide1 == 'r':
        print('el objeto es rojo')
        robot.MoveJ(pick5)
        clo.RunProgram()
        robot.MoveJ(aprox5)
        robot.MoveJ(aproxred5)
        robot.MoveJ(red5)
        opn.RunProgram()
        robot.MoveJ(aproxred5)
    else:
        print('el objeto es azul')
        robot.MoveJ(pick5)
        clo.RunProgram()
        robot.MoveJ(aprox5)
        robot.MoveJ(aproxblue5)
        robot.MoveJ(blue5)
        opn.RunProgram()
        robot.MoveJ(aproxblue5)

##-- POSICIÓN 6
elif D > cx1 > C and Y > cy1 > X:
    robot.MoveJ(aprox6)
    if ide1 == 'r':
        print('el objeto es rojo')
        robot.MoveJ(pick6)
        clo.RunProgram()
        robot.MoveJ(aprox6)
        robot.MoveJ(aproxred6)
        robot.MoveJ(red6)
        opn.RunProgram()
        robot.MoveJ(aproxred6)
    else:
        print('el objeto es azul')
        robot.MoveJ(pick6)
        clo.RunProgram()
        robot.MoveJ(aprox6)
        robot.MoveJ(aproxblue6)
        robot.MoveJ(blue6)
```

```
        opn.RunProgram()
        robot.MoveJ(aproxblue6)

##-- POSICIÓN 7
elif  B > cx1 > A and Z > cy1 > Y:
    robot.MoveJ(aprox7)
    if ide1 == 'r':
        print('el objeto es rojo')
        robot.MoveJ(pick7)
        clo.RunProgram()
        robot.MoveJ(aprox7)
        robot.MoveJ(aproxred7)
        robot.MoveJ(red7)
        opn.RunProgram()
        robot.MoveJ(aproxred7)
    else:
        print('el objeto es azul')
        robot.MoveJ(pick7)
        clo.RunProgram()
        robot.MoveJ(aprox7)
        robot.MoveJ(aproxblue7)
        robot.MoveJ(blue7)
        opn.RunProgram()
        robot.MoveJ(aproxblue7)


##-- POSICIÓN 8
elif  C > cx1 > B and Z > cy1 > Y:
    robot.MoveJ(aprox8)
    if ide1 == 'r':
        print('el objeto es rojo')
        robot.MoveJ(pick8)
        clo.RunProgram()
        robot.MoveJ(aprox8)
        robot.MoveJ(aproxred8)
        robot.MoveJ(red8)
        opn.RunProgram()
        robot.MoveJ(aproxred8)
    else:
        print('el objeto es azul')
        robot.MoveJ(pick8)
        clo.RunProgram()
        robot.MoveJ(aprox8)
        robot.MoveJ(aproxblue8)
        robot.MoveJ(blue8)
        opn.RunProgram()
        robot.MoveJ(aproxblue8)

##-- POSICIÓN 9
elif  D > cx1 > C and Z > cy1 > Y:
    robot.MoveJ(aprox9)
    if ide1 == 'r':
```

```
            print('el objeto es rojo')
            robot.MoveJ(pick9)
            clo.RunProgram()
            robot.MoveJ(aprox9)
            robot.MoveJ(aproxred9)
            robot.MoveJ(red9)
            opn.RunProgram()
            robot.MoveJ(aproxred9)
        else:
            print('el objeto es azul')
            robot.MoveJ(pick9)
            clo.RunProgram()
            robot.MoveJ(aprox9)
            robot.MoveJ(aproxblue9)
            robot.MoveJ(blue9)
            opn.RunProgram()
            robot.MoveJ(aproxblue9)
    else:
        return False, i_

    robot.MoveJ(target)
    return True, i_

cap = cv2.VideoCapture(1)

i = 0
n_frames = 20
vectors, labels = [], []

while True:
    ret, frame = cap.read()

    ##-- GUÍAS PARA UBICACIÓN DE LA CÁMARA
    x1, y1 = frame.shape[1]/2, frame.shape[0]/2
    w1, h1 = 400, 350

    A = int(x1-w1/2)
    B = int(x1-w1/6)
    C = int(x1+w1/6)
    D = int(x1+w1/2)

    W = int(y1-h1/2)
    X = int(y1-h1/4)
    Y = int(y1+h1/8)
    Z = int(y1+h1/2)

