Desarrollo de un robot crupier con inteligencia artificial

Denís Montoya del Canto

Resumen—Combinando técnicas de sistemas multimedia, visión por computador y mecánica, se ha logrado diseñar un robot físico capaz de ejercer la función de crupier en una partida de póker modalidad Texas holdem. Este es el caso de un robot de tipo SCARA de 4 grados de libertad y junto con aplicaciones sencillas de varios algoritmos, es capaz de desplazarse a las zonas deseadas y de detectar cartas y fichas, con las cuales puede operar. Utilizando la red neuronal conocida como YOLO, el robot identifica las cartas en base a los elementos que la componen, mientras que con la libreía OpenCV de Python, detecta las fichas en base a su forma y color. Juntando dichos módulos con una API de conversión de audio a texto, el robot es capaz de realizar un seguimiento adecuado de la partida, permitiendo a los jugadores realizar las acciones que normalmente harían en una partida de póker normal. Al final de la partida, el propio robot determina quien es el ganador de esta gracias a algoritmos de combinatoria y una API de texto a voz para informar a los jugadores.

Paraules clau— SCARA, Inteligencia Artificial, Visión por Computador, Sistemas multimedia, Reconocimiento de forma y color, Python, Arduino, Texto a Voz, Voz a Texto, YOLO, OpenCV, VEX.

Abstract—Combining multimedia knowledge techniques, computer vision and mechanics, it has been possible to design a physical robot capable of performing the role of dealer in a Texas holdem poker game. This case is about a SCARA robot with 4 axes and when simple applications of various algorithms are applied, it can move to the desired areas and detect cards and tokens, which it can operate. Using the neural network known as YOLO, the robot identifies the cards based on the elements that compose them, while with the Python OpenCV library, it detects the cards based on their shape and color. By pairing these modules with a speech-to-text API, the robot is able to adequately track the game, allowing players to perform the actions they would normally do in a normal poker game. At the end of the game, the robot says who's the winner thanks to combinatorial algorithms and a text-to-speech API.

Index Terms—SCARA, Artificial Inteligence, Computer Vision, Multimedia Knowledge, Form and color recognition, Python, Arduino, Text-to-Speech, Speech-to-Text, YOLO, OpenCV, Vex.

1 Introducción

El mundo de la robótica es extenso y variado. Existen una gran variedad de tipos de autómatas, pero en este trabajo se ha utilizado uno de tipo articulado o también conocido como brazo robótico. Dentro de esta familia de robots, el modelo SCARA[1][2] (Selective Compilant Assembly Robot Arm) el cual se caracteriza por ser un manipulador sencillo de pocos ejes normalmente utilizado para operaciones de tipo *pick and place*.

Por otro lado, existe el campo de inteligencia artificial conocido como visión por computador o *computer vision* en inglés, el cual se caracteriza por otorgar a un ordenador la capacidad de "ver" y analizar imágenes o videos.

Junto a estos dos módulos principales, se ha diseñado un manipulador de tipo SCARA que junto a módulos de visión por computador y de voz, es capaz de gestionar y ejercer la función de un crupier en partidas de póker, modalidad texas holdem[3]. Dicha modalidad se explica con detalle en el apéndice A1.

La motivación de este trabajo nace de un proyecto en grupo desarrollado en mi tercer año de carrera en la asignatura de Robótica, Lenguaje y Planificación (RLP), en la que se realizó un proyecto de robótica en un entorno simulado[5] en vez de en un entorno físico a causa de la pandemia de 2020 – 2021 provocada por la SARS-CoV-2. En este caso, aquel proyecto es el que se detella en este informe salvo que en este caso el robot es físico y no simulado. Así pues, contemplé este trabajo como una oportunidad para obtener la experiencia que perdí.

Se considera que este modelo de robot es más que suficiente para ejercer la basta mayoría de funciones que puede tener un crupier humano. De este modo, se determinan los siguientes módulos a implementar los cuales constituyen los objetivos del proyecto:

- Diseñar y montar el robot con las medidas necesarias para una zona de juego estándar de póker. En este caso, una mesa semicircular.
- Establecer un algoritmo o función que permita realizar cinemática inversa al autómata de manera óptima.
- Entrenar un modelo de detección de cartas que detecte todas las variantes de esta utilizadas en una partida de póker. En total es una baraja de 52 naipes.

- Generar un modelo de identificación de fichas para hallar e identificar tanto la ficha como su valor en base a su color.
- Implementar un sistema de detección de voz para indicar a la inteligencia artificial en qué situaciones debe realizar una acción u otra y permitir también el seguimiento de la partida y de los jugadores.

