Universidad del País Vasco Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

MÁSTER UNIVERSITARIO EN

INGENIERÍA INDUSTRIAL

TRABAJO FIN DE MÁSTER

APLICACIÓN DE UN ROBOT COLABORATIVO DE DOS BRAZOS PARA LA RESOLUCIÓN DEL CUBO DE RUBIK BASADO EN VISIÓN ARTIFICIAL

Estudiante *Fernández, Gutiérrez de Rozas, Garazi*
Director/Directora *Cabanes, Axpe, Itziar*
Departamento Ingeniería de Sistemas y Automática
Curso académico *2020/21*

Bilbao, 21, Septiembre, 2021

RESUMEN

Con objeto de entretener y acercar al usuario al mundo de la robótica, este trabajo presenta una célula robotizada para la resolución de cubos de Rubik. El sistema está formado por el robot colaborativo YuMi de dos brazos y una cámara de visión artificial. Gracias a esta cámara, el sistema es capaz de detectar la posición y la configuración del cubo de Rubik, además de identificar si se trata de un cubo de 2x2 o de 3x3. Una vez detectado el cubo, el robot utiliza los dos brazos para resolver el cubo con el mínimo número de giros y movimientos posibles. Adicionalmente, con el fin de aumentar la comunicación entre el sistema y el usuario, se le ha añadido una pantalla táctil y un módulo de sonido.

Palabras clave: Robótica colaborativa, visión artificial, Matlab, RobotStudio, cubo de Rubik.

ABSTRACT

In order to entertain and get robotics closer to the user, this work presents a robotic cell for the resolution of Rubik's cubes. The system consists of the YuMi two-armed collaborative robot and an artificial vision camera. Thanks to this camera, the system is able to detect the position and configuration of the Rubik's cube, in addition to identifying whether it is a 2x2 or 3x3 cube. Once the cube is detected, the robot uses its both arms to solve the cube with the minimum number of turns and movements possible. Additionally, in order to increase communication between the system and the user, a touch screen and a sound module have been added.

Keywords: Collaborative robotics, computer vision, Matlab, RobotStudio, Rubik's cube.

LABURPENA

Robotikaren mundura hurbiltzeko eta entretenitzeko asmoz, Rubiken kuboak ebazteko zelula robotizatu bat aurkezten du lan honek. Sistema bi besoko YuMi robot kolaboratzaileak eta ikusmen artifizialeko kamera batek osatzen dute. Kamera honi esker, sistema Rubiken kuboaren posizioa eta konfigurazioa detektatzeko gai da, 2x2ko kuboa edo 3x3koa den identifikatzeaz gain. Kuboa detektatu ondoren, robotak bi besoak erabiltzen ditu kuboa ahalik eta bira eta mugimendu gutxienekin ebazteko. Gainera, sistemaren eta erabiltzailearen arteko komunikazioa areagotzeko, ukimen-pantaila bat eta soinu-modulu bat gehitu zaizkio.

Hitz gakoak: Elkarlaneko robotika, ikusmen artifiziala, Matlab, RobotStudio, Rubiken kuboa.

INDICE

ILUSTRACIONES

TABLAS

1. INTRODUCCIÓN

El proyecto que se presenta a continuación se ha realizado en la Universidad del País Vasco (UPV), más concretamente, en el Departamento de Ingeniería y Sistemas y Automática de la Escuela de Ingeniería de Bilbao. El presente proyecto se formula como continuación de otro proyecto desarrollado con anterioridad mediante el robot colaborativo YuMi, al cual se propone realizar una serie de mejoras para optimizar y agilizar su funcionamiento. De esta manera se pretenden demostrar los beneficios que aporta la robótica colaborativa.

En primer lugar, se describe el contexto en el que se enmarca este trabajo, junto con el objetivo principal y los objetivos secundarios propuestos para este proyecto. Seguidamente se expone la evolución y el estado actual de la tecnología con respecto al tema, más concretamente se habla sobre la cuarta revolución industrial, la robótica colaborativa y la visión artificial.

Posteriormente, se realiza un análisis de alternativas en el que se seleccionarán distintos factores de especial relevancia para el desarrollo del proyecto. Tras ello, se pasará a detallar la descripción de la solución que estará compuesta por cuatro apartados principales: (1) Elementos principales del sistema, (2) Diseño de la zona de trabajo, (3) Implementación del sistema y (4) Validación.

Una vez se haya descrito la solución se realizará la descripción del procedimiento, donde se expondrán las diferentes tareas a realizar y el tiempo de duración. Además, a posteriori también se realizará un análisis de riesgos para tratar de minimizar la afección ante cualquier posible problema dado a lo largo del proyecto. Seguidamente se presentará el presupuesto estimado para llevar a cabo el trabajo.

A continuación, se realizará un análisis de resultados en el que se comprobará el correcto funcionamiento de la aplicación y su eficacia. Por último, se expondrán las conclusiones obtenidas a lo largo del proyecto.

2. CONTEXTO

En las últimas décadas los robots han sido ampliamente utilizados en diferentes ámbitos de la industria, como en aplicaciones de soldadura, ensamblaje o empaquetamiento. Estos, resultan muy útiles para operaciones repetitivas y producción en grandes lotes. Sin embargo, para procesos productivos con alta variabilidad e incertidumbre son más eficaces las personas. Este es el motivo por el cual en la cuarta revolución industrial se desarrollan los robots colaborativos o también denominados *cobots* [1] y se incorporan a los procesos de producción dando respuesta a esa producción de alta variabilidad, de lotes más pequeños y con una programación e interacción más cercana y a la vez sencilla con los operarios.

A pesar de que la industria es un área donde la incorporación de la robótica responde con éxito a la automatización de tareas y al aumento de la productividad, no todos los robots desarrollados son para trabajar en este campo. Ante la reciente aparición de los robots colaborativos y su inclusión a la industria, también existen muchos cobots que dan respuesta a la sociedad, en diferentes sectores como el sanitario y el del entretenimiento.

Hoy en día la sociedad tiene una necesidad cada vez mayor de buscar entretenimiento, es por ello que cada vez existen más robots dirigidos a dar compañía tanto a las personas de tercera edad como a los pequeños. Estos robots cumplen diversas tareas, como educar, ayudar en la cocina y tareas del hogar, recordar eventos y participar en juegos. Algunos de los robots a los que se les pueden atribuir estas cualidades son los siguientes: Buddy, SAM, Lynx, AIBO, etc [2].

Debido a esta tendencia se ha decidido realizar este proyecto basado en el campo del entretenimiento. Dentro de esta área, una actividad que despierta el interés y la atención de las personas es la realización de rompecabezas, siendo uno muy conocido el cubo de Rubik.

El cubo de Rubik es un mecanismo de varios ejes que permite que cada una de sus caras gire independientemente, de manera que los colores se mezclen, ver ilustración 1. Para solucionar el rompecabezas, las piezas que conforman cada una de sus caras deben ser del mismo color. Completar dicho rompecabezas 3D requiere de una serie de habilidades tales como concentración, memoria, coordinación óculo-manual, matemáticas, buenas destrezas psicomotoras y agilidad mental, entre otras [3].

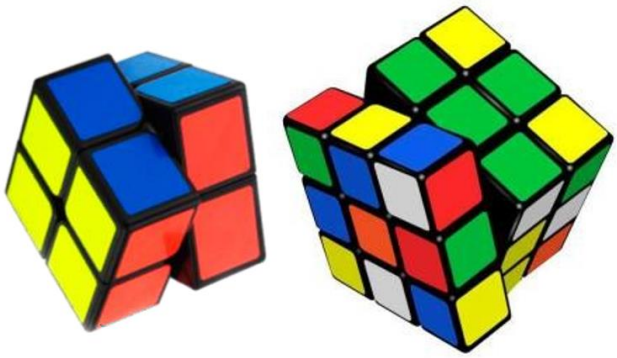


Ilustración 1: Cubos de Rubik de 2x2 y 3x3

Hoy en día son muchos los ingenios que en torno a este puzzle 3D han surgido. Actualmente, uno de los más destacados es el Sub1 Reload, ya que fue el primer mecanismo capaz de resolver el cubo de Rubik en menos de 1 segundo (exactamente 0,64 segundos), ver ilustración 2 . El Sub1 Reload está equipado con seis motores (uno para cada cara del cubo) y dos webcams que capturan tres lados del cubo cada una. Las imágenes recogidas de cada cara se envían a un ordenador que se encarga de procesarlas y de calcular la mejor solución mediante el algoritmo de dos fases de Herbert Kociemba, para tras ello efectuar los movimientos necesarios para resolver el cubo. Dado que la tecnología avanza a pasos agigantados, ya existen otros mecanismos que han batido este récord, como es el creado por el estudiante de robótica del MIT Ben Katz, capaz de resolver el cubo en 0,38 segundos [4].

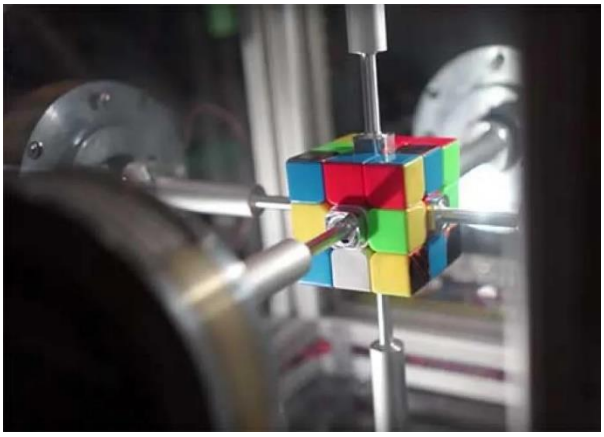


Ilustración 2: Sub1 Reload [3]

Con el reto de abordar la resolución del cubo de Rubik con un robot colaborativo, en el curso 2019-2020 se diseñó en el Departamento de Ingeniería de Sistemas y Automática, de la Escuela de Ingeniería de Bilbao (UPV/EHU), una primera versión de un robot colaborativo capaz de resolver los cubos de Rubik de 2x2 y 3x3.

La aplicación en cuestión, consiste en depositar un cubo de Rubik frente al robot para que este pueda cogerlo, sacar fotos a cada una de sus caras, procesar dichas imágenes para averiguar el tipo de cubo

a resolver (2x2 o 3x3) y los movimientos a realizar por el robot para la resolución del cubo y finalizar la aplicación realizando con el robot los movimientos y giros necesarios a las caras del cubo para resolverlo.

Para el desarrollo del presente proyecto se ha tomado como base dicha aplicación, con el objetivo de mejorarla e incrementar sus prestaciones.

3. OBJETIVOS Y ALCANCE DEL TRABAJO

En esta aplicación el usuario deberá depositar uno de los cubos (el de 2x2 o 3x3) en la mesa de recepción, tras ello el robot calculara la posición en la que se encuentra el cubo y lo tomara. Una vez tenga el cubo, haciendo uso de los dos brazos realizará las fotos a cada una de las caras de este, las cuales se procesarán para determinar el tipo de cubo a resolver y como esta mezclado. Cuando el procesamiento haya terminado y ya se conozca la disposición del cubo se enviará esta información a un algoritmo de resolución mediante el cual se obtendrán los movimientos a realizar por el robot para la resolución final. Por último, el robot realizará los giros que han sido previamente calculados y finalizará la aplicación depositando el cubo nuevamente en la mesa.

3.1 Objetivos

El objetivo principal del presente trabajo es la optimización y mejora del sistema robótico actual capaz de resolver los cubos de Rubik de 2x2 y 3x3.

Para lograr este objetivo principal, es necesario lograr los siguientes objetivos secundarios:

- Diseño de la célula y fabricación de componentes

Actualmente el robot colaborativo YuMi está ubicado en una célula de trabajo con varios escenarios integrados, puesto que se ha utilizado para diferentes proyectos. Además, esta célula robotizada es móvil y bajo la mesa sobre la que trabaja el robot se encuentra toda la aparamenta eléctrica que lo compone. Por lo que se considera de gran importancia organizar y optimizar toda esta zona de trabajo.

Para hacer operativa y eficaz la célula también es necesario modificar varios de los componentes que posee actualmente y añadir alguno nuevo. La carcasa de la cámara y el agarre que tiene ahora mismo el robot son excesivamente grandes, por lo que pueden dificultar la facilidad de movimiento del YuMi. Por tanto, uno de los retos se centra en el diseño de nuevos componentes que optimicen el espacio de trabajo, maximizando la movilidad del robot, la interacción con el usuario y la flexibilidad de la célula. Concretamente, el diseño de una nueva carcasa y soporte para la cámara (minimizando sus dimensiones para poder aprovechar mejor el espacio de trabajo), el diseño de una mesa multifuncional en la que el robot pueda depositar el cubo tras haberlo cogido y la instalación de una minitorre en la que guardar los programas de la aplicación.

Para el diseño de estos componentes será necesario adquirir conocimientos para el manejo de la herramienta AutoCAD, capaz de realizar los planos en 2D y 3D de los componentes mencionados. Dichos planos serán utilizados en una impresora 3D donde se fabricarán los componentes del nuevo escenario.

- Optimización de la programación

En la aplicación que se plantea, donde el robot interacciona con la cámara y con los componentes del escenario, es necesario programar varias herramientas que den respuesta al sistema robotizado

completo. Para cumplir esta función, en primer lugar se modificará la programación para la toma del cubo de la mesa, adecuándolo a la nueva disposición. En segundo lugar, se optimizará la toma, envío y procesamiento de imágenes (detección de colores y posiciones en las caras del cubo, ubicación de las esquinas, etc.). De esta manera se pretenden minimizar los tiempos de espera y mejorar la precisión del proceso. En tercer lugar, se tratará de mejorar la algoritmia de resolución del cubo de Rubik, donde se obtienen los movimientos a realizar por el robot. Por último, se optimizarán los movimientos a realizar por el robot para hacerlos más eficaces y adecuados al nuevo sistema.

- Interacción con el usuario e integración

Una de las grandes mejoras que se proponen en la nueva aplicación es la interacción con el usuario. Para ello, se pretende incorporar un módulo de sonido compuesto por dos altavoces con objeto de que el robot utilice mensajes orales durante la interacción con los que atraerá la atención de los que le rodeen. Además, también se integrará una pantalla táctil con la que se podrá visualizar la aplicación, mostrar mensajes e interactuar de manera más próxima al usuario.

- Validación del Sistema Robotizado

Tras realizar los pasos anteriores, es fundamental realizar y comprobar la correcta implantación del nuevo sistema con las funcionalidades ya mencionadas (integración de nuevos componentes, optimizaciones en la programación, incorporación de módulo de sonido, etc.). Para ello, se definirán unos casos de estudio donde se valide la calidad y el tiempo de la resolución, el grado de aceptación e interacción con las personas que lo rodean, la flexibilidad y movilidad de la célula, etc.

- Foro nacional de Automática

El último objetivo es presentar el proyecto en las Jornadas de Automática de 2021 (JJAA21), para ello se realizará la escritura de un artículo y la realización de un poster.

3.2 Alcance

En cuando al alcance de este trabajo fin de máster, este viene definido por los siguientes desarrollos:

- Diseño e impresión de los componentes 3D: Carcasa de la cámara, sujeción de la carcasa y mesa de recepción del cubo.

- Diseño de la célula: Además de los componentes 3D también se integrarán nuevos elementos para mejorar la célula (pantalla, minitorre, altavoces, etc.)

- Optimización de la toma del cubo de la mesa de recepción: Se modificará la programación para recoger el cubo, mejorándola y amoldándola a la nueva disposición.

- Optimización de la toma, envío y procesamiento de imágenes: Se modificará el método de toma y envío de imágenes para mejorar los tiempos de ejecución. También se modificará el procesamiento de imágenes para optimizarlo y adecuarlo a la nueva situación.

- Programación e implementación del módulo de sonido: Se programará un módulo de sonido

mediante el cual el robot será capaz de decir varias frases de manera oral en tres idiomas.

- Optimización de los movimientos del robot: Se modificarán los movimientos del robot para hacerlos más eficaces y adecuados al nuevo sistema.

- Implementación y validación del sistema final: Se realizarán una serie de pruebas mediante las cuales se comprobarán los tiempos de resolución, el grado de aceptación, la interacción, etc.

- Redacción del artículo y del póster para las Jornadas de Automática: Se redactarán los documentos necesarios para poder participar en las Jornadas de Automática de 2021.

4. EVOLUCIÓN Y ESTADO ACTUAL DE LA TECNOLOGÍA

Desde que en el Neolítico los humanos desarrollaron las primeras tecnologías agrícolas, la tecnología no ha hecho más que avanzar hasta el día de hoy, en la denominada la cuarta revolución industrial. En concreto, los robots son uno de los grandes descubrimientos, puesto que desde la creación del primer robot programable (1954), la robótica ha evolucionado cada día más, aumentando tanto el número de robots existentes como sus capacidades y funciones. Actualmente los citados robots se utilizan en una inmensidad de áreas: industrial, educativa, de entretenimiento, sanitaria, etc.

4.1 INDUSTRIA 4.0

Hoy en día, los procesos de fabricación se encuentran inmersos en un proceso de transformación digital, generado por el desarrollo y avance de las tecnologías de la información, más concretamente de la informática y el software. Es lo que se denomina Industria 4.0 o también recibe nombres como “fabrica inteligente” o “internet industrial”.

Las principales causas de esta cuarta revolución industrial son:

- Por un lado, la globalización de la economía: la cual ha ejercido una gran presión sobre la industria europea y su competitividad. Tras la crisis de 2007 se ha considerado de vital importancia para el beneficio de la economía el avance en el sector industrial, puesto que esto supone gran desarrollo de nuevas tecnologías y la aparición de puestos de trabajo.
- Por otro lado, las nuevas necesidades que demandan los clientes también han colaborado a la hora de impulsar esta revolución. Por esa razón, actualmente los mercados se basan en la personalización y creación de nuevos productos, junto con servicios más innovadores, experiencias más individualizadas y la creación de productos con capacidad de actualizarse, estando la informática (software y conectividad) presente en cualquier producto [5].

A la cuarta revolución industrial se le considera como la era de la digitalización en la producción y está impulsada por el aumento de la capacidad de volumen de datos, la potencia en los sistemas computacionales y la conectividad. Una de las diferencias más significativas que se encuentra en la industria 4.0 con respecto a las anteriores es la forma en la que se combinan las tecnologías para generar disrupciones significativas. De hecho, en [6] se detallan los pilares fundamentales de la industria 4.0 (ver ilustración 3), resumiendo a continuación cada una de las tecnologías que proponen.

Ilustración 3: Pilares de la industria 4.0

Sistema de Integración

Gracias a los sistemas de integración se presenta la posibilidad de conectar máquinas con otras máquinas y productos. Trabajando sobre la gestión interna de la empresa son capaces de integrar las diferentes áreas de la unidad productiva. Otra de las capacidades que aporta es utilizar plataformas digitales, habilitar la conexión entre la empresa y diferentes usuarios de su cadena de valor (por ejemplo, entre proveedores, actores del sistema de logística y trasporte, clientes, etc.).

Máquinas y sistemas autónomos (robots)

Para lograr el objetivo de conseguir fabricas inteligentes se trabaja tanto en máquinas inteligentes, capaces de automatizar tareas que antes solo las realizaban los humanos, como en la robótica colaborativa mediante la cual se quiere lograr que las empresas trabajen de forma conectada y con la capacidad de automatizar la mayoría de las tareas.

Internet de las cosas (IoT)

Mediante el IoT se logra la comunicación multidireccional entre personas, máquinas y productos. Además, también facilita la toma de decisiones puesto que esta tecnología permite recoger información de su entorno. De esta forma se lograrán productos más personalizados que se adapten mejor a las necesidades del cliente.

Fabricación aditiva

Mediante esta tecnología se logra la creación de piezas directamente desde un modelo virtual, sin necesidad de utilizar moldes, ya que se realiza mediante la superposición de capas de distintos materiales tomando como referencia un diseño previo. Gracias a esta tecnología se logra insertar un mayor componente de servicios y software a la fabricación, puesto que se descentralizan las etapas de diseño y desarrollo de los productos. Esta fabricación aditiva se utiliza en gran parte para prototipar y producir componentes individuales muy específicos en lotes pequeños.

Big data y análisis de grandes datos

Son datos de gran volumen, los cuales además de por su gran cantidad de datos también se caracterizan por su velocidad y variedad (datos estructurados y no estructurados). Los presentes datos pueden transferirse desde distintas fuentes (empresas, proveedores, clientes y redes sociales) y recibidos por diferentes dispositivos y sistemas (máquinas, equipos, sensores, cámaras, micrófonos, teléfonos móviles, software de producción). En la actualidad la capacidad de tomar decisiones en tiempo real es clave para facilitar el acceso a nuevos mercados y permitir alcanzar mejores estándares de calidad de producto y procesos. Por ello es imprescindible realizar el análisis de estos datos (grandes datos o Big data) mediante algoritmos avanzados.

Computación en la nube

Mediante la nube se pueden almacenar, acceder y hacer uso de servicios informáticos en línea. Gracias a la tecnología proporcionada por la nube, las empresas cuentan con las ventajas de poder acceder a los recursos informáticos de manera flexible, desde cualquier dispositivo, capaz de mantenerse en línea con un muy bajo esfuerzo administrativo, por lo que se ofrece agilidad, interoperabilidad y escalabilidad. Permite que gran parte de las aplicaciones que anteriormente necesitaban la instalación de un programa en el servidor de la empresa no sean necesarios y tengan la capacidad de ejecutarse de manera remota.

Simulación de entornos virtuales

Con la simulación de entornos virtuales se puede realizar la representación del funcionamiento del grupo de máquinas, procesos y personas deseados previamente a la puesta en marcha. De esta manera se logra prevenir averías, ahorrar tiempo y evaluar el resultado final que se obtendría antes de realizarlo. Además, los costos relacionados con los procesos de aprendizaje disminuyen, ya que con los entornos virtuales se pueden realizar pruebas de diseño de nuevos productos así como de distintas configuraciones en las operaciones de plantas productivas, hasta lograr la configuración o diseño final más óptimo.

Inteligencia artificial

Tiene como base el desarrollo de algoritmos que permiten procesar datos a una gran velocidad y lograr un aprendizaje automático. Estos algoritmos van aprendiendo a base de datos y experiencias, por lo

que se van perfeccionando, otorgándole a la maquina en cuestión capacidades cognitivas que poseen los seres humanos (visión, lenguaje, planificación, comprensión y decisión). En el ámbito industrial suelen utilizarse para el procesamiento de imágenes (reforzando la seguridad y el control de calidad), la predicción de series temporales de consumo eléctrico, el desarrollo de estrategias de control para la gestión optimizada de estaciones de producción, entre otros.

Ciberseguridad

Dado que se han aumentado los dispositivos conectados y el traspaso de información entre ellos, la ciberseguridad ha ganado gran importancia a la hora de evitar ataques tanto externos como internos. Por ello es clave en la fase de digitalización ya que permitirá una buena integración y uso de los datos y aplicaciones.

Realidad aumentada

Se utiliza para diseñar objetos digitales que puedan componer la situación real. La realidad aumentada combina distintos sistemas como son la simulación, modelado y la virtualización. De esta forma se logra mayor flexibilidad y rapidez en las cadenas de producción, además de nuevas fórmulas para la organización de procesos y diseño de productos. Se prevé que esta tecnología vaya en aumento para lograr mejorar la toma de decisiones, proporcionar información en tiempo real y optimizar los procesos de producción.

4.2 ROBÓTICA COLABORATIVA

En las últimas décadas los robots han sido ampliamente utilizados en diferentes ámbitos de la industria, como en aplicaciones de soldadura, ensamblaje o empaquetamiento. Estos, resultan muy útiles para operaciones repetitivas y producción en grandes lotes. Sin embargo, para procesos productivos con alta variabilidad e incertidumbre no es tan bueno su rendimiento. Este es el motivo por el cual en la cuarta revolución industrial se desarrollan los robots colaborativos o cobots [1].

Gracias a ellos se ha conseguido una mayor colaboración entre humanos y personas, eliminando las barreras físicas que los han separado durante muchos años [7]. Además, la utilización de sensores de fuerza, presión o tacto adquiere especial relevancia en estos robots, ya que al estar incorporados en la estructura del robot, permiten medir y controlar la fuerza y la velocidad, lo que garantiza que no superen los umbrales definidos en caso de producirse un contacto, sea intencionado o por accidente, manteniendo en todo momento la seguridad del usuario y de los componentes de su entorno.

Además, del acercamiento y mejora en la interacción con los usuarios, los cobots permiten reducir de forma significativa los problemas de espacio en su instalación (por la eliminación de barreras), mejorar la facilidad y rapidez en la programación, la seguridad y agilizar la adaptación ante cambios en la producción [8].

Si se analiza la historia de los cobots en el tiempo, se puede encontrar que la definición de cobot se crea en 1999. Ese año una empresa americana presentó la siguiente patente: *“un aparato y método*

para la integración física directa entre una persona y un manipulador con propósito general controlado por un ordenador." Por otro lado, General Motors fue el impulsor de los primeros cobots, puesto que realizó una gran cantidad de avances en el área de la robótica, tratando de implantar ésta en el sector de los automóviles. Logró crear un dispositivo capaz de colaborar con los humanos en las operaciones de montaje, sin embargo sus capacidades eran limitadas, puesto que por razones de seguridad el sistema no poseía una fuente interna que le facilitase el auto movimiento. En 2004 crearon el primer cobot ligero, capaz de moverse de forma autónoma. Cuatro años después, en 2008, los avances de la robótica permitían integrar los cobots en empresas no solo de gran tamaño, si no también pequeñas y medianas, sin necesidad de un excesivo coste o cambios significativos en las plantas de producción. Siguiendo con esta tendencia, hoy en día los cobots, concretamente en el sector industrial, tienen un crecimiento del 50% [9,11].

A pesar de que en el párrafo anterior se mencionan varios de los grandes momentos para la robótica colaborativa, en la ilustración 4 se amplían otros hitos más, como es la creación del KUKA en 1973, del robots SCARA en 1979 y del ASIMO en 2011.

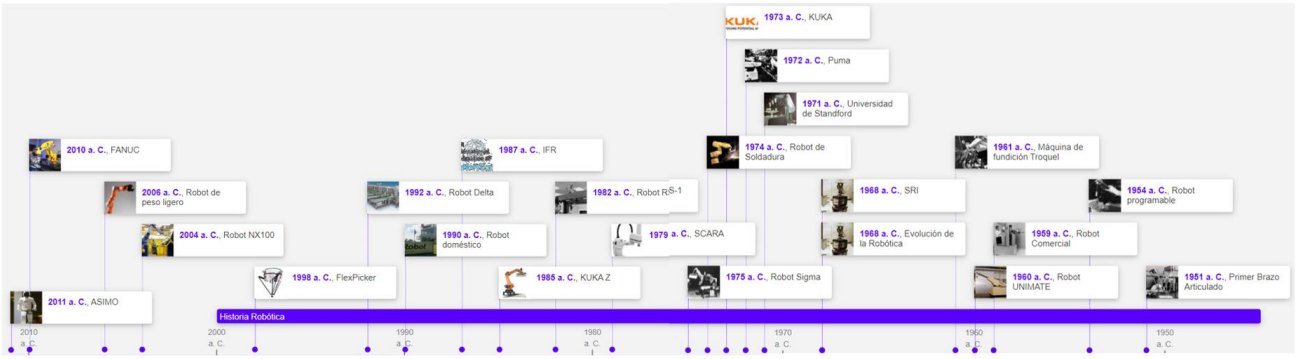


Ilustración 4: Evolución de la robótica [10]

La tendencia futura en cuanto al uso de los cobots es creciente, siendo conscientes de que aún hay mucho que mejorar y muchos retos que superar. Uno de ellos, a modo de ejemplo, se centra en las capacidades para tomar decisiones de forma autónoma y eficaz. Por ello la mayoría de las empresas fabricantes de robots ya están desarrollando cobots con sistemas integrados de visión y procesadores más rápidos, con objeto de que logren mayor grado de autonomía y productividad [9,11].

A pesar de que el sector de la industria es el más conocido como beneficiario de la robótica colaborativa, existen muchos más sectores a los que les afecta y que prosperan a través de los avances de esta. Entre ellos se encuentran el entretenimiento, la sanidad, el deporte, la limpieza, entre otros.

A continuación se tratará de analizar y exponer ejemplos de varios de los cobots implementados en dichos sectores.

4.2.1 INDUSTRIA

Como se ha mencionado anteriormente uno de los ámbitos que más se está beneficiando de la robótica colaborativa es el de la industria. Puesto que gracias a los cobots se logra automatizar la mayoría de los procesos industriales, ver ilustración 5. A diferencia de los robots comunes, esta nueva generación de robot permite la iteración con humanos, haciendo así más óptimo y eficaz el trabajo.

Muchas son las aplicaciones para las que ya se están utilizando los cobots en la industria [12,13,14], recalcando que son especialmente útiles en líneas de producción.

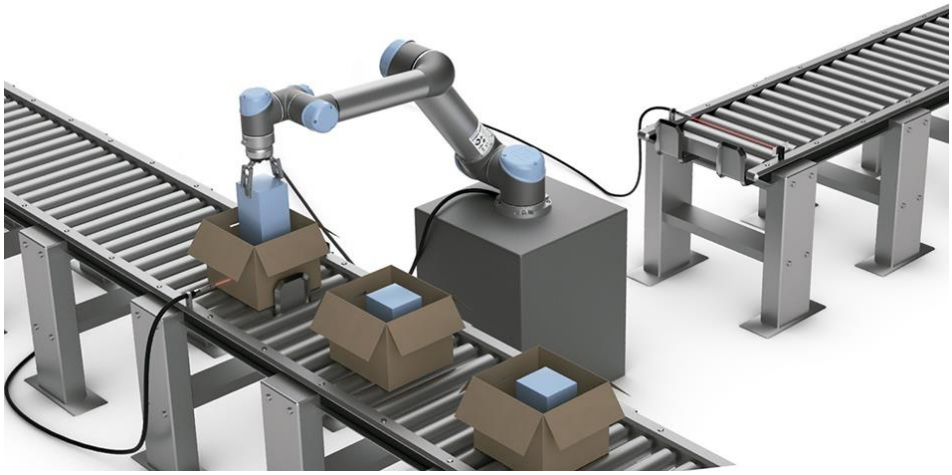


Ilustración 5: Cobots en la industria

Los cobots son la mejor opción de las empresas más pequeñas para poder competir con el resto, ya que como se ha dicho anteriormente son asequibles, versátiles y fáciles de integrar sin suponer un gran coste [15].

Empresas como Universal Robots fueron las pioneras advirtiendo el potencial que poseían los cobots para darles un impulso a las compañías pequeñas, fábricas de países en desarrollo y todo tipo de negocios con mano de obra escasa. Entre estas pequeñas industrias beneficiarias de los cobots desarrollados por Universal Robots se encuentran Craft y Technik, las cuales utilizaron los cobots para realizar operaciones de carga y descarga e inspecciones automáticas. Otras de las pequeñas empresas favorecidas por estos robots son las de Voodoo Manufacturing y Multi-Wing CZ, dedicadas a impresión y producción de ventiladores, las cuales han percibido un cambio notable en la reducción de los costos y el aumento de la capacidad de producción [15].

Por otro lado, las grandes empresas como Repsol, tampoco están desaprovechando esta oportunidad y están haciendo uso de los cobots para realizar distintas tareas. Se pone como ejemplo el centro de investigación de Repsol Technology Lab que ha automatizado gran cantidad de sus procesos de laboratorio a través de la robótica colaborativa (se han dispuesto cobots para llenar jeringas con diferentes muestras y manipularlas), ver ilustración 6. Gracias a la tecnología que ofrece Universal

Robots han logrado reducir los tiempos de puesta globales y liberar el tiempo de los técnicos de la planta, con objetivo de que estos últimos puedan dedicarse a otro tipo de tareas [16].

Ilustración 6: Cobots en Repsol Technology Lab

4.2.2 SANIDAD

El área de la sanidad está haciendo uso cada vez de más cobots para colaborar tanto con médicos como con enfermeros en la realización de diferentes tareas.

Uno de los robots más utilizados en este campo son los robots quirúrgicos, ver ilustración 7, los cuales emplean sistemas teleoperados para manejar instrumental quirúrgico o la automatización para tareas de diferentes intervenciones. Varios de los estudios más destacados se centran en la Cirugía Mínimamente Invasiva (CMI). Esta se ha vuelto especialmente popular en los últimos años, ya que presenta beneficios de gran relevancia como: la reducción de la dimensión de las incisiones, acortar el tiempo de convalecencia posoperatoria y de otras complicaciones, minimizar el tiempo que el paciente permanece en el hospital y otro tipo de beneficios tanto sociales como económicos [16].

Gracias a los cobots los cirujanos pueden poner toda su atención en realizar los procedimientos quirúrgicos mientras estos le ayudan de forma colaborativa. Entre las técnicas más conocidas se encuentra el Guiado Visual, el cual se utiliza para la realización de los movimientos del endoscopio en la cirugía cardiaca [17]. Por otro lado, se pueden encontrar trabajos en los que los robots colaborativos mediante la automatización de maniobras ayudan al cirujano de forma activa con tareas como la realización de procedimientos de hilado y anudado [18], la intervención de colecistectomía sin información preoperatoria [19], la conversión automática de navegación laparoscópica a movimientos de cirugía abierta [20].

Ilustración 7: Robot quirúrgico

Otro tipo de robot colaborativo demandado en el ámbito de la sanidad es el robot social que interacciona con las personas siguiendo comportamientos, patrones y normas sociales. Más concretamente se han realizado estudios para utilizar este tipo de cobot con objetivo de mitigar el deterioro de habilidades cognitivas en niños con problemas psicoafectivos [21]. Varios son los centros hospitalarios que ya han puesto en marcha esta idea, entre ellos el Hospital de Rehabilitación Glenrose de Edmonton en Canadá, en el cual se implementa una plataforma robótica de manipulación para facilitar el aprendizaje de niños con dificultad severa en el lenguaje y en la captura de objetos [22], ver ilustración 8.

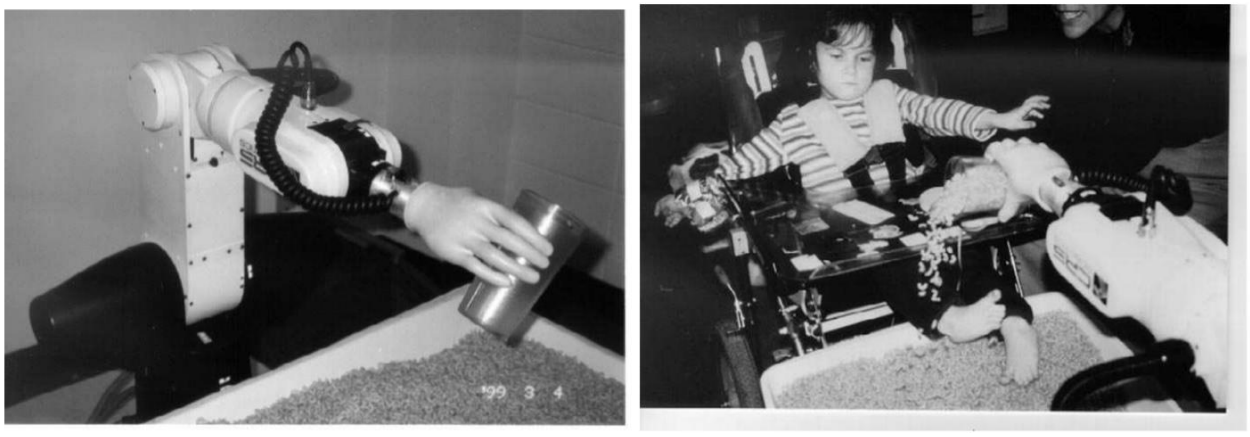


Ilustración 8: Plataforma robótica de manipulación pilotada en Edmonton, Canadá

4.2.3 ENTRETENIMIENTO

Por último, pese a de primeras no parecer un área tan importante como puede ser la de la sanidad y la industria, otro sector en el que despierta interés los cobots es el del entretenimiento.

El entretenimiento siempre ha sido un pilar fundamental en la sociedad, y aún lo es más hoy en día con los avances tecnológicos que se están dando en estos últimos años. Se pueden encontrar gran variedad de robots musicales, como es el robot cantante de ocho brazos (Shimon) [23], el cual ha trabajado con Gil Weinberg, músico y director del Georgia Tech Center for Music. La investigación que se realizó para obtener dicho cobot se basa en el aprendizaje automático y la psicología, pasando por la teoría musical, ver ilustración 9. Este robot es capaz de tocar varios instrumentos, siendo el primero de ellos la marimba, además ha desarrollado su propia voz y tiene la capacidad de cantar.

Ilustración 9: Robot cantante de 8 brazos

Otro de los robots colaborativos que logra entretener mediante la música se ha desarrollado dentro del proyecto BertsoBot, concretamente varios investigadores de la UPV/EHU (Universidad del País Vasco) han logrado un método automático con el que clasificar las piezas musicales y generar nuevas melodías. Para lograr que estos robots canten en verso han trabajado en diferentes aspectos como la comprensión de la señal de voz, la visión mediante el ordenador, la navegación, generación de nuevas melodías musicales entre otros [24].

La de los deportes es otra área que logra entretener en gran medida a los espectadores. Actualmente ya existen muchos robots que practican diferentes deportes, entre ellos se pueden encontrar robots que juegan a baloncesto [25], al ping pon [26], a futbol [27] y hasta que son gimnastas [28]. En la compañía Boston Dynamics han logrado desarrollar un robot con la capacidad de hacer gimnasia, puede realizar desde diferentes piruetas (volteretas, giros, etc.) hasta distintos saltos, sin suponerle esfuerzo alguno.

Dado que el área del entretenimiento se considera un pilar fundamental para incentivar a diferentes grupos de personas, este TFM trata sobre una aplicación cuyo objetivo es entretener y atraer a los alumnos y demás usuarios para lograr que se interesen por este campo tecnológico.

4.3 VISIÓN ARTIFICIAL

La visión artificial (VA) es otro de los puntos clave en la cuarta revolución industrial, concretamente la VA se basa en hacer uso de un software específico para procesar las imágenes captadas por una cámara. Contando con este proceso existen una gran variedad de aplicaciones [29]: inspección automática, reconocimiento de objetos, mediciones, robótica (dotándoles a los robots industriales de una mayor adaptabilidad, flexibilidad y capacidad de reorganización), etc.