    # frame = cv2.flip(frame, 1)
    if i < n_frames:
        if ret==True:
            frameHSV = cv2.cvtColor(frame,cv2.COLOR_BGR2HSV)

            #### UMBRALES DE COLOR
```

```
            ##-- UMBRALES PARA EL COLOR ROJO
            rojo_bajo1 = np.array([0, 100, 20], np.uint8)
            rojo_alto1 = np.array([10, 255, 255], np.uint8)
            rojo_bajo2 = np.array([175, 100, 20], np.uint8)
            rojo_alto2 = np.array([180, 255, 255], np.uint8)

            ##-- UMBRALES PARA EL COLOR AZUL
            azul_bajo = np.array([90, 240, 70], np.uint8)
            azul_alto = np.array([120, 255, 255], np.uint8)

            #### MÁSCARAS DE COLOR
            maskRojo1 = cv2.inRange(frameHSV, rojo_bajo1, rojo_alto1)
            maskRojo2 = cv2.inRange(frameHSV, rojo_bajo2, rojo_alto2)
            maskrojo = cv2.add(maskRojo1, maskRojo2)
            maskazul = cv2.inRange(frameHSV, azul_bajo, azul_alto)

            ##### DETECCIÓN DE CONTORNOS
            contorno1,    hierarchy    =    cv2.findContours(maskrojo,
            cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE
            contorno2,    hierarchy    =    cv2.findContours(maskazul,
            cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

            ##### EXTRACCIÓN DE CARACTERÍSTICAS
            for c in contorno1:
                area1 = cv2.contourArea(c)
                if area1 > 1000:
                    ### --- Centros
                    M = cv2.moments(c)
                    x = int(M["m10"]/M["m00"])
                    y = int(M["m01"]/M["m00"])
                    cv2.circle(frame, (x,y), 7, (0,255,0), -1)
                    cv2.putText(frame,  '{},{}'.format(x,  y),  (x-60,
y+50), cv2.FONT_ITALIC, 1, (255, 255, 255), 2)
                    cv2.putText(frame,    "Rojo",    (x-30,    y-30),
cv2.FONT_ITALIC, 1, (255, 255, 255), 2)
                    ##-- ENCERRAR CONTORNO
                    nuevoContorno = cv2.convexHull(c)
                    cv2.drawContours(frame,  [nuevoContorno],  0,  (0,
255, 0), 3)
                    ##-- AGRUPACIÓN DE CENTROS
                    vectors.append([x, y])
                    labels.append(['r'])

            for h in contorno2:
                area2 = cv2.contourArea(h)
                if area2 > 1000:
                    ### --- Centros
                    M1 = cv2.moments(h)
                    cx = int(M1["m10"]/M1["m00"])
                    cy = int(M1["m01"]/M1["m00"])
                    cv2.circle(frame, (cx, cy), 7, (0, 255, 0), -1)
                    cv2.putText(frame, '{},{}'.format(cx, cy), (cx-60,
cy+50), cv2.FONT_ITALIC, 1, (255, 255, 255), 2)
```

```
                    cv2.putText(frame,    "Azul",    (cx-30,    cy-30),
cv2.FONT_ITALIC, 1, (255, 255, 255), 2)
                    ##-- ENCERRAR CONTORNO
                    nuevoContorno = cv2.convexHull(h)
                    cv2.drawContours(frame,  [nuevoContorno],  0,  (0,
255, 0), 3)
                    ##-- AGRUPACIÓN DE CENTROS
                    vectors.append([cx, cy])
                    labels.append(['b'])

            i += 1
    else:
        if len(vectors) > 1:
            vectors, labels = np.array(vectors), np.array(labels)
            vectors_ord       =       np.array(sorted(vectors.tolist(),
key=itemgetter(0)))
            object_current = vectors_ord[:n_frames, :]
            ide_object = np.where(vectors == object_current[0, :])[0]
            ide_color = labels[ide_object]
            vals_x    =    np.array([float(values)    for    values    in
object_current[:, 0]])
            vals_y    =    np.array([float(values)    for    values    in
object_current[:, 1]])
            cx_, cy_ = np.mean(vals_x), np.mean(vals_y)

            response, i = trajectory(cx_, cy_, ide_color[0], i)
            if response:
                i = 0
                #vectors = []
                vectors, labels = [], []
        else:
            print('No existen objetos')
            i = 0