Del mismo modo, existe la función de barajar las cartas la cual un SCARA no puede implementar, pero este módulo se ha obviado dado que no es vital para la ejecución de una partida completa y existen artilugios que permiten realizar esta acción de manera rápida y sencilla.

Finalmente, una vez generados todas las funcionalidades necesarias de manera individual, se agrupan en un único módulo de inteligencia artificial el cual permitirá relacionarlas todas entre si y, de este modo, combinar y generar nuevas funciones.

2 Planificación

Para la correcta y eficiente organización del proyecto, se han dividido los módulos a desarrollar en dos grandes grupos: El grupo de hardware y el de software. El primero incluye todos los módulos relacionados con el robot y su movimiento, lo que incluiría su montaje y los algoritmos de cinemática inversa. El segundo grupo es el que contiene todos los objetivos de desarrollo de programas de visión por computador y de detección de voz.

Se ha utilizado la metodología ágil o *Agile* debido que muchos de estos módulos están relacionados entre si y también se pueden desarrollar elementos de ambos grandes grupos de forma paralela. Así pues, utilizando un diagrama de Gantt como el de la Figura 1, se han distribuido las tareas en "sprints" para obtener una mayor organización.

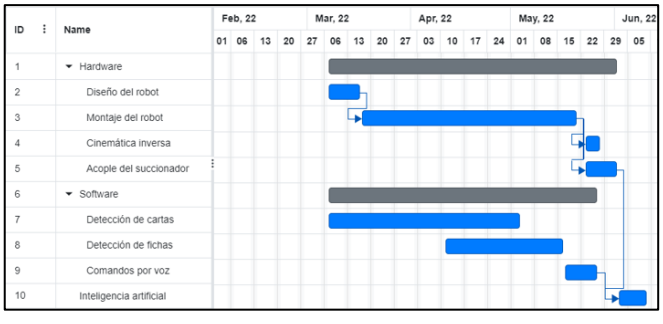


Fig. 1: Diagrama de Gantt del proyecto.

3 Materiales

Al querer generar un robot en un entorno físico y no simulado, es imprescindible disponer del material necesario para su desarrollo. Dependiendo del módulo, hará falta un tipo de material u otro.

De este modo, para el desarrollo del robot en si, se han utilizado los siguientes materiales: placas de aluminio y metal de la marca VEX para la estructura del robot; engranajes y piezas dentales de la misma marca para generar rotaciones y movimientos en un eje; motores continuos y servomotores para generar movimiento en los ejes; una placa Shield de motores adafruit 1 combinada con una placa Arduino UNO para poder conectar y controlar los motores; un final de carrera como sensor de proximidad; una cámara Logitech C270 HD; un relé y una regleta de conexión o borne de alimentación; y finalmente un solenoide que junto a una ventosa, permite mantener una zona sin circulación de aire o zona de vacío para coger y manipular objetos.

En cuanto al material del juego, tan solo es necesario un set de cartas que contenga las 13 cartas de cada palo (corazones, picas, diamantes y tréboles) y un pequeño set de fichas de póker.

Además, para ambos casos, se utilizan piezas personalizadas generadas por una impresora 3D para situar o fijar los demás componentes en una posición favorable. Estas piezas son: un volteador de cartas que, mediante un par de pequeñas rampas, permite al robot dar la vuelta a la carta con el uso de la gravedad; piezas para acomplar motores, el solenoide y la ventosa al robot de manera cómoda; y soportes para las cartas y las fichas con formas que permitan al robot acceder a ellas fácilmente.

4 Desarrollo

En esta sección se detallará el método de desarrollo y las técnicas utilizadas para satisfacer las necesidades del proyecto. Tal como se ha explicado en apartados anteriores, este apartado se dividirá en dos grandes grupos: Hardware y software y, dentro de estos, se explicará el proceso que se ha seguido para lograr generar cada uno de los módulos.

4.1 Hardware

El desarrollo del robot se divide en cuatro grandes módulos: El diseño previo del robot, el montaje del SCARA, los algoritmos de cinemática inversa y el acople del succionador. Estos algoritmos se han decidido añadir a la sección de hardware porque sin ellos no se podrían interactuar con los elementos físicos de la escena. Por tanto, todo módulo o programa que esté directamente relacionado con la placa Arduino se considerará como elemento hardware del proyecto.

4.1.1 Diseño previo del robot

Las funciones de movimiento y manipulación necesarias dependen de elementos externos como el área de juego y la disposición de los elementos en esta. La zona de juego siempre será una mesa con superficie llana y los objetos pueden estar en distintas alturas en cada ocasión. Por tanto, un SCARA es ideal para este proyecto.