Con esta tecnología se pretende lograr máquinas autónomas que sean capaces de interaccionar de manera inteligente con el entorno. Actualmente, gracias a la VA con la que cuentan algunas cámaras, los cobots ya son capaces de completar numerosas aplicaciones y tomar decisiones basadas en distintos parámetros de calidad [30].

Las cámaras se pueden situar en distintas ubicaciones del robot, una de las más comunes es la que se ve en la ilustración 10, que se trata de una cámara de VA situada en el brazo de un robot colaborativo.

Ilustración 10: Cámara de VA en brazo robótico

La cámara mencionada se convierte en los ojos del brazo robótico, aportándole así autonomía para gran cantidad de procesos industriales y de logística, garantizando la trazabilidad de los procesos de producción, permitiendo el guiado automático y proporcionando rapidez, constancia y rigor.

Los sistemas robóticos implementados con visión artificial son perfectos para aplicaciones como las siguientes: pick & place, empaquetado y paletizado, control de calidad y montaje [30].

En la actualidad la mayoría de los robots ya van acompañados de VA y hay muchos ejemplos de ello, a continuación se detallan algunos.

Como se ha citado, debido a la VA hoy en día se están implementando distintos robots, entre ellos los denominados “Robots espaciales” [31]. Estos robots humanoides están diseñados para estar en constante relación con los humanos y ser capaces de adaptarse al mismo entorno que estos, además de utilizar las mismas herramientas y paneles de control.

Además de los robots espaciales se están desarrollando muchos más robots para diferentes áreas haciendo uso de la VA. Entre ellos se pueden encontrar robots recoge pelotas [32] (para utilizarlo en partidos y entrenamientos de tenis), y robots para invernaderos [33] (prototipo de vehículo pulverizador que se desplaza de forma autónoma por el interior de invernaderos) ver ilustración 11.

Ilustración 11: Robot para invernadero

5. ANÁLISIS DE ALTERNATIVAS

Para cumplir el objetivo principal de lograr un sistema robótico optimizado capaz de resolver los cubos de Rubik de 2x2 y 3x3, es necesario realizar un análisis de alternativas, en el que se analicen las diferentes opciones para llevar a cabo el sistema.

Para esta aplicación se considera necesario realizar un razonamiento de la selección del robot (que se encargará de manipular el cubo), de la cámara (que realizará la toma de fotos para poder coger el cubo y resolverlo) y su ubicación en la célula, de la configuración del sistema y software (mediante los cuales se realizaran las conexiones y la programación) y del PC a utilizar (que tendrá instalados los programas necesarios para el desarrollo de la aplicación).

5.1 SELECCIÓN DEL ROBOT

Ya que la aplicación consiste en un robot capaz de resolver el cubo de rubik este componente es el más importante del sistema y también el que representa un mayor valor económico. Para la elección del robot se han tenido en cuenta aspectos de maniobrabilidad, seguridad y económicos.

Dado que el sistema robotizado se pretende que interactúe con personas sin la presencia de un vallado, se descartan los robots industriales, dado que no presentan la seguridad necesaria para este ámbito. Con este criterio, la mejor opción es un robot colaborativo, debido a que además de presentar la seguridad necesaria para la interacción con humanos también consta de mayor facilidad de programación, aumento de flexibilidad y menos coste económico que un robot industrial.

Una vez elegido el tipo de robot (colaborativo), es importante analizar sus características. Para la resolución de diferentes cubos de rubik se considera importante que el robot tenga dos brazos, con el fin de facilitar su resolución. De no ser así, para lograr una correcta manipulación del cubo se requeriría la ayuda de diferentes elementos auxiliares. Otra característica a tener en cuenta es que el robot no debe ser excesivamente grande ni pesado, puesto que otro de los objetivos del trabajo es poder desplazarlo para mostrarlo en las jornadas de automática y en diferentes actividades académicas.

Tras analizar los requisitos principales se han expuesto dos alternativas: El robot Yumi – IRB1400 de la empresa ABB y el Hiro Nextage de la impresa Kawada Industries.

ROBOT HIRO/NEXTAGE:

Este robot (ver ilustración 12) creado por la empresa Kawada Industries de Japón, en 2009, cuenta con una plataforma móvil y dos brazos robóticos. Cada uno de los brazos consta de 6 grados de libertad (gdl), a los cuales se les suma el grado de libertad que el robot posee en su cintura y los dos grados de libertad que forman el cuello, haciendo un total de 15 gdl. Gracias a estas características posee un espacio de trabajo enorme.

En cuanto a la visión artificial se pude decir que es un robot bastante completo y versátil, ya que cuenta con cuatro cámaras, situadas dos de ellas en la cabeza y las otras dos una en cada brazo.

Resaltar que tiene una capacidad de carga de brazo bastante elevada, 1,5 kg, y una repetitividad de 0,03mm.

Ya que en la aplicación se da la necesidad de que el robot interactúe con las personas, es importante mencionar que cuenta con el suficiente nivel de seguridad para poder realizar esta interacción sin peligro alguno.

Ilustración 12: Hiro Nextage

ROBOT YUMI IRB 14000:

El robot YuMi (ver ilustración 13) creado por la empresa ABB en 2015 es un robot colaborativo de dos brazos. Cada uno de sus brazos cuanta con siete grados de libertad, que en total suponen 14 gdl. Estos hacen que el YuMi posea un gran espacio de trabajo y mucha facilidad de movimiento. Además de permitirle manipular cargas de hasta 0,5 kg con una repetitividad de 0,02mm.

El YuMi presenta una gran seguridad para los usuarios y para trabajar con otros robots, puesto que posee un diseño de recubrimiento acolchado y varios sistemas de seguridad integrados.

Como valor añadido, es importante mencionar que este robot posee dos pinzas colaborativas Smart Gripper las cuales integran una cámara de visión artificial para facilitar la localización de los objetos y unas ventosas que ayudan a coger y trasladar dichos componentes.

Ilustración 13: Robot YuMi – IRB1400

Una vez analizadas las características de ambos robots se ha hecho una tabla con los datos más relevantes para realizar la elección del mismo, marcando en color verde las más ventajosas y en rojo las que menos, ver tabla 1.

Tabla 1: Comparación HIRO/YUMI

ROBOT	PRECIO (€)	REPETITIVIDAD (mm)	CARGA POR BRAZO (Kg)	GDL
HIRO	50.000	0.03	1.5	15
YUMI	45.000	0.02	0.5	14

SELECCIÓN

Se puede observar que los dos modelos presentan la misma cantidad de características a favor que en contra. Sin embargo, no todas las características tienen el mismo valor para el desarrollo del proyecto.

- Uno de los datos más relevantes es el **precio**. Como se puede observar el robot YUMI es más barato que el HIRO, por lo que es un aspecto de gran valor.
- La **repetitividad** también es un factor de gran importancia en este proyecto, puesto que se requiere de una extrema precisión a la hora de coger y manipular el cubo. En este aspecto vemos que el robot YUMI es superior, ya que presenta una repetitividad de 0.02mm.
- En cuanto a la **carga** por brazo, se ve que el HIRO tiene 1kg más de capacidad. Sin embargo, ya que la combinación del cubo de rubik y las pinzas no supera los 0,4kg no se considera una característica determinante.
- Por último, en cuando a los **grados de libertad** se observa que es superior el robot HIRO. Sin embargo, si lo analizamos detalladamente el HIRO consta de 2 gdl en el cuello, 1 gdl en la cadera y 6 gdl en cada brazo. Por el contrario, el robot YUMI consta únicamente de 7 gdl en cada brazo. Por lo que, a pesar de que el HIRO tenga más gdl en total, para esta aplicación son más relevantes los gdl de los brazos del YUMI, los cuales realizarán la maniobrabilidad con mayor destreza (los gdl del cuello y la cadera no serán necesarios).

Por lo tanto, tras realizar este análisis se ha decidido que el robot más adecuado para la presente aplicación sea el **ROBOT YUMI IRB 14000 de ABB**.

5.2 SELECCIÓN DE LA CÁMARA DE VISIÓN ARTIFICIAL Y SU UBICACIÓN

Puesto que el proyecto se basa en la resolución del cubo de rubik, para esa tarea es necesario que el robot sea capaz de identificar donde se encuentra el cubo y como están organizados los bloques de colores. Por lo tanto, tras el robot, el componente más importante es la cámara.

En primer lugar se detallan las especificaciones de las que debe constar la cámara, para así poder reducir la magnitud de opciones de esta.

ESPECIFICACIONES:

- Fotografías en color: Es necesario que la cámara sea capaz de identificar y sacar fotografías a color puesto que para la resolución del cubo hace falta identificar los colores de las piezas y su posición en cada cara del cubo.

- Resolución mínima 2K: Debido a que habrá que realizar un procesamiento de imágenes, es importante que la cámara tenga una buena resolución. Sin embargo, a mayor resolución mayor es el precio. Por lo que se ha estimado una resolución mínima de 2K que cumple con la función deseada y no compromete mucho el precio.

- Interfaz gigE: Dado que el robot YUMI dispone de un puerto RJ45 situado en la controladora, esto supone una posibilidad para conectar una cámara externa al mismo. Por esta razón se quiere que la cámara seleccionada tenga interfaz gigE (Gigabit Ethernet).

A pesar de reducir el campo de búsqueda gracias a las especificaciones marcadas, sigue habiendo demasiadas cámaras capaces de cumplir dichos requisitos en el mercado. Por lo tanto, dado que el robot elegido para la aplicación es de ABB ya posee una cámara del fabricante Infaimon (es de baja resolución y no detecta colores), para facilitar la integración del sistema se ha optado por elegir una cámara del mismo fabricante.

Dentro del catálogo de Infaimon, se ha decidido elegir una cámara de la familia de UI 5584LE debido a que estas cámaras cumplen con las especificaciones mínimas mencionadas, no suponen un coste excesivo y además aportan una gran flexibilidad en cuanto a la alimentación e intercambio de datos. Destacar que esta característica se logra debido a que la cámara posee un puerto RJ45 (ya mencionado), alimentación de dos polos y un puerto ZIF de 10 polos mediante el cual se pueden realizar conexiones con placas externas de todo tipo.

Tras elegir la familia, se ha pasado a detallar el modelo, para ello se ha buscado una cámara que se caracterice principalmente por su precisión y nitidez de colores, para que resulte más fácil distinguir los colores del cubo de rubik. En este caso se ha escogido la UI 5584LE-C-HQ, que posee una gran sensibilidad lumínica, ver ilustración 14.

Ilustración 14: Cámara UI 5584LE-C-HQ

Otra de las elecciones importantes es la de la situación de la cámara en la célula, ya que de la posición de esta dependerá el posterior procesamiento de imágenes, los movimientos que deberá desempeñar el robot para la toma de fotos, etc.

Es por ello que para seleccionar la ubicación de la cámara se han tenido en cuenta las siguientes especificaciones:

- Tiene que ser capaz de enfocar toda la mesa de trabajo donde se recepciona el cubo.

- No puede suponer un obstáculo para el robot a la hora de realizar los movimientos.

- Tiene que estar ubicada de tal forma que el robot pueda acercarle el cubo de manera sencilla para la toma de fotos.

- La ubicación no debe presentar problemas de iluminación.

Actualmente la cámara se encuentra demasiado inmersa en la célula requiriendo así una sujeción demasiado voluminosa que le quita espacio y elegancia al montaje. Además, por el momento la ubicación es tal que sería muy complicado situar una pantalla en un lugar adecuado haciendo uso de la misma sujeción.

Por lo tanto, se ha estimado la necesidad de modificar la ubicación de dicha cámara y situarla justo en la parte superior del robot, sin llegar a adentrarse en el campo de trabajo, considerando que desde ese punto se cumplen todas las especificaciones detalladas.

5.3 SELECCIÓN DE LA CONFIGURACIÓN DEL SISTEMA Y SOFTWARE

Una vez elegida la cámara para visión artificial y el robot para llevar a cabo la aplicación deseada, queda analizar cómo se realizará la comunicación entre estos.

Las dos opciones son: (1) realizando la conexión de la cámara directamente al robot mediante la interfaz gigE, y (2) utilizando un ordenador de pasarela que enlace la cámara y el robot.

CONEXIÓN DIRECTA CÁMARA-ROBOT

Ventajas: Esta primera opción cuenta con la ventaja de que no es necesario un ordenador de pasarela para conectar el robot y la cámara. Esta conexión se puede llevar a cabo debido a que la cámara seleccionada para la aplicación consta de una interfaz gigE con la cual se puede conectar directamente al robot creando una conexión Ethernet IP.

Desventajas: Esta opción implica tener que programar tanto la cámara, como el procesamiento de imágenes, el algoritmo de resolución del cubo de rubik y el módulo de sonido desde el RobotStudio. Esto se considera una desventaja ya que por un lado, RobotStudio no cuenta con la versatilidad ni potencia que pueden aportar otros programas para llevar a cabo el procesamiento de imágenes necesario para la aplicación. Por otro lado, a pesar de que es posible realizar la programación del algoritmo de resolución del cubo de rubik mediante RobotStudio, esto implica lograr un algoritmo tan óptimo y eficaz con el lenguaje Rapid, el cual no está adaptado para ello.

CONEXIÓN CÁMARA-ROBOT MEDIANTE ORDENADOR

Ventajas: Al utilizar un ordenador de pasarela entre la cámara y el robot, esta segunda opción permite hacer uso de un programa específico para el procesamiento de imágenes, para el algoritmo de resolución y para el módulo de sonido, suficientemente potente como para desempeñar la programación de estos de manera óptima y eficaz. Para esta función se plantean las herramientas de Matlab y Python, dado que tienen la capacidad de crear sus propias aplicaciones para realizar procesamiento de imágenes.

Desventajas: Como desventaja se encuentra la que es obvia, es decir, la necesidad de utilizar un PC de nexo. Esto supone tener que llevar dicho PC en la célula del robot aumentando así su peso y ocupando espacio.

Tras analizar las ventajas y desventajas de las dos opciones, se ha optado por utilizar la segunda opción. Debido a que tener que implantar un PC en la célula del robot no supone una gran desventaja, ya que en la parte inferior hay sitio suficiente para hacerlo. Sin embargo, si se considera una desventaja importante la pérdida de eficacia y tiempo a la hora de realizar las programaciones necesarias, por lo que se ha descartado la primera opción.

Una vez tomada la decisión de utilizar un ordenador como pasarela entre la cámara y el robot es importante realizar la elección del software para llevar a cabo la conexión y los procesamientos que se estimen necesarios.

ROBOTSTUDIO

Al elegir el robot Yumi de ABB el fabricante aporta esta herramienta de programación, por lo que el coste de la licencia entraría dentro del precio del robot. Además, esta herramienta es necesario usarla para programar los movimientos del robot.

Por ello, esta es la herramienta que se ha decidido utilizar para realizar la programación de los movimientos del robot.

MATLAB

Posee todas las funciones requeridas para realizar la programación necesaria para la aplicación (procesamiento de imágenes, algoritmo de resolución del cubo, módulo de sonido). Tiene una gran capacidad para manejar y resolver cálculos matemáticos, posee una gran cantidad de toolboxes relacionados con la inteligencia artificial y el procesamiento de imágenes, permite desarrollar programas y aplicaciones compatibles con prácticamente cualquier dispositivo y cuenta con un gran desarrollo en las funciones de matrices que se considera imprescindible para este trabajo. Además, está el factor de que el alumno a desarrollar esta aplicación posee conocimientos previos de esta herramienta.

Por estas razones se ha decidido utilizar Matlab como software complementario para realizar la aplicación de la resolución del cubo de rubik. Quedando una imagen como la siguiente, en la que se aprecia la comunicación de los componentes principales del sistema (robot, cámara y ordenador) mediante el software de Matlab, ver ilustración 15.

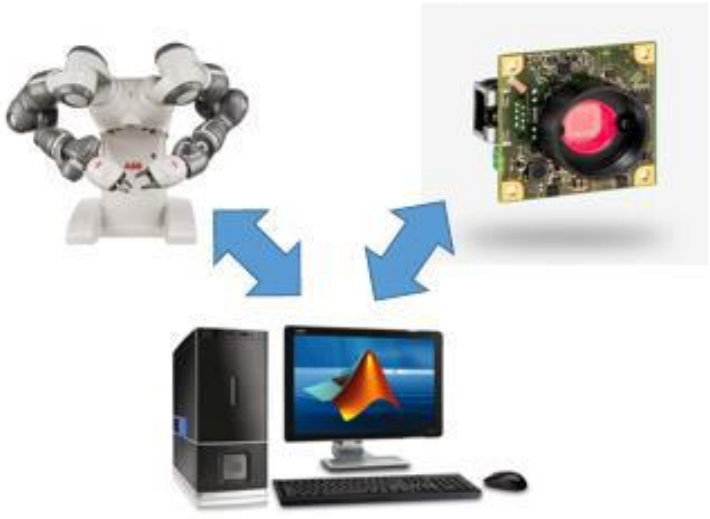


Ilustración 15: Conexión del sistema

5.4 SELECCIÓN DEL PC

Tras decidir que la configuración del sistema se realizará mediante un PC y el tipo de software a utilizar, se plantea la selección del PC que actuará como pasarela entre el robot y la cámara.

En primer lugar se han analizado las siguientes tres opciones:

ORDENADOR DE MESA

La primera opción es utilizar un ordenador de mesa, que se encuentre situado junto al robot, en el cual irán instaladas todos los programas y herramientas necesarias para el desarrollo de la aplicación.

Ventajas: Una de las grandes ventajas que presenta un ordenador de mesa es que por menor coste se encuentran PCs de mayor capacidad y mejor procesador.

Desventajas: La desventaja más notoria trata de que con un ordenador de mesa el traslado de la célula robotizada se hace más complejo, puesto que al ser el PC de mayor tamaño es complicado implantarlo en esta y habría que trasladarlo por separado.

MINITORRE Y PANTALLA TACTIL

La segunda opción consiste en la utilización de una minitorre para albergar los programas necesarios para la aplicación y una pantalla táctil para la visualización.

Ventajas: A diferencia de en la anterior opción, aquí se presenta la ventaja de facilitar el traslado de la célula robotizada. Esta capacidad se considera de vital importancia para el proyecto, puesto que uno de los objetivos es poder desplazar el robot a diferentes lugares, como las jornadas de automática. Además, gracias a la pantalla táctil se mejora la interacción del usuario y el robot, pudiendo visualizar en dicha pantalla desde la aplicación hasta los mensajes que se deseen.

Desventajas: Al ser el PC más pequeño, por el mismo coste que un PC de tamaño normal se encuentra una capacidad de almacenamiento menor y un procesador más lento.

ORDENADOR PORTATIL

La tercera opción que se ha planteado ha sido la de un ordenador portátil.

Ventajas: Este PC es fácil de trasladar e integrar en la célula, siempre y cuando no sea de gran tamaño. Además cuenta con una pantalla propia, por lo que no habría que adquirir una por separado.

Desventajas: Dependiendo de las características que se necesiten puede suponer costoso. Por otro lado, es necesario un tamaño reducido, ya que se quiere que la pantalla sea pequeña, esto disminuye las opciones, puesto que a menor tamaño menos capacidades tienen o más costosos son.

Comparando las distintas alternativas, la primera opción se ha descartado, dado que se considera requisito indispensable tener facilidad para el traslado de la célula robotizada. Tras ello, solo quedan las opciones de la minitorre más pantalla táctil y la del ordenador portátil.

ESPECIFICACIONES

Antes de analizar diferentes modelos de cada tipo se han detallado unas especificaciones que deben cumplir y en las que hay que basarse para tomar la decisión:

- Basta con un procesador de Intel i5 o i3, ya que para ejecutar bien Matlab se considera

suficiente.
- La pantalla que lleve incorporada el portátil o en el otro caso la pantalla táctil no podrá ser superior a 14”.
- La memoria RAM tiene que ser mínimo de 8GB o de ser menor tener la capacidad de realizarle una ampliación de memoria.

MODELOS

Partiendo de estas especificaciones se han buscado diferentes modelos de cada una de las opciones, con objetivo de elegir el más óptimo para el proyecto.

Tabla 2: Alternativas ordenadores portátiles

PORTÁTILES				
MODELO	**PROCESADOR**	**PULGADAS**	**MEMORIA**	**PRECIO**
	Intel Core i5	14”	16 GB RAM	799€
	Intel Core i5	13,9”	16GB RAM	1.050€

Tabla 3: Alternativas minitorres

MINITORRE				
MODELO	**PROCESADOR**	**DIMENSIONES (alto*ancho*profundo)**	**MEMORIA**	**PRECIO**
 HP Slim Desktop S01-PF1006NS	Intel Core i5	270x95x303mm	8GB RAM	450€

 HP Desktop M01-F1030ns	Intel Core i7	303x155,4x337,4mm	8GB RAM	600€
 Optiplex 7010	Intel Core i3	235*65*240mm	4GB RAM	170€

SELECCIÓN

Los anteriores modelos cumplen con las especificaciones detalladas. La mayor diferencia que se puede apreciar entre los portátiles y las minitorres seleccionadas es el precio, siendo muy superior el de los portátiles. Por esta razón se ha decidido optar por la opción de una minitorre que tenga las características deseadas.

Dentro de las minitorres, a la hora de tomar una decisión también ha prevalecido el precio, ya que el último modelo presentado (Optiplex 7010) es mucho más económico que los otros. Este modelo posee menor capacidad de RAM, sin embargo tiene la posibilidad de en caso de ser necesario ampliar dicha memoria por un coste mínimo, por otro lado el procesador del que dispone se considera suficiente para los programas que son necesarios para llevar a cabo la aplicación. Además, se destaca como ventaja que esta minitorre es la de menores dimensiones, por lo que facilita su integración en la célula.

Por todas las razones anteriores se ha elegido para la aplicación una **minitorre Optiplex 7010**.

Puesto que se ha optado por una minitorre para la realización del proyecto, será necesaria una pantalla mediante la cual visualizar las funciones y procedimientos a desempeñar.

Esta pantalla se situará en la parte superior del robot, justo encima de la cámara para la visión artificial, y utilizará la misma sujeción que esta. Por lo tanto, es importante que la pantalla sea pequeña y ligera. Además, ya que se quiere lograr que el usuario tenga la posibilidad de interactuar con el robot es preferible que la pantalla sea táctil, de esta forma tampoco se requerirá de ratón para manipularla.

Se puede ver que en el mercado hay un amplio abanico de posibilidades que cumplen con estas características, por lo que se han analizado varias de ellas (las más competitivas y económicas), y tras ello se ha llegado a la conclusión de que la mejor opción es la siguiente: El modelo TOGUARD Monitor ES, ver ilustración 16.

Ilustración 16: Pantalla TOGUARD

La pantalla seleccionada cuenta con 10.1 pulgadas, la capacidad de manejo táctil y la posibilidad de poner y quitar el apoyo al antojo del usuario. Además su precio es muy económico, puesto que solo cuesta 110€ y es ultra ligera. Por ello, ha sido la elegida para el proyecto.

6. DESCRIPCIÓN DE LA SOLUCIÓN

En este apartado se detalla el desarrollo seguido para optimizar la aplicación. Primeramente se realiza una breve explicación sobre los elementos principales del sistema; en segundo lugar se detalla el diseño de la zona de trabajo; en tercer lugar la implementación del sistema, en la que se recoge la configuración de varios componentes y la programación realizada; y por último se realizará la validación de la aplicación en el robot real.

Elementos principales	Diseño zona de trabajo	Implementación sistema	Validación
- Robot YUMI - Cámara - PC	- Mesa - Carcasa - Fijación - Altavoces	- Configuración - Programación	- Programación robot real

6.1 ELEMENTOS PRINCIPALES DEL SISTEMA

Como se ha mencionado en apartados anteriores los elementos fundamentales del sistema son los siguientes: el robot colaborativo YUMI, la cámara de visión artificial UI 5584 LE-C-HQ y el PC (en este caso, minitorre). Cada uno de estos componentes tiene una función relevante para llevar a cabo la aplicación y además de ser indispensable su funcionamiento por separado también lo es una correcta comunicación y sincronización entre ellos, ver ilustración 17.

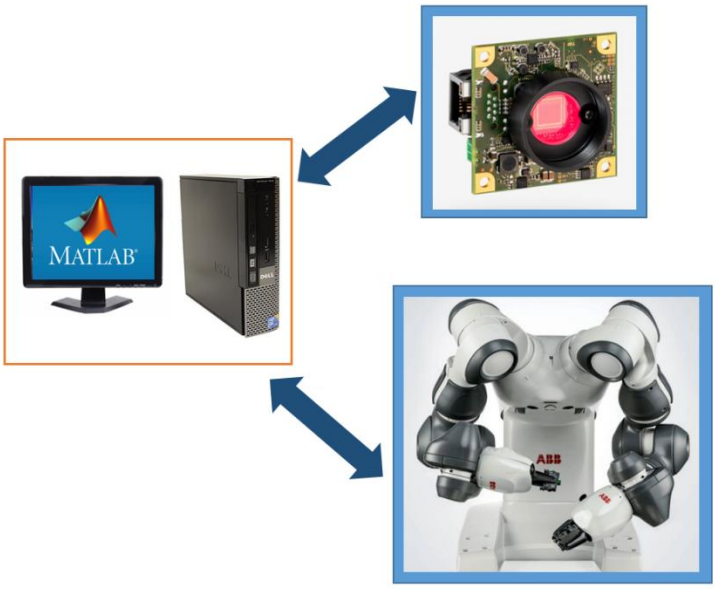


Ilustración 17: Componentes principales del sistema

En cuanto a la secuencia de acciones a realizar en esta aplicación, en la ilustración 18 se muestra un esquema. Así, en primer lugar la cámara se encarga de sacar una foto inicial a la mesa de recepción del cubo, la cual se la envía al PC para procesarla mediante el programa Matlab. Este por su parte, calcula la ubicación del cubo y el tipo de cubo a resolver, trasladándoselo al robot Yumi que mediante la programación con RobotStudio se encarga de realizar los movimientos necesarios para cogerlo. Tras ello, el robot posiciona el cubo en las 6 ubicaciones diferentes para realizar la toma de fotos de las 6 caras, en cada ubicación la cámara realizara una captura de imagen la cual es enviada a Matlab para realizar su procesamiento y obtener los movimientos necesarios para la resolución del cubo, que serán enviados al robot. Además, la herramienta Matlab también se encargará de reproducir las notas de voz correspondientes en el momento asignado, o en las circunstancias especificadas.

A modo resumen se podría decir que la cámara se encarga de realizar las fotos para obtener información del cubo, Matlab de interpretar dicha información y trasladársela al robot y el Yumi de realizar los movimientos necesarios para llevar a cabo la aplicación.

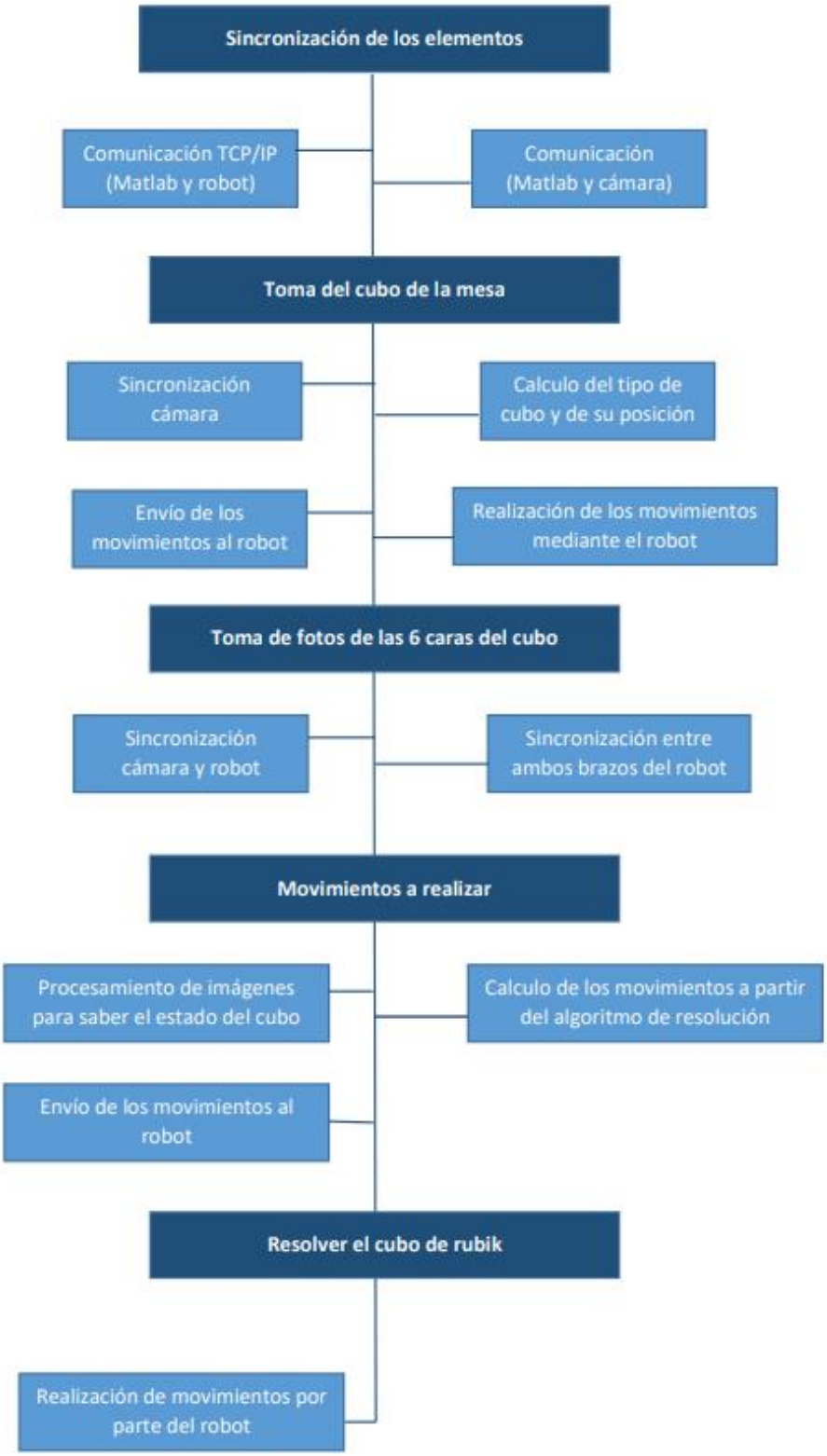


Ilustración 18: Esquema de interacción elementos principales

6.2 DISEÑO DE LA ZONA DE TRABAJO

Para esta aplicación se está utilizando una célula de montaje implementada por otros estudiantes en el curso 2018-2019. Esta célula además de para este proyecto, también se utiliza para diferentes aplicaciones, por lo que es necesario diseñar una zona de trabajo que no suponga alteraciones en el correcto funcionamiento de otros proyectos y no interfiera con sus escenarios.

6.2.1 Mesa de recepción del cubo de Rubik

Previamente a la realización de esta aplicación, la mesa de trabajo del robot era la que se muestra en la ilustración 19, en la que se pueden visualizar instalados en ella diferentes componentes que son utilizados en otras aplicaciones.

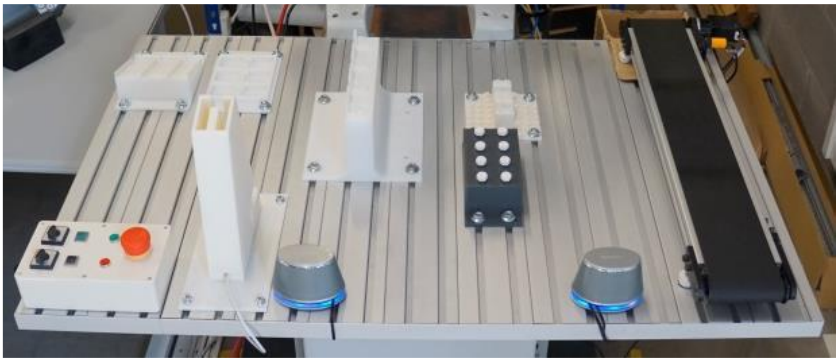


Ilustración 19: Escenario previo a la aplicación

El anterior alumno partiendo de este escenario realizó una mesa provisional para que el robot pudiera tomar y depositar el cubo de rubik en ella. Sin embargo, esta mesa no se consideraba valida dado que era bastante inestable y poseía una sola pata que además estaba hecha de poliespán. Por esta razón, se ha visto la necesidad de diseñar una superficie plana que no interfiera en el escenario de las demás aplicaciones, pero que sea duradera y adecuada para este proyecto.

El diseño de la mesa se ha realizado mediante la herramienta AutoCAD y cuenta con tres componentes.

- La parte superior de la mesa: Este componente es cuadrado y plano, es el lugar donde se deposita el cubo y donde el robot va a cogerlo.

- Las patas de la mesa: La mesa consta de dos patas las cuales se sitúan en los extremos de la parte superior de la mesa y se utilizan para anclar todo el conjunto a la mesa de la célula del robot.

- El refuerzo: Para darle más estabilidad a la mesa y disminuir las posibilidades de ruptura se le ha añadido un refuerzo que une las dos patas de la mesa, consiguiendo de esta forma una mayor robustez.

Cabe destacar que todas las esquinas han sido levemente redondeadas y se han realizado todos los chaflanes posibles, minimizando de esta manera la posibilidad de rupturas en caso de choques.

Finalmente el diseño 3D que se ha obtenido de la mesa es el que se presenta en la ilustración 20.

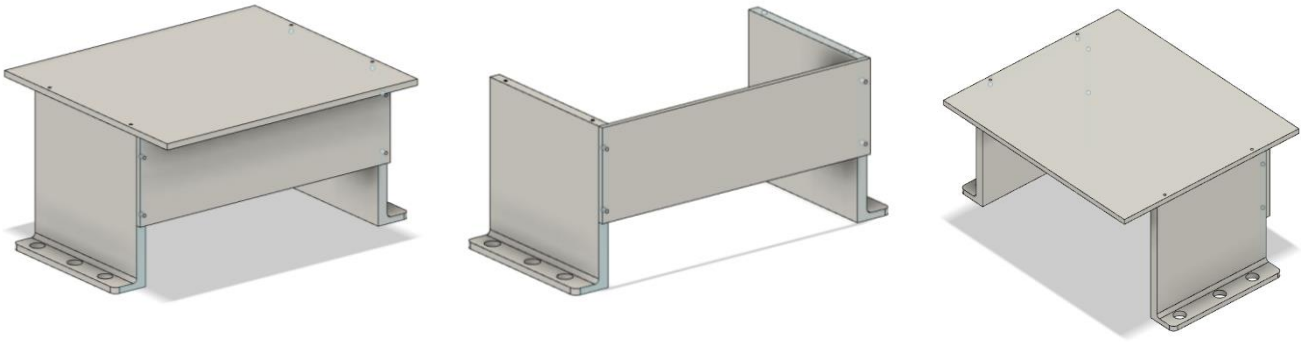


Ilustración 20: Mesa de recepción del cubo

Es importante mencionar que la mesa se ha impreso en color gris, sin embargo, en la parte superior (superficie plana), se ha cubierto con color blanco. Esto se debe a que se ha diseñado un procesamiento de imágenes en el que no se detecta el blanco, por lo que al tomar la primera foto para buscar la ubicación del cubo en la mesa, con la superficie blanca se descarta la posibilidad de error por detectar otro color o por efecto de algún brillo, ver ilustración 21.

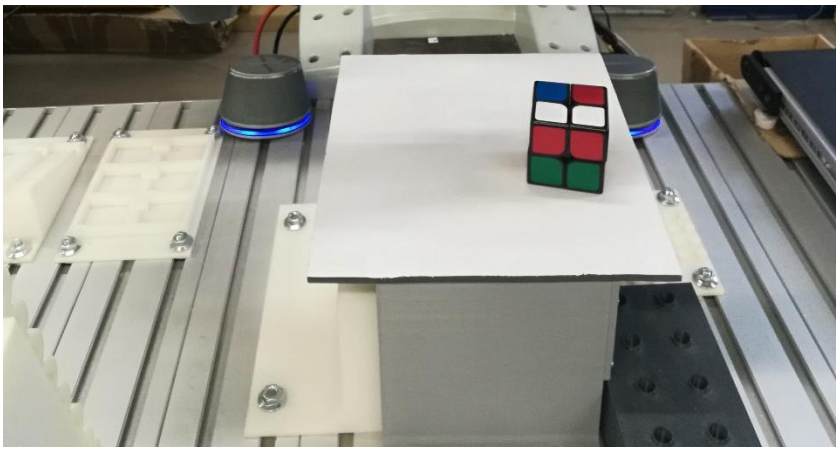


Ilustración 21: Mesa final de recepción del cubo de rubik

6.2.2 Carcasa para la cámara de visión artificial

La cámara suministrada por Infaimon, UI 5584 LE-C-HQ, únicamente la componen el objetivo y la placa base, no tiene ningún tipo de superficie ni carcasa que la protejan de posibles golpes o de la suciedad, como puede ser el polvo. La carcasa anterior era demasiado grande y voluminosa, y se adentraba en exceso en el campo de movimiento del robot, como se puede ver en la ilustración 22.

Ilustración 22: Antigua carcasa ubicada en el robot

Debido a esto se ha optado por diseñar una nueva carcasa, que se ajuste mejor a la cámara (no dejando margen de movimiento para evitar posibles golpes o modificaciones de posición). Además, se han reducido todo lo posible las dimensiones de esta y para ello el ventilador que tenía junto a la placa base se ha desplazado a la parte de fuera de la carcasa, dando la posibilidad de minimizarla todavía más.

Destacar que al igual que en el apartado anterior, el diseño de los componentes 3D se ha realizado utilizando el programa de AutoCAD, y se han realizado varios diseños de prueba antes de lograr el diseño más óptimo que ha sido el final.

Como se puede observar en la ilustración 23, en la parte posterior de la carcasa se han realizado una especie de agujeros, ya que es donde irá situado el ventilador. En la parte inferior también se ha realizado un agujero cuadrado que sirve para meter y sacar los cables de conexión de la cámara (el cable Ethernet).