        cv2.rectangle(frame,        (int(x1-w1/2),        int(y1-h1/2)),
(int(x1+w1/2), int(y1+h1/2)), (0, 255, 0), 3)
        cv2.rectangle(frame,        (int(x1-w1/6),        int(y1-h1/2)),
(int(x1+w1/6), int(y1+h1/2)), (255, 0, 255), 2)
        cv2.rectangle(frame,        (int(x1-w1/2),        int(y1-h1/4)),
(int(x1+w1/2), int(y1+h1/8)), (255, 0, 0), 2)

    cv2.imshow('Video', frame)
    k = cv2.waitKey(1)
    if k == 27:
        break
```

Anexo C: Datasheet del robot KUKA KR10 R900 Sixx.

KUKA

KR 10 R900 sixx C

Technical data

Maximum reach	901.5 mm
Maximum payload	10 kg
Pose repeatability (ISO 9283)	± 0.03 mm
Number of axes	6
Mounting position	Ceiling
Footprint	320 mm x 320 mm
Weight	approx. 52 kg

Axis data

Motion range	
A1	±170 °
A2	-190 ° / 45 °
A3	-120 ° / 156 °
A4	±185 °
A5	±120 °
A6	±350 °

Operating conditions

Ambient temperature during operation	5 °C to 45 °C (278 K to 318 K)

Protection rating

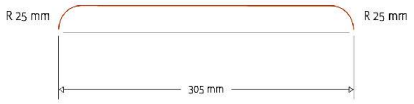