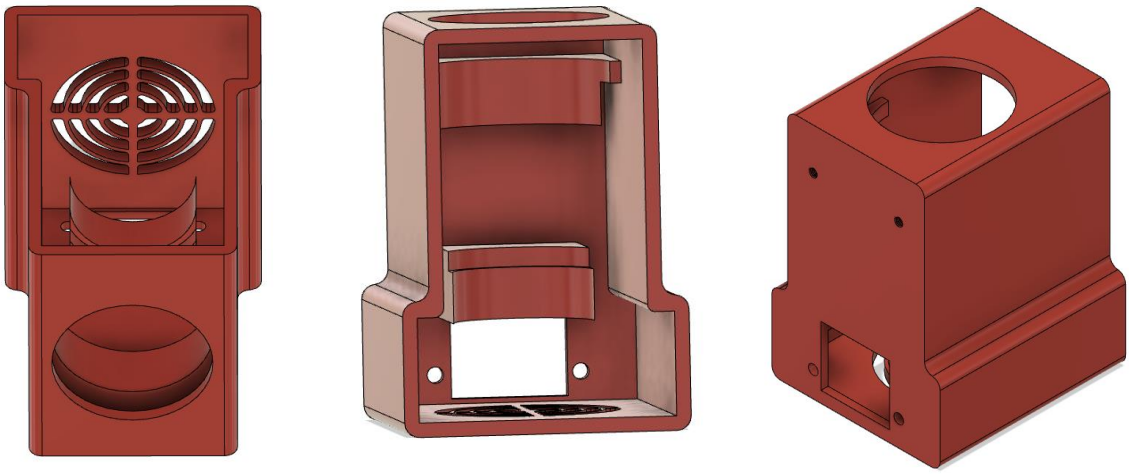


Ilustración 23: Carcasa para la cámara

Además de la carcasa, también se ha diseñado una tapa para cubrirla, y evitar que entre polvo en la cámara. La tapa se ha diseñado de tal forma que encaje perfectamente con la carcasa, ver ilustración 24.

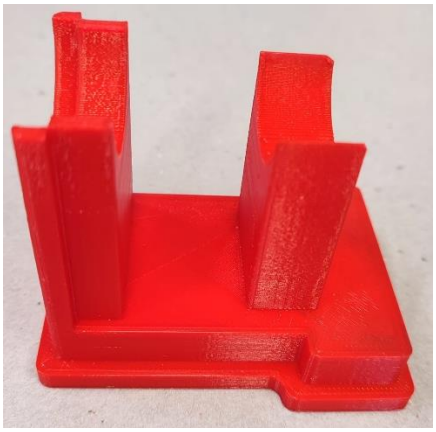


Ilustración 24: Tapa de la carcasa

6.2.3 Fijación de la carcasa y pantalla a la célula

Como se ha mostrado en el apartado anterior es preciso un componente que integre la cámara en el robot. Actualmente el sistema ya cuenta con una sujeción de ese tipo, sin embargo, es demasiado grande y se adentra en el espacio de trabajo del robot, por lo que se ha considerado diseñar una nueva fijación.

En este caso, ya que la pantalla también irá situada en la parte superior del robot, se ha desarrollado un componente que incluya las fijaciones tanto de la pantalla como de la carcasa de la cámara, quedando de esta manera unidos a la estructura del robot.

Este soporte se ha diseñado mediante el software de AutoCAD, haciendo uso de sus funciones para poder llevar a cabo a posteriori la impresión 3D del componente (ver ilustración 25).

Ilustración 25: Fijación de la carcasa y pantalla a la célula

Destacar que como unión entre la carcasa y la fijación se ha colocado una especie de tornillo deslizante, con el cual se puede modificar la inclinación de la cámara para adecuarla al ángulo y posición deseados. De esta manera se facilita cualquier alteración necesaria.

La ranura que se observa en la parte superior de la sujeción es donde irá anclada la pantalla táctil, para que se tenga una buena visión de esta.

6.2.4 Altavoces

Una de las mejoras de la aplicación es dotarle de voz al robot, por lo que para ello es necesario un amplificador de sonido, en este caso altavoces.

Por lo tanto se integrará un par de mini altavoces (ver ilustración 26) a los lados del robot YuMi. Los altavoces serán los encargados de reproducir las notas de voz cuando el programa se lo indique, dando la sensación de que el que está hablando es el propio robot.

Ilustración 26: Altavoces

6.3 IMPLEMENTACIÓN DEL SISTEMA

En este apartado se va a detallar el procedimiento seguido para la implementación del sistema.

Tras aprobar estos diseños, se ha pasado a la impresión de los mismos mediante una impresora 3D que posee el departamento.

Primeramente se ha realizado la impresión de la carcasa y la tapa de la cámara, y tras comprobar que encajan correctamente se ha pasado a imprimir la fijación. Una vez impresa la fijación se ha realizado el montaje de esta con la carcasa y se ha verificado que su unión sea correcta. Por lo tanto, se ha integrado tanto la fijación como la carcasa en la célula del robot.

En segundo lugar, se ha comprobado el diseño de la mesa y su impresión 3D. Una vez impresos los 3 elementos de la mesa, se ha realizado el montaje y se ha instalado en la célula del robot.

En tercer lugar, se ha montado la pantalla en la sujeción diseñada y se ha instalado la minitorre sobre su propia fijación, con las debidas conexiones.

Por último, se han colocado los altavoces en su lugar correspondiente y se han conectado a la minitorre.

Tras el montaje del escenario se ha obtenido un resultado como el que se puede observar en la ilustración 27.

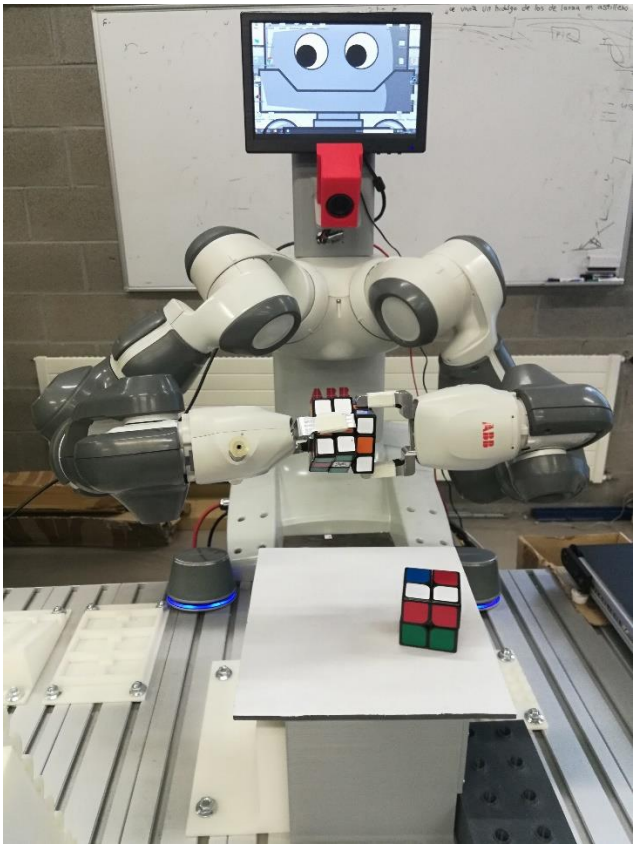


Ilustración 27: Célula robotizada de entretenimiento

En los siguientes apartados se detalla, por un lado, la configuración realizada para los diferentes componentes del sistema y por otro lado la programación para la optimización de la aplicación. En las ilustraciones 28 y 29 se presenta un esquema de cómo están estructurados dichos apartados.

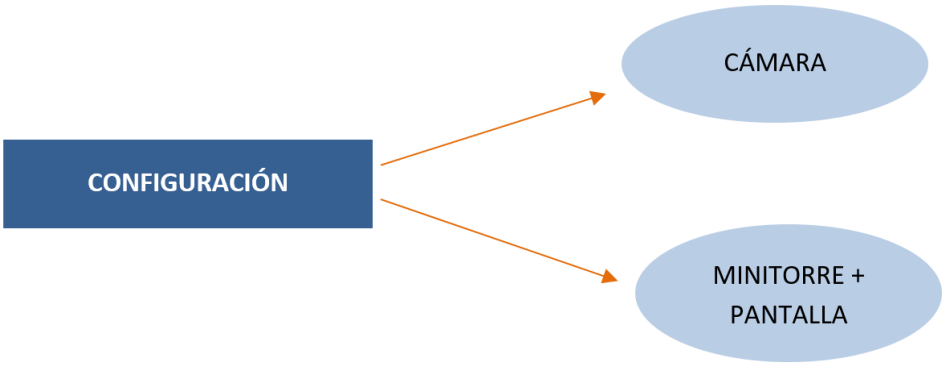


Ilustración 28: Desglose configuración

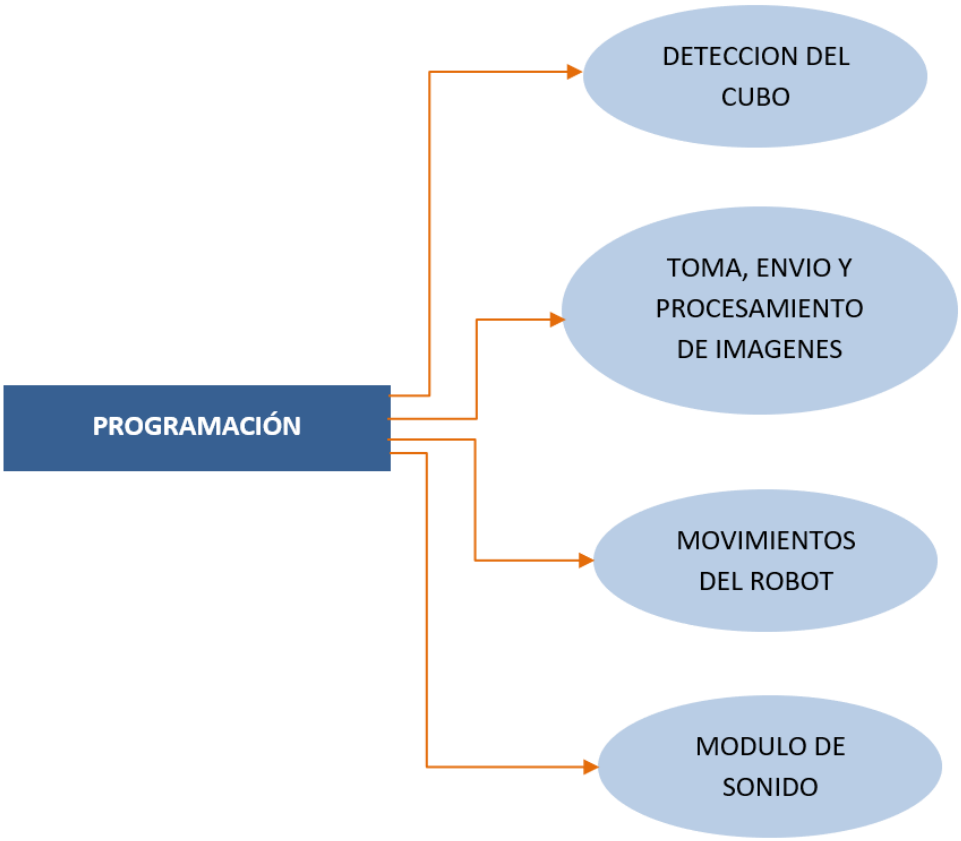


Ilustración 29: Desglose programación

6.3.1 Configuración de la cámara

Como se ha citado anteriormente la cámara es el modelo UI 5584LE-C-HQ de Infaimon. En el análisis de alternativas se ha detallado que este modelo en concreto posee una interfaz gigE (Gigabyte Ethernet) con la que mediante un cable de ethernet se conecta al ordenador en cuestión con un conector RJ45. Además, la cámara se alimenta mediante dos cables (conectados a la célula del robot), que le dan a la placa base la energía necesaria para funcionar.

Tras alimentar y conectar la cámara con el ordenador se ha realizado la configuración de la misma, para lo que Infaimon ofrece varias aplicaciones. Primeramente se hace uso de la aplicación IDS Camera Manager, mediante la cual se puede gestionar la cámara una vez conectada al PC, teniendo acceso a su información general y estado de funcionamiento. Además, mediante esta aplicación se puede modificar la IP de la cámara para configurarla en la red que se desee de manera sencilla y rápida, ya que una de las ventajas de esta aplicación es que detecta la cámara automáticamente una vez conectada al sistema. En la siguiente imagen se puede observar como la aplicación detecta la cámara y la dirección IP que esta posee. En la parte derecha de la ilustración 30 también se pueden ver los distintos parámetros de la cámara, desde las características generales hasta la configuración y datos de red.

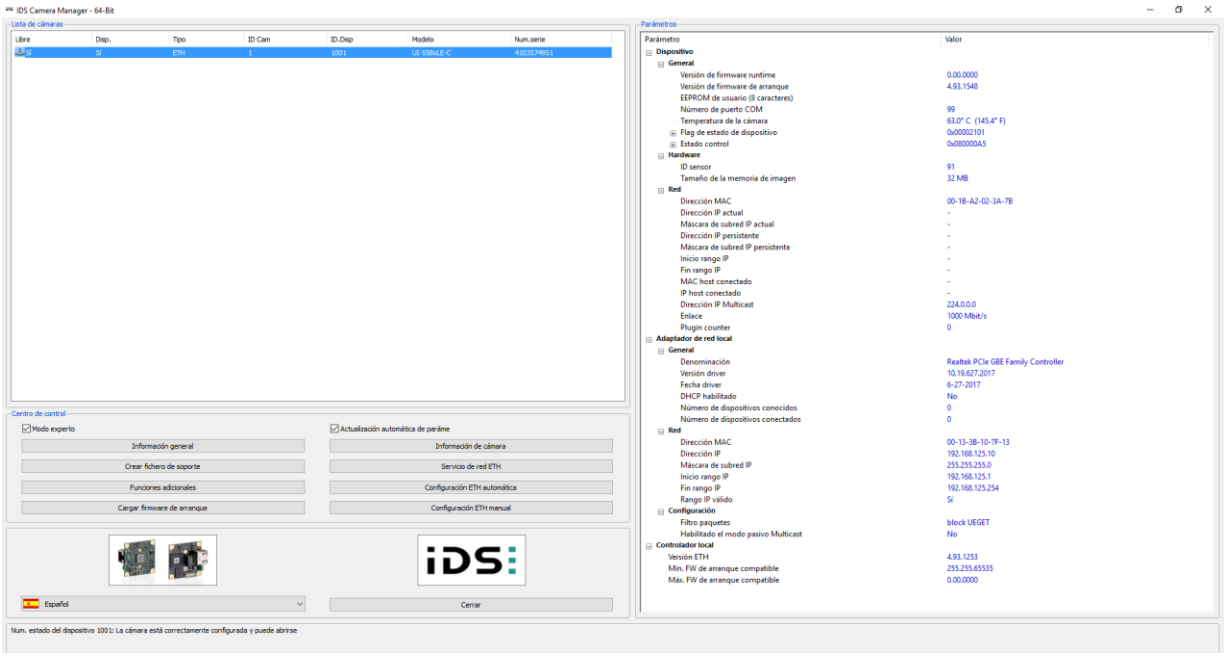


Ilustración 30: Ventana de aplicación IDS Camera

En segundo lugar, y tras haber realizado la configuración ethernet de la cámara mediante la aplicación anterior, se pasará a utilizar la aplicación Ueye cockpit. Esta aplicación de Infaimon presenta la posibilidad de modificar diferentes parámetros de la cámara muy útiles para este proyecto, entre ellos: el modo de disparo, la resolución, el pixel clock, etc. Ya que la cámara viene de fabrica con una configuración estándar que probablemente no sea la adecuada para las necesidades del proyecto, es muy beneficioso contar con la posibilidad de editar los parámetros presentados, además de la capacidad de guardarlos. Mediante uEye Cockpit se ha obtenido una visión en tiempo real de lo que capta la cámara (en este caso la mesa y el cubo de Rubik) para modificar como se ve en la ilustración 31 los valores de pixel clock, frame rate y exposure time (además de otras tantas posibilidades más).

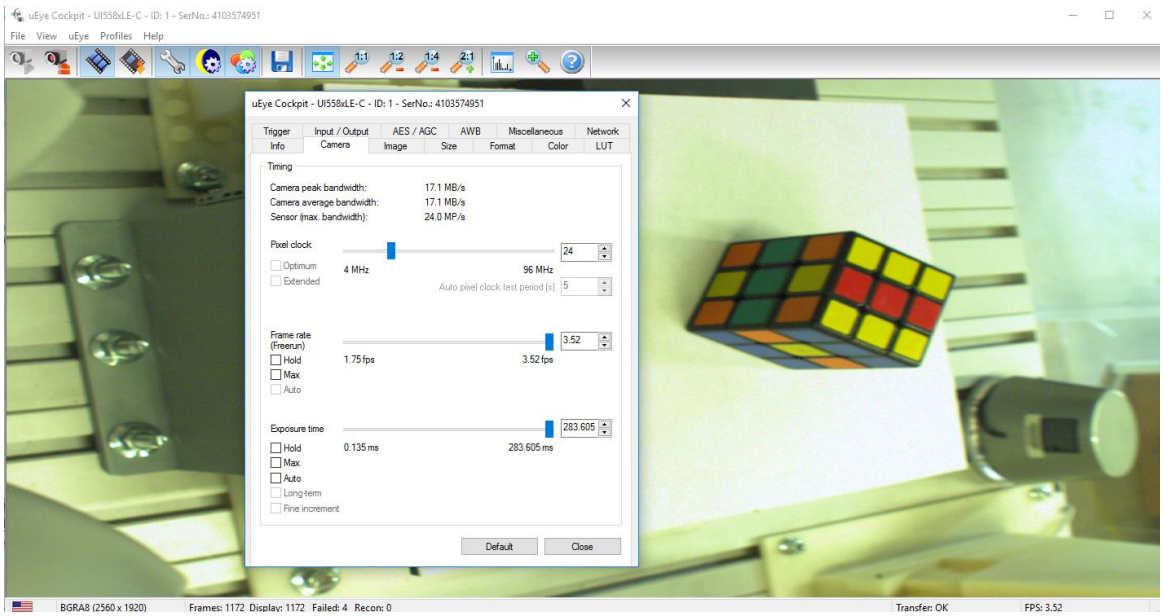


Ilustración 31: Ventana de la aplicación uEye Cockpit

6.3.2 Configuración de la minitorre y pantalla táctil

Primeramente, como elemento de pasarela entre la cámara y el robot se utilizaba un ordenador de mesa. Sin embargo, como ya se ha comentado, debido a que uno de los objetivos del proyecto es facilitar el traslado de la aplicación, se van a configurar tanto una minitorre como una pantalla táctil que se puedan situar en la célula robotizada.

Por un lado, en la minitorre se han instalado todos los programas y licencias necesarias para llevar a cabo el proceso de resolución del cubo:

(1) RobotStudio: para realizar la programación de los movimientos del robot y de esta manera poder manipular el cubo de Rubik de la forma deseada.

(2) Matlab: mediante este programa se realiza el procesamiento de imágenes y el módulo de sonido. Para ello dentro de la misma es imprescindible instalar diferentes toolboxes (Image Adquisition Toolbox, Image Processing Toolbox, etc.).

(3) IDS Camera manager: como se ha comentado anteriormente esta aplicación se utiliza para darle a la cámara la IP deseada, por lo que es necesario realizar su instalación en la minitorre.

(4) uEye Cockpit: mediante esta herramienta se modifican el modo de disparo, la resolución y demás parámetros de la cámara, esto la hace imprescindible para la realización del trabajo, y en consecuencia se tiene que instalar en el ordenador.

Por otro lado, el robot va a contar con una pantalla táctil, mediante la cual se podrán lanzar mensajes y mostrar videos e imágenes. Además, dado que la pantalla escogida es táctil, el usuario podrá manipular y visualizar los programas de la aplicación de forma fácil. En este caso, con objetivo de atraer en mayor medida la atención de diferente público se ha decidido mostrar mediante dicha pantalla la cara de un robot.

6.3.3 Programación para la detección del cubo

Para dar comienzo a la aplicación se sitúa el cubo de Rubik (de 2x2 o 3x3) en la mesa de recepción que se ha diseñado e impreso previamente. Una vez que la cámara detecte el cubo el robot se encarga de recogerlo. Para ello es de vital importancia lograr referenciar el cubo respecto al origen del robot. Dado que hay un programa con el que se logran referenciar las 3 esquinas del cubo que aparecen en la imagen, pero en pixeles, es necesario desarrollar un programa con el que se conviertan dichas medidas en cm. El procedimiento que se explica a continuación se ejecuta 3 veces, una vez para cada esquina detectada del cubo.

En este caso, se ha referenciado todo el proceso sobre la esquina inferior izquierda de la mesa, de la cual se han tomado las coordenadas partiendo de la base del robot. Dado que al modificar la posición de la cámara esta se encuentra más inclinada y alejada que anteriormente, se ha considerado necesario generar un script en Matlab con el que se detecten de manera más precisa las coordenadas de la posición del cubo en la mesa.

Con ese objetivo, se ha diseñado una hoja cuadriculada en la cual cada pequeño cuadrado tiene las siguientes medidas: 1x1cm. En total, cumpliendo con las dimensiones de la mesa, hay 24 cuadrados a lo ancho y 26 a lo largo, es decir, 24cm en el eje X y 26cm en el eje Y.

Debido a que la cámara no se encuentra en un ángulo perpendicular con la mesa, al tomar la foto los cuadrados no tienen la medida real, por lo que se han cogido varias medidas de referencia.

Primeramente, concretar que en Matlab el origen del eje de coordenadas se encuentra en la parte superior izquierda de la imagen, por lo que las medidas tomadas serán respecto a ese punto.

En la ilustración 32 se puede ver un ejemplo sencillo de como con la herramienta *data tips*, colocando el cursor en los puntos deseados se obtiene el valor de las coordenadas X e Y (entre otras).

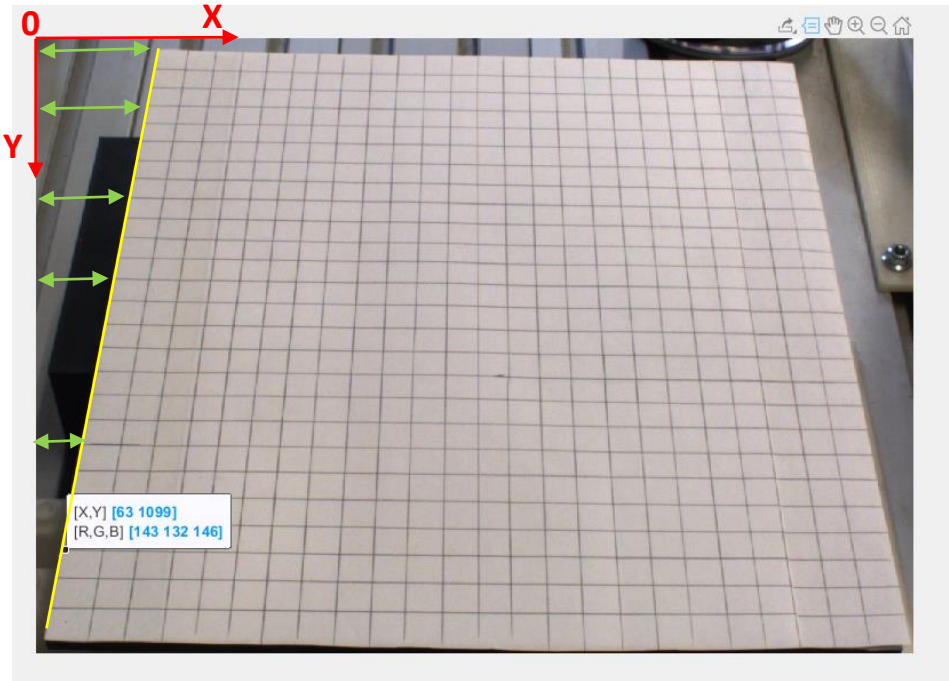


Ilustración 32: Hoja de cuadricula para posicionar el cubo

Haciendo uso de esta herramienta se han tomado diferentes valores para después realizar iteraciones. Para empezar, se han tomado todos los valores de Y de las intersecciones que se encuentran en la primera columna (línea amarilla), comenzando por la más lejana a la referencia, que sería la de la parte inferior de la imagen. Como se puede ver en la ilustración que se adjunta a continuación, los valores de Y se han guardado en un vector llamado *VecY*, en el que hay un total de 27 valores. Tras ello haciendo uso del valor de *posCuboY_Calculo*, que es el valor en pixeles de la posición en Y de una de las esquinas del cubo (se ha obtenido de otro subprograma: *Posicion_cubo*), mediante una estructura de *while* se logra deducir entre qué puntos de la cuadricula esta esa esquina del cubo. Para ello se compara el valor de *posCuboY_Calculo* con los diferentes valores del vector *VecY*, hasta que se encuentra un valor en el vector *VecY* que es menor que el guardado en *posCuboY_Calculo*. Tras ello se guardará el valor de *j* en la variable: *Ya*, que corresponderá con el último punto que cumple el *while,* ver ilustración 33.

```
VecY= [1288 1220 1161 1099 1043 984 927 873 817 763 716 662 613 562 515 473 427 387 347 299 262 220 188 147 105 70 29];
j=1;
while posCuboY_Calculo<VecY(j) & j<27
    j=j+1;
end
Ya=j;
```

Ilustración 33: Valores de Y

Para continuar, se han obtenido dos valores. Por un lado, lo que va aumentando la distancia entre columnas de una fila a otra, para ello se ha cogido la sucesión de valores de la misma columna y diferentes filas y se ha realizado la resta con los valores posterior y previo, obteniendo así una serie de valores con los que se ha logrado la media, en este caso de 0.4449, que sería el valor que va

aumentando el cuadrado inicial (el más cercano a la referencia), a medida que se aleja en Y. Por otro lado, se ha medido la distancia entre columnas en la primera fila (la más cercana a la referencia) y se ha realizado la media, obteniendo así el valor de 60. Como se puede ver en la siguiente ilustración estos dos valores se han guardado en las variables *distX* y *distX1.*

Una vez obtenidos dichos valores se toman los datos para crear el vector *Xini*, que observando la ilustración 34 se logran como en el caso del vector *VecY*, consultando los valores que se obtienen en los puntos de intersección entre la línea amarilla y las filas de la cuadricula pero para X. Para mejorar la explicación se pueden ver una flechas verdes que indicarían los valores dentro del vector *Xini,* ver ilustración 32.

Llegados a este punto ya se tienen los valores de toda la primera columna, tanto de X como de Y, además del valor medio de cada columna en la primera fila y el valor a sumar a la X de cada fila para pasar a las diferentes columnas.

Para lograr los valores de todos los puntos de la cuadricula se programan dos sentencias de *for*. En las cuales se hace uso de los datos obtenidos anteriormente para ir rellenando diferentes vectores, cada uno correspondiente a una fila y así entre ellos generar una matriz de vectores. El detalle de esta programación se puede ver en la siguiente imagen.

```
distX=0.44449; %lo que aumenta la distancia entre columnas de una fila a otra
distX1=60;%distancia entre columnas arriba

Xini= [265 257 247 244 236 228 218 212 202 192 186 178 167 155 147 137 121 116 108 95 86 76 65 53 41 28 13];
%los valores desde la parte izquierda de la foto hasta las diferentes filas de la mesa
for k=1:length(Xini)
VecX1(1)=Xini(k);
for i=2:length(VecY)
    Xini(k)=Xini(k)+distX1;
    VecX1(i)=Xini(k);
end
distX1=distX1+distX;
VecX2(:,:,k)=VecX1;
end
```

Ilustración 34: Valores de X y demás puntos de la cuadricula

A continuación se compara el valor de *posCuboX_Calculo* que es la posición de una de las esquinas del cubo en X, con los valores del vector *VecXa* y en el momento que este último sea mayor parará el *while* y se guardará la posición en la que se encontraba dentro del vector, es decir, el valor de *j*.

Tras esta programación ya se tienen las posiciones de los puntos entre los que se encuentra la esquina del cubo, estos puntos forman uno de los cuadrados de la cuadricula y realizando una iteración se calcula el punto medio de este cuadrado que será donde se estima que esta una de las esquinas del cubo.

```
VecXa=VecX2(:,:,29-Ya);
j=1;
while posCuboX_Calculo>VecXa(j) & j<25
    j=j+1;
end
Xa=j+1;

Xa2=VecXa(j);

Xa;Ya;
Xb=Xa-1;
Yb=Ya-1;

Y1_Cubo=Yb+ (posCuboY_Calculo-VecY(Yb))/(VecY(Ya)-VecY(Yb))-1;
X1_Cubo=Xb+ (posCuboX_Calculo-VecXa(Xb))/(VecXa(Xa)-VecXa(Xb))-1;
```

Ilustración 35: Obtención de las coordenadas de la esquina del cubo

Como se ha visto en la ilustración 35, de la programación realizada se obtienen dos valores: *X1_Cubo* y *Y1_Cubo*. Estos valores representan la posición de la esquina del cubo en los ejes X e Y.

A modo de resumen sobre la programación presentada en este apartado, se muestra el siguiente esquema (ver ilustración 36) en el cual se puede ver la interacción del programa *Calculo_pos_cubo,* el cual se ha explicado con detalle en este apartado, con el programa *Posicion_cubo.*

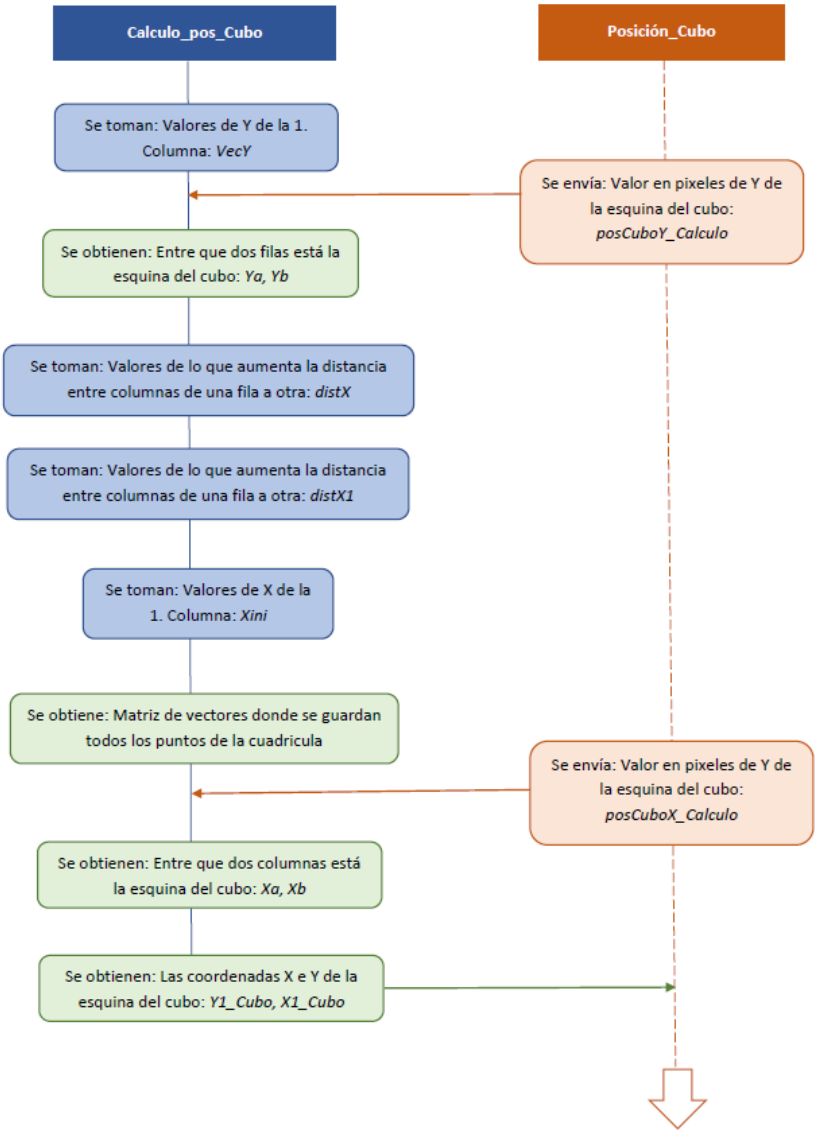


Ilustración 36: Iteración de Calculo_pos_Cubo con Posicion_cubo

6.3.4 Programación para la Toma, Envío y procesamiento de imágenes

Otro de los aspectos importantes a mejorar es la toma, envío y procesamiento de imágenes, debido a que en la antigua aplicación el robot tardaba mucho al tomar las fotos. Esto se debe a que tras coger el cubo de la mesa el robot se posicionaba en la ubicación de la primera foto, después avisaba a Matlab de que ya podía realizar la primera foto y Matlab al terminar de realizar la foto volvía a avisar al robot de que ya podía moverse a la siguiente posición. Además, el procesamiento de cada imagen tomada se realizaba tras la toma de cada fotografía, alargando a un más los tiempos de espera.

Para solucionar este problema y mejorar los tiempos de resolución del cubo de Rubik se han hecho una serie de pruebas con las que se ha llegado a la conclusión de que la secuencia para la toma y envío de imágenes más optima es la que se muestra en las ilustraciones 37 y 38.

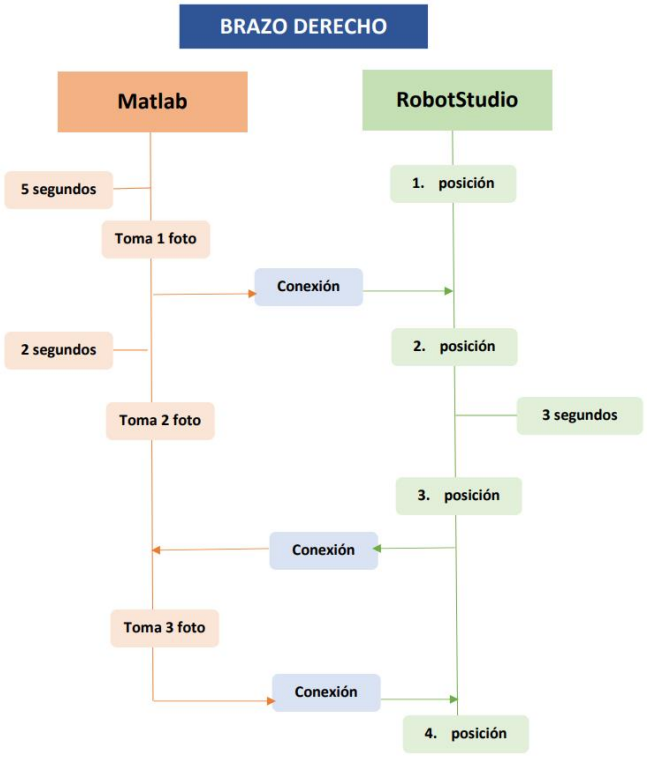


Ilustración 37: Toma y envío de imágenes brazo derecho

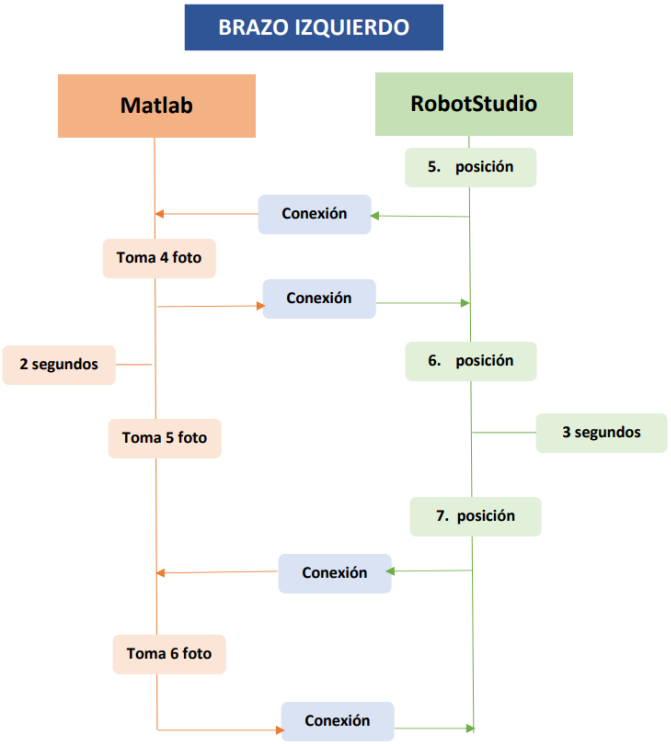


Ilustración 38: Toma y envío de imágenes brazo izquierdo

Foto 1:

Las primeras fotos se toman cuando el robot tiene el cubo agarrado con el brazo derecho, por lo que la primera parte de la programación en la que hay que centrarse tiene que ver con este brazo. Para empezar el robot se sitúa en la posición de la foto número 1, en la que espera hasta que Matlab le envía una señal de que ya puede cambiar de posición. En este caso, se ha visto la necesidad de que Matlab avise al robot para cambiar de posición debido a que el procedimiento que realiza Matlab en la primera foto es más tedioso que en otros casos, puesto que analiza el tipo de cubo a resolver, por lo que calcular el tiempo de espera necesario es muy complejo.

En la ilustración 39 se puede ver la programación para la toma de la primera foto de RobotStudio.

```
PROC Principal()
    g_Init \maxSpd:=10 \ holdForce:=5 \Calibrate; ! Calibracion de la pinza, indicación
de la fuerza máxima en N y velocidad máxima en mm/s
    waitTime 1;
    g_GripOut;

! IF A = 1 THEN
    !Crear comunicacion
    SocketCreate server1;
    !SocketBind server, "10.109.253.189",14044;
    SocketBind server1, "192.168.125.1",14044;
    SocketListen server1;
    SocketAccept server1,client1;

  Foto1;
! Waittime 3;
  !enviar mensaje al cliente
 !SocketSend client1,\str:="A";
  !recivir mensaje
  SocketReceive client1,\RawData:=data;
  UnpackRawBytes data,1,foto\Ascii:=2;
  oki:=StrtoVal(foto,CuboTres);
 ! WaitUntil CuboTres=0 OR CuboTres=1;
```

Ilustración 39: Programación 1 foto RobotStudio

Mientras el Robot se encuentra en la primera posición, como ya se ha comentado, realiza la toma de la primera foto, sin necesidad de ser avisado por RobotStudio, en este caso esperando un tiempo de 5 segundos, que tras realizar varias pruebas es el mínimo conseguido. Además, tras la toma de la imagen se analiza el tipo de cubo a resolver (2x2 o 3x3).

En la ilustración 40 se puede ver la programación de Matlab para la toma de la primera imagen.

```
foto=50;%equivale a 2

tc=tcpip("192.168.125.1",14044);
pause(20);
fopen(tc);

pause(5)%num_foto=fread(tc); En lugar de leer la variable que indica que el robot llegó a la posición
vid = videoinput('winvideo', 1, 'RGB24_2560x1920');
vid.TriggerRepeat = 1
start(vid);

Cara1 = getsnapshot(vid);%Tomar la primera foto
CaraX=Cara1;
 Y=imrotate(X,90);
 Z=fliplr(Y);

Copy_2_of_Proc_CaraF;%Se procesa la foto para saber si el cubo es de 2x2 o 3x3


if numItemRed == 0 & numItemBlue == 0 & numItemYellow ==0  & numItemGreen == 0 & numItemOrange == 0
    Cubo3=48;%En caso de que el cubo sea de 3x3 se cambia la variable a 48 (0 en ascii)
end
fwrite(tc,Cubo3);%se envia la información al robot
pause(1);
close all
imshow(Z)
```

Ilustración 40: Programación Matlab 1 foto

Foto 2:

Una vez el robot se haya posicionado en la toma de la segunda foto, espera un tiempo de 3 segundos para pasar a la posición 3. Esto se debe a que se ha calculado que Matlab solo necesita ese tiempo para la toma de la foto.

En la ilustración 41 se puede ver como se han suprimido los comandos para realizar el intercambio de información con Matlab.

```
Foto2;
Waittime 3;
 !enviar mensaje al cliente
!SocketSend client1,\str:="B";
 !recivir mensaje
! SocketReceive client1,\RawData:=data;
!UnpackRawBytes data,1,foto\Ascii:=2;
! oki:=StrtoVal(foto,numfoto);
! WaitUntil numfoto=3;
```

Ilustración 41: Programación 2 foto RobotStudio

Para la toma de la segunda foto, Matlab espera 2 segundos a que el robot se haya colocado en el lugar adecuado y tras ese tiempo saca la foto. Ver ilustración 42.

```
pause(2);%se espera un tiempo en vez de recibir la informacion de que el robot esta en la posicion
Cara2 = getsnapshot(vid);%se saca la segunda foto
X=Cara2;
Y=imrotate(X,90);
Z=fliplr(Y);
imshow(Z)%se muestra
```

Ilustración 42: Programación 2 foto Matlab

Foto 3:

El robot pasa a la posición de la tercera foto y tras ello avisa a Matlab y se sincroniza con este, dado que la 4 foto será tomada con el otro brazo del robot. La sincronización además de realizarse con Matlab también se realiza con el otro brazo, de esta manera una vez le haya dado el cubo al brazo izquierdo, el derecho se coloca en la posición Foto4, que a pesar de su nombre es una simple posición para alejarse del centro de la célula y no molestar al brazo que se está ejecutando. Ver ilustración 43.

```
Foto3;
!waittime 3;
!enviar mensaje al cliente
SocketSend client1,\str:="C";
!recivir mensaje
SocketReceive client1,\RawData:=data;
UnpackRawBytes data,1,foto\Ascii:=2;
oki:=StrtoVal(foto,numfoto);

 WaitUntil numfoto=4;

SocketSend client1,\str:="Z";


!Cerrar comunicacion
SocketClose server1;
SocketClose client1;

WaitSyncTask sync_RL_A, ListaTareas;

waittime 2;

Foto4;
 WaitSyncTask sync_RL_B, ListaTareas;
 waitTime 3;
```

Ilustración 43: Programación 3 foto y 4 posición RobotStudio

Debido a que tras la toma de la tercera foto Matlab cerrara la conexión y el robot cambiara el cubo de mano, para la toma de esta foto se recibe información de la posición del robot y también se le avisa una vez se haya sacado la foto. Ver ilustración 44.

```
foto=foto+2
num_foto=fread(tc);%se recibe la informacion de que el robot está en la posicion de la 3 foto

Cara3 = getsnapshot(vid);%se saca la foto
fwrite(tc,foto);%se envia la información de que el robot puede pasar a la 4 foto
X=Cara3;
Y=imrotate(X,90);
Z=fliplr(Y);
imshow(Z)
fclose(tc);%se cierra la conexión
```

Ilustración 44: Programación 3 foto Matlab

Ubicaciones brazo derecho:

Las ubicaciones a las que corresponden cada posición de las fotos se recogen en la siguiente programación y tras realizar varias pruebas se han modificado a la ubicación más optima y eficaz, ver ilustración 45.

```
PROC Foto1()
    g_Gripin \holdForce:=10;
    waitTime 1;
    GripLoad cubo3;
        waitTime 1;
    MoveJ Pos_Cubo_2,v200,fine,Servo_R;
    MoveJ Tomar_Fotos11,v300,z100,Servo_R\WObj:=wobj0;
        MoveJ Tomar_Fotos,v300,z100,Servo_R\WObj:=wobj0;
    waittime 2;
      ENDPROC
      PROC Foto2()
     WaitTime 2;
        MoveL Tomar_Fotos2,v20,z100,Servo_R\WObj:=wobj0;
    waittime 2;
      ENDPROC
      PROC Foto3()
            MoveL TomarFoto3,v20,z100,Servo_R\WObj:=wobj0;
    waittime 2;

      ENDPROC
      PROC Foto4()
    waitTime 2;
    g_GripOut;
        WaitTime 0.2;
    MoveJ Apro_Fotos,v200,z100,Servo_R\WObj:=wobj0;
            MoveJ Apro_R5,v200,z100,Servo_R\WObj:=wobj0;
            waitTime 1;
        waitTime 1;
      ENDPROC
```

Ilustración 45: Ubicaciones brazo derecho

Foto 4:

Como ya se ha comentado, para la toma de la cuarta foto el robot tiene que cambiar el cubo de mano, por lo que cuando el brazo izquierdo coge el cubo pasa a colocarse en la posición para la toma de esta cuarta foto. Para la realización de esta foto, si habrá una comunicación entre Matlab y RobotStudio, ya que al tener que realizar el cambio de mano del cubo no era posible estimar con concreción el tiempo de espera necesario para la toma de la foto, por lo que se ha decidido que la mejor opción era realizar la comunicación.

En la ilustración 47 se puede ver que Matlab no saca la foto hasta que RobotStudio no se lo indica, y de la misma manera el brazo izquierdo no pasa a la siguiente posición hasta que Matlab le dice que puede hacerlo, ver ilustración 46.

```
PROC Principal()

    WaitSyncTask sync_RL_A, ListaTareas;
    !Crear comunicacion
    SocketCreate server1;
    SocketBind server1, "192.168.125.1",14043;
     SocketListen server1;
      SocketAccept server1,client1;

    Foto1;
     !enviar mensaje al cliente
    SocketSend client1,\str:="D";
    !recivir mensaje
    SocketReceive client1,\RawData:=data;
    UnpackRawBytes data,1,fotos\Ascii:=2;
    oki:=StrtoVal(fotos,CuboTres);
   WaitUntil CuboTres=0 OR CuboTres=1;
```

Ilustración 46: Programación 4 foto RobotStudio

```
tc=tcpip("192.168.125.1",14043);%se vuelve a abrir la conexión
fopen(tc);

foto=foto+1
num_foto=fread(tc);%se recibe la información de que el robot esta en la posicion de la 4 foto

Cara4 = getsnapshot(vid);%se saca la foto
stop(vid);
fwrite(tc,Cubo3);
X=Cara4;
Y=imrotate(X,90);
Z=fliplr(Y);
imshow(Z)
```

Ilustración 47: Programación 4 foto Matlab

Foto 5:

Tras llegar a la posición de la quinta foto, el robot esperara un tiempo de 3 segundos (ver ilustración 48), para que Matlab saque la foto de esa cara del cubo.

```
!waitTime 1;
Foto2;
 Waittime 3;
!enviar mensaje al cliente
! SocketSend client1,\str:="E";
!recivir mensaje
! SocketReceive client1,\RawData:=data;
! UnpackRawBytes data,1,fotos\Ascii:=2;
! oki:=StrtoVal(fotos,numfoto);
! WaitUntil numfoto=6;
```

Ilustración 48: Programación 5 foto RobotStudio

En el caso de Matlab, se esperan 2 segundos para que el robot llegue a la quinta posición, tras ello, sin realizar ningún tipo de comunicación pasa a sacar la foto de la cara. Ver ilustración 49.

```
pause(2);%se espera un tiempo para que el robot vaya a la 5° posicion
Cara5 = getsnapshot(vid);%se saca la foto
X=Cara5;
Y=imrotate(X,90);
Z=fliplr(Y);
Z2=imcrop(Z2,[665,457,1018,1011]);
imshow(Z)%se muestra
```

Ilustración 49: Programación 5 foto Matlab

Foto 6:

Una vez el brazo se haya situado en la posición para la toma de la última foto se establecerá conexión con Matlab, dado que tras Matlab realizar la sexta foto se finalizará esta fase del procesamiento. Ver ilustración 50 y 51.

```
 Foto3;
! Waittime 3;
 !enviar mensaje al cliente
 SocketSend client1,\str:="F";
 !recivir mensaje
 SocketReceive client1,\RawData:=data;
 UnpackRawBytes data,1,fotos\Ascii:=2;
oki:=StrtoVal(fotos,numfoto);
WaitUntil numfoto=7;
waitTime 1;

 !Cerrar comunicacion
 SocketClose server1;
 SocketClose client1;
WaitSyncTask sync_RL_B, ListaTareas;
MoveJ Agarre_L_X,v30,z100,Servo_L\WObj:=WO_Cubo;
waittime 5;
g_GripOut;
waittime 1;


MoveJ Apro_Agarre_L_X,v30,z100,Servo_L\WObj:=WO_Cubo;
MoveJ Apro_L_1,v30,z100,Servo_L\WObj:=WO_Cubo;
waittime 1;
```

Ilustración 50: Programación 6 foto RobotStudio

```
foto=foto+2
num_foto=fread(tc);%se recibe la informacion de que el robot esta en la ultima posicion

Cara6 = getsnapshot(vid);%se saca la foto
stop(vid);

fwrite(tc,foto);
X=Cara6;
Y=imrotate(X,90);
Z=fliplr(Y);
imshow(Z)
fclose(tc);%se cierra la conexion
```

Ilustración 51: Programación 6 foto Matlab

Ubicaciones brazo izquierdo:

Las ubicaciones a las que corresponden cada posición de las fotos del brazo izquierdo se recogen en la siguiente programación y tras realizar varias pruebas se han modificado a la ubicación más optima y eficaz, ver ilustración 52.

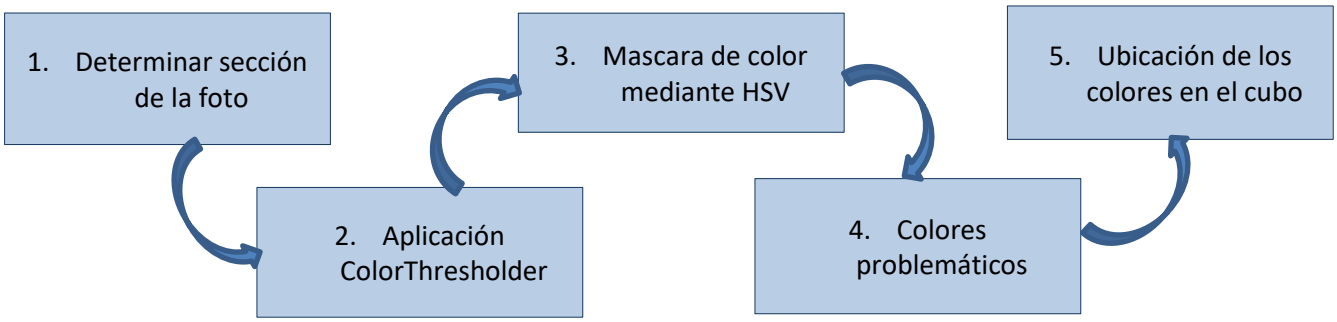
```
PROC Foto1()
    MoveJ Apro_Coger_CuboL,v200,z100,Servo_L\WObj:=WO_Cubo;
        MoveJ Tomar_Cubo,v200,z100,Servo_L\WObj:=WO_Cubo;
        waitTime 1;
        g_Gripin;
    GripLoad cubo3;
    Waittime 4;
        MoveJ Tomar_Fotos3,v50,z100,Servo_L\WObj:=WO_Cubo;
        waitTime 1;
      ENDPROC

      PROC Foto2()
        MoveL Tomar_Fotos2,v20,z100,Servo_L\WObj:=WO_Cubo;
        waitTime 1;
      ENDPROC
      PROC Foto3()
        MoveJ Tomar_Fotos,v20,z100,Servo_L\WObj:=WO_Cubo;
        waitTime 1;
      ENDPROC
```

Ilustración 52: Ubicaciones brazo izquierdo

PROCESAMIENTO DE IMAGENES

En consecuencia de haber movido la ubicación de la cámara se han tenido que realizar modificaciones en el procesamiento de imágenes, ya que la sección de la imagen a analizar varía y también la luz y los reflejos que esta recibe. Este apartado se estructura de la siguiente manera:



Por lo que, en primer lugar mediante la toolbox de Matlab “ImageAcquisition”, se ha determinado la sección de interés de cada foto para poder realizar su posterior recorte antes de realizar el procesamiento de imágenes, ver ilustración 53.

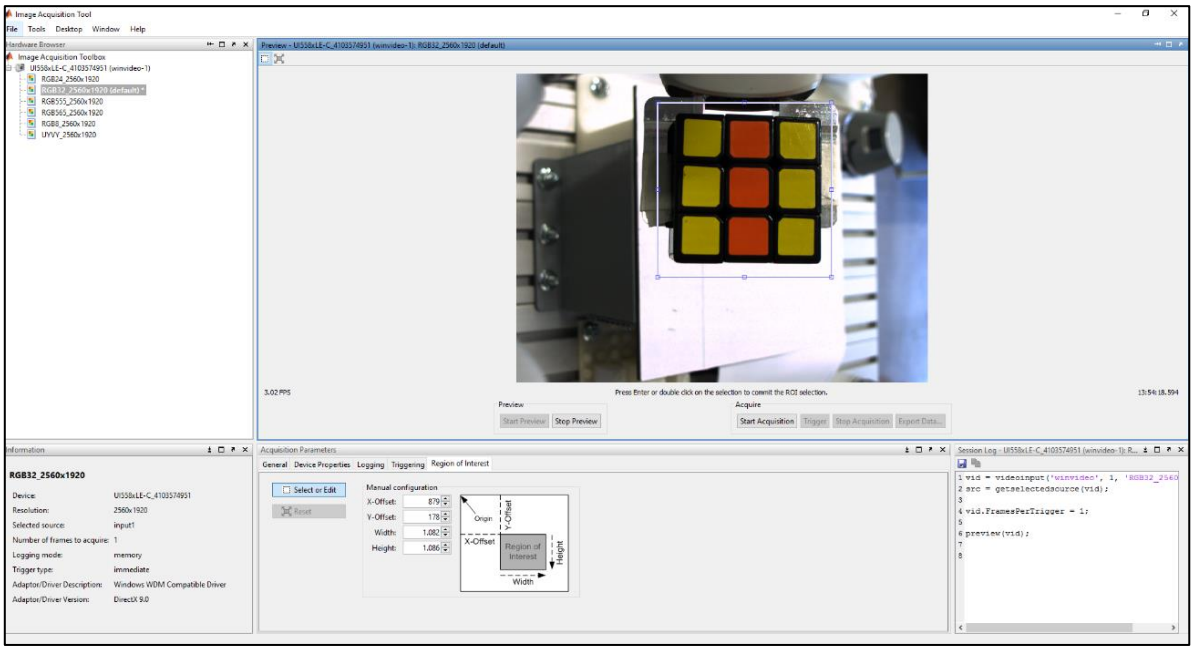


Ilustración 53: ImageAdquisition Toolbox

Tras delimitar la sección a analizar se ha pasado a realizar el procesamiento de color, para ello se ha utilizado la aplicación de "ColorThresholder". Mediante esta aplicación se puede generar una máscara que aísle diferentes colores, por lo que si se selecciona por ejemplo el rojo solo se mostrarán los elementos que aparezcan en este color en la imagen, dejando el resto en negro. Para la selección del rango de colores hay cuatro opciones: RGB, HSV, YCbCr y L*a*b, ver ilustración 54.

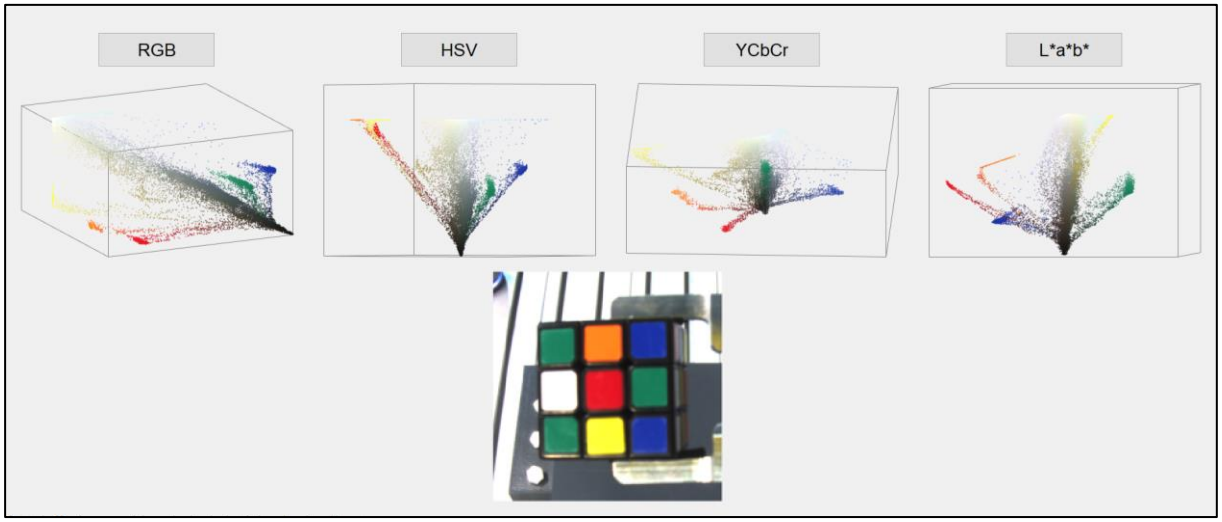


Ilustración 54: Espacios de color

En este caso se ha optado por utilizar la opción de HSV, dado que el procesamiento de colores se puede realizar de forma sencilla. Como se puede ver en la ilustración 55, el espacio de color se presenta como un cono invertido. Dependiendo de los valores que se le asignen a Hue, Saturation y Value (matiz, saturación y valor) se obtiene en este cono el color deseado.

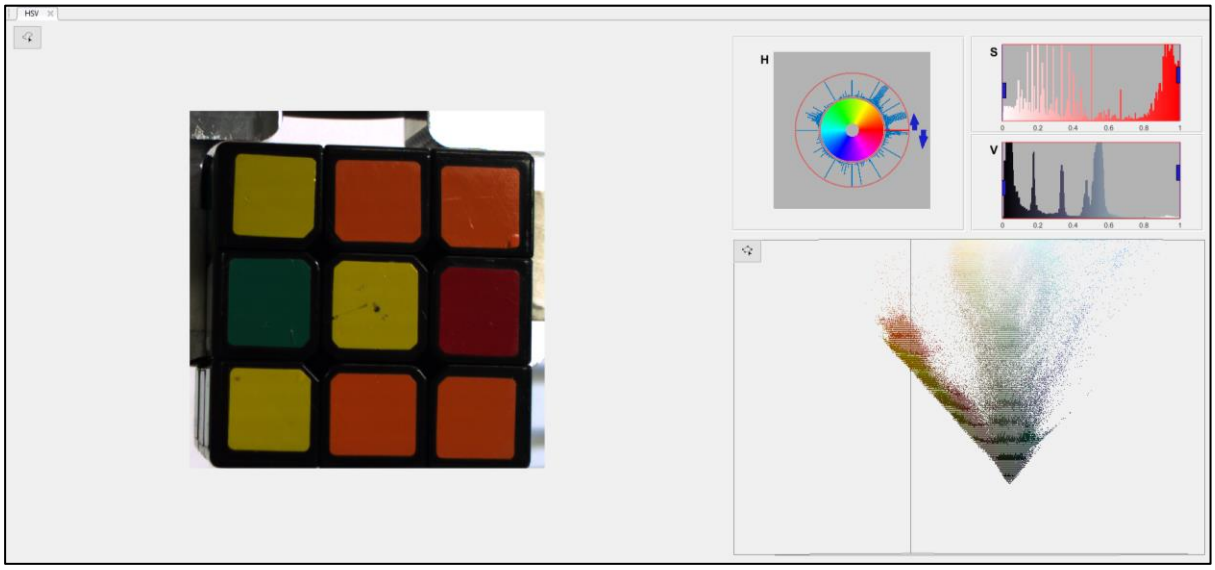


Ilustración 55: Mascara de color con HSV

A continuación se explica cada uno de los parámetros a modificar.

Hue: Se muestra como un alguno, cuyos diferentes valores representan distintos colores, estos valores van de 0 a 360 grados. Mediante esta herramienta se puede seleccionar un único grado o un rango de ellos (para aislar más colores).

Saturation: En este caso los valores varían entre 0 y 1, y representan la intensidad del color de la imagen. Siendo 0 el más oscuro y 1 el que más color tenga. Cuanto mayor sea el valor más intenso será el color, sin embargo, cuanto más se acerque a 0 más decolorado estará.

Value: Los valores varían entre 0 y 1, siendo 0 la parte de abajo del triángulo y 1 la parte de arriba. De esta manera cuanto más se acerquen los valores a 0 menos accesibles serán los colores de la parte de arriba del triángulo, que son los más claros.

En la ilustración 56 se puede ver la representación visual de los valores que pueden tomar cada uno de los parámetros.

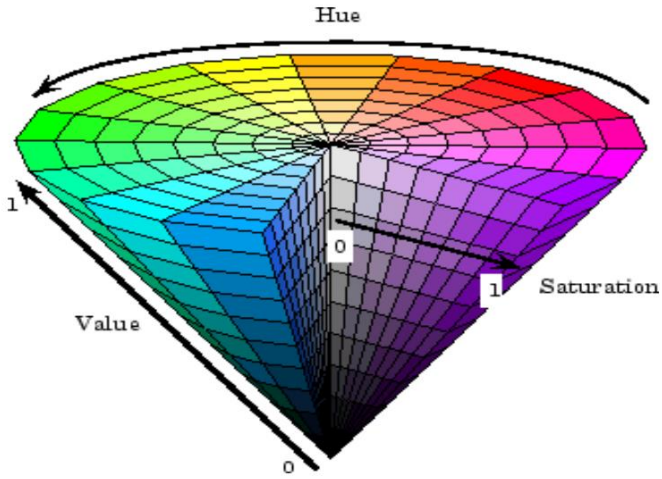


Ilustración 56: Cono del espacio de color HSV

Tras realizar varias pruebas, dos de los colores que más problemas presentan dependiendo del tipo de luz que haya en la sala son el rojo y el naranja, ya que se parecen bastante entre sí. Por lo que se recomienda prestar especial atención a estos. En la ilustración 57 se muestra un ejemplo de los parámetros a editar para lograr una máscara que aísle el color naranja.

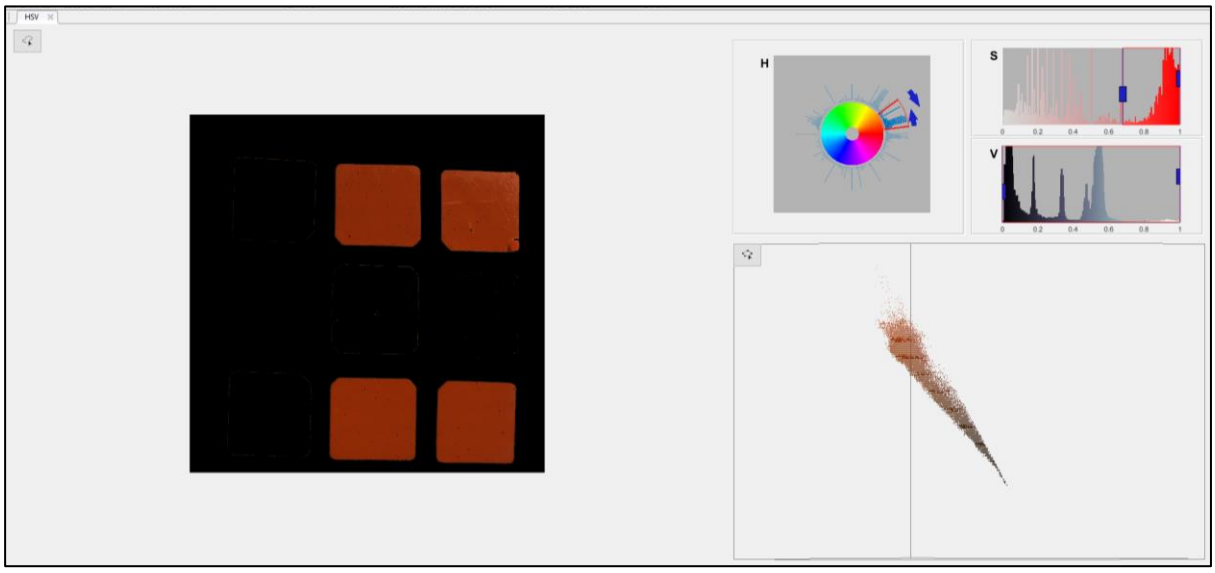


Ilustración 57: Mascara de color naranja con HSV

Por último, se han modificado los script para ubicar cada uno de los colores de las caras en su posición correspondiente dentro de la matriz, para esto se han tomado coordenadas (X,Y) de las filas y columnas del cubo en las imágenes. En la ilustración 58 se puede ver un ejemplo para el color rojo, habiendo un script diferente para cada uno de los colores.

```
if posItemRedX~=0

for i=1:length(posItemRedX)
        %CARA F
if  posItemRedX(i) <= 300;
    if  posItemRedY(i) < 375;
    Cara(1,1)=1;
    end
    if posItemRedY(i) > 375 & posItemRedY(i) < 625;
    Cara(2,1)=1;
    end
    if posItemRedY(i) > 625;
    Cara(3,1)=1;
    end

end

if posItemRedX(i) > 300 & posItemRedX(i) < 600;
    if  posItemRedY(i) < 375;
    Cara(1,2)=1;
    end
    if posItemRedY(i) > 375 & posItemRedY(i) < 625;
    Cara(2,2)=1;
    end
    if posItemRedY(i) > 625;
    Cara(3,2)=1;
    end

end

if posItemRedX(i) >=600;
    if  posItemRedY(i) < 375;
    Cara(1,3)=1;
    end
    if posItemRedY(i) > 375 & posItemRedY(i) < 625;
    Cara(2,3)=1;
    end
    if posItemRedY(i) > 625;
    Cara(3,3)=1;
    end

end
end
end
```

Ilustración 58: Pos_Rojo_Cara

6.3.5 Programación movimientos del robot

Con objetivo de mejorar los tiempos de resolución del cubo de Rubik, se ha querido implementar otro algoritmo de resolución capaz de resolver el cubo en menos movimientos que el de Joren Heit. Sin embargo, tras realizar un minucioso estudio sobre los diferentes algoritmos no se ha podido encontrar uno que mejore la aportación del algoritmo actual a la aplicación.

Destacar que ahora mismo se utiliza el algoritmo de resolución Solve45 que se puede encontrar en Matlab. Este algoritmo asegura una resolución del cubo en menos de 45 movimientos, siendo la media de 31. El algoritmo de Joren Heit está diseñado para ser utilizado por un ordenador. Se basa en unas tablas de poda generadas automáticamente y formadas por más de un millón de entradas que le permiten elegir la mejor solución en función de la forma en la que esté mezclado el cubo.

A pesar de no haberse logrado optimizar la cantidad de movimientos a realizar por el robot para la resolución del cubo, sigue habiendo varios factores por los que es de vital importancia modificar varios de los movimientos del robot, además de para mejorar los tiempos de la aplicación.

En primer lugar, al modificarse la ubicación de la cámara en la célula es necesario modificar los movimientos del robot a la hora de realizar la toma de fotos.

En segundo lugar, en ocasiones el robot presentaba problemas al realizar los giros de las caras del cubo, ya que no lo cogía de forma muy precisa. Debido a esto se ha decidido que el robot tome el cubo 5mm más abajo, para lograr así una mayor sujeción.

En tercer lugar, destacar que la aplicación daba problemas cuando para resolver el cubo era necesario realizar dos giros iguales seguidos, por lo que se han creado en Matlab nuevos scrips sencillos que eliminen este problema y posibiliten la realización de estos dobles giros.

6.3.6 Programación módulo de sonido

El principal objetivo de este apartado es dotarle de voz al robot, que reproducirá diferentes frases a medida que esté ejecutando las diferentes tareas, de manera que el usuario pueda interactuar con él y conocer en que etapa del proceso se encuentra. Para ello, se barajaron dos propuestas:

- Utilizar un lector de textos: Esta propuesta utilizaría un lector de textos para reproducir las frases que previamente el usuario elija.
- Utilizar audios pregrabados: En esta propuesta se reproducen audios que previamente han sido grabados.

Pese a que en un primer instante se planteó la primera propuesta, finalmente se ha optado por implementar la segunda opción debido a que se quiere que el robot tenga la capacidad de hablar en 3 idiomas (Castellano, Inglés y Euskera).

FUNCIONAMIENTO

Ya que se ha optado por la opción de utilizar audios pregrabados, el primer paso para dotar de voz al robot es grabar dichos audios. Estos audios pueden ser grabados mediante distintos programas software disponibles como son Audacity, Recording Studio, etc. Sin embargo, ya que el algoritmo de resolución del cubo de rubik utiliza Matlab, se opta por grabar los audios con el propio Matlab, evitando así la instalación de programas externos en la torre que controla al robot.

La grabación de los audios se hace mediante la función *audiorecorder.* Esta función permite grabar el audio recibido de un micrófono para que sea procesada en Matlab. El código utilizado para ello es el que se encuentra en la ilustración 59. En ella, se puede ver como cuando se ejecute el scrip, el usuario debe presionar la tecla “s” para comenzar a grabar el audio. Para ello, se crea la variable global *fs,* que

corresponde con la frecuencia de muestreo con la que se grabará el audio. Destacar, que Matlab tiene establecido ciertos límites para la frecuencia de muestreo, teniendo que estar esta comprendida entre los 1000 Hz y los 384000 Hz. Para la aplicación a desarrollar se optar por utilizar una frecuencia de 44100 Hz, dado que una inferior puede presentar problemas de sonido y tampoco es necesario utilizar una muy superior ya que requeriría de mayor espacio.

```
aviso = input('Presione s para empezar','s')%mostrar en pantalla
global fs %creamos una variable global de la frecuencia de muestreo
fs=44100;%establecemos frecuencia de muestreo (si ponemos una demasiado alta ocupa demasiado,
                                               %si ponemos una frecuancia muy pequeña se escucha mal

if aviso == 's' %para empezar a grabar pulsamos S
    recObj = audiorecorder(fs,16,2); %funcion que graba el audio a la frecuencia correspondiente
    disp('Iniciando grabacion 1') %mostrar en pantalla
    disp('Grabe su voz durante 4 seg') %mostrar en pantalla
    recordblocking(recObj,4) %finaliza la grabacion al de 4 segundos
    disp('Fin de grabacion')%muestra pantalla

end
```

Ilustración 59: Grabar audio

Una vez el usuario pulsa la tecla "s", se llama a la función *audiorecorder* la cual necesita las siguientes tres entradas: frecuencia de muestreo (fs), numero de bits por muestra y numero de canales. El número de bits por muestra se establece en 16 y el número de canales se establece en 2 ya que se instalan un par de altavoces, en caso de instalar un solo altavoz esto no tendría sentido, ya que solo se requeriría de un canal. La función, guarda los datos en una variable asignada que este caso es llamada *recObj.*

Tras ejecutar esta parte de código, se muestran en pantalla los textos de "Iniciando Grabación 1" junto con "Grabe su voz durante 4 seg". El tiempo de grabación es establecido por el usuario con la función *recordblocking* cuyas entradas son la variable donde se guardan los datos y el tiempo de duración de la grabación en segundos. Una vez termina de grabarse el audio, aparece el mensaje de "Fin de la grabación" para que el usuario sepa que ha terminado la grabación.

Para convertir el audio en un vector numérico y así guardarlo en un fichero .mat que facilite poder utilizarlo en cualquier scrip, se utiliza la función *getaudiodata.* Con esta función el audio se guarda en un vector de 176400x2. El vector es doble ya que la opción seleccionada es la de dos canales, en caso de haber seleccionado un solo canal se crearía un vector de 176400x1. Una vez convertido a vector, es guardado en un fichero .mat para su posterior utilización.

Para esta aplicación se han grabado un total de 6 frases (se podrían grabar todas las deseadas), que utilizar el robot para hacer saber en cada momento sus intenciones y/o en qué momento de la aplicación se encuentra. Los sonidos grabados son los siguientes:

- *¿Quién se atreve a retarme con el cubo de Rubik? / ¿Who dares to challenge me with the Rubik's cube? / Nor ausartzen da Rubik kuboa egiten?*

Esta primera frase se reproduce al inicio de la aplicación, para animar al usuario a depositar un cubo de Rubik (tanto de 2x2 como de 3x3) en la mesa de recepción y que el robot inicie el programa.

- *Necesito conocer las 6 caras. / I need to know the six faces. / Aurpegi guztiak esagutu behar ditut.*

  Esta frase es la segunda que se escucha, ya que se reproduce tras detectar un cubo en la mesa y antes de que el robot vaya a por el para sacar las fotos de las 6 caras del cubo.

- *Voy a hacer el cubo de 2x2. / I'm going to do the 2 by 2 cube. / 2x2-ko kuboa egingo dut.*

  Tras realizar la toma de las fotos y procesarlas, el robot sabe qué tipo de cubo tiene entre manos, de ser el de 2x2 se escuchará esta frase.

- *Voy a hacer el cubo de 3x3. / I'm going to do the 3 by 3 cube. / 3x3-ko kuboa egingo dut.*

  Si en vez de ser de 2x2 el cubo que posee el robot es el de 3x3, la frase que se reproducirá será esta.

- *¡Veamos lo rápido que puedo resolverlo! / ¡Let's see how fast I can solve this! / Goaz ikustera zein askar egiten dudan!*

  Una vez detectado el tipo de cubo y obtenidos los movimientos a realizar por el robot para resolver el cubo, el robot dirá esta frase y se dispondrá a resolverlo.

- *¡He acabado! ¡Ha salido genial! / ¡I'm finished! ¡It turned out great! / Bukatu dut! Oso ondo atera da!*

  Cuando el robot haya terminado de resolver el cubo lo celebrará y dirá esta frase final antes de volver a iniciarse el proceso.

Para reproducir los audios, se debe de cargar los ficheros .mat en los que se encuentran junto con la frecuencia de muestreo con la que se han grabado. Una vez cargados los archivos, para reproducirlos, se insertan en los tramos del algoritmo correspondiente, que se han detallado anteriormente, mediante la función *sound*. Esta función necesita dos entradas, por un lado, el vector en el que sea guardado el audio y por otro la frecuencia de muestreo, ver ilustración 60.

En la siguiente ilustración se puede ver un ejemplo del código a insertar en el programa principal, como se ha dicho, este se insertará en los tramos deseados, y dependiendo del idioma en el que se quiera ejecutar la aplicación se cargarán los diferentes sonidos.

```
%sonido
load('fs.mat');
load('Sonido1.mat');
%load('Sonido7.mat'); %INGLES
%load('Sonido13.mat');%EUSKERA
sound(audio1,fs);
```

Ilustración 60: Reproducir audios

6.4 VALIDACIÓN

Una vez configurado el escenario con la mejor ubicación de los componentes, y realizado el código de programación en el entorno Matlab y RobotStudio, se valida la solución.

Dado que uno de los objetivos principales era retar al robot para lograr que la resolución del cubo sea lo más rápida posible, se han realizado varias pruebas con los cubos de 2x2 y 3x3. De esta manera se han obtenido diferentes resultados de tiempos y cantidad de movimientos que se necesitan para resolver los distintos cubos.

En cuanto al módulo de sonido, se ha comprobado que la aplicación se ejecuta correctamente con los 3 idiomas y con todos los audios a reproducir. Además, también se ha verificado el correcto funcionamiento de la pantalla táctil y la minitorre.

Sobre los movimientos del robot se ha probado que son óptimos para la realización del proyecto y que el robot coge y ubica correctamente el cubo.

7. DESCRIPCIÓN DEL PROCEDIMIENTO

A continuación se va a presentar la planificación para llevar a cabo el presente proyecto, para ello se deben completar las siguientes tareas:

TAREAS GANTT:

1. Estudio y búsqueda de documentación: Se realizará un estudio minucioso sobre la documentación de la aplicación anterior y se buscará información sobre el tema.
2. Estudio de la herramienta de AutoCAD: Se trabajará con la herramienta de AutoCAD para conseguir un pleno dominio de esta, con objetivo de realizar el diseño de los componentes 3D.
3. Diseño e impresión de piezas 3D: Una vez se logre el control de la herramienta AutoCAD, se pasará a realizar el diseño de los componentes deseados, para ello se tomaran medidas de la cámara y de la célula del robot. Tras realizar los diseños y verificar que las medidas son las correctas se pasará a imprimir dichos componentes.
    a. Carcasa y tapa
    b. Sujeción
    c. Mesa
4. Estudio de las herramientas Matlab y RobotStudio: Dado que gran parte de la programación se realiza haciendo uso de la herramienta Matlab, es necesario tener un pleno dominio sobre esta. Teniendo en cuenta los conocimientos previos de la alumna este aprendizaje no requerirá un excesivo tiempo. Además, para controlar el robot es necesario utilizar la herramienta RobotStudio, en la que se programa mediante el lenguaje Rapid, por lo que se dedicara un tiempo al estudio de esta.
5. Búsqueda de nuevos algoritmos de resolución del cubo de Rubik: Con objetivo de optimizar los tiempos de resolución del cubo de Rubik se propone la tarea de la búsqueda de un algoritmo más rápido.
6. Búsqueda de nuevos componentes: Se realizará una búsqueda y análisis de diferentes minitorres y pantallas táctiles, con objetivo de elegir las más económicas y que mejor se ajusten a la aplicación.
7. Montaje de piezas 3D: Tras obtener las piezas 3D se montaran y ajustaran en la célula del robot.
8. Montaje y configuración de nuevos componentes: Se configurarán los nuevos componentes para adecuarlos a las necesidades del proyecto y tras ellos se instalarán en la célula.
9. Configuración de la cámara: Se configurará la cámara para una correcta toma de imágenes.