Protection rating (IEC 60529)	IP54
Protection rating, robot wrist (IEC 60529)	IP54

Controller

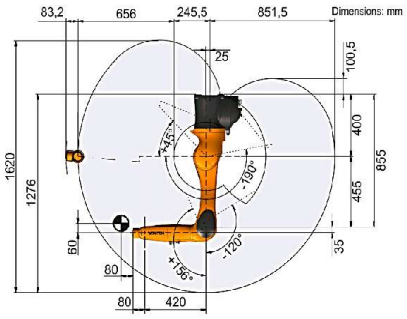
Controller	KR C4 smallsize-2; KR C4 compact

Cycle time

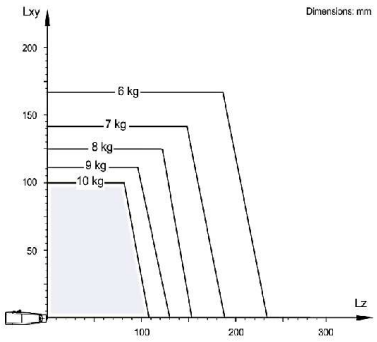
131 cycles per minute (25 mm / 305 mm / 25 mm, 1 kg)



Workspace graphic

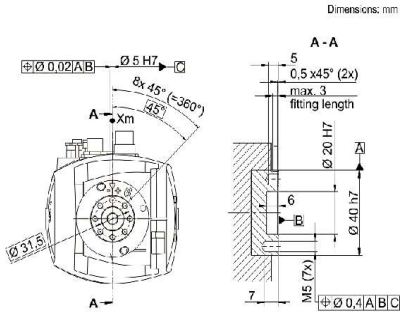


Payload diagram

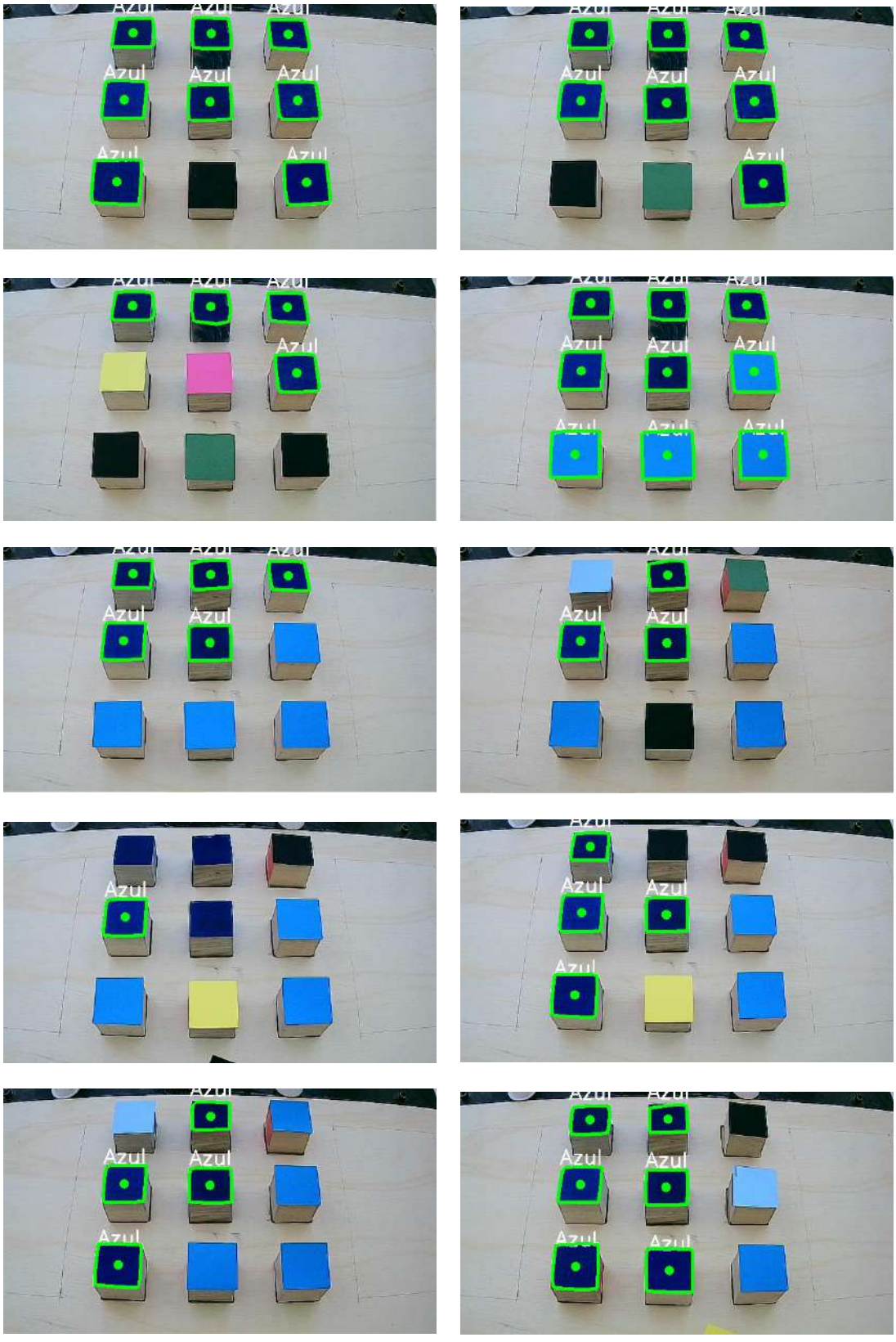


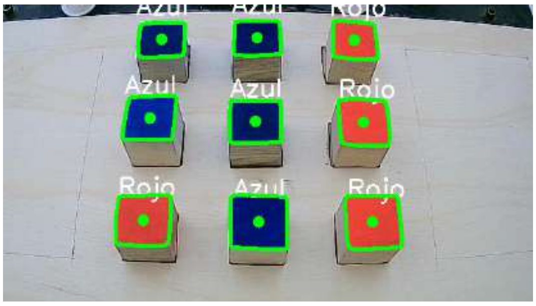
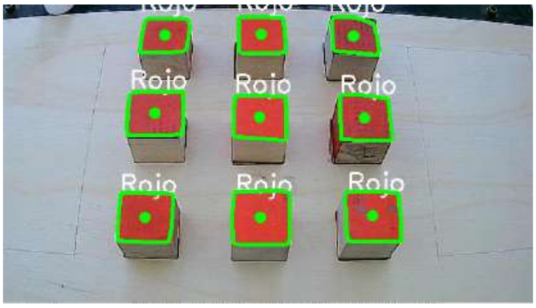
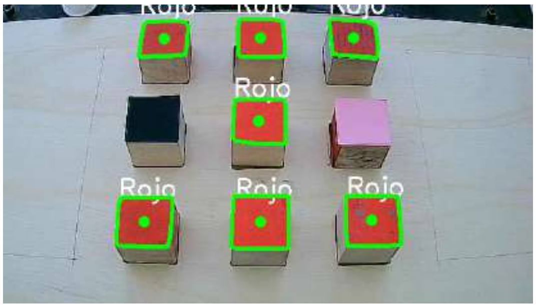
The KR 10 R900 sixx C is designed for a rated payload of 5 kg in order to optimize the dynamic performance of the robot. With reduced load center distances and favorable supplementary loads, a maximum payload of up to 10 kg can be mounted. The specific KUKA Load case must be verified using KUKA. For further consultation, please contact KUKA Service.

Mounting flange

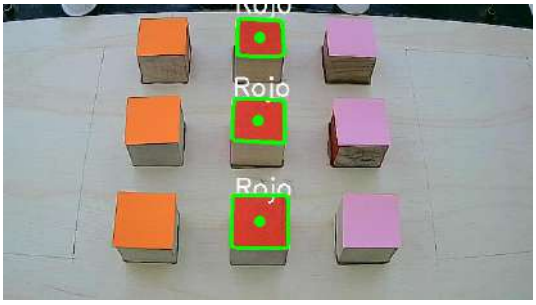