10. Programación de toma del cubo de la mesa: Haciendo uso de RobotStudio y Matlab se realizará la programación necesaria para que el robot coja el cubo de Rubik de la mesa con las nuevas modificaciones.
11. Optimización de los movimientos del robot: Se modificaran los movimientos para la toma de imágenes del cubo, de manera que resulten más eficaces.
12. Optimización de la toma, envío y procesamiento de imágenes: Se modificará la toma y envío de imágenes entre RobotStudio y Matlab para realizarlo en el menor tiempo posible. Además, también se modificará el procesamiento de imágenes para que sea más preciso y eficaz.
13. Programación módulo de sonido: Mediante la herramienta Matlab se programará un script que sea capaz de grabar notas de voz y procesarlas. Tras ello se realizará la programación necesaria a lo largo del código, en los puntos en los que se quiera que el robot hable.
14. Realización del artículo y poster: Se trabajará en el desarrollo del artículo y poster para presentar la aplicación a las jornadas de automática de 2021.
15. Pruebas del sistema final: Se realizarán varias pruebas finales para comprobar que la aplicación trabaja como es debido y analizar los tiempos de resolución del cubo y compararlos con los anteriores.

FASES GANTT:

Para la realización del diagrama de GANTT y por ende la planificación del proyecto, se han marcado varias fases que estarán compuestas por las tareas anteriormente detalladas.

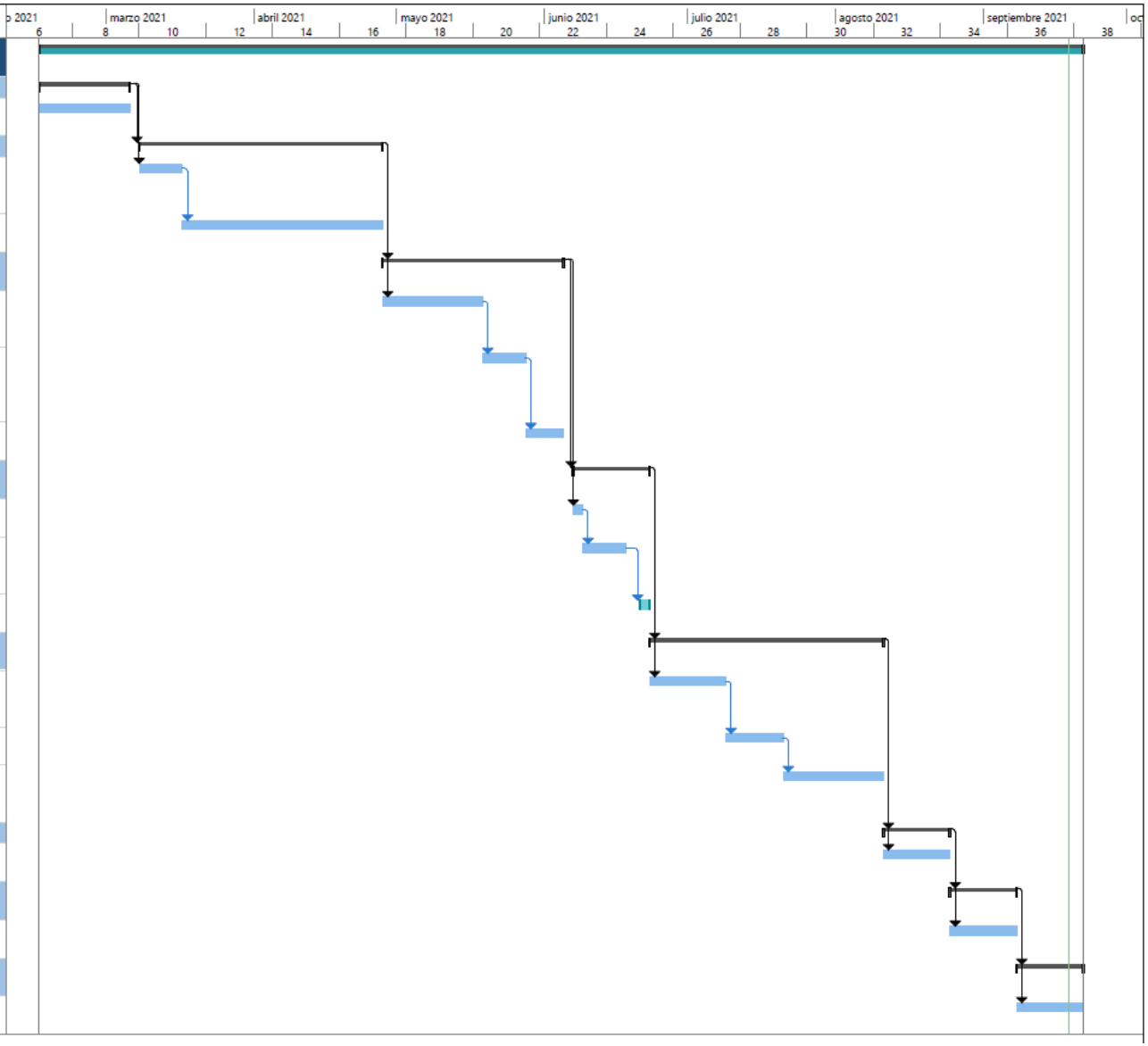
Las fases son las siguientes:

Tabla 4: FASES GANTT

FASE	NUMERO DE TAREA
1. INICIO	1
2. COMPONENTES 3D	2,3
3. ESTUDIO Y BUSQUEDA	4,5,6
4. CONFIGURACIÓN DE COMPONENTES Y MONTAJE	7,8,9
5. PROGRAMACIONES Y OPTIMIZACIONES	10,11,12
6. MÓDULO DE SONIDO	13
7. JORNADAS DE AUTOMÁTICA	14
8. PRUEBAS Y DOCUMENTACIÓN FINAL	15

A continuación se presenta el diagrama de GANTT con todas las tareas mencionadas dentro de cada una de sus fases.

Id	Nombre de tarea	Duración	Comienzo	Fin
1	**Desarrollo TFM -Garazi Fernandez**	**157 días**	**lun 15/02/21**	**mar 21/09/21**
2	**FASE 1: Inicio**	**15 días**	**lun 15/02/21**	**vie 05/03/21**
3	Estudio y busqueda de la documentación	15 días	lun 15/02/21	vie 05/03/21
4	**FASE 2: Componentes 3D**	**37 días**	**lun 08/03/21**	**mar 27/04/21**
5	Estudio de la herramienta de AutoCAD	7 días	lun 08/03/21	mar 16/03/21
6	Diseño e impresión de piezas 3D	30 días	mié 17/03/21	mar 27/04/21
7	**FASE 3: Estudio y busqueda**	**28 días**	**mié 28/04/21**	**vie 04/06/21**
8	Estudio de la herramienta Matlab y RobotStudio	15 días	mié 28/04/21	mar 18/05/21
9	Busqueda de nuevos algoritmos de resolucion del cubo de rubik	7 días	mié 19/05/21	jue 27/05/21
10	Busqueda de nuevos componentes	6 días	vie 28/05/21	vie 04/06/21
11	**FASE 4: Configuración de componentes y montaje**	**12 días**	**lun 07/06/21**	**mar 22/06/21**
12	Montaje de las piezas 3D	2 días	lun 07/06/21	mar 08/06/21
13	Configuración y montaje de nuevos componentes	7 días	mié 09/06/21	jue 17/06/21
14	Configuración de la cámara	2 días	lun 21/06/21	mar 22/06/21
15	**FASE 5: Programaciones y optimizaciones**	**35 días**	**mié 23/06/21**	**mar 10/08/21**
16	Programación de la toma del cubo de la mesa	12 días	mié 23/06/21	jue 08/07/21
17	Optimización movimientos del robot	8 días	vie 09/07/21	mar 20/07/21
18	Optimización toma, envio y procesamiento de imágenes	15 días	mié 21/07/21	mar 10/08/21
19	**FASE 6: Módulo de sonido**	**10 días**	**mié 11/08/21**	**mar 24/08/21**
20	Programación módulo de sonido	10 días	mié 11/08/21	mar 24/08/21
21	**FASE 7: Jornadas de automática**	**10 días**	**mié 25/08/21**	**mar 07/09/21**
22	Realización del artículo y poster	10 días	mié 25/08/21	mar 07/09/21
23	**FASE 8: Pruebas y documentación final**	**10 días**	**mié 08/09/21**	**mar 21/09/21**
24	Pruebas del sistema final	10 días	mié 08/09/21	mar 21/09/21



8. ANÁLISIS DE RIESGOS

En el presente apartado se va a pasar a analizar los posibles riesgos que pueden afectar de manera negativa o ralentizar el desarrollo del trabajo.

En primer lugar se identificarán dichos riegos, en segundo lugar se evaluará las posibilidades que tienen de suceder y el impacto en consecuencia, en tercer lugar se realizará un plan de prevención para cada uno de ellos y en cuarto y último lugar se establecerán las prioridades.

A continuación se realiza la evaluación para los riesgos identificados: ruptura de equipos, retraso de objetivos, error en los diseños de componentes 3D y bajas de los trabajadores.

8.1 Ruptura de equipos (A)

Este riesgo se da cuando alguno de los equipos necesarios para el desarrollo del proyecto no funciona debidamente o en su caso extremo no funciona. Para la realización de este proyecto hay varios equipos que podrían ser dañados, entre ellos: la cámara, el robot YuMi, la pantalla táctil, la minitorre, los altavoces, la carcasa y la sujeción.

Dependiendo del tipo de elemento la posibilidad de ruptura varia, así como el impacto que genera en el normal desarrollo del proyecto.

Para evitar esta situación hay que tomar una serie de precauciones a la hora de utilizar las máquinas. Antes de empezar a hacer uso de cualquier dispositivo hay que aprender bien cómo funciona, y solo una vez entendido esto se pasará a utilizarlo en la práctica. Además, cada vez que se termine de utilizar alguno de los equipos se deberá apagar.

8.1.1 Cámara (A.1)

Que la cámara se pueda dañar se considera posible, puesto que esta no consta de una carcasa propia (de primeras se visualiza la placa base y objetivo sin ninguna protección), además es uno de los elementos que más se manipulan a lo largo del proyecto.

El impacto que supondría la ruptura de este componente es muy elevado, puesto que es indispensable para el desarrollo del proyecto y su sustitución requeriría rehacer partes de la programación, una nueva carcasa, etc. Por el contrario, si se diese el caso de que el elemento tuviera arreglo (en vez de sustituirlo por otro) también ralentizaría todo el trabajo.

8.1.2 Robot YUMI (A.2)

Es muy raro que el robot YUMI se dañe, puesto que consta de una estructura bastante robusta y ha sido desarrollado por ABB el cual es un fabricante muy fiable.

En caso de darse algún problema con el robot el impacto sería catastrófico, puesto que sin él no hay ninguna posibilidad de desarrollar el proyecto.

8.1.3 Pantalla táctil (A.3)

Hay posibilidades de que se estropee la pantalla táctil o de que sufra algún problema, sin embargo estas posibilidades no son elevadas.

En este caso el impacto ocasionado sería menor, ya que se podría realizar su sustitución de manera sencilla.

8.1.4 Minitorre (A.4)

La ruptura de la minitorre es poco probable, puesto que elementos como este no suelen presentar problemas.

En caso de darse esta situación si acarrearía un impacto en el desarrollo del proyecto puesto que habría que volver a configurar una minitorre, sin embargo el impacto sería moderado.

8.1.5 Altavoces (A.5)

La posibilidad de que los altavoces implantados en el sistema se rompan es muy elevada, puesto que al no ser unos altavoces de gama alta si se les diera un golpe podrían presentar problemas.

El impacto sería casi menor, ya que es un elemento de fácil sustitución y/o arreglo.

8.1.6 Carcasa y sujeción (A.6)

Tanto la carcasa de la cámara la sujeción están hechas con un material resistente pero no irrompible, por lo que en caso de caída brusca o fuerte choque puede darse la situación de ruptura.

El impacto de esta situación es bastante elevado, puesto que requeriría la elaboración de nuevos componentes y para ello tiempo y coste de la impresora 3D.

8.2 Retraso de objetivos (B)

Este riesgo se da cuando no se cumple con los tiempos planificados. Hay muchas probabilidades de que suceda este problema, sin embargo, el impacto no es muy grande ya que puede solucionarse invirtiendo más horas otro día.

Para prevenir este riesgo se va a hacer una planificación con un amplio margen de cambios, es decir a cada tarea se le atribuirán más horas de las necesarias para en caso de darse algún problema, tener tiempo suficiente para arreglarlo.

8.3 Error diseño de componentes 3D (C)

Este riesgo es casi seguro que suceda, ya que es muy complicado realizar los diseños de los componentes bien a la primera.

En cuanto al impacto se considera menor, puesto que se habilitara un tiempo extra para el diseño de componentes para que de darse el problema no afecte al desarrollo del proyecto.

8.4 Bajas de los trabajadores (D)

Debido a la situación actual (Covid-19), es muy probable que algunos de los participantes en el proyecto tenga que coger la baja.

Este riesgo acarrearía un retraso importante en el desarrollo del trabajo, es por ello que se debe prever y organizar los tiempos de manera que pueda solventarse y no afecte en exceso.

8.5 Resumen del análisis de riesgos

Los riegos anteriores se han clasificado en una matriz de probabilidad y consecuencias, para dependiendo de la gravedad poder identificar los riesgos prioritarios, ver tabla 5.

Tabla 5: Matriz de probabilidad y consecuencias

		Probabilidad				
		Raro (0,05)	Poco Probable (0,1)	Posible (0,2)	Muy probable (0,4)	Casi seguro (0,8)
Consecuencias	Despreciable (0,1)					
	Menores (0,3)		**A.4** (0,03)	**A.3** (0,06)	**A.5** (0,12)	**B, C** (0,24)
	Moderados (0,5)				**D** (0,2)	
	Mayores (0,7)			**A.6** (0,14)		
	Catastróficas (0,9)	**A.2** (0,045)		**A.1** (0,18)		

8.6 Riesgos prioritarios

Utilizando la tabla anterior se van a ordenar los distintos riesgos explicados previamente de mayor a menor gravedad:

1) B y C: Retraso de objetivos y error de diseño de componentes 3D
2) D: Bajas de los trabajadores
3) A.1: Ruptura de equipos - Cámara
4) A.6: Ruptura de equipos – Carcasa y sujeción
5) A.5: Ruptura de equipos - Altavoces
6) A.3: Ruptura de equipos – Pantalla táctil
7) A.2: Ruptura de equipos – Robot YUMI
8) A.4: Ruptura de equipos - Minitorre

9. PRESUPUESTO

En este apartado se realizará un análisis de los costes que ha supuesto la realización del proyecto. Los costes se dividen mayormente en tres partes: Materiales, maquinaria y mano de obra. Además, también se realizará un análisis de las amortizaciones con objetivo de ver el coste que dichos elementos le han supuesto al proyecto.

Cabe destacar que para la realización de los costes se ha utilizado la herramienta Excel, con objetivo de hacer el trabajo más óptimo y preciso.

9.1 Materiales

En la siguiente tabla se muestran los materiales utilizados a lo largo del proyecto, con sus respectivos precios y amortizaciones.

Siendo la fórmula utilizada para el cálculo de la amortización la siguiente:

$$Amortización\ (€) = \frac{Precio\ del\ material\ (€)}{Vida\ util\ (años) * 12\ (\frac{meses}{años})} * Tiempo\ de\ uso\ (meses)$$

Tabla 6: Costes de Materiales

MATERIALES	PRECIO (€)	VIDA ÚTIL (AÑOS)	TIEMPO DE USO (MESES)	AMORTIZACIÓN (€)
ROBOT YUMI	45.000,00 €	10	7	2.625,00 €
CAMARA IDS	1.581,57 €	8	7	115,32 €
OBJETIVO	200,00 €	8	7	14,58 €
CARCASA 3D	100,00 €	6	5	6,94 €
SUJECION 3D	100,00 €	6	5	6,94 €
MESA 3D	100,00 €	6	5	6,94 €
MINITORRE	170,00 €	5	3	8,50 €
PANTALLA TACTIL	110,00 €	5	3	5,50 €
ALTAVOCES	30,00 €	5	4	2,00 €
MESA DE TRABAJO	1.500,00 €	10	7	87,50 €
CUADRO ELECTRICO	172,00 €	5	7	20,07 €
PANEL DE CONTROL	154,00 €	5	7	17,97 €
MS OFFICE	118,00 €	1	7	68,83 €
MATLAB	1.985,00 €	3	7	385,97 €
AUTOCAD	3.194,40 €	1	2	532,40 €

Siendo el total de las amortizaciones de los materiales el siguiente:

Tabla 7: Costes total de Materiales

TOTAL AMORTIZACIONES DE MATERIALES	3.904,48 €

9.2 Maquinaria

Para la ejecución del proyecto también se ha hecho uso de diferentes maquinarias, siendo las principales las que se exponen en la siguiente tabla. Como en el caso anterior, esta tabla muestra los precios y amortizaciones de cada elemento.

Siendo la fórmula utilizada para el cálculo de la amortización la siguiente:

$$Amortización\ (€) = \frac{Precio\ del\ material\ (€)}{Vida\ util\ (años) * 12\ (\frac{meses}{años})} * Tiempo\ de\ uso\ (meses)$$

Tabla 8: Costes de Maquinaria

MAQUINARIA	PRECIO (€)	VIDA UTIL (AÑOS)	TIEMPO DE USO (MESES)	AMORTIZACION (€)
ORDENADOR DE SOBRE MESA	980,00 €	5	7	114,33 €
ORDENADOR PORTATIL	850,00 €	5	7	99,17 €
SET DE HERRAMIENTAS	60,00 €	5	5	5,00 €

Realizando el sumatorio de las amortizaciones se obtiene el siguiente valor, que equivale al valor de amortización total de la maquinaria.

Tabla 9: Coste total de Maquinaria

TOTAL AMORTIZACIONES DE MAQUINARIA	218,50 €

9.3 Mano de obra

Por último, uno de los apartados más relevantes en cuando al presupuesto del proyecto consiste en la mano de obra. Para el desarrollo de este han tomado parte cuatro perfiles diferentes: ingeniero junior, técnico de impresión, directora del proyecto y técnico eléctrico.

En la siguiente tabla se muestra el coste por hora de cada perfil, las horas dedicas al proyecto y el coste final de cada uno de ellos.

Tabla 10: Costes de Mano de obra

MANO DE OBRA	COSTE UNITARIO (€/h)	HORAS (h)	COSTE (€)
DIRECTORA DEL PROYECTO	60	45	2.700,00 €
INGENIERO JUNIOR	28	600	16.800,00 €
TECNICO ELECTRICO	38	12	456,00 €
TECNICO DE IMPRESIÓN	38	50	1.900,00 €

Teniendo en cuenta el coste de los diferentes perfiles el coste total es el siguiente:

Tabla 11: Coste total de Mano de obra

COSTE TOTAL DE MANO DE OBRA	21.856,00 €

10. ANÁLISIS DE RESULTADOS

Como se ha comentado en el apartado de validación, finalmente, se han realizado las pruebas de resolución del cubo de 2x2 y 3x3, ejecutando un total de 10 pruebas con cada uno de ellos. En la tabla 12 se resume el número de movimientos realizados (Mov) para resolver cada una de las 10 configuraciones de partida con cada cubo y el tiempo que tarda el robot en conseguirlo (T).

Tabla 12: Ensayos con cubos de 2x2 y 3x3

Cubo de Rubik 2x2										
Mov	10	12	13	15	11	19	14	12	16	9
T (min)	2,3	3,16	3,4	3,5	2,5	4,4	3,2	3,16	4,1	2
Cubo de Rubik 3x3										
Mov	29	34	35	30	28	33	29	27	32	34
T (min)	7	8,1	8,4	7,2	6,7	8	7	6,5	7,7	8,1

Se puede concluir que el tiempo medio que necesita el robot para resolver el cubo de 2x2 es de 3,17 minutos y la media del número de movimientos es de 13,1. En el caso del cubo 3x3, el tiempo medio es de 7,47 minutos y la media de movimientos es de 31,1.

Como era de esperar, el robot tarda mucho menos tiempo y necesita menos movimientos para resolver el cubo de 2x2, puesto que este posee un menor número de combinaciones posibles.

Otro de los resultados a destacar es que se ha cumplido el objetivo de publicar un paper y un poster en las jornadas de automática, lo cual da la posibilidad de participar activamente en ellas con la aplicación descrita. El poster y el paper realizados a lo largo del proyecto se pueden encontrar en el anexo I y anexo II.

11. CONCLUSIONES

En este trabajo se ha diseñado y construido una célula robótica de entretenimiento que permite resolver cubos de Rubik de 2x2 y 3x3 de forma automática.

El sistema propuesto está compuesto por el robot colaborativo YuMi de dos brazos, una cámara de visión artificial, un ordenador de procesamiento con Matlab y elementos auxiliares como la mesa, el soporte de la cámara o la carcasa que están impresos con una impresora 3D.

Adicionalmente, con la intención de atraer a los usuarios, se le ha añadido una pantalla táctil y un sistema de sonido que aumenta la interacción entre el robot y el usuario.

Después de realizar varios ensayos experimentales, se puede concluir que el sistema propuesto es capaz de resolver el cubo de Rubik de 2x2 en un tiempo medio de 3,17 y el de 3x3 en 7,47 minutos.

Los resultados de este trabajo se han difundido en las Jornadas de Automática a través de un paper y un poster.

REFERENCIAS

[1] Schwab, K. (2016). The fourth industrial revolution. Ed. Crown Business.

[2] T. (2021, 24 febrero). *Estos son los 12 robots que no te dejarán solo*. Robotoide. Enlace. Acceso 08/07/2021

[3] Julita, J. R., & Viejo, V. A. (2015). Cubo de Rubik. XVIII Concurso de Trabajos Estudiantiles (EST 2015)-JAIIO 44.

[4] Arteaga, S. (2017, 7 marzo). *Este robot resuelve el cubo de Rubik en poco más de medio segundo. ComputerHoy.* Enlace. Acceso 08/07/2021

[5] del Val Román, J. L. (2016, March). Industria 4.0: la transformación digital de la industria. In *Valencia: Conferencia de Directores y Decanos de Ingeniería Informática, Informes CODDII*.

[6] Basco, A. I., Beliz, G., Coatz, D., & Garnero, P. (2018). *Industria 4.0: fabricando el futuro* (Vol. 647). Inter-American Development Bank.

[7] Moctezuma Gutiérrez, S. G., Cruz Pazarán, A., Galicia Mejía, R., & Oliva Moreno, L. N. (2018). Desarrollo de plataforma para implementación de robots colaborativos. Visión electrónica, 12(1), 22–31. Enlace. Acceso 09/07/2021

[8] G. Du, M. Chen, C. Liu, B. Zhang, and P. Zhang, "Online Robot Teaching With Natural Human–Robot Interaction", IEEE Transactions on Industrial Electronics, vol. 65, no. 12, pp. 9571–9581, Dec. 2018.

[9] M. (2021a, febrero 2). *Evolución y tendencias de los robots colaborativos*. Camp Tecnologico. https://camptecnologico.com/evolucion-tendencias-robots-colaborativos/

[10] Timetoast. (2021). *Historia de la Robótica timeline*. Timetoast Timelines. Enlace. Acceso 15/07/2021

[11] Pape, J. P. (2015). Los robots colaborativos: una nueva era en la automatización industrial. *Tecnoalimen: tecnología alimentaria y packaging*, (12), 50-51.

[12] Tabuenca Alcusón, D. (2017). Implantación de robots colaborativos en linea de producción.

[13] Riveros López, J. A. (2016). Diseño e implementacion de un prototipo de robotica colaborativa de ensamble de producto.

[14] Mínguez Moretón, J. (2020). Estudio, diseño e integración de una estación industrial colaborativa.

[15] *3 empresas manufactureras incluyeron cobots en su producción*. (2020). Metal Metálica Internacional. Enlace. Acceso 16/07/2021

[16] Pelegrí, J. (2021). Repsol Technology Lab personaliza sus robots colaborativos para automatizar las tareas de laboratorio. *Industria química*, (89), 46-47.

[17] H. Elhawary, y A. Popovic, "Robust feature tracking in the beating heart for a roboticguided endoscope". Int. J. Medical Robotics and Computer Assisted Surgery, 7, pp. 459-468, 2011.

[18] A. Knoll, H. Mayer, C. Staub, y R. Bauernschmitt, "Selective automation and skill transfer in medical robotics: a demonstration on surgical knot-tying". Int. J. Medical Robotics and Computer Assisted Surgery, D.O.I. 10.1002/res.149, 2012.

[19] C. Staub, S. Can, A. Knoll, V. Nitsch, I. Karl, y B. Farber, "Implementation and evaluation of a gesture-based input method in robotic surgery". IEEE Int. Workshop on Haptic Audio Visual Environments and Games (HAVE), 2011.

[20] M. Kranzfelder, A. Schneider, S. Gillen, y H. Feussner, "New technologies for information retrieval to achieve situational awareness and higher patient safety in the surgical operating room: the MRI institutional approach and review of the literature". Surgical Endoscopy, 25, pp. 696-705, 2011.

[21] Rey Serrano, J. L., Lázaro Cornejo, J. L., Massana Guitart, F., Yuste Marco, A., Bustos García, P., Plana Farnós, S., ... & Loichate Cid, M. (2009). La robótica al servicio de la salud: Caso de aplicación robótica para la atención sanitaria de la salud infantil. I+ S. Informática y Salud, (77), 30-42.

[22] Cook A., Meng M., Gu J. and Howery K (2002). "Development of a robotic device for facilitating learning by children who have severe disabilities", IEEE Transactions on Neural Systems and Rehabilitation Engineering, 10 (3), p. 178-187, ISSN: 15344320.

[23] Pérez, M. H. (2021, 9 abril). *El robot cantante de ocho brazos y otras máquinas capaces de hacer música.* EL País. Enlace. Acceso 20/07/2021

[24] *Robots que cantan en verso ayudan a clasificar la música*. (2020). Agencia SINC. https://www.agenciasinc.es/Noticias/Robots-que-cantan-en-verso-ayudan-a-clasificar-la-musica

[25] Kuka. Timo Boll – The spin o flife, 2020. Enlace. Acceso 22/07/2021

[26] Toyota. The development of CUE, the AI Basketbal robot, 2020. Enlace. Acceso 23/07/2021

[27] Nao – Team HTWK, 2020. https://www.htwk-robots.de/

[28] Domínguez, R. G. (2019, 24 septiembre). *Estos robots ya saben hacer gimnasia*. AS.com. Enlace. Acceso 23/07/2021

[29] Malpartida, E. A. S. (2011). *Sistema de visión artificial para el reconocimiento y manipulación de objetos utilizando un brazo robot*. Pontificia Universidad Catolica del Peru-CENTRUM Catolica (Peru).

[30] *Universal Robots - Blog*. (s. f.). Universal Robots. Recuperado 19 de septiembre de 2021, de Enlace. Acceso 24/07/2021

[31] I. (2018, 23 marzo). *Robots espaciales: la necesidad de la visión artificial*. Revolución artificial. Enlace. Acceso 24/07/2021

[32] Guerrero Pilco, C. G. (2020). *Diseño e implementación de un robot recolector de pelotas de tenis de campo mediante visión artificial* (Bachelor's thesis, Escuela Superior Politécnica de Chimborazo).

[33] Jiménez, F. J., Moreno, J. C., González, R., Díaz, F. R., & Sánchez-Hermosilla, J. (2008). Sistema de visión de apoyo a la navegación de un robot móvil en invernaderos. *CEA Jornadas de Automática*, 10-17.

ANEXOS

ANEXO I: PAPER JJAA21

APLICACIÓN DE UN ROBOT COLABORATIVO DE DOS BRAZOS PARA RESOLVER EL CUBO DE RUBIK

Garazi Fernández, Asier Herrán, Aitziber Mancisidor, César Pérez, Itziar Cabanes

gfernandez051@ikasle.ehu.eus, aherran014@ikasle.ehu.eus, aitziber.mancisidor@ehu.eus, itziar.cabanes@ehu.eus
Departamento de Ingeniería de Sistemas y Automática
Escuela de Ingeniería de Bilbao, UPV/EHU

Resumen

Con objeto de entretener y acercar al usuario al mundo de la robótica, este trabajo presenta una célula robotizada para la resolución de cubos de Rubik. El sistema está formado por el robot colaborativo YuMi de dos brazos y una cámara de visión artificial. Gracias a esta cámara, el sistema es capaz de detectar la posición y la configuración del cubo de Rubik, además de identificar si se trata de un cubo de 2x2 o de 3x3. Una vez detectado el cubo, el robot utiliza los dos brazos para resolver el cubo con el mínimo número de giros y movimientos posibles. Adicionalmente, con el fin de aumentar la comunicación entre el sistema y el usuario, se le ha añadido una pantalla táctil y un módulo de sonido.

Palabras clave: Robótica colaborativa, visión artificial, Matlab, RobotStudio, cubo de Rubik.

1 INTRODUCCIÓN

En las últimas décadas los robots han sido ampliamente utilizados en diferentes ámbitos de la industria, como es en aplicaciones de soldadura, ensamblaje o empaquetamiento. Estos, resultan muy útiles para operaciones repetitivas y producción en grandes lotes. Sin embargo, por motivos de seguridad, no es posible la directa interacción entre el operario y los robots. Este es el motivo por el cual en la cuarta revolución industrial se desarrollan los robots colaborativos o cobots [12].

Los robots colaborativos traen integrados sensores de fuerza, presión o tacto que posibilitan la colaboración entre los robots y sus usuarios, eliminando las barreras físicas que los han separado durante muchos años [6]. Estos sensores permiten medir y controlar la fuerza y la velocidad, lo que garantiza que no superen los umbrales definidos en caso de producirse un contacto, sea intencionado o por accidente, manteniendo en todo momento la seguridad del usuario y de los componentes de su entorno.

Además, los cobots permiten reducir de forma significativa los problemas de espacio en su instalación (por la eliminación de barreras), mejorar la facilidad y rapidez en la programación y agilizar la adaptación ante cambios en la producción [2].

Uno de los sectores que más se beneficia de estos robots es el de la industria. A día de hoy los cobots son ampliamente utilizados en las líneas de producción, realizando diversas tareas desde ensamblado hasta clasificación de objetos, pasando por la fabricación [1, 11], entre otras.

No obstante, uno de los grandes beneficos de la robótica colaborativa ha sido la posibilidad de utilizar los robots fuera de la industria. Hoy en día se pueden encontrar robots colaborativos en el sector sanitario [4], en terapias de rehabilitación [13] o en el ámbito de la enseñanza [10, 7].

Otra de las grandes aplicaciones de estos robots es el entretenimiento. En el mercado existen una gran cantidad de robots destinados al entretenimiento. Entre ellos, los más populares son las mascotas robóticas las cuales poseen un gran rango de funcionalidades; desde las más simples que únicamente se mueven por la casa, hasta las más complejas, dotadas de inteligencia artificial y equipadas con cámaras que les permiten reconocer comandos de voz, esquivar obstáculos y hasta ayudar en las tareas del hogar [5].

En este trabajo, se presenta una aplicación de un robot colaborativo de dos brazos capaz de resolver los cubos de Rubik de 2x2 y 3x3. Con ello se pretende lograr una aplicación llamativa que genere interés, entretenga al usuario y atraiga al usuario al mundo de la robótica.

El resto del artículo se estructura de la siguiente manera. En la sección 2 se realiza una descripción del diseño de la aplicación. En la sección 3 se detalla la programación e implementación de la misma, seguida de la sección 4 donde se resumen los resultados. Finalmente, en la sección 5, se resumen las conclusiones y trabajos futuros.

https://doi.org/10.17979/spudc.9788497498043.611

2 DISEÑO DE LA APLICACIÓN

2.1 ESPECIFICACIÓN Y FUNCIONALIDAD

El cubo de Rubik es un mecanismo de ejes que permite que cada una de sus caras gire independientemente, de manera que los colores se mezclen. Para solucionar el rompecabezas, cada una de sus caras debe estar compuesta por un solo color. Completar dicho rompecabezas 3D requiere de una serie de habilidades tales como concentración, memoria, coordinación óculo manual, matemáticas, buenas destrezas psicomotoras y agilidad mental, entre otras [3].

Con objeto de entretener y lograr la resolución del cubo de Rubik de forma automática, se plantea el diseño de una aplicación robótica, de manera que permita al usuario interactuar y retar al robot sin peligro alguno.

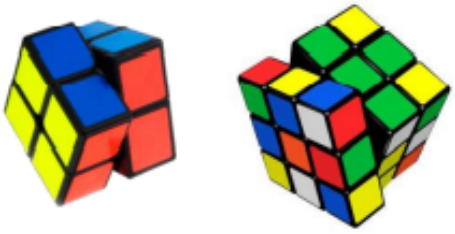


Figura 1: Cubos de Rubik de 2x2 y 3x3

Para ello, se selecciona el robot colaborativo de dos brazos Yumi, del fabricante ABB, con el que los usuarios puedan interactuar, de manera que logre atraer su atención. Una vez que el usuario deposite uno de los cubos de Rubik en cualquier punto arbitrario del escenario, el robot lo recogerá e identificará el tipo de cubo que es, considerando las opciones de 2x2 y de 3x3 (ver Figura 1). Una vez reconocido, realizará fotos de las 6 caras del cubo para conocer la configuración inicial, es decir saber con precisión la ubicación de las piezas de las diferentes caras que componen el cubo. Tras procesar dichas imágenes, el robot conocerá la posición y el color de cada una de las piezas móviles que componen el cubo y será necesario aplicar un algoritmo que calcule la mejor estrategia a seguir. Esto es, el mínimo número de giros y movimientos que se deberán realizar en el cubo para completar las 6 caras del mismo color respectivamente, y cómo debe realizarse (dirección y sentido). Finalmente, con toda la información adquirida, el robot mediante sus dos brazos ejecutará los movimientos calculados para resolver el cubo.

Adicionalmente, con objeto de incrementar la interacción y empatía del robot con el usuario, la aplicación contará con voz y con una pantalla táctil para interaccionar de forma sencilla con él.

En la Figura 2 se muestran los componentes hardware para llevar a cabo esta aplicación: 1) el robot colaborativo Yumi de dos brazos, del fabricante ABB, que realizará los movimientos y giros deseados en el cubo como lo haría una persona; 2) una cámara de visión artificial, UI 5584LE-C-HQ de Infaimon, para identificar dónde deposita el usuario el cubo, de qué tipo de cubo se trata (2x2 ó 3x3) y también se encargará de realizar las fotos de las diferentes caras para saber la configuración inicial del cubo; 3) un PC el cual permite procesar las imágenes y desarrollar la algoritmia que minimice los giros para la resolución del cubo de Rubik mediante el entorno Matlab.

Figura 2: Relación de componentes

2.2 DISEÑO Y SELECCIÓN DE COMPONENTES

Para afrontar las tareas y funciones definidas en el sistema, es necesario realizar el diseño de varios componentes del escenario, los cuales se detallan en este apartado.

2.2.1 Componentes auxiliares

Para que el robot pueda localizar de forma autónoma el cubo, cogerlo, resolverlo, y depositarlo una vez resuelto, hace falta una superficie plana donde hacerlo. Debido a que hay más elementos en el escenario que forman parte de otras células de montaje, se ha optado por diseñar una mesa mediante la herramienta AutoCAD que tenga la facilidad de integrarse cuando se requiera. En dicha mesa el usuario depositará el cubo en un punto arbitrario para que el robot lo recoja y comience con su función.

La ubicación de la cámara es muy importante para esta aplicación. Concretamente, para el sistema de visión (cámara, placa base, lente y un ventilador), se ha realizado el diseño de una carcasa que además de

https://doi.org/10.17979/spudc.9788497498043.611

protegerla ayude a ubicarla en la célula de trabajo, permitiendo la orientación de la cámara en los ángulos que se desee sin que entorpezca el correcto funcionamiento del robot. Esta carcasa consta de dos componentes, tapa y base, las cuales han sido diseñadas mediante AutoCAD. Adicionalmente, también se ha diseñado un soporte que sujete de forma conjunta la pantalla táctil al robot. En la Figura 3 se muestra dicho componente.

Finalmente, puesto que es necesario que el robot sea capaz de recoger, girar y manipular con destreza el cubo de Rubik, se han diseñado unos dedos con mayor capacidad de apertura y mejor adecuación que los que poseía el Yumi (ver Figura 3).

Tras realizar el diseño, se han generado todos los componentes mediante una impresora 3D (Artillery X1 Sidewinder).

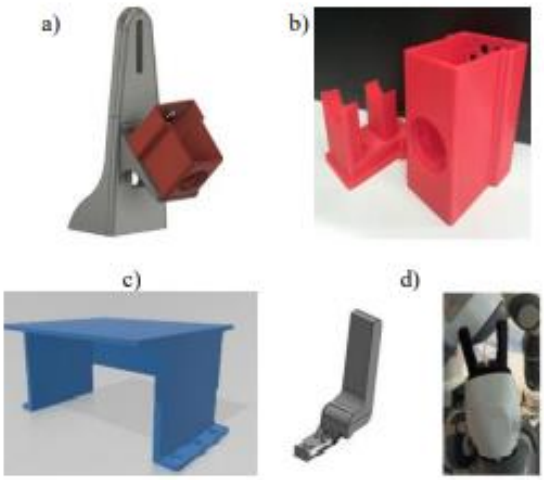


Figura 3: Componentes auxiliares diseñados: a) soporte para la carcasa de la cámara y la pantalla táctil, b) carcasa donde se aloja la cámara, c) mesa donde se recoge y deposita el cubo, d) dedos de la pinza.

2.2.2 Componentes interacción

Debido a que uno de los objetivos principales es lograr la interacción usuario-robot, se ha instalado una pantalla táctil, TFT-LCD Monitor, mediante la cual se podrá interactuar con el robot, controlando tanto los movimientos del robot como el procesamiento de las imágenes.

Adicionalmente, para lograr una mayor relación con el robot se han integrado un módulo de sonido, el cual será utilizado por el Yumi para anunciar al usuario sus futuras acciones. El módulo de sonido consta de dos altavoces, a los cuales se les envían los datos mediante Matlab.

En la Figura 4 se muestra el escenario completo, con todos los componentes detallados en esta sección.

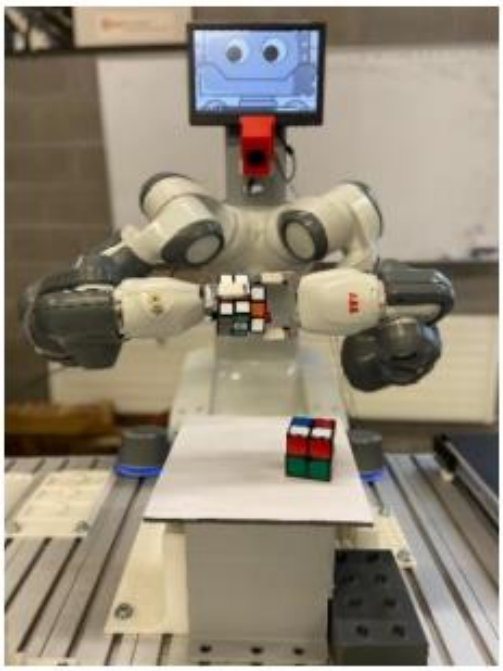


Figura 4. Escenario completo.

3 PROGRAMACIÓN

Para llevar a cabo la resolución del cubo de Rubik hace falta realizar una serie de instrucciones y códigos. Estos deben permitir tanto, obtener las imágenes y procesarlas, extrayendo de ellas la información necesaria para trasladarle al robot los giros a realizar.

3.1 PROCESAMIENTO DE IMÁGENES

Al detectar un cubo de Rubik en la mesa, el sistema, mediante la cámara de visión artificial y el software Matlab, efectúa la primera fotografía la cual se utiliza para localizar la posición exacta del cubo.

En este primer procesamiento de la imagen, se aplica una máscara de color con objetivo de aislar los colores del cubo y seguido se binariza la imagen. Posteriormente utilizando funciones como *strel*, *cornermetric* e *imregionalmax* se logra filtrar el ruido y obtener el contorno de cada uno de los objetos, para después mediante el código creado con el comando *regionprops* conseguir las posiciones de los puntos (ver Figura 5).

https://doi.org/10.17979/spudc.9788497498043.611

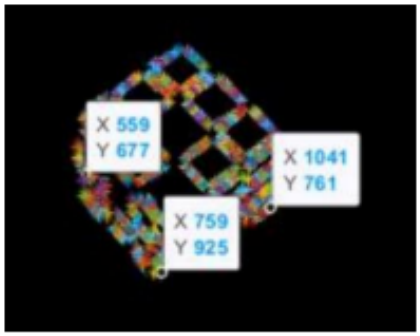


Figura 5: Procesamiento mediante *regionprops*

Por otro lado, una vez que el robot ha recogido el cubo, se realiza la toma de imágenes de las caras del mismo (ver Figura 6), siendo un total de 6 fotos, una por cada cara. Tras esta toma de imágenes del cubo, se realiza un procesamiento de los colores de cada cara que permita conocer la posición de cada una de las nueve piezas que las forman y, por ende, la forma en la que está mezclado el cubo. Para este procesamiento de color se ha utilizado la función *ColorThresholder*, que genera una máscara que muestra solo los colores seleccionados (ver Figura 6).

Figura 6: Procesamiento de imágenes

Una vez procesados todos los colores de la imagen, se genera la matriz Cara donde cada color se representa mediante un número siendo, Rojo = 1; Azul = 2, Naranja = 3; Verde = 4; Blanco = 5 y Amarillo = 6. Por tanto, la fotografía original de la cara se reduce a una matriz de dimensión 3x3. Por último, la información de cada Cara se traslada a un array multidimensional R que contiene toda la información del cubo mezclado. Dicho array está compuesto por seis matrices de 3x3, una por cara. Este array se enviará al algoritmo que se presenta en el siguiente apartado para calcular la resolución del cubo.

3.2 ALGORITMO RESOLUCIÓN CUBO DE RUBIK

Una vez identificadas las posiciones de los bloques de colores del cubo, es necesario desarrollar el algoritmo de resolución, con el que se obtienen los giros a realizar.

En este caso, se aborda la resolución del cubo mediante la estrategia de Joren Heit [9]. A partir de imágenes tomadas, se lograran los movimientos para resolver el cubo mediante la función *Solve45*, asegurando una media de 31 movimientos para la resolución completa del cubo (siendo 45 su máximo). El algoritmo se basa en unas tablas de poda generadas automáticamente y formadas por más de un millón de entradas que le permiten elegir la mejor solución en función de la forma en la que esté mezclado el cubo.

La implementación y ejecución del algoritmo conlleva cuatro fases:
1) Orientación de los bordes.
2) Orientación de las esquinas y colocar los bordes de las caras derecha e izquierda.
3) Colocar el resto de bordes y esquinas.
4) Resolver el cubo.

Para cada una de dichas fases el algoritmo calcula mediante las tablas de poda el número mínimo de movimientos. Este número de movimientos mínimo se denomina d, y tras elegir un giro, se comprueba cómo varía. Si d disminuye, el movimiento es correcto, ya que se encuentra un movimiento más cerca de la siguiente fase. Cuando d sea igual a 0, el algoritmo pasa de fase. Este algoritmo devuelve un array de celdas en las que cada una hace referencia a un movimiento a realizar para resolver el cubo.

3.3 MOVIMIENTOS ROBOT

El robot posee dos brazos, que trabajarán de forma sincronizada a la hora de realizar movimientos. El brazo derecho se encarga de coger el cubo de la mesa y llevar el cubo hasta la posición de las 3 primeras fotos. El brazo izquierdo deberá coger el cubo en la posición de la tercera foto, y trasladarlo para realizar las 3 restantes.

Posteriormente, para que los brazos ejecuten los movimientos necesarios para la resolución del cubo, es necesario enviar al robot colaborativo YuMi los movimientos codificados obtenidos. Para ello hace falta realizar la comunicación entre Matlab y el robot de forma que permita el intercambio de datos por ambas partes. En este caso se ha optado por una comunicación mediante protocolo TCP/IP.

Una vez establecida la comunicación, se procede al envío de los datos al robot. En primer lugar, es necesario contar los movimientos a enviar y guardar la información en una variable. El siguiente paso es enviar el número de movimientos al robot de forma que sepa la cantidad de caracteres que tiene que recibir y procesar. En caso de ser menos de 26 movimientos se realiza un único envío, mientras que si son más, se realizarán dos.

https://doi.org/10.17979/spudc.9788497498043.611

4 DISCUSIÓN Y RESULTADOS

Una vez configurado el escenario con la mejor ubicación de los componentes, y realizado el código de programación en el entorno Matlab y RobotStudio, se valida la solución.

Dado que uno de los objetivos principales era retar al robot para lograr que la resolución del cubo sea lo más rápida posible, se han realizado varias pruebas con los cubos de 2x2 y 3x3. De esta manera se han obtenido diferentes resultados de tiempos y cantidad de movimientos que se necesitan para resolver los distintos cubos.

Finalmente, se han realizado las pruebas de resolución del cubo de 2x2 y 3x3, ejecutando un total de 10 pruebas con cada uno de ellos. En la tabla 1 se resume el número de movimientos realizados (Mov) para resolver cada una de las 10 configuraciones de partida con cada cubo y el tiempo que tarda el robot en conseguirlo (T).

Tabla 1: Ensayos con cubos de Rubik de 2x2 y 3x3

Cubo de Rubik 2x2										
Mov	10	12	13	15	11	19	14	12	16	9
T (min)	2,3	3,16	3,4	3,5	2,5	4,4	3,2	3,16	4,1	2
Cubo de Rubik 3x3										
Mov	29	34	35	30	28	33	29	27	32	34
T (min)	7	8,1	8,4	7,2	6,7	8	7	6,5	7,7	8,1

Se puede concluir que el tiempo medio que necesita el robot para resolver el cubo de 2x2 es de 3,17 minutos y la media del número de movimientos es de 13,1. En el caso del cubo 3x3, el tiempo medio es de 7,47 minutos y la media de movimientos es de 31,1.

Como era de esperar, el robot tarda mucho menos tiempo y necesita menos movimientos para resolver el cubo de 2x2, puesto que este posee un menor número de combinaciones posibles.

5 CONCLUSIONES

En este trabajo se ha diseñado y construido una célula robótica de entretenimiento que permite resolver cubos de Rubik de 2x2 y 3x3 de forma automática.

El sistema propuesto está compuesto por el robot colaborativo YuMi de dos brazos, una cámara de visión artificial, un ordenador de procesamiento con Matlab y elementos auxiliares como la mesa, el soporte de la camara o dedos de la pinza del robot impresos con una impresora 3D.

Adicionalmente, con la intención de atraer a los usuarios, se le ha añadido una pantalla táctil y un sistema de sonido que aumenta la interacción entre el robot y el usuario.

Después de realizar varios ensayos experimentales, se puede concluir que el sistema propuesto es capaz de resolver el cubo de Rubik de 2x2 en un tiempo medio de 3,17 y el de 3x3 en 7,47 minutos.

Agradecimientos

Este trabajo ha sido parcialmente financiado por el proyecto GIU19/045 de la UPV/EHU.

English summary

APPLICATION OF A TWO-ARMED COLLABORATIVE ROBOT TO SOLVE RUBIK'S CUBE

Abstract

In order to entertain and get the robotics closer to the user, this work presents a robotic cell for the resolution of Rubik's cubes. The system consists of the YuMi two-armed collaborative robot and an artificial vision camera. Thanks to this camera, the system is able to detect the position and configuration of the Rubik's cube, in addition to identifying whether it is a 2x2 or 3x3 cube. Once the cube is detected, the robot uses its both arms to solve the cube with the minimum number of turns and movements possible. Additionally, in order to increase communication between the system and the user, a touch screen and a sound module have been added.

Keywords: Collaborative robotics, computer vision, Matlab, RobotStudio, Rubik's cube.

Referencias

[1] Gómez, E., & Andrés, D. Diseño e implementación de una celda colaborativa robotizada mediante robots móviles y humanoides para clasificación de objetos. Recuperado el 22 de junio de 2021, de Edu.ec

[2] G. Du, M. Chen, C. Liu, B. Zhang, and P. Zhang, "Online Robot Teaching With Natural Human–Robot Interaction", IEEE Transactions on Industrial

https://doi.org/10.17979/spudc.9788497498043.611

Electronics, vol. 65, no. 12, pp. 9571–9581, Dec. 2018.

[3] Julita, J. R., & Viejo, V. A. (2015). Cubo de Rubik. XVIII Concurso de Trabajos Estudiantiles (EST 2015) - JAIIO 44.

[4] J.F. Avila-Tomás, M.A. Mayer-Pujadas, & V.J. Quesada-Varela (2020). La inteligencia artificial y sus aplicaciones en medicina I: introducción antecedentes a la IA y robótica. *Atención Primaria, 52(10), 778-784.*

[5] López Rodríguez, R., & Ospina Saldaña, J. (2019). Robot domótico con control remoto y cámaras, para pruebas de Mayordomo y Jardinero: Rob-Erto. Letras ConCiencia TecnoLógica, (15), 13-17. Recuperado de https://revistas.itc.edu.co/index.php/letras/article/view/155

[6] Moctezuma Gutiérrez, S. G., Cruz Pazarán, A., Galicia Mejía, R., & Oliva Moreno, L. N. (2018). Desarrollo de plataforma para implementación de robots colaborativos. Visión electrónica, 12(1), 22–31. https://doi.org/10.14483/22484728.13308

[7] Moreno, I., Muñoz, L., Serracín, J. R., Quintero, J., Patiño, K. P., & Quiel, J. (2012). La robótica educativa, una herramienta para la enseñanza-aprendizaje de las ciencias y las tecnologías. *Teoría de la Educación. Educación y Cultura en la Sociedad de la Información, 13(2),* 74-90.

[9] Rubik's cube simulator and solver - file exchange - MATLAB central. (2011, octubre 23). Recuperado el 23 de junio de 2021, de Mathworks.com website: https://es.mathworks.com/matlabcentral/fileexchange/31672-rubik-s-cube-simulator-and-solver

[10] Salamanca, M. L. P., Lombana, N. B., & Holguín, W. J. P. (2010). Uso de la robótica educativa como herramienta en los procesos de enseñanza. *Ingeniería Investigación y Desarrollo: I2+ D, 10(1),* 15-23.

[11] Salimbeni, S., & Mamani, D. (2020). Marco de referencia para la incorporación de Cobots en líneas de manufactura. *Podium*, *38*(38), 159–180.

[12] Schwab, K. (2016). The fourth industrial revolution. Ed. Crown Business.

[13] Tucan, D. (2021). Ankle Rehabilitation of Stroke Survivors Using Kuka LBR Iiwa. In *New Trends in Medical and Service Robotics* (pp. 29–36). Springer International Publishing.

https://doi.org/10.17979/spudc.9788497498043.611

ANEXO II: POSTER JJAA21

APLICACIÓN DE UN ROBOT COLABORATIVO DE DOS BRAZOS PARA RESOLVER EL CUBO DE RUBIK

Garazi Fernández, Asier Herrán, Aitziber Mancisidor, Cesar Perez y Itziar Cabanes

Departamento de Ingeniería de Sistemas y Automática, Escuela Técnica Superior de Ingeniera de Bilbao, Universidad del País Vasco (UPV/EHU)

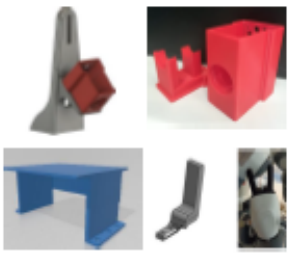
gfernandez051@ikasle.ehu.eus, aherran014@ikasle.ehu.eus, aitziber.mancisidor@ehu.eus, cesar.perez@ehu.eus, itziar.cabanes@ehu.eus

Actualmente, en la 4. revolución industrial, los robots colaborativos están en auge, cada día presentan mas capacidades y mayor autonomía, por ello se están utilizando en una gran diversidad de sectores: industrial, sanitario, educativo, de entretenimiento, etc. Ante esta situación, con el fin de entretener y lograr la resolución del cubo de Rubik de forma automática, este trabajo presenta el diseño de una aplicación robótica, de manera que permita al usuario interactuar y retar al robot sin peligro alguno. Para ello se hace uso de un robot colaborativo YUMI de dos brazos, una cámara de visión artificial y un PC el cual permite procesar las imágenes y desarrollar la algoritmia que minimice la resolución del cubo de Rubik mediante Matlab.

DISEÑO DE LA APLICACIÓN

Los componentes hardware para llevar a cabo la aplicación:

- Robot colaborativo YUMI
- Cámara de visión artificial UI 5584LE-C-HQ de Infaimon
- PC para procesamiento de imágenes y resolución de algoritmia mediante Matlab



Los componentes auxiliares del escenario:

- Soporte para la carcasa de la cámara y pantalla
- Carcasa donde se aloja la cámara
- Mesa donde se recoge y deposita el cubo
- Dedos de la pinza

Los componentes de interacción:

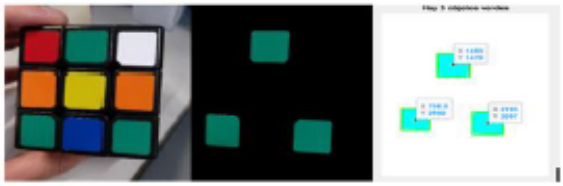
- Pantalla táctil
- Modulo de sonido

En la imagen superior se encuentra el escenario completo

PROGRAMACIÓN

PROCESAMIENTO DE IMÁGENES

- Se detecta el cubo en la mesa y se procesa la imagen para sacar las coordenadas de la posición.
- Tras la toma de imágenes de las 6 caras se realiza un procesamiento de los colores de cada cara.



- La información de cada cara se traslada a un array multidimensional que contiene la información del cubo mezclado

ALGORITMO RESOLUCIÓN CUBO DE RUBIK

- Mediante la estrategia de Joren Heit (asegura una media de 31 movimientos para el cubo de 3x3).
  - orientación de los bordes.
  - orientación de las esquinas y colocar los bordes de las caras derecha e izquierda.
  - colocar el resto de bordes y esquinas.
  - resolver el cubo.
- El algoritmo devuelve un array de celdas (cada celda hace referencia a un movimiento), en la siguiente imagen se ve la nomenclatura.

	Giro horario	Giro anti horario	Giro doble
Cara F	10	11	12
Cara R	20	21	22
Cara B	30	31	32
Cara L	40	41	42
Cara U	50	51	52
Cara D	60	61	62

MOVIMIENTOS ROBOT

Brazo derecho:

- Coger el cubo de la posición situada en la mesa
- Tres primeras posiciones de las fotos

Brazo Izquierdo:

- Coger cubo de la posición de la tercera foto
- Trasladarlo para realizar las tres restantes

- Comunicación entre Matlab y el robot mediante protocolo TCP/IP.
- Se guardan los movimientos en un array y se envían la robot

ANALISIS Y DISCUSION DE RESULTADOS

- 10 pruebas con el cubo de 2x2
- 10 pruebas con el cubo de 3x3

Cubo de Rubik 2x2										
Mov.	10	12	13	15	11	19	14	12	16	9
Min.	2,3	3,16	3,4	3,5	2,5	4,4	3,2	3,16	4,1	2
Cubo de Rubik 3x3										
Mov.	29	34	35	30	28	33	29	27	32	34
Min.	7	8,1	8,4	7,2	6,7	8	7	6,5	7,7	8,1

Análisis de la media obtenida de los tiempos y movimientos

- No supera la cantidad de movimientos estimada por el algoritmo para el cubo de 2x2
- No supera la cantidad de movimientos estimada por el algoritmo para el cubo de 3x3

Media Cubo 2x2	
Mov.	13,1
Min.	3,17
Media Cubo 3x3	
Mov.	31,1
Min.	7,47

EUSKO JAURLARITZA GOBIERNO VASCO
euskampus
tecnalia

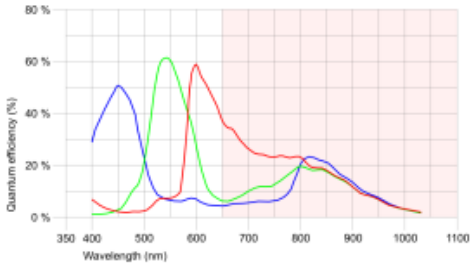
ANEXO III: CÁMARA DE VISION ARTIFICIAL UI-5584LE-C-HQ

UI-5584LE-C-HQ (AB00469)

Especificación

Sensor

Tipo de sensor	CMOS Color
Sistema de obturador	Rolling Shutter / Global Start Shutter
Characteristic	Lineal
Método de lectura del sensor	Progressive scan
Clase de píxeles	QSXGA
Resolución	4,92 Mpx
Resolución (h x v)	2560 x 1920 Pixel
Relación de aspecto	4:3
CAD	12 bit
Profundidad de color (camára)	12 bit
Clase de sensor óptico	1/2"
Superficie optica	5,632 mm x 4,224 mm
Diagonal del sensor óptico	7,04 mm (1/2,27")
Tamaño de píxel	2,2 µm
Fabricante	ON Semiconductor
Denominación del sensor	MT9P006STC
Ganancia (total/RGB)	12.2x/5.8x
AOI horizontal	mayor frecuencia de imagen
AOI vertical	mayor frecuencia de imagen
AOI ancho de imagen / ancho de paso	32 / 4
AOI alto de imagen / ancho de paso	4 / 2
AOI cuadrícula de posición (horizontal/vertical)	4 / 2
Binning horizontal	mayor frecuencia de imagen
Binning vertical	mayor frecuencia de imagen
Método binning	Color
Factor binning	2 / 3 / 4 / 6
Subsampling horizontal	mayor frecuencia de imagen
Subsampling vertical	mayor frecuencia de imagen
Método subsampling	Color
Factor subsampling	2, 3, 4, 5, 6



UI-5584LE-C-HQ (AB00469)

Modelo

Rango de frecuencia de píxeles	4 MHz - 96 MHz
Frecuencia de imagen en modo libre	14
Frecuencia de imágenes disparador (máxima)	14
Tiempo de exposición (mínimo - máximo)	0.034 ms - 3404 ms
Consumo de potencia	2,6 W - 3,1 W
Memoria gráfica	60 MB

Condiciones ambientales

Las temperaturas mencionadas describen la temperatura del aparato exterior de la carcasa de la cámara. Para versiones de placa tenga en cuenta las indicaciones específicas que figuran en la documentación pertinente.

Temperatura del aparato durante el funcionamiento	0 °C - 55 °C / 32 °F - 131 °F
Temperatura del aparato durante el almacenamiento	-20 °C - 60 °C / -4 °F - 140 °F
Humedad (relativa, sin condensación)	20 % - 80 %

Conexiones

Conexión de interfaz	GigE RJ45
Conexión I/O	Conector Molex de 10 polos (Pico Blade)
Alimentación	12 V - 24 V

Asignación de pins conexión I/O

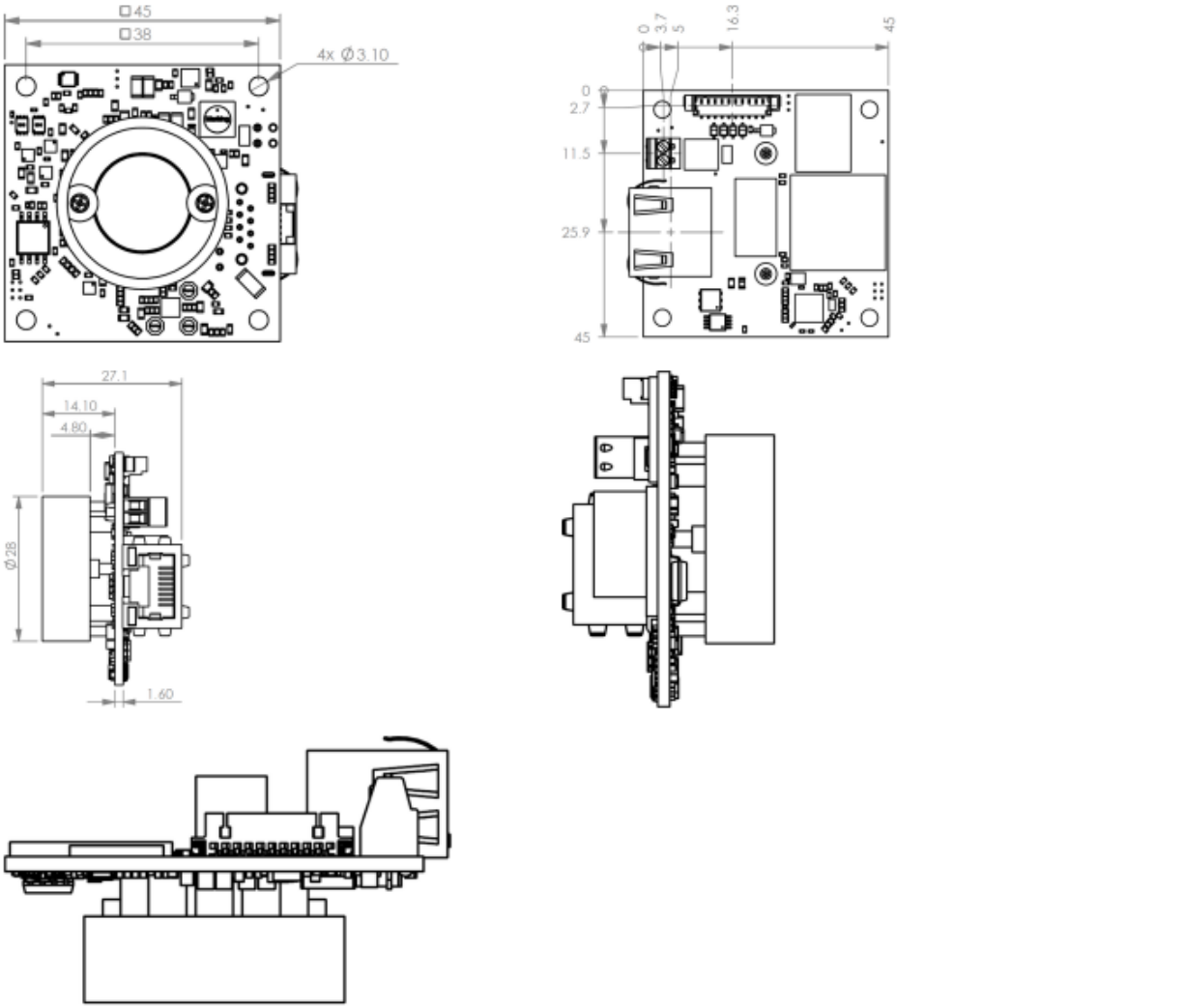
1	Masa (GND)
2	Vout 3,1 V máx. 100 mA
3	Entrada de disparador sin optoacoplador
4	Salida de flash sin optoacoplador
5	General Purpose I/O (GPIO) 1
6	General Purpose I/O (GPIO) 2
7	Señal de reloj bus I2C
8	Señal de datos bus I2C
9	Vin+ 12 V (160 mA) - 24 V (90 mA)
10	Vin- (GND)

Vista de la cámara (vista posterior)

Diseño

Conexión del objetivo	Montura CS / Montura C
Grado de protección IP	-
Dimensiones	45,0 mm x 45,0 mm x 27,1 mm
Peso	24 g

UI-5584LE-C-HQ (AB00469)



ANEXO IV: ROBOT YUMI IRB 14000

ROBOTICS

YuMi® creating an automated future together.

You and me.

YuMi is the first truly collaborative dual armed robot, designed for a world in which humans and robots work together. It heralds a new era of robotic co-workers which are able to work side-by-side on the same tasks as humans with extreme accuracy while ensuring the safety of those around it.

Collaboration
YuMi is designed to meet the flexible and agile production needs required for small parts assembly in the electronics industry. It is also well suited to other small parts environments, including the manufacture of watches, toys and automotive components. All of this thanks to its dual-arms, flexible hands, universal parts feeding system, camera-based part location and state-of-the-art motion control.

Redefining safety
YuMi has a lightweight yet rigid magnesium skeleton covered with a floating plastic casing wrapped in soft padding, which absorbs the force of any unexpected impacts to a very high degree. YuMi has no pinch points so that sensitive ancillary parts cannot be crushed between two opposing surfaces as the axes open and close.

If YuMi senses an unexpected impact or change in its environment such as a collision with a coworker, it can pause its motion within milliseconds to prevent injury, and the motion can be restarted again as easily as pressing play on a remote control.

YuMi is very precise and fast, returning to the same point in space over and over again to within 0.02 mm accuracy and moving at a maximum velocity of 1,500 mm/sec. This ensures the safety of human co-workers on production lines and in fabricating cells.

Total solution concept
ABB also develops software and manufactures hardware, peripheral equipment, process equipment and modular manufacturing cells. This "total solution" concept is evident in YuMi's breakthrough design.

Features
- The fifth-generation, integrated IRC5 controller with TrueMove and QuickMove™ motion control technology commands accuracy, speed, cycle-time, programmability and synchronization with external devices.
- I/O interfaces include Ethernet IP, Profibus, USB ports, DeviceNet™, communication port, emergency stop and air-to-hands. YuMi accepts a wide range of HMI devices including ABB's teach pendant, industrial displays and commercially available tablets.
- The 100-240 volt power supply plugs into any power socket for worldwide versatility.

Benefits
- Can operate equally effectively side-by-side or face-to-face with human coworkers.
- Servo grippers (the "hands") include options for built-in cameras.
- Real-time algorithms set a collision-free path for each arm customized for the required task.
- Padding protects coworkers in high-risk areas by absorbing force if contact is made.

Specification

Robot version	Reach (mm)	Payload (g)	Armload
IRB 14000-0.5/0.5	559	500	No armloads
Number of axes	14		
Protection	Std: IP30 and Clean Room		
Mounting	Table		
Controller	Integrated		
Integrated signal and power supply	24V Ethernet or 4 Signals		
Integrated air supply	1 per Arm on tool Flange (4 Bar)		
Integrated Ethernet	One 100/10 Base-TX ethernet port/per arm		

Performance (according to ISO 9283)

IRB 14000-0.5/0.5	
0.5 kg picking cycle	
25* 300 * 25 mm	0.86s
Max TCP Velocity	1.5 m/s
Max TCP Acceleration	11 m/s*s
Acceleration time 0-1m/s	0.12s
Position repeatability	0.02 mm

Technical information

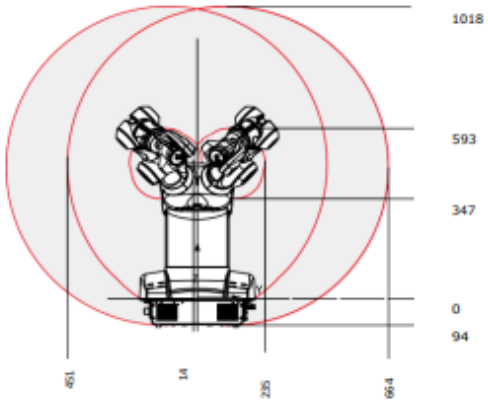
Physical	
Robot base	399 x 496 mm
Robot toes	399 x 134 mm
Weight	38 kg
Environment	
Ambient temperature for mechanical unit	
During operation	+5°C i (41°F) to +40°C (104°F)
During transportation and storage	-10°C (14°F) to +55°C (131°F)
Relative humidity	Max. 85%
Noise level	< 70 dB
Safety	PL b Cat B

Data and dimensions may be changed without notice.

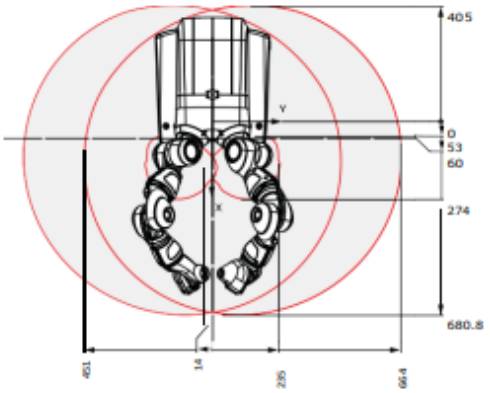
Movement

Axis movement	Working range	Axis max. speed
Axis 1 arm rotation	-168.5° to +168.5°	180°/s
Axis 2 arm bend	-143.5° to +43.5°	180°/s
Axis 7 arm rotation	-168.5° to +168.5°	180°/s
Axis 3 arm bend	-123.5° to +80°	180°/s
Axis 4 wrist rotation	-290° to +290°	400°/s
Axis 5 wrist bend	-88° to +138°	400°/s
Axis 6 flange rotation	-229° to +229°	400°/s