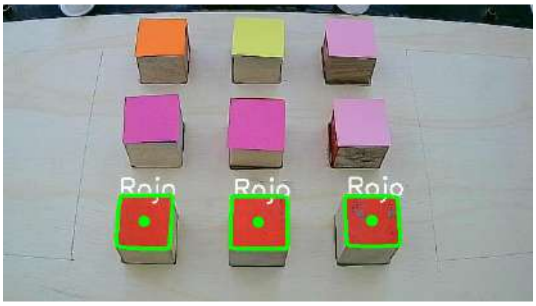
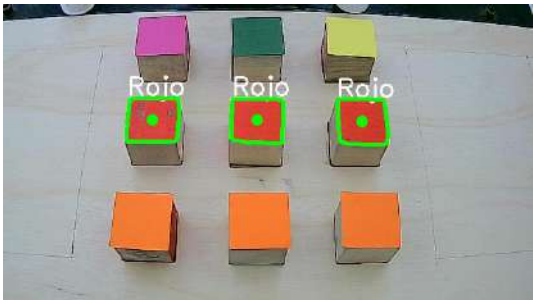
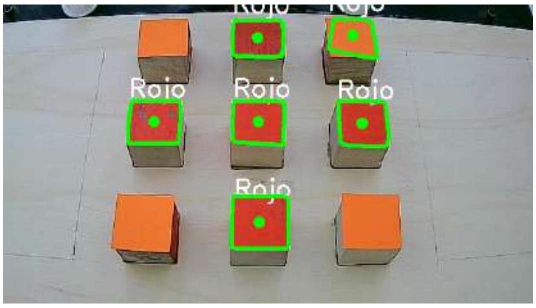
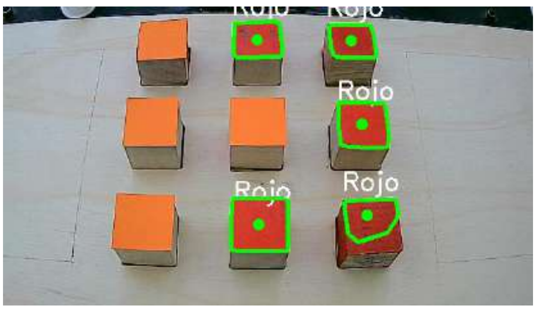


Anexo D: Muestras capturadas para el analisis estadistico.




Azul
Azul
Rojo
Rojo
Azul
Rojo

Rojo
Rojo
Rojo
Rojo
Rojo
Rojo

Rojo
Rojo
Rojo
Rojo

Rojo
Rojo
Rojo
Rojo

Rojo

Rojo
Rojo

Rojo
Rojo
Rojo

Rojo
Rojo
Rojo

Rojo
Rojo
Rojo
Rojo

Rojo
Rojo
Rojo

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

DIRECCIÓN DE BIBLIOTECAS Y RECURSOS DEL APRENDIZAJE

UNIDAD DE PROCESOS TÉCNICOS

REVISIÓN DE NORMAS TÉCNICAS, RESUMEN Y BIBLIOGRAFÍA

Fecha de entrega: 22/ 11/ 2023

INFORMACIÓN DE LOS AUTORES
Nombres – Apellidos: Jonathan Francisco Pilco Villa
INFORMACIÓN INSTITUCIONAL
Facultad: Informática Y Electrónica
Carrera: Electrónica Y Automatización
Título a optar: Ingeniero En Electrónica Y Automatización
f. Analista de Biblioteca responsable: Ing. Fernanda Arévalo M.

1704-DBRA-UPT-2023