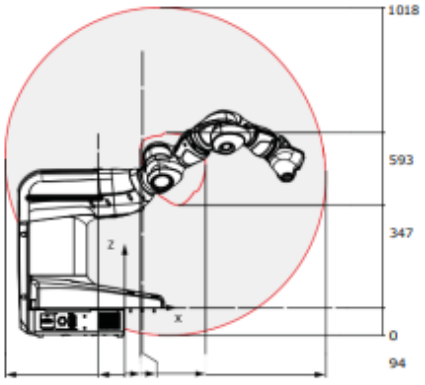
Working range, front view



Working range, top view



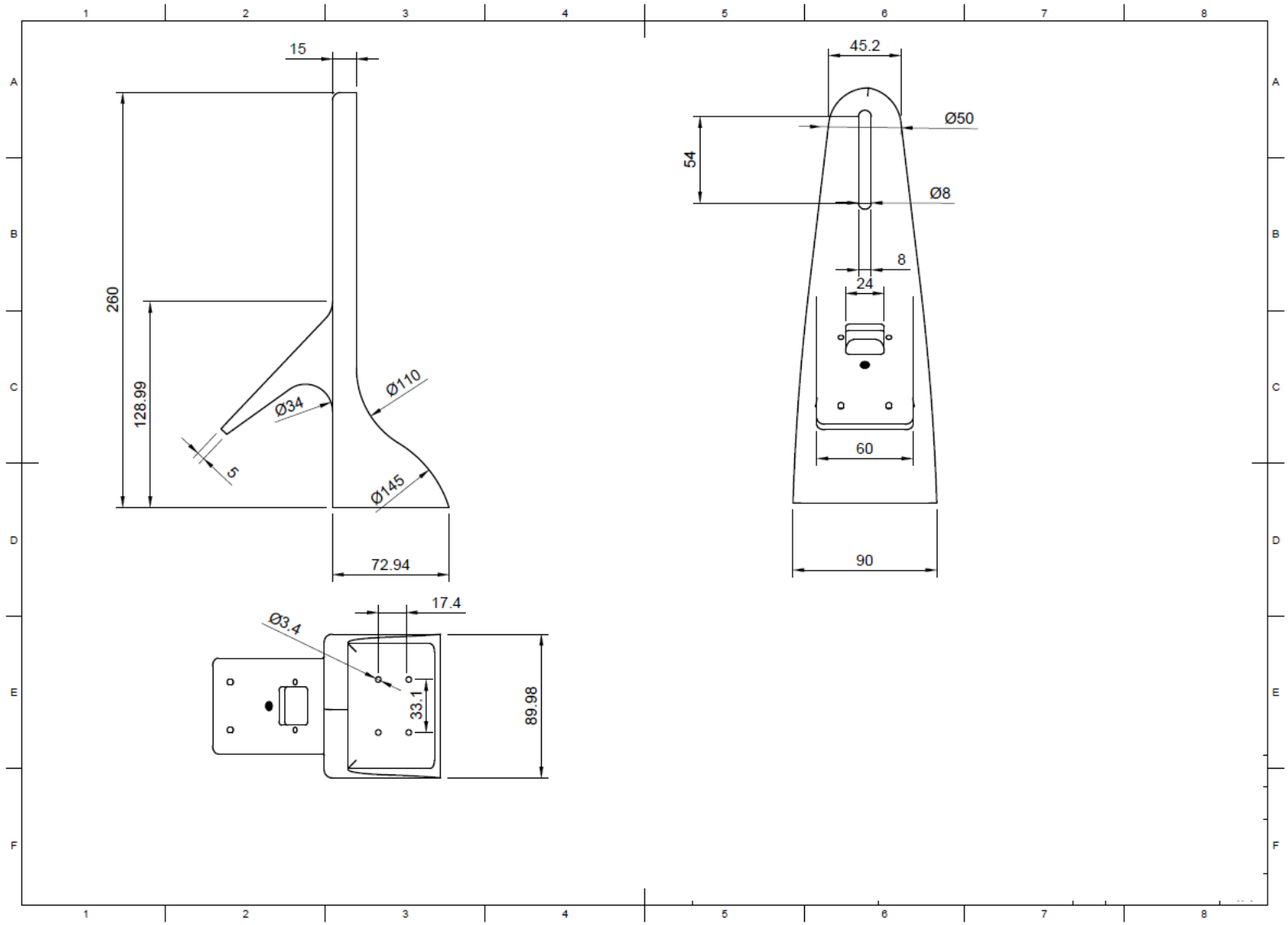
Working range, side view

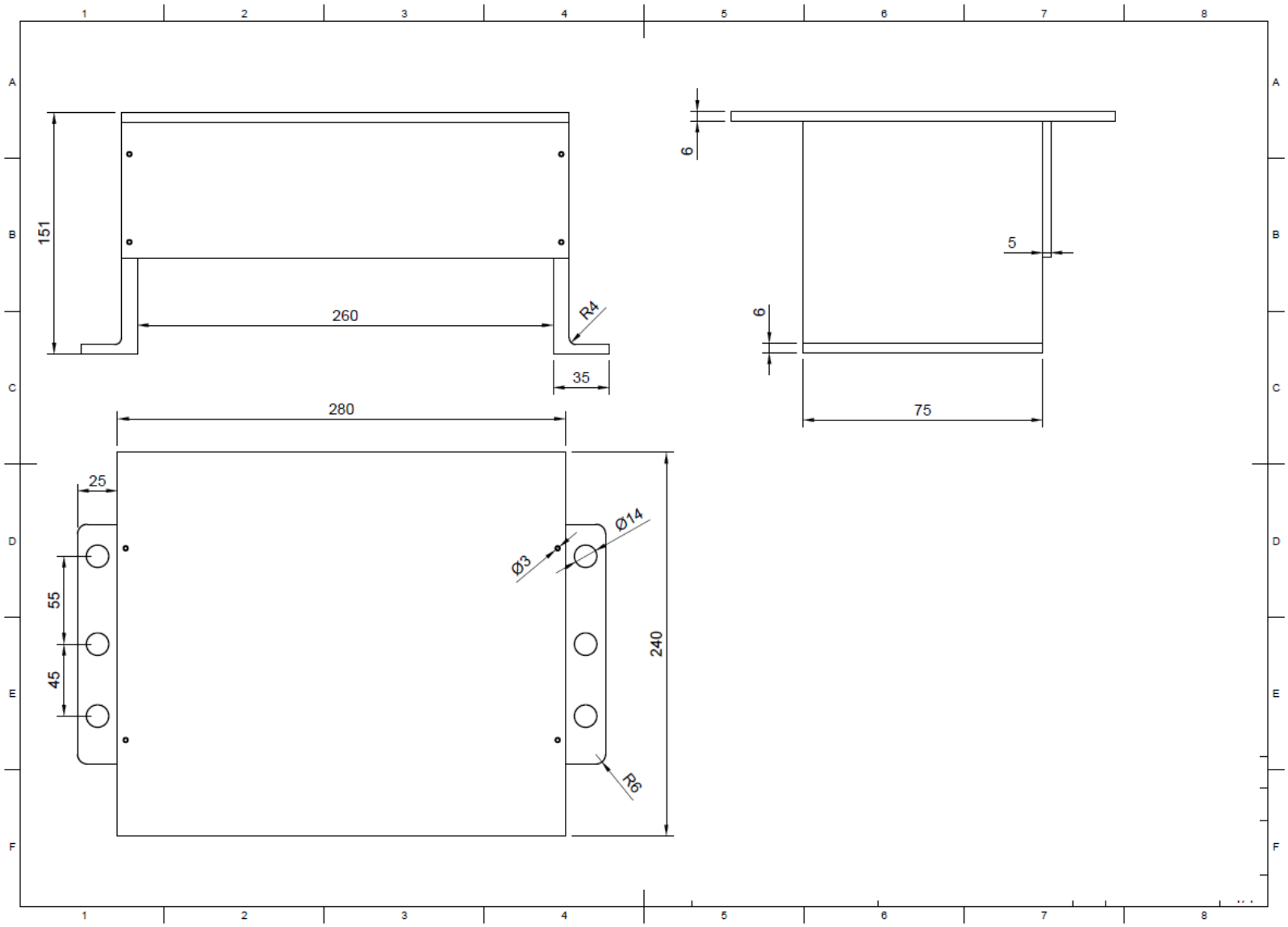


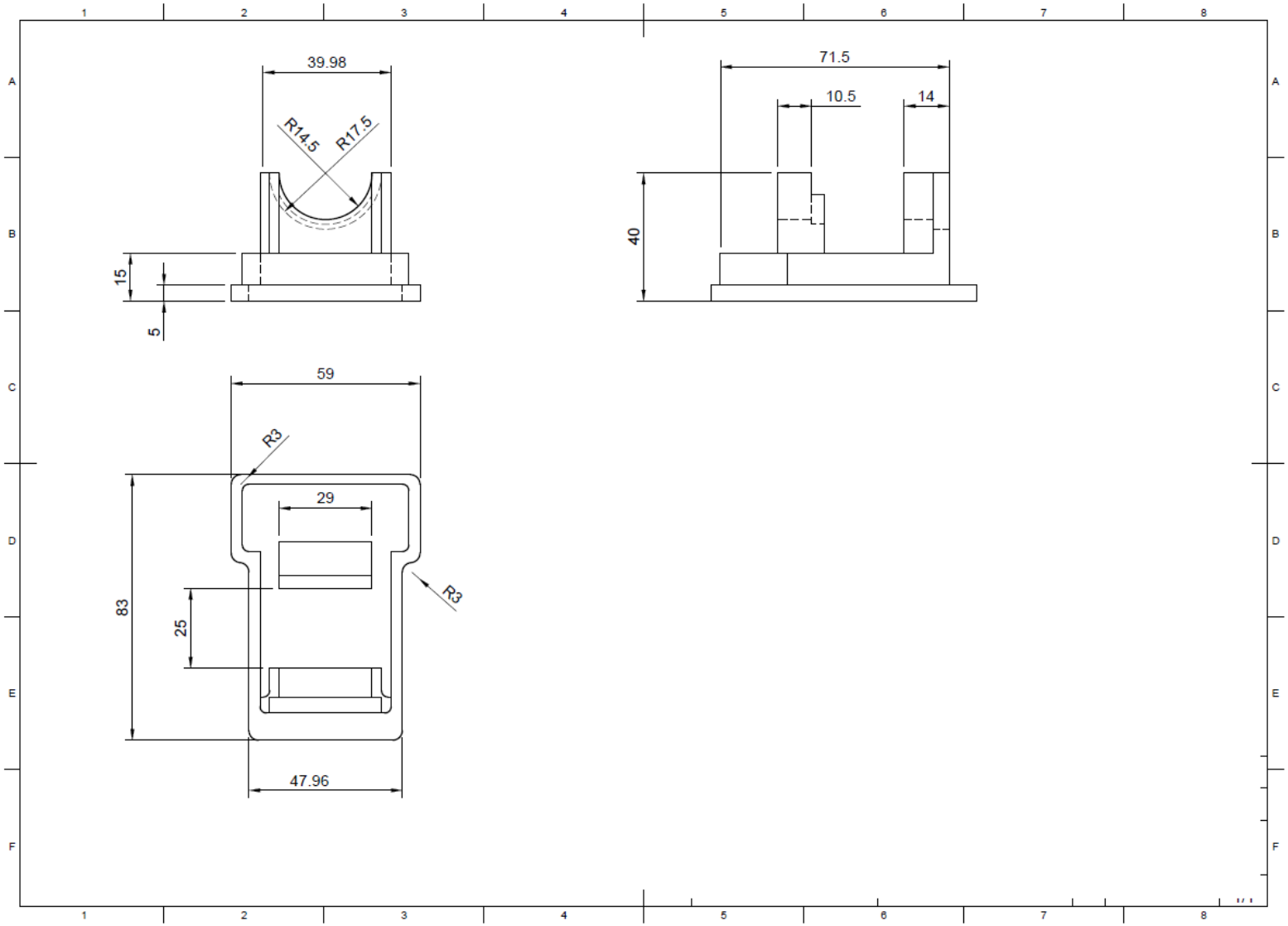
ANEXO V: PLANOS DE COMPONENTES 3D

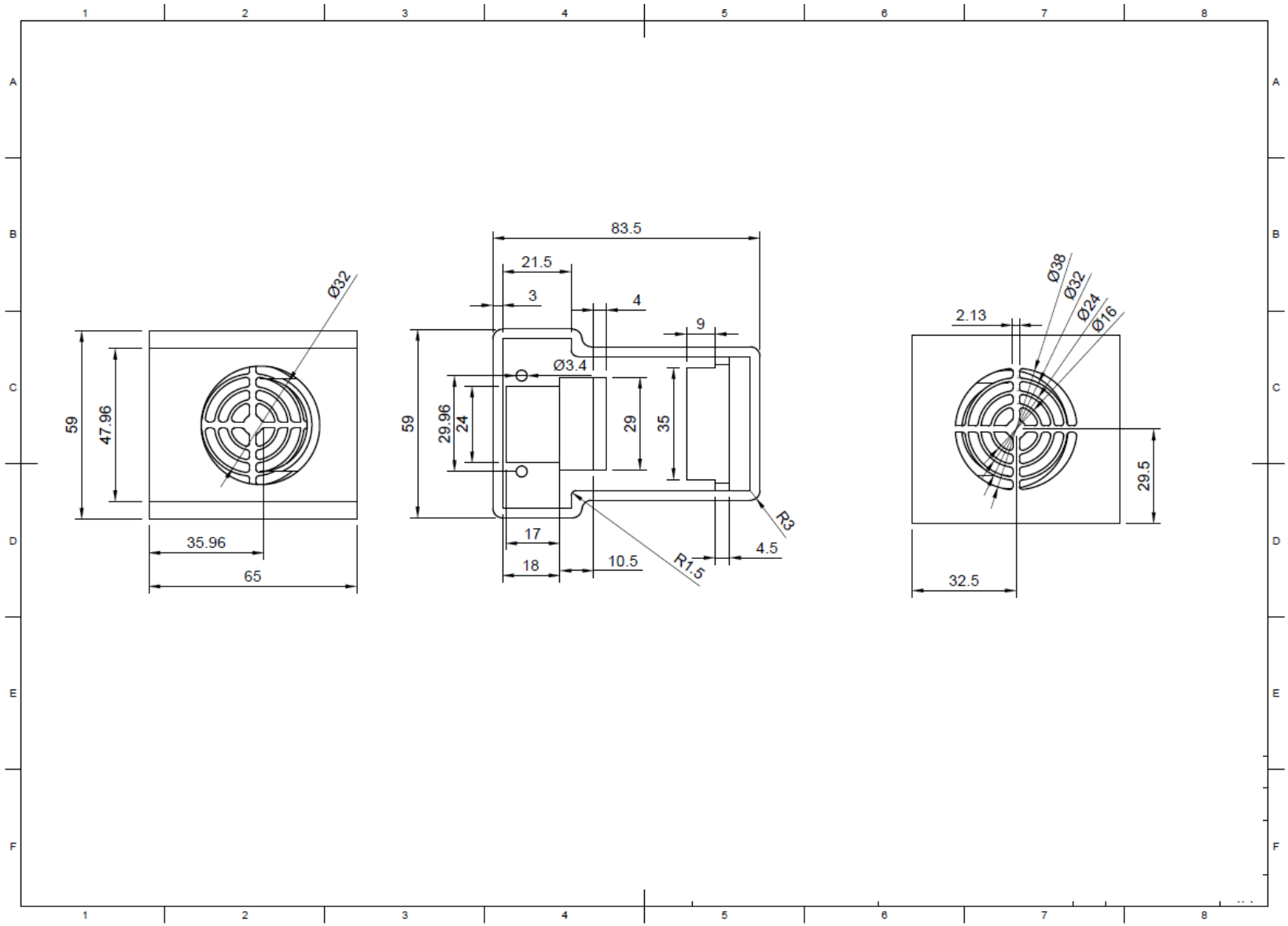
A lo largo de este anexo se presentan los planos de los siguientes componentes:

- Sujeción de la carcasa y pantalla.
- Mesa de recepción del cubo.
- Tapa de la carcasa de la cámara.
- Carcasa de la cámara.


15
260
128.99
Ø34
Ø110
5
Ø145
72.94
45.2
Ø50
54
Ø8
8
24
60
90
17.4
Ø3.4
33.1
89.98


151
260
R4
35
280
6
5
6
75
25
55
45
Ø3
Ø14
240
R6


39.98
R14.5
R17.5
15
5
71.5
10.5
14
40
59
R3
29
83
25
R3
47.96


Ø32
59
47.96
35.96
65
83.5
21.5
3
4
9
Ø3.4
59
29.96
24
29
35
17
18
10.5
R1.5
4.5
R3
Ø38
Ø32
Ø24
Ø16
2.13
29.5
32.5

ANEXO VI: PROGRAMACIÓN

En el anexo VI se presenta el código de Matlab y RobotStudio añadido y modificado para la realización de esta aplicación.

CÓDIGO DEL ROBOT YUMI

Brazo izquierdo

En primer lugar se presentan las partes del código de RobotStudio que han sido añadidas o modificadas.

En este caso se han modificado las posiciones para la toma de fotos y la forma en la que realizar el cambio de posición de una foto a otra.

T_ROB_L/Garazi:

[...]

```
        PROC Foto1()
    MoveJ Apro_Coger_CuboL,v200,z100,Servo_L\WObj:=WO_Cubo;
          MoveJ Tomar_Cubo,v200,z100,Servo_L\WObj:=WO_Cubo;
          waitTime 1;
          g_Gripin;
    GripLoad cubo3;
    Waittime 4;
          MoveJ Tomar_Fotos3,v50,z100,Servo_L\WObj:=WO_Cubo;
          waitTime 1;
        ENDPROC

        PROC Foto2()
          MoveL Tomar_Fotos2,v20,z100,Servo_L\WObj:=WO_Cubo;
          waitTime 1;
        ENDPROC
        PROC Foto3()
          MoveJ Tomar_Fotos,v20,z100,Servo_L\WObj:=WO_Cubo;
          waitTime 1;
        ENDPROC

  PROC Principal()

  WaitSyncTask sync_RL_A, ListaTareas;
    !Crear comunicacion
    SocketCreate server1;
    SocketBind server1, "192.168.125.1",14043;
    SocketListen server1;
    SocketAccept server1,client1;

  Foto1;
   SocketSend client1,\str:="D";
  !recivir mensaje
  SocketReceive client1,\RawData:=data;
  UnpackRawBytes data,1,fotos\Ascii:=2;
  oki:=StrtoVal(fotos,CuboTres);
 WaitUntil CuboTres=0 OR CuboTres=1;
```

```
Foto2;
 Waittime 3;

Foto3;
!enviar mensaje al cliente
SocketSend client1,\str:="F";
!recivir mensaje
SocketReceive client1,\RawData:=data;
UnpackRawBytes data,1,fotos\Ascii:=2;
oki:=StrtoVal(fotos,numfoto);
WaitUntil numfoto=7;
waitTime 1;

!Cerrar comunicacion
SocketClose server1;
SocketClose client1;
WaitSyncTask sync_RL_B, ListaTareas;
MoveJ Agarre_L_X,v30,z100,Servo_L\WObj:=WO_Cubo;
waittime 5;
g_GripOut;
waittime 1;
MoveJ Apro_Agarre_L_X,v30,z100,Servo_L\WObj:=WO_Cubo;
MoveJ Apro_L_1,v30,z100,Servo_L\WObj:=WO_Cubo;
waittime 1;
```

[…]

Brazo derecho

A continuación se presenta el código para el brazo derecho del robot realizado en RobotStudio.

Por un lado se han modificado las coordenadas para realizar la toma del cubo de la mesa, las posiciones para la toma de fotos y la forma en la que realizar el cambio de posición de una foto a otra.

T_ROB_R/Garazi:

[…]

```
PROC Coger_Cubo_Mesa()
    g_Init \maxSpd:=25 \holdForce:=1 \Calibrate; ! Calibracion de la pinza, indicación de la fuerza máxima en N y velocidad máxima en mm/s
    Pos_Cubo:=[[PosCuboY*10+287,115-PosCuboX*10,275],[0,orient6,orient8,0],[1,-2,1,4],[-145.293,9E+09,9E+09,9E+09,9E+09,9E+09]];
    Pos_Cubo.Trans:=[PosCuboY*10+287,115-PosCuboX*10,275];
    MoveJ Pos_Cubo,v200,fine,Servo_R;
    g_GripOut \holdForce:=10;
    waittime 1;

    Pos_Cubo_0:=[[PosCuboY*10+287,115-PosCuboX*10,200],[0,orient6,orient8,0],[1,-2,1,4],[-145.293,9E+09,9E+09,9E+09,9E+09,9E+09]];
    Pos_Cubo_0.Trans:=[PosCuboY*10+287,115-PosCuboX*10,200];
    MoveJ Pos_Cubo_0,v200,fine,Servo_R;
    g_GripIn;
    g_GripOut;

    WaitTime 1;
    MoveJ Pos_Cubo,v200,fine,Servo_R;
```

```
        WaitTime 1;

        Pos_Cubo_2:=[[PosCuboY*10+287,115-PosCuboX*10,275],[0,Orient2,orient4,0],[1,-2,1,4],[-
145.293,9E+09,9E+09,9E+09,9E+09,9E+09]];
        Pos_Cubo_2.Trans:=[PosCuboY*10+287,115-PosCuboX*10,275];
        MoveJ Pos_Cubo_2,v200,fine,Servo_R;
        Waittime 1;

        Pos_Cubo_3:=[[PosCuboY*10+287,115-PosCuboX*10,235],[0,Orient2,orient4,0],[1,-2,1,4],[-
145.293,9E+09,9E+09,9E+09,9E+09,9E+09]];
        Pos_Cubo_3.Trans:=[PosCuboY*10+287,115-PosCuboX*10,235];
        MoveJ Pos_Cubo_3,v200,fine,Servo_R;

        Pos_Cubo_1:=[[PosCuboY*10+287,115-PosCuboX*10,200],[0,Orient2,orient4,0],[1,-2,1,4],[-
145.293,9E+09,9E+09,9E+09,9E+09,9E+09]];
        Pos_Cubo_1.Trans:=[PosCuboY*10+287,115-PosCuboX*10,200];
        MoveJ Pos_Cubo_1,v200,fine,Servo_R;
        Waittime 1;
        g_gripin;

    ENDPROC


    PROC Principal()
        g_Init \maxSpd:=10 \ holdForce:=5 \Calibrate; ! Calibracion de la pinza, indicación de la fuerza máxima en N y
velocidad máxima en mm/s
        waitTime 1;
        g_GripOut;

  ! IF A = 1 THEN
        !Crear comunicacion
        SocketCreate server1;
        !SocketBind server, "10.109.253.189",14044;
        SocketBind server1, "192.168.125.1",14044;
        SocketListen server1;
        SocketAccept server1,client1;

    Foto1;
    SocketReceive client1,\RawData:=data;
    UnpackRawBytes data,1,foto\Ascii:=2;
    oki:=StrtoVal(foto,CuboTres);


    Foto2;
    Waittime 3;
     !enviar mensaje al cliente

    Foto3;
    SocketSend client1,\str:="C";
    !recivir mensaje
   SocketReceive client1,\RawData:=data;
    UnpackRawBytes data,1,foto\Ascii:=2;
    oki:=StrtoVal(foto,numfoto);

     WaitUntil numfoto=4;

    SocketSend client1,\str:="Z";

    !Cerrar comunicacion
    SocketClose server1;
    SocketClose client1;
```

```
WaitSyncTask sync_RL_A, ListaTareas;

waittime 2;
Foto4;
 WaitSyncTask sync_RL_B, ListaTareas;
 waitTime 3;

        PROC Foto1()
  g_Gripin \holdForce:=10;
  waitTime 1;
  GripLoad cubo3;
          waitTime 1;
  MoveJ Pos_Cubo_2,v200,fine,Servo_R;
  MoveJ Tomar_Fotos11,v300,z100,Servo_R\WObj:=wobj0;
          MoveJ Tomar_Fotos,v300,z100,Servo_R\WObj:=wobj0;
  waittime 2;
       ENDPROC
       PROC Foto2()
   WaitTime 2;
          MoveL Tomar_Fotos2,v20,z100,Servo_R\WObj:=wobj0;
  waittime 2;
       ENDPROC
       PROC Foto3()
                MoveL TomarFoto3,v20,z100,Servo_R\WObj:=wobj0;
  waittime 2;

       ENDPROC
       PROC Foto4()
  waitTime 2;
  g_GripOut;
          WaitTime 0.2;
  MoveJ Apro_Fotos,v200,z100,Servo_R\WObj:=wobj0;
                MoveJ Apro_R5,v200,z100,Servo_R\WObj:=wobj0;
                waitTime 1;
          waitTime 1;
       ENDPROC

ENDMODULE
```

[…]

CÓDIGO DE MATLAB

El código de Matlab consta de un programa principal, mediante el cual se les va llamando a los diferentes subprogramas. A lo largo de este programa principal se han realizado varias modificaciones y se han añadido diferentes trozos de código. Se destacan los siguientes:

- Se ha añadido el código para que el robot diga las diferentes frases.
- Se ha modificado la secuencia para la toma de fotos.
- Se han modificado algunas de las matrices de las caras.

Programa_Principal:

```
%PROGRAMA FINAL
inicio=1;

while inicio > 0
```

```
conexion=1;
num=1;
Cubo3=49;
cubo_mesa=48;

%sonido
load('fs.mat');
load('Sonido1.mat');
%load('Sonido7.mat'); %INGLES
%load('Sonido13.mat');%EUSKERA
sound(audio1,fs);
%clear(audio1);
%sonido



while conexion > 0
    try
    tc=tcpip("192.168.125.1",14045);
    fopen(tc);
    conexion=0;
    catch
       inicio = 1;
    end
end


Copy_of_Posicion_Cubo;


while numItemMesa < 100
 fread(tc,num);
fwrite(tc,cubo_mesa);
Copy_of_Posicion_Cubo;
close all hidden
end

%sonido
load('fs.mat');
load('Sonido6.mat');
%load('Sonido12.mat'); %INGLES
%load('Sonido18.mat');%EUSKERA
sound(audio1,fs);
%clear(audio1);
%sonido

cubo_mesa=49;
fwrite(tc,cubo_mesa);
% pause(5);
fclose(tc);

Posicion_Cubo;


foto=50;%equivale a 2
%pause(5);

tc=tcpip("192.168.125.1",14044);
pause(20);
fopen(tc);

pause(5)
%Toma_Imagenes;
```

```
%pause(3);
vid = videoinput('winvideo', 1, 'RGB24_2560x1920');
%src = getselectedsource(vid);
vid.TriggerRepeat = 1

start(vid);

Cara1 = getsnapshot(vid);
CaraX=Cara1;
 Y=imrotate(X,90);
 Z=fliplr(Y);
Copy_2_of_Proc_CaraF;


if numItemRed == 0 & numItemBlue == 0 & numItemYellow ==0  & numItemGreen == 0 &
numItemOrange == 0
    Cubo3=48;
end
fwrite(tc,Cubo3);
pause(1);
close all

imshow(Z)



%foto=foto+1 %seria 3

pause(2);

Cara2 = getsnapshot(vid);
X=Cara2;
Y=imrotate(X,90);
Z=fliplr(Y);
imshow(Z)




foto=foto+2 % seria 4
num_foto=fread(tc);

Cara3 = getsnapshot(vid);

fwrite(tc,foto);
X=Cara3;
Y=imrotate(X,90);
Z=fliplr(Y);
imshow(Z)



fclose(tc);

tc=tcpip("192.168.125.1",14043);

fopen(tc);

foto=foto+1 %tiene que ser 5
num_foto=fread(tc);

Cara4 = getsnapshot(vid);
stop(vid);
```

```
fwrite(tc,Cubo3);
X=Cara4;
Y=imrotate(X,90);
Z=fliplr(Y);
imshow(Z)


pause(2);

Cara5 = getsnapshot(vid);
X=Cara5;
Y=imrotate(X,90);
Z=fliplr(Y);
Z2=imcrop(Z2,[665,457,1018,1011]);
imshow(Z)



foto=foto+2
num_foto=fread(tc);

Cara6 = getsnapshot(vid);
stop(vid);
fwrite(tc,foto);
X=Cara6;
Y=imrotate(X,90);
Z=fliplr(Y);

imshow(Z)


fclose(tc);

%-------------------------------------------------------------
Cubo3=Cubo3-48;
if Cubo3==0

%sonido
load('fs.mat');
load('Sonido3.mat');
%load('Sonido9.mat');%INGLES
%load('Sonido15.mat');%EUSKERA
sound(audio1,fs);
%clear(audio1);
%sonido

CaraX=Cara1;
Copy_of_Proc_CaraF;

Cara_D=Cara;
Cubo(:,:,6)=Cara_D;

CaraX=Cara2;
Copy_of_Proc_CaraF;
Cara_L(1,1)=Cara(3,1);Cara_L(2,1)=Cara(3,2);Cara_L(3,1)=Cara(3,3);
Cara_L(1,2)=Cara(2,1);Cara_L(2,2)=Cara(2,2);Cara_L(3,2)=Cara(2,3);
Cara_L(1,3)=Cara(1,1);Cara_L(2,3)=Cara(1,2);Cara_L(3,3)=Cara(1,3);
Cubo(:,:,4)=Cara_L;

CaraX=Cara3;
Copy_of_Proc_CaraF;

Cara_U=Cara;
```

```
Cubo(:,:,5)=Cara_U;

CaraX=Cara4;
Copy_of_Proc_CaraF;
Cara_F=Cara;
Cubo(:,:,1)=Cara_F;

CaraX=Cara5;
Copy_of_Proc_CaraF;
%Cara_R=Cara;
Cara_R(1,1)=Cara(1,3);Cara_R(2,1)=Cara(1,2);Cara_R(3,1)=Cara(1,1);
Cara_R(1,2)=Cara(2,3);Cara_R(2,2)=Cara(2,2);Cara_R(3,2)=Cara(2,1);
Cara_R(1,3)=Cara(3,3);Cara_R(2,3)=Cara(3,2);Cara_R(3,3)=Cara(3,1);
Cubo(:,:,2)=Cara_R;

CaraX=Cara6;
Copy_of_Proc_CaraF;
Cara_B(1,1)=Cara(3,3);Cara_B(2,1)=Cara(2,3);Cara_B(3,1)=Cara(1,3);
Cara_B(1,2)=Cara(3,2);Cara_B(2,2)=Cara(2,2);Cara_B(3,2)=Cara(1,2);
Cara_B(1,3)=Cara(3,1);Cara_B(2,3)=Cara(2,1);Cara_B(3,3)=Cara(1,1);
Cubo(:,:,3)=Cara_B;


elseif Cubo3==1

%sonido
load('fs.mat');
load('Sonido2.mat');
%load('Sonido8.mat');%INGLES
%load('Sonido14.mat');%EUSKERA
sound(audio1,fs);
%clear(audio1);
%sonido

CaraX=Cara1;
Proc_Cara_2x2;
Cara_D=Cara_2x2;
Cubo_2x2(:,:,6)=Cara_D;

CaraX=Cara2;
Proc_Cara_2x2;
Cara_L(1,1)=Cara_2x2(2,1);Cara_L(2,1)=Cara_2x2(2,2);
Cara_L(1,2)=Cara_2x2(1,1);Cara_L(2,2)=Cara_2x2(1,2);
Cubo_2x2(:,:,4)=Cara_L;

CaraX=Cara3;
Proc_Cara_2x2;
Cara_U=Cara_2x2;
Cubo_2x2(:,:,5)=Cara_U;

CaraX=Cara4;
Proc_Cara_2x2;
Cara_F=Cara_2x2;
Cubo_2x2(:,:,1)=Cara_F;


CaraX=Cara5;
Proc_Cara_2x2;
Cara_R(1,1)=Cara_2x2(1,2);Cara_R(2,1)=Cara_2x2(1,1);
Cara_R(1,2)=Cara_2x2(2,2);Cara_R(2,2)=Cara_2x2(2,1);
Cubo_2x2(:,:,2)=Cara_R;

CaraX=Cara6;
```

```
Proc_Cara_2x2;
Cara_B(1,1)=Cara_2x2(2,2);Cara_B(2,1)=Cara_2x2(1,2);
Cara_B(1,2)=Cara_2x2(2,1);Cara_B(2,2)=Cara_2x2(1,1);
Cubo_2x2(:,:,3)=Cara_B;

end

 %-------------------------------------------------------------------------
------------------------

 %%

  if Cubo3==0
R=Cubo;
sol = Solve45_2(R);

 elseif Cubo3==1
R=Cubo_2x2;
[rot sol] = Solve222(R);
 end

solmat=cell2mat(sol);
movimientosR=zeros(length(sol));
movimientosR=diag(movimientosR);
movimientosR=movimientosR';
solmat(length(solmat)+1)='W';

for i=1:length(solmat)-1

if solmat(i)=='F' & solmat(i+1) ~= 'F'  & solmat(i+1) ~= 'F2' & solmat(i+1) ~= 'R'
& solmat(i+1) ~= 'R2' & solmat(i+1) ~= 'B'  & solmat(i+1) ~= 'B2' & solmat(i+1) ~=
'L'  & solmat(i+1) ~= 'L2' & solmat(i+1) ~= 'U'  & solmat(i+1) ~= 'U2' &
solmat(i+1) ~= 'D'  & solmat(i+1) ~= 'D2' & solmat(i+1) ~= 'W'
movimientosR(i)=11;
elseif solmat(i)=='F' & solmat(i+1)=='2'
movimientosR(i)=12;
elseif solmat(i)=='F'
 movimientosR(i)=10;

elseif solmat(i)=='R' & solmat(i+1) ~= 'F'  & solmat(i+1) ~= 'F2' & solmat(i+1) ~=
'R'  & solmat(i+1) ~= 'R2' & solmat(i+1) ~= 'B'  & solmat(i+1) ~= 'B2' &
solmat(i+1) ~= 'L'  & solmat(i+1) ~= 'L2' & solmat(i+1) ~= 'U'  & solmat(i+1) ~=
'U2' & solmat(i+1) ~= 'D'  & solmat(i+1) ~= 'D2' & solmat(i+1) ~= 'W'
movimientosR(i)=21;
elseif solmat(i)=='R'  & solmat(i+1)=='2'
movimientosR(i)=22;
elseif solmat(i)=='R'
movimientosR(i)=20;

elseif solmat(i)=='B' & solmat(i+1) ~= 'F'  & solmat(i+1) ~= 'F2' & solmat(i+1) ~=
'R'  & solmat(i+1) ~= 'R2' & solmat(i+1) ~= 'B'  & solmat(i+1) ~= 'B2' &
solmat(i+1) ~= 'L'  & solmat(i+1) ~= 'L2' & solmat(i+1) ~= 'U'  & solmat(i+1) ~=
'U2' & solmat(i+1) ~= 'D'  & solmat(i+1) ~= 'D2' & solmat(i+1) ~= 'W'
movimientosR(i)=31;
elseif solmat(i)=='B' & solmat(i+1)=='2'
movimientosR(i)=32;
elseif solmat(i)=='B'
movimientosR(i)=30;

elseif solmat(i)=='L' & solmat(i+1) ~= 'F'  & solmat(i+1) ~= 'F2' & solmat(i+1) ~=
'R'  & solmat(i+1) ~= 'R2' & solmat(i+1) ~= 'B'  & solmat(i+1) ~= 'B2' &
solmat(i+1) ~= 'L'  & solmat(i+1) ~= 'L2' & solmat(i+1) ~= 'U'  & solmat(i+1) ~=
'U2' & solmat(i+1) ~= 'D'  & solmat(i+1) ~= 'D2' & solmat(i+1) ~= 'W'
```

```
movimientosR(i)=41;
elseif solmat(i)=='L'  & solmat(i+1)=='2'
movimientosR(i)=42;
elseif solmat(i)=='L'
movimientosR(i)=40;

elseif solmat(i)=='U' & solmat(i+1) ~= 'F'  & solmat(i+1) ~= 'F2' & solmat(i+1) ~=
'R'  & solmat(i+1) ~= 'R2' & solmat(i+1) ~= 'B'  & solmat(i+1) ~= 'B2' &
solmat(i+1) ~= 'L'  & solmat(i+1) ~= 'L2' & solmat(i+1) ~= 'U'  & solmat(i+1) ~=
'U2' & solmat(i+1) ~= 'D'  & solmat(i+1) ~= 'D2' & solmat(i+1) ~= 'W'
movimientosR(i)=51;
elseif solmat(i)=='U' & solmat(i+1)=='2'
movimientosR(i)=52;
elseif solmat(i)=='U'
movimientosR(i)=50;

elseif solmat(i)=='D' & solmat(i+1) ~= 'F'  & solmat(i+1) ~= 'F2' & solmat(i+1) ~=
'R'  & solmat(i+1) ~= 'R2' & solmat(i+1) ~= 'B'  & solmat(i+1) ~= 'B2' &
solmat(i+1) ~= 'L'  & solmat(i+1) ~= 'L2' & solmat(i+1) ~= 'U'  & solmat(i+1) ~=
'U2' & solmat(i+1) ~= 'D'  & solmat(i+1) ~= 'D2' & solmat(i+1) ~= 'W'
movimientosR(i)=61;
elseif solmat(i)=='D' & solmat(i+1)=='2'
movimientosR(i)=62;
elseif solmat(i)=='D'
movimientosR(i)=60;

end
end

noblancos=0;
for j=1:length(movimientosR)
if  movimientosR(j)~=0
noblancos=noblancos+1;
end
end

movimientosR1=zeros(noblancos);
movimientosR1=diag(movimientosR1);
movimientosR1(1)=99;
k=2;

if Cubo3==1
Rotaciones;
movimientosR1(2)=rotR(1);movimientosR1(3)=rotR(2);
k=4;
end

for j=1:length(movimientosR)
if  movimientosR(j)~=0
movimientosR1(k)=movimientosR(j);
k=k+1;
end
end

z=1;
for i=1:length(movimientosR1)
if  movimientosR1(i)~=70 & movimientosR1(i)~=80 & movimientosR1(i)~=90
    movimientosR3(z)=movimientosR1(i);
    z=z+1;
end
end

movimientosR1=movimientosR3
```

```
% %------------------------------------------------------
%%

num_mov=length(movimientosR1);
datos=mat2str(num_mov);

if length(movimientosR1) > 26
    j=1;
    for i=27:length(movimientosR1)
        movimientosR2(j)=movimientosR1(i);
        j=j+1;
    end
    for i=length(movimientosR1):-1:27
        movimientosR1(i)=[];
    end
end

if num_mov < 27
datos1=mat2str(movimientosR1);

tc=tcpip("192.168.125.1",14044);
pause(5);
fopen(tc);
pause(1);
fwrite(tc,datos);
pause(1);
fwrite(tc,datos1);

else
datos1=mat2str(movimientosR1);
datos2=mat2str(movimientosR2);

tc=tcpip("192.168.125.1",14044);

fopen(tc);

fwrite(tc,datos);
pause(1);
fwrite(tc,datos1);
pause(1);
fwrite(tc,datos2);
pause(2);
end

pause(10);
%Brazo Izquierdo

if num_mov < 27

datos1=mat2str(movimientosR1);

tc=tcpip("192.168.125.1",14043);

fopen(tc);

pause(1);
fwrite(tc,datos);
pause(1);
fwrite(tc,datos1);
pause(2);

else
```

```
datos1=mat2str(movimientosR1);
datos2=mat2str(movimientosR2);

tc=tcpip("192.168.125.1",14043);

fopen(tc);

fwrite(tc,datos);
pause(1);
fwrite(tc,datos1);
pause(1);
fwrite(tc,datos2);
end

fclose(tc);

 close all hidden
 close all

   Gen_Caras;

%sonido
load('fs.mat');
load('Sonido4.mat');
%load('Sonido10.mat');%INGLES
%load('Sonido16.mat');%EUSKERA
sound(audio1,fs);
%clear(audio1);
%sonido

   Resolver_ApartirdeR;

%sonido
load('fs.mat');
load('Sonido5.mat');
%load('Sonido11.mat');%INGLES
%load('Sonido17.mat');%EUSKERA
sound(audio1,fs);
%clear(audio1);
%sonido

close all hidden
 close all
 clear all


inicio=1;

end
```

Subprogramas:

Mediante el programa principal se llama a diferentes subprogramas. A continuación se muestran los subprogramas modificados o en los que se ha añadido parte de código.

Copy_of_Posicion_Cubo:

```
vid = videoinput('winvideo', 1, 'RGB24_2560x1920');
vid.TriggerRepeat = 1
```

```
start(vid);


Mesa11 = getsnapshot(vid);
stop(vid);
X=Mesa11;
Y=imrotate(X,90);
Z=fliplr(Y);

Z2=imcrop(Z,[19 506 1883 1320]);

I = rgb2hsv(Z2);

% Define thresholds for channel 1 based on histogram settings
channel1Min = 0.000;
channel1Max = 1.000;

% Define thresholds for channel 2 based on histogram settings
channel2Min = 0.350;
channel2Max = 1.000;

% Define thresholds for channel 3 based on histogram settings
channel3Min = 0.000;
channel3Max = 1.000;


% Create mask based on chosen histogram thresholds
sliderBW = (I(:,:,1) >= channel1Min ) & (I(:,:,1) <= channel1Max) & ...
    (I(:,:,2) >= channel2Min ) & (I(:,:,2) <= channel2Max) & ...
    (I(:,:,3) >= channel3Min ) & (I(:,:,3) <= channel3Max);
BW = sliderBW;

BW = medfilt2(BW);
se=strel('square',15);
BW=imopen(BW,se);
```

[…]

Posicion_Cubo:

```
vid = videoinput('winvideo', 1, 'RGB24_2560x1920');
vid.TriggerRepeat = 1

start(vid);

Mesa11 = getsnapshot(vid);
X=Mesa11;
Y=imrotate(X,90);
Z=fliplr(Y);

Z2=imcrop(Z,[19 506 1883 1320]);

I = rgb2hsv(Z2);

% Define thresholds for channel 1 based on histogram settings
channel1Min = 0.000;
channel1Max = 1.000;

% Define thresholds for channel 2 based on histogram settings
channel2Min = 0.350;
channel2Max = 1.000;
```

```
% Define thresholds for channel 3 based on histogram settings
channel3Min = 0.000;
channel3Max = 1.000;


% Create mask based on chosen histogram thresholds
sliderBW = (I(:,:,1) >= channel1Min ) & (I(:,:,1) <= channel1Max) & ...
    (I(:,:,2) >= channel2Min ) & (I(:,:,2) <= channel2Max) & ...
    (I(:,:,3) >= channel3Min ) & (I(:,:,3) <= channel3Max);
BW = sliderBW;

BW = medfilt2(BW);
se=strel('square',15);
BW=imopen(BW,se);
```

[…]

Calculo_pos_Cubo:

```
VecY= [1288 1220 1161 1099 1043 984 927 873 817 763 716 662 613 562 515 473 427 387
347 299 262 220 188 147 105 70 29];
j=1;
while posCuboY_Calculo<VecY(j) & j<27
    j=j+1;
end

Ya=j;


distX=0.44449; %lo que aumenta la distancia entre columnas de una fila a otra
distX1=60;%distancia entre columnas arriba
Xini= [265 257 247 244 236 228 218 212 202 192 186 178 167 155 147 137 121 116 108
95 86 76 65 53 41 28 13];%los valores desde la parte izquierda de la foto hasta las
diferentes filas de la mesa
for k=1:length(Xini)
VecX1(1)=Xini(k);
for i=2:length(VecY)
    Xini(k)=Xini(k)+distX1;
    VecX1(i)=Xini(k);
end
distX1=distX1+distX;
VecX2(:,:,k)=VecX1;
end

VecXa=VecX2(:,:,29-Ya);
j=1;
while posCuboX_Calculo>VecXa(j) & j<25
    j=j+1;
end
Xa=j+1;

Xa2=VecXa(j);

Xa;Ya;
Xb=Xa-1;
Yb=Ya-1;

Y1_Cubo=Yb+ (posCuboY_Calculo-VecY(Yb))/(VecY(Ya)-VecY(Yb))-1;
X1_Cubo=Xb+ (posCuboX_Calculo-VecXa(Xb))/(VecXa(Xa)-VecXa(Xb))-1;
```

Copy_2_of_Proc_CaraF:

```
%PROCESAMIENTO CARA

Copy_2_of_Proc_Rojo_Cara1;
Copy_2_of_Proc_Azul_Cara;
Copy_2_of_Proc_Naranja_Cara;
Copy_2_of_Proc_Verde_Cara;
Copy_2_of_Proc_Amarillo_Cara;
```

Copy_2_of_Proc_Rojo_Cara1:

```
X=CaraX;
Y=imrotate(X,90);
Z=fliplr(Y);
Z=imcrop(Z,[665,457,1018,1011]);
% Convert RGB image to chosen color space
I = rgb2hsv(Z);

% Define thresholds for channel 1 based on histogram settings
channel1Min = 0.944;
channel1Max = 0.020;

% Define thresholds for channel 2 based on histogram settings
channel2Min = 0.336;
channel2Max = 1.000;

% Define thresholds for channel 3 based on histogram settings
channel3Min = 0;
channel3Max = 1.000;

% Create mask based on chosen histogram thresholds
sliderBW = ( (I(:,:,1) >= channel1Min) | (I(:,:,1) <= channel1Max) ) & ...
    (I(:,:,2) >= channel2Min ) & (I(:,:,2) <= channel2Max) & ...
    (I(:,:,3) >= channel3Min ) & (I(:,:,3) <= channel3Max);
BW = sliderBW;

[L,Ne] = bwlabel(BW);

figure
imshow(label2rgb(BW ))

prop=regionprops(L);
numItemRed =0;
posItemRed =zeros(9,2);
posItemRedX=0;
posItemRedY=0;
hold on

for n=1:length(prop);
    x=prop(n).Centroid(1);
    y=prop(n).Centroid(2);
    if prop(n).Area> 50000
    rectangle('Position',prop(n).BoundingBox,'EdgeColor','Red','LineWidth',2);
   % plot(x,y,'*');
    numItemRed =numItemRed+1;
    posItemRed (numItemRed)=(x);
    posItemRed (numItemRed,2)=(y);

    posItemRedX (numItemRed)= (x);
```

```
    posItemRedY (numItemRed)= (y);
    end
end

title (['Hay ' num2str(numItemRed )   ' objetos rojos'])
hold off
```

Copy_2_of_Proc_Azul_Cara:

```
X=CaraX;
Y=imrotate(X,90);
Z=fliplr(Y);
Z=imcrop(Z,[665,457,1018,1011]);
% Convert RGB image to chosen color space
I = rgb2hsv(Z);

% Define thresholds for channel 1 based on histogram settings
channel1Min = 0.522;
channel1Max = 0.738;

% Define thresholds for channel 2 based on histogram settings
channel2Min = 0.497;
channel2Max = 1;

% Define thresholds for channel 3 based on histogram settings
channel3Min = 0.000;
channel3Max = 1.000;

% Create mask based on chosen histogram thresholds
sliderBW = (I(:,:,1) >= channel1Min ) & (I(:,:,1) <= channel1Max) & ...
    (I(:,:,2) >= channel2Min ) & (I(:,:,2) <= channel2Max) & ...
    (I(:,:,3) >= channel3Min ) & (I(:,:,3) <= channel3Max);
BW = sliderBW;


[L,Ne] = bwlabel(BW);

figure
imshow(label2rgb(BW))

prop=regionprops(L);
numItemBlue=0;
posItemBlue=zeros(9,2);
posItemBlueX=0;
posItemBlueY=0;
hold on

for n=1:length(prop);

    x=prop(n).Centroid(1);
    y=prop(n).Centroid(2);
    if prop(n).Area> 50000
    rectangle('Position',prop(n).BoundingBox,'EdgeColor','blue','LineWidth',2);
    plot(x,y,'*');
    numItemBlue=numItemBlue+1;
    posItemBlue(numItemBlue)=(x);
    posItemBlue(numItemBlue,2)=(y);

    posItemBlueX(numItemBlue)= (x);
    posItemBlueY(numItemBlue)= (y);
    end
```

```
end

title (['Hay ' num2str(numItemBlue)  ' objetos azules'])
hold off
```

Copy_2_of_Proc_Naranja_Cara:

```
X=CaraX;
Y=imrotate(X,90);
Z=fliplr(Y);
Z=imcrop(Z,[665,457,1018,1011]);
% Convert RGB image to chosen color space
I = rgb2hsv(Z);

% Define thresholds for channel 1 based on histogram settings
channel1Min = 0.016;
channel1Max = 0.078;

% Define thresholds for channel 2 based on histogram settings
channel2Min = 0.721;
channel2Max = 1;

% Define thresholds for channel 3 based on histogram settings
channel3Min = 0.000;
channel3Max = 1.000;


% Create mask based on chosen histogram thresholds
sliderBW = (I(:,:,1) >= channel1Min ) & (I(:,:,1) <= channel1Max) & ...
    (I(:,:,2) >= channel2Min ) & (I(:,:,2) <= channel2Max) & ...
    (I(:,:,3) >= channel3Min ) & (I(:,:,3) <= channel3Max);
BW = sliderBW;

[L,Ne] = bwlabel(BW);


prop=regionprops(L);
numItemOrange=0;
posItemOrange =zeros(9,2);
posItemOrangeX=0;
posItemOrangeY=0;

for n=1:length(prop);
    x=prop(n).Centroid(1);
    y=prop(n).Centroid(2);
    if prop(n).Area> 50000

rectangle('Position',prop(n).BoundingBox,'EdgeColor',[0.91,0.41,0.17],'LineWidth',2
);
    %plot(x,y,'*');
    numItemOrange =numItemOrange+1;
    posItemOrange (numItemOrange)=(x);
    posItemOrange (numItemOrange,2)=(y);

    posItemOrangeX (numItemOrange)= (x);
    posItemOrangeY (numItemOrange)= (y);
    end
end

title (['Hay ' num2str(numItemOrange )  ' objetos naranjas'])
hold off
```

Copy_2_of_Proc_Verde_Cara:

```
X=CaraX;
Y=imrotate(X,90);
Z=fliplr(Y);
Z=imcrop(Z,[665,457,1018,1011]);
% Convert RGB image to chosen color space
I = rgb2hsv(Z);

% Define thresholds for channel 1 based on histogram settings
channel1Min = 0.226;
channel1Max = 0.491;

% Define thresholds for channel 2 based on histogram settings
channel2Min = 0.392;
channel2Max = 1;

% Define thresholds for channel 3 based on histogram settings
channel3Min = 0;
channel3Max = 1;

% Create mask based on chosen histogram thresholds
sliderBW = (I(:,:,1) >= channel1Min ) & (I(:,:,1) <= channel1Max) & ...
    (I(:,:,2) >= channel2Min ) & (I(:,:,2) <= channel2Max) & ...
    (I(:,:,3) >= channel3Min ) & (I(:,:,3) <= channel3Max);
BW = sliderBW;


[L,Ne] = bwlabel(BW);


prop=regionprops(L);
numItemGreen=0;
posItemGreen=zeros(9,2);
posItemGreenX=0;
posItemGreenY=0;
%hold on

for n=1:length(prop);
    x=prop(n).Centroid(1);
    y=prop(n).Centroid(2);
    if prop(n).Area> 50000
    rectangle('Position',prop(n).BoundingBox,'EdgeColor','green','LineWidth',2);
    %plot(x,y,'*');
    numItemGreen=numItemGreen+1;
    posItemGreen(numItemGreen)=(x);
    posItemGreen(numItemGreen,2)=(y);

    posItemGreenX (numItemGreen)= (x);
    posItemGreenY (numItemGreen)= (y);
    end
end
```

Copy_2_of_Proc_Amarillo_Cara:

```
X=CaraX;
Y=imrotate(X,90);
Z=fliplr(Y);
Z=imcrop(Z,[665,457,1018,1011]);
% Convert RGB image to chosen color space
I = rgb2hsv(Z);
```

```
% Define thresholds for channel 1 based on histogram settings
channel1Min = 0.104;
channel1Max = 0.209;

% Define thresholds for channel 2 based on histogram settings
channel2Min = 0.562;
channel2Max = 1.000;

% Define thresholds for channel 3 based on histogram settings
channel3Min = 00;
channel3Max = 1;


% Create mask based on chosen histogram thresholds
sliderBW = (I(:,:,1) >= channel1Min ) & (I(:,:,1) <= channel1Max) & ...
    (I(:,:,2) >= channel2Min ) & (I(:,:,2) <= channel2Max) & ...
    (I(:,:,3) >= channel3Min ) & (I(:,:,3) <= channel3Max);
BW = sliderBW;


[L,Ne] = bwlabel(BW);

prop=regionprops(L);
numItemYellow =0;
posItemYellow = zeros(9,2);
posItemYellowX=0;
posItemYellowY=0;
hold on

for n=1:length(prop);
    x=prop(n).Centroid(1);
    y=prop(n).Centroid(2);
    if prop(n).Area> 50000
    rectangle('Position',prop(n).BoundingBox,'EdgeColor','Yellow','LineWidth',2);
%     plot(x,y,'*');
    numItemYellow =numItemYellow+1;
    posItemYellow (numItemYellow)=(x);
    posItemYellow (numItemYellow,2)=(y);

    posItemYellowX (numItemYellow)= (x);
    posItemYellowY (numItemYellow)= (y);
    end
end
```

Copy_of_Proc_CaraF

```
%PROCESAMIENTO CARA

Cara=[5 5 5;5 5 5;5 5 5];

Copy_of_Proc_Rojo_Cara1;
Pos_Rojo_Cara;

Copy_of_Proc_Azul_Cara;
Pos_Azul_Cara;

Copy_of_Proc_Naranja_Cara;
Pos_Naranja_Cara;

Copy_of_Proc_Verde_Cara;
Pos_Verde_Cara;
```

```
Copy_of_Proc_Amarillo_Cara;
Pos_Amarillo_Cara;
```

Copy_of_Proc_Rojo_Cara1:

```
X=CaraX;
Y=imrotate(X,90);
Z=fliplr(Y);
Z=imcrop(Z,[665,457,1018,1011]);%[644,514,974,933]
% Convert RGB image to chosen color space
I = rgb2hsv(Z);

% Define thresholds for channel 1 based on histogram settings
channel1Min = 0.944;
channel1Max = 0.020;

% Define thresholds for channel 2 based on histogram settings
channel2Min = 0.336;
channel2Max = 1.000;

% Define thresholds for channel 3 based on histogram settings
channel3Min = 0.000;
channel3Max = 1.000;

% Create mask based on chosen histogram thresholds
sliderBW = ( (I(:,:,1) >= channel1Min) | (I(:,:,1) <= channel1Max) ) & ...
    (I(:,:,2) >= channel2Min ) & (I(:,:,2) <= channel2Max) & ...
    (I(:,:,3) >= channel3Min ) & (I(:,:,3) <= channel3Max);
BW = sliderBW;

[L,Ne] = bwlabel(BW);

figure
imshow(label2rgb(BW ))

prop=regionprops(L);
numItemRed =0;
posItemRed =zeros(9,2);
posItemRedX=0;
posItemRedY=0;
hold on

for n=1:length(prop);
    x=prop(n).Centroid(1);
    y=prop(n).Centroid(2);
    if prop(n).Area> 10000
    rectangle('Position',prop(n).BoundingBox,'EdgeColor','Red','LineWidth',2);
    plot(x,y,'*');
    numItemRed =numItemRed+1;
    posItemRed (numItemRed)=(x);
    posItemRed (numItemRed,2)=(y);

    posItemRedX (numItemRed)= (x);
    posItemRedY (numItemRed)= (y);
    end
end

title (['Hay ' num2str(numItemRed )  ' objetos rojos'])
hold off
```

Pos_Rojo_Cara:

```
if posItemRedX~=0

for i=1:length(posItemRedX)
          %CARA F
if  posItemRedX(i) <= 300;
    if  posItemRedY(i) < 375;
    Cara(1,1)=1;
    end
    if posItemRedY(i) > 375 & posItemRedY(i) < 625;
    Cara(2,1)=1;
    end
    if posItemRedY(i) > 625;
    Cara(3,1)=1;
    end

end

if posItemRedX(i) > 300 & posItemRedX(i) < 600;
    if  posItemRedY(i) < 375;
    Cara(1,2)=1;
    end
    if posItemRedY(i) > 375 & posItemRedY(i) < 625;
    Cara(2,2)=1;
    end
    if posItemRedY(i) > 625;
    Cara(3,2)=1;
    end

end

if posItemRedX(i) >=600;
    if  posItemRedY(i) < 375;
    Cara(1,3)=1;
    end
    if posItemRedY(i) > 375 & posItemRedY(i) < 625;
    Cara(2,3)=1;
    end
    if posItemRedY(i) > 625;
    Cara(3,3)=1;
    end


end
  %--------------------------------------------------------------------------

end
end
```

Copy_of_Proc_Azul_Cara:

```
X=CaraX;
Y=imrotate(X,90);
Z=fliplr(Y);
Z=imcrop(Z,[665,457,1018,1011]);
% Convert RGB image to chosen color space
I = rgb2hsv(Z);

% Define thresholds for channel 1 based on histogram settings
```

```
channel1Min = 0.522;
channel1Max = 0.738;

% Define thresholds for channel 2 based on histogram settings
channel2Min = 0.497;
channel2Max = 1;

% Define thresholds for channel 3 based on histogram settings
channel3Min = 0.000;
channel3Max = 1.000;

% Create mask based on chosen histogram thresholds
sliderBW = (I(:,:,1) >= channel1Min ) & (I(:,:,1) <= channel1Max) & ...
    (I(:,:,2) >= channel2Min ) & (I(:,:,2) <= channel2Max) & ...
    (I(:,:,3) >= channel3Min ) & (I(:,:,3) <= channel3Max);
BW = sliderBW;
[L,Ne] = bwlabel(BW);

imshow(label2rgb(BW))

prop=regionprops(L);
numItemBlue=0;
posItemBlue=zeros(9,2);
posItemBlueX=0;
posItemBlueY=0;
hold on

for n=1:length(prop);

    x=prop(n).Centroid(1);
    y=prop(n).Centroid(2);
    if prop(n).Area> 10000
    rectangle('Position',prop(n).BoundingBox,'EdgeColor','blue','LineWidth',2);
    plot(x,y,'*');
    numItemBlue=numItemBlue+1;
    posItemBlue(numItemBlue)=(x);
    posItemBlue(numItemBlue,2)=(y);

    posItemBlueX(numItemBlue)= (x);
    posItemBlueY(numItemBlue)= (y);
    end
end

title (['Hay ' num2str(numItemBlue)  ' objetos azules'])
hold off
```

Pos_Azul_Cara:

```
if posItemBlueX~=0

for i=1:length(posItemBlueX)
        %CARA F
if  posItemBlueX(i) <= 300;
    if  posItemBlueY(i) < 375;
    Cara(1,1)=2;
    end
    if posItemBlueY(i) > 375 & posItemBlueY(i) < 625;
    Cara(2,1)=2;
    end
    if posItemBlueY(i) > 625;
    Cara(3,1)=2;
```

```
    end

end

if posItemBlueX(i) > 300 & posItemBlueX(i) < 600;
    if  posItemBlueY(i) < 375;
    Cara(1,2)=2;
    end
    if posItemBlueY(i) > 375 & posItemBlueY(i) < 625;
    Cara(2,2)=2;
    end
    if posItemBlueY(i) > 625;
    Cara(3,2)=2;
    end

end

if posItemBlueX(i) >=600;
    if  posItemBlueY(i) < 375;
    Cara(1,3)=2;
    end
    if posItemBlueY(i) > 375 & posItemBlueY(i) < 625;
    Cara(2,3)=2;
    end
    if posItemBlueY(i) > 625;
    Cara(3,3)=2;
    end


end
  %--------------------------------------------------------------------------

end
end
```

Copy_of_Proc_Naranja_Cara:

```
X=CaraX;
Y=imrotate(X,90);
Z=fliplr(Y);
Z=imcrop(Z,[665,457,1018,1011]);
% Convert RGB image to chosen color space
I = rgb2hsv(Z);

% Define thresholds for channel 1 based on histogram settings
channel1Min = 0.016;
channel1Max = 0.078;

% Define thresholds for channel 2 based on histogram settings
channel2Min = 0.721;
channel2Max = 1;

% Define thresholds for channel 3 based on histogram settings
channel3Min = 0.000;
channel3Max = 1.000;


% Create mask based on chosen histogram thresholds
sliderBW = (I(:,:,1) >= channel1Min ) & (I(:,:,1) <= channel1Max) & ...
    (I(:,:,2) >= channel2Min ) & (I(:,:,2) <= channel2Max) & ...
    (I(:,:,3) >= channel3Min ) & (I(:,:,3) <= channel3Max);
```

```
BW = sliderBW;


[L,Ne] = bwlabel(BW);

figure
imshow(label2rgb(BW ))

prop=regionprops(L);
numItemOrange=0;
posItemOrange =zeros(9,2);
posItemOrangeX=0;
posItemOrangeY=0;
hold on

for n=1:length(prop);
    x=prop(n).Centroid(1);
    y=prop(n).Centroid(2);
    if prop(n).Area> 10000

rectangle('Position',prop(n).BoundingBox,'EdgeColor',[0.91,0.41,0.17],'LineWidth',2
);
    plot(x,y,'*');
    numItemOrange =numItemOrange+1;
    posItemOrange (numItemOrange)=(x);
    posItemOrange (numItemOrange,2)=(y);

    posItemOrangeX (numItemOrange)= (x);
    posItemOrangeY (numItemOrange)= (y);
    end
end

title (['Hay ' num2str(numItemOrange )  ' objetos naranjas'])
hold off
```

Pos_Naranja_Cara:

```
if posItemOrangeX~=0

for i=1:length(posItemOrangeX)
          %CARA F
if  posItemOrangeX(i) <= 300;
    if  posItemOrangeY(i) < 375;
    Cara(1,1)=3;
    end
    if posItemOrangeY(i) > 375 & posItemOrangeY(i) < 625;
    Cara(2,1)=3;
    end
    if posItemOrangeY(i) > 625;
    Cara(3,1)=3;
    end

end

if posItemOrangeX(i) > 300 & posItemOrangeX(i) < 600;
    if  posItemOrangeY(i) < 375;
    Cara(1,2)=3;
    end
    if posItemOrangeY(i) > 375 & posItemOrangeY(i) < 625;
    Cara(2,2)=3;
    end
```

```
    if posItemOrangeY(i) > 625;
    Cara(3,2)=3;
    end

end

if posItemOrangeX(i) >=600;
    if  posItemOrangeY(i) < 375;
    Cara(1,3)=3;
    end
    if posItemOrangeY(i) > 375 & posItemOrangeY(i) < 625;
    Cara(2,3)=3;
    end
    if posItemOrangeY(i) > 625;
    Cara(3,3)=3;
    end


end
  %--------------------------------------------------------------------------

end
end
```

Copy_of_Proc_Verde_Cara:

```
X=CaraX;
Y=imrotate(X,90);
Z=fliplr(Y);
Z=imcrop(Z,[665,457,1018,1011]);
% Convert RGB image to chosen color space
I = rgb2hsv(Z);

% Define thresholds for channel 1 based on histogram settings
channel1Min = 0.226;
channel1Max = 0.491;

% Define thresholds for channel 2 based on histogram settings
channel2Min = 0.392;
channel2Max = 1;

% Define thresholds for channel 3 based on histogram settings
channel3Min = 0;
channel3Max = 1;

% Create mask based on chosen histogram thresholds
sliderBW = (I(:,:,1) >= channel1Min ) & (I(:,:,1) <= channel1Max) & ...
    (I(:,:,2) >= channel2Min ) & (I(:,:,2) <= channel2Max) & ...
    (I(:,:,3) >= channel3Min ) & (I(:,:,3) <= channel3Max);
BW = sliderBW;


[L,Ne] = bwlabel(BW);

figure
imshow(label2rgb(BW))

prop=regionprops(L);
numItemGreen=0;
posItemGreen=zeros(9,2);
posItemGreenX=0;
```

```
posItemGreenY=0;
hold on

for n=1:length(prop);
    x=prop(n).Centroid(1);
    y=prop(n).Centroid(2);
    if prop(n).Area> 10000
    rectangle('Position',prop(n).BoundingBox,'EdgeColor','green','LineWidth',2);
    plot(x,y,'*');
    numItemGreen=numItemGreen+1;
    posItemGreen(numItemGreen)=(x);
    posItemGreen(numItemGreen,2)=(y);

    posItemGreenX (numItemGreen)= (x);
    posItemGreenY (numItemGreen)= (y);
    end
end

title (['Hay ' num2str(numItemGreen)  ' objetos verdes'])
hold off
```

Pos_Verde_Cara;

```
if posItemGreenX~=0

for i=1:length(posItemGreenX)
          %CARA F
if  posItemGreenX(i) <= 300;
    if  posItemGreenY(i) < 375;
    Cara(1,1)=4;
    end
    if posItemGreenY(i) > 375 & posItemGreenY(i) < 625;
    Cara(2,1)=4;
    end
    if posItemGreenY(i) > 625;
    Cara(3,1)=4;
    end

end

if posItemGreenX(i) > 300 & posItemGreenX(i) < 600;
    if  posItemGreenY(i) < 375;
    Cara(1,2)=4;
    end
    if posItemGreenY(i) > 375 & posItemGreenY(i) < 625;
    Cara(2,2)=4;
    end
    if posItemGreenY(i) > 625;
    Cara(3,2)=4;
    end

end

if posItemGreenX(i) >=600;
    if  posItemGreenY(i) < 375;
    Cara(1,3)=4;
    end
    if posItemGreenY(i) > 375 & posItemGreenY(i) < 625;
    Cara(2,3)=4;
    end
    if posItemGreenY(i) > 625;
```

```
    Cara(3,3)=4;
    end


end
  %--------------------------------------------------------------------

end
end
```

Copy_of_Proc_Amarillo_Cara;

```
X=CaraX;
Y=imrotate(X,90);
Z=fliplr(Y);
Z=imcrop(Z,[665,457,1018,1011]);
% Convert RGB image to chosen color space
I = rgb2hsv(Z);

% Define thresholds for channel 1 based on histogram settings
channel1Min = 0.104;%0.13
channel1Max = 0.209;%0.186

% Define thresholds for channel 2 based on histogram settings
channel2Min = 0.562;%0.7
channel2Max = 1.000;

% Define thresholds for channel 3 based on histogram settings
channel3Min = 0.000;
channel3Max = 1.000;


% Create mask based on chosen histogram thresholds
sliderBW = (I(:,:,1) >= channel1Min ) & (I(:,:,1) <= channel1Max) & ...
    (I(:,:,2) >= channel2Min ) & (I(:,:,2) <= channel2Max) & ...
    (I(:,:,3) >= channel3Min ) & (I(:,:,3) <= channel3Max);
BW = sliderBW;


[L,Ne] = bwlabel(BW);
figure
imshow(label2rgb(BW ))

prop=regionprops(L);
numItemYellow =0;
posItemYellow = zeros(9,2);
posItemYellowX=0;
posItemYellowY=0;
hold on

for n=1:length(prop);
    x=prop(n).Centroid(1);
    y=prop(n).Centroid(2);
    if prop(n).Area> 10000
    rectangle('Position',prop(n).BoundingBox,'EdgeColor','Yellow','LineWidth',2);
    plot(x,y,'*');
    numItemYellow =numItemYellow+1;
    posItemYellow (numItemYellow)=(x);
    posItemYellow (numItemYellow,2)=(y);

    posItemYellowX (numItemYellow)= (x);
```

```
    posItemYellowY (numItemYellow)= (y);
    end
end

title (['Hay ' num2str(numItemYellow )  ' objetos amarillos'])
hold off
```

Pos_Amarillo_Cara;

```
if posItemYellowX~=0

for i=1:length(posItemYellowX)
        %CARA F
if  posItemYellowX(i) <= 300;
    if  posItemYellowY(i) < 375;
    Cara(1,1)=6;
    end
    if posItemYellowY(i) > 375 & posItemYellowY(i) < 625;
    Cara(2,1)=6;
    end
    if posItemYellowY(i) > 625;
    Cara(3,1)=6;
    end

end

if posItemYellowX(i) > 300 & posItemYellowX(i) < 600;
    if  posItemYellowY(i) < 375;
    Cara(1,2)=6;
    end
    if posItemYellowY(i) > 375 & posItemYellowY(i) < 625;
    Cara(2,2)=6;
    end
    if posItemYellowY(i) > 625;
    Cara(3,2)=6;;
    end

end

if posItemYellowX(i) >=600;
    if  posItemYellowY(i) < 375;
    Cara(1,3)=6;
    end
    if posItemYellowY(i) > 375 & posItemYellowY(i) < 625;
    Cara(2,3)=6;
    end
    if posItemYellowY(i) > 625;
    Cara(3,3)=6;
    end


end
  %--------------------------------------------------------------------

end
end
```

Proc_Cara_2x2:

```
%PROCESAMIENTO CARA

Cara_2x2=[5 5;5 5];

Copy_3_of_Proc_Azul_Cara;
Copy_of_Pos_Azul_Cara;

Copy_3_of_Proc_Rojo_Cara1;
Copy_of_Pos_Rojo_Cara;

Copy_3_of_Proc_Naranja_Cara;
Copy_of_Pos_Naranja_Cara;

Copy_3_of_Proc_Verde_Cara;
Copy_of_Pos_Verde_Cara;

Copy_3_of_Proc_Amarillo_Cara;
Copy_of_Pos_Amarillo_Cara;
```

Copy_3_of_Proc_Azul_Cara:

```
X=CaraX;
Y=imrotate(X,90);
Z=fliplr(Y);
Z=imcrop(Z,[455,440,1195,1108]);%[230,500,850,750]
% Convert RGB image to chosen color space
I = rgb2hsv(Z);

% Define thresholds for channel 1 based on histogram settings
channel1Min = 0.472;
channel1Max = 0.738;

% Define thresholds for channel 2 based on histogram settings
channel2Min = 0.521;
channel2Max = 1;

% Define thresholds for channel 3 based on histogram settings
channel3Min = 0;
channel3Max = 1.000;

% Create mask based on chosen histogram thresholds
sliderBW = (I(:,:,1) >= channel1Min ) & (I(:,:,1) <= channel1Max) & ...
    (I(:,:,2) >= channel2Min ) & (I(:,:,2) <= channel2Max) & ...
    (I(:,:,3) >= channel3Min ) & (I(:,:,3) <= channel3Max);
BW = sliderBW;


[L,Ne] = bwlabel(BW);

figure
imshow(label2rgb(BW))

prop=regionprops(L);
numItemBlue=0;
posItemBlue=zeros(9,2);
posItemBlueX=0;
posItemBlueY=0;
hold on
```

```
for n=1:length(prop);

    x=prop(n).Centroid(1);
    y=prop(n).Centroid(2);
    if prop(n).Area> 50000
    rectangle('Position',prop(n).BoundingBox,'EdgeColor','blue','LineWidth',2);
    plot(x,y,'*');
    numItemBlue=numItemBlue+1;
    posItemBlue(numItemBlue)=(x);
    posItemBlue(numItemBlue,2)=(y);

    posItemBlueX(numItemBlue)= (x);
    posItemBlueY(numItemBlue)= (y);
    end
end

title (['Hay ' num2str(numItemBlue)   ' objetos azules'])
hold off
```

Copy_of_Pos_Azul_Cara:

```
if posItemBlueX~=0
for i=1:length(posItemBlueX);
       %Cara_2x2 F
if  posItemBlueX(i) <= 650;
    if  posItemBlueY(i) <= 530;
    Cara_2x2(1,1)=2;
    end

    if posItemBlueY(i) >= 530;
    Cara_2x2(2,1)=2;
    end

end


if posItemBlueX(i) >=650;
    if  posItemBlueY(i) <= 530;
    Cara_2x2(1,2)=2;
    end

    if posItemBlueY(i) >= 530;
    Cara_2x2(2,2)=2;
    end


end
  %-------------------------------------------------------------------------

end
end
```

Copy_3_of_Proc_Rojo_Cara1:

```
X=CaraX;
Y=imrotate(X,90);
Z=fliplr(Y);
Z=imcrop(Z,[455,440,1195,1108]);
```

```
% Convert RGB image to chosen color space
I = rgb2hsv(Z);

% Define thresholds for channel 1 based on histogram settings
channel1Min = 0.921;
channel1Max = 0.017;

% Define thresholds for channel 2 based on histogram settings
channel2Min = 0.521;
channel2Max = 1.000;

% Define thresholds for channel 3 based on histogram settings
channel3Min = 0.000;
channel3Max = 1.000;

% Create mask based on chosen histogram thresholds
sliderBW = ( (I(:,:,1) >= channel1Min) | (I(:,:,1) <= channel1Max) ) & ...
    (I(:,:,2) >= channel2Min ) & (I(:,:,2) <= channel2Max) & ...
    (I(:,:,3) >= channel3Min ) & (I(:,:,3) <= channel3Max);
BW = sliderBW;

[L,Ne] = bwlabel(BW);

figure
imshow(label2rgb(BW ))

prop=regionprops(L);
numItemRed =0;
posItemRed =zeros(9,2);
posItemRedX=0;
posItemRedY=0;
hold on

for n=1:length(prop);
    x=prop(n).Centroid(1);
    y=prop(n).Centroid(2);
    if prop(n).Area> 50000
    rectangle('Position',prop(n).BoundingBox,'EdgeColor','Red','LineWidth',2);
    plot(x,y,'*');
    numItemRed =numItemRed+1;
    posItemRed (numItemRed)=(x);
    posItemRed (numItemRed,2)=(y);

    posItemRedX (numItemRed)= (x);
    posItemRedY (numItemRed)= (y);
    end
end

title (['Hay ' num2str(numItemRed )  ' objetos rojos'])
hold off
```

Copy_of_Pos_Rojo_Cara:

```
if posItemRedX~=0
for i=1:length(posItemRedX);

       %Cara_2x2 F
if  posItemRedX(i) <= 650;
    if  posItemRedY(i) <= 530;
    Cara_2x2(1,1)=1;
    end
```

```
    if posItemRedY(i) >= 530;
    Cara_2x2(2,1)=1;
    end

end


if posItemRedX(i) >=650;
    if  posItemRedY(i) <= 530;
    Cara_2x2(1,2)=1;
    end

    if posItemRedY(i) >= 530;
    Cara_2x2(2,2)=1;
    end


end
  %--------------------------------------------------------------------------

end
end
```

Copy_3_of_Proc_Naranja_Cara:

```
X=CaraX;
Y=imrotate(X,90);
Z=fliplr(Y);
Z=imcrop(Z,[455,440,1195,1108]);
% Convert RGB image to chosen color space
I = rgb2hsv(Z);

% Define thresholds for channel 1 based on histogram settings
channel1Min = 0.042;
channel1Max = 0.121;

% Define thresholds for channel 2 based on histogram settings
channel2Min = 0.473;
channel2Max = 1;

% Define thresholds for channel 3 based on histogram settings
channel3Min = 0;
channel3Max = 1;


% Create mask based on chosen histogram thresholds
sliderBW = (I(:,:,1) >= channel1Min ) & (I(:,:,1) <= channel1Max) & ...
    (I(:,:,2) >= channel2Min ) & (I(:,:,2) <= channel2Max) & ...
    (I(:,:,3) >= channel3Min ) & (I(:,:,3) <= channel3Max);
BW = sliderBW;


[L,Ne] = bwlabel(BW);

figure
imshow(label2rgb(BW ))

prop=regionprops(L);
numItemOrange=0;
posItemOrange =zeros(9,2);
```

```
posItemOrangeX=0;
posItemOrangeY=0;
hold on

for n=1:length(prop);
    x=prop(n).Centroid(1);
    y=prop(n).Centroid(2);
    if prop(n).Area> 50000

rectangle('Position',prop(n).BoundingBox,'EdgeColor',[0.91,0.41,0.17],'LineWidth',2
);
    plot(x,y,'*');
    numItemOrange =numItemOrange+1;
    posItemOrange (numItemOrange)=(x);
    posItemOrange (numItemOrange,2)=(y);

    posItemOrangeX (numItemOrange)= (x);
    posItemOrangeY (numItemOrange)= (y);
    end
end

title (['Hay ' num2str(numItemOrange )  ' objetos naranjas'])
hold off
```

Copy_of_Pos_Naranja_Cara:

```
if posItemOrangeX~=0
for i=1:length(posItemOrangeX);

       %Cara_2x2 F
if  posItemOrangeX(i) <= 650;
    if  posItemOrangeY(i) <= 530;
    Cara_2x2(1,1)=3;
    end

    if posItemOrangeY(i) >= 530;
    Cara_2x2(2,1)=3;
    end

end


if posItemOrangeX(i) >=650;
    if  posItemOrangeY(i) <= 530;
    Cara_2x2(1,2)=3;
    end

    if posItemOrangeY(i) >= 530;
    Cara_2x2(2,2)=3;
    end


end
  %-------------------------------------------------------------------------

end
end
```

Copy_3_of_Proc_Verde_Cara:

```
X=CaraX;
Y=imrotate(X,90);
Z=fliplr(Y);
Z=imcrop(Z,[455,440,1195,1108]);
% Convert RGB image to chosen color space
I = rgb2hsv(Z);

% Define thresholds for channel 1 based on histogram settings
channel1Min = 0.243;
channel1Max = 0.503;

% Define thresholds for channel 2 based on histogram settings
channel2Min = 0.218;
channel2Max = 1;

% Define thresholds for channel 3 based on histogram settings
channel3Min = 0.000;
channel3Max = 1.000;

% Create mask based on chosen histogram thresholds
sliderBW = (I(:,:,1) >= channel1Min ) & (I(:,:,1) <= channel1Max) & ...
    (I(:,:,2) >= channel2Min ) & (I(:,:,2) <= channel2Max) & ...
    (I(:,:,3) >= channel3Min ) & (I(:,:,3) <= channel3Max);
BW = sliderBW;


[L,Ne] = bwlabel(BW);

figure
imshow(label2rgb(BW))

prop=regionprops(L);
numItemGreen=0;
posItemGreen=zeros(9,2);
posItemGreenX=0;
posItemGreenY=0;
hold on

for n=1:length(prop);
    x=prop(n).Centroid(1);
    y=prop(n).Centroid(2);
    if prop(n).Area> 50000
    rectangle('Position',prop(n).BoundingBox,'EdgeColor','green','LineWidth',2);
    plot(x,y,'*');
    numItemGreen=numItemGreen+1;
    posItemGreen(numItemGreen)=(x);
    posItemGreen(numItemGreen,2)=(y);

    posItemGreenX (numItemGreen)= (x);
    posItemGreenY (numItemGreen)= (y);
    end
end

title (['Hay ' num2str(numItemGreen)  ' objetos verdes'])
hold off
```

Copy_of_Pos_Verde_Cara:

```
if posItemGreenX~=0
```

```
for i=1:length(posItemGreenX);

       %Cara_2x2 F
if  posItemGreenX(i) <= 650;
    if  posItemGreenY(i) <= 530;
    Cara_2x2(1,1)=4;
    end

    if posItemGreenY(i) >= 530;
    Cara_2x2(2,1)=4;
    end

end


if posItemGreenX(i) >=650;
    if  posItemGreenY(i) <= 530;
    Cara_2x2(1,2)=4;
    end

    if posItemGreenY(i) >= 530;
    Cara_2x2(2,2)=4;
    end


end
  %--------------------------------------------------------------------

end
end
```

Copy_3_of_Proc_Amarillo_Cara:

```
X=CaraX;
Y=imrotate(X,90);
Z=fliplr(Y);
Z=imcrop(Z,[455,440,1195,1108]);
% Convert RGB image to chosen color space
I = rgb2hsv(Z);

% Define thresholds for channel 1 based on histogram settings
channel1Min = 0.117;
channel1Max = 0.236;

% Define thresholds for channel 2 based on histogram settings
channel2Min = 0.500;
channel2Max = 1.000;

% Define thresholds for channel 3 based on histogram settings
channel3Min = 0;
channel3Max = 1;


% Create mask based on chosen histogram thresholds
sliderBW = (I(:,:,1) >= channel1Min ) & (I(:,:,1) <= channel1Max) & ...
    (I(:,:,2) >= channel2Min ) & (I(:,:,2) <= channel2Max) & ...
    (I(:,:,3) >= channel3Min ) & (I(:,:,3) <= channel3Max);
BW = sliderBW;


[L,Ne] = bwlabel(BW);
```

```
figure
imshow(label2rgb(BW ))

prop=regionprops(L);
numItemYellow =0;
posItemYellow = zeros(9,2);
posItemYellowX=0;
posItemYellowY=0;
hold on

for n=1:length(prop);
    x=prop(n).Centroid(1);
    y=prop(n).Centroid(2);
    if prop(n).Area> 10000
    rectangle('Position',prop(n).BoundingBox,'EdgeColor','Yellow','LineWidth',2);
    plot(x,y,'*');
    numItemYellow =numItemYellow+1;
    posItemYellow (numItemYellow)=(x);
    posItemYellow (numItemYellow,2)=(y);

    posItemYellowX (numItemYellow)= (x);
    posItemYellowY (numItemYellow)= (y);
    end
end

title (['Hay ' num2str(numItemYellow )  ' objetos amarillos'])
hold off
```

Copy_of_Pos_Amarillo_Cara:

```
if posItemYellowX~=0
for i=1:length(posItemYellowX);

       %Cara_2x2 F
if  posItemYellowX(i) <= 650;
    if  posItemYellowY(i) <= 530;
    Cara_2x2(1,1)=6;
    end

    if posItemYellowY(i) >= 530;
    Cara_2x2(2,1)=6;
    end

end


if posItemYellowX(i) >=650;
    if  posItemYellowY(i) <= 530;
    Cara_2x2(1,2)=6;
    end

    if posItemYellowY(i) >= 530;
    Cara_2x2(2,2)=6;
    end


end
  %--------------------------------------------------------------------

end
end
```

Girar_B2:

```
Girar_B;
Girar_B;
```

Girar_D2:

```
Girar_D;
Girar_D;
```

Girar_F2:

```
Girar_F;
Girar_F;
```

Girar_L2:

```
Girar_L;
Girar_L;
```

Girar_R2:

```
Girar_R;
Girar_R;
```

Girar_U2:

```
Girar_U;
Girar_U;
```

MÓDULO DE SONIDO

El siguiente código es el realizado mediante Matlab para la grabación de las frases que dirá oralmente el robot.

```
aviso = input('Presione s para empezar','s')%mostrar en pantalla
global fs %creamos una variable global de la frecuencia de muestreo
fs=44100;%establecemos frecuencia de muestreo (si ponemos una demasiado alta ocupa
demasiado, si ponemos una frecuencia muy pequeña se escucha mal)

if aviso == 's' %para empezar a grabar pulsamos S
    recObj = audiorecorder(fs,16,2); %funcion que graba el audio a la frecuencia
correspondiente
    disp('Iniciando grabacion 1') %mostrar en pantalla
    disp('Grabe su voz durante 4 seg') %mostrar en pantalla
    recordblocking(recObj,4) %finaliza la grabacion al de 4 segundos
    disp('Fin de grabacion')%muestra pantalla

end
```

```
play(recObj) %para hacer la prueba de reproducir el sonido

audio1 = getaudiodata(recObj); %para guardar en una variable el sonido

%CASTELLANO
%save('Sonido1','audio1') %para guardar en un fichero la variable
%save('Sonido2','audio1')
%save('Sonido3','audio1')
%save('Sonido4','audio1')
%save('Sonido5','audio1')
%save('Sonido6','audio1')

%INGLES

% save('Sonido7','audio1')
% save('Sonido8','audio1')
% save('Sonido9','audio1')
% save('Sonido10','audio1')
% save('Sonido11','audio1')
% save('Sonido12','audio1')

% Euskera

% save('Sonido13','audio1')
% save('Sonido14','audio1')
% save('Sonido15','audio1')
% save('Sonido16','audio1')
% save('Sonido17','audio1')
% save('Sonido18','audio1')

%save('fs','fs')
%% comeentario de como cargar las cosas para reproducirla

% paso 1) Cargar fs:  load('fs.mat')
% paso 2) Cargar el archivo de audio: load('SonidoXX.mat')  [XX es el número de
audio que quieres reproducir]
% paso 3) Ordenar reproducion: sound(audio1,fs)  [Aqui SIEMPRE DEJAR audio1 , NO
cambiar el 1!]
% paso 4) Para evitar problemas con los audios que vienen despues borrar la
% variable  despues de ordenar reprodir el audio1 : clear('audio1')

sound(audio1,fs)

clear all

load('fs.mat')

%load('Sonido1.mat')
%load('Sonido2.mat')
%load('Sonido3.mat')
%load('Sonido4.mat')
%load('Sonido5.mat')
%load('Sonido6.mat')
```