Este robot tiene una zona de trabajo equivalente a un

semicírculo en caso de utilizar servomotores de 180 grados, pero es capaz de conseguir una zona más amplia en caso de incrementar los ángulos de dicho motor. En este proyecto se utiliza como zona de juego una mesa semicircular por tanto no es necesario sobrepasar los 180 grados de rotación.

En cuanto a la altura, es necesario determinar de manera previa cuál será el elemento con mayor elevación por la que el robot deberá cruzar a través o bien donde deberá situar su manipulador. El elemento más alto que hay en la zona de juego es el volteador de cartas, donde el robot deberá situar las cartas en la parte superior de este para poder darles la vuelta. Así pues, la altura mínima que debe tener es la altura del propio volteador más una pequeña distancia de seguridad. La figura 2 muestra el diseño inicial planteado para este proyecto, considerando que la zona de juego sería de aproximadamente 40 centímetro de radio. Las proporciones no necesariamente deben ser exactas al diagrama, pero es recomendable generar un diseño previo con proporciones razonables para posteriormente crear un modelo físico adecuado.

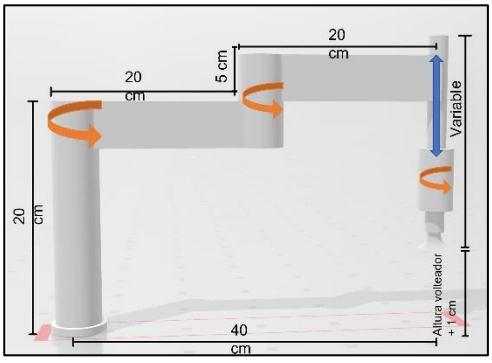


Fig. 2: Diseño inicial del SCARA de 4 ejes.

Es necesario destacar que la colocación de los ejes no necesariamente debe ser la mostrada en la figura 2, pero utilizar este formato puede facilitar la realización de tareas posteriores.

4.1.2 Montaje del robot

A la hora de ensamblar el robot se destacan 3 grandes grupos:

1. El propio brazo o la zona que contiene la gran mayoría de ejes que permiten rotar las distintas articulaciones de este.
2. La base la cual soporta el peso del brazo y permite que haya la mínima inclinación posible en base a este.
3. La zona del manipulador donde, en este caso, está situado el solenoide el cual succiona las distintas piezas.

Para la primera parte, es vital tener en cuenta la distancia máxima que se desea que el robot recorra. Para este proyecto se ha utilizado una distancia de 15 centímetros para la primera articulación seguida de otra de 25 centímetros, permitiendo operar al robot en una zona de trabajo de 40 centímetros de radio. Para cada articulación se utilizan dos piezas de aluminio conectadas entre si para mejorar la estabilidad de este como se muestra en la figura 3.

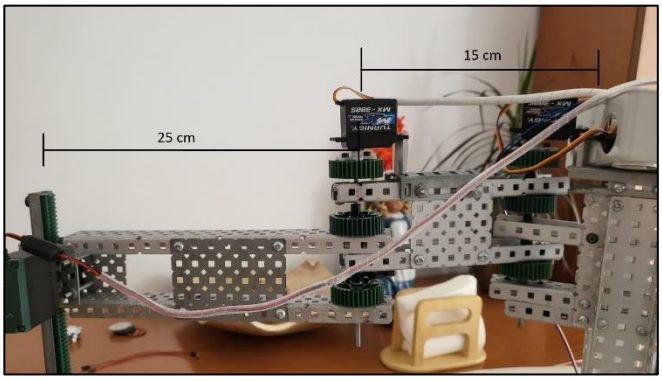


Fig. 3: Brazo del robot.

Al utilizar placas de aluminio en vez de metal se consigue reducir el peso en las extremidades y poder contrarrestarlo más fácilmente utilizando piezas más pesadas en la base. Para la base es fundamental utilizar una estructura que permita equilibrar y estabilizar el conjunto entero. Utilizando una forma en "C" y aumentando la densidad con piezas de metal en vez de aluminio en la base, obtenemos este equilibro deseado.

El manipulador también necesita tener una altura fija y calculada para cumplir con los requisitos necesarios. En cuanto al eje prismático, este proyecto lo tiene situado en la parte del manipulador.

Finalmente, se fusionan todas las piezas para obtener el resultado mostrado en la figura 4, el cual corresponde a la arquitectura hardware utilizada en el proyecto. Todos los componentes hardware están conectados a la placa Shield de motores vinculada con la placa Arduino. Esta última se conecta mediante USB al ordenador el cual contiene todos los componentes relacionados con los módulos de visión por computador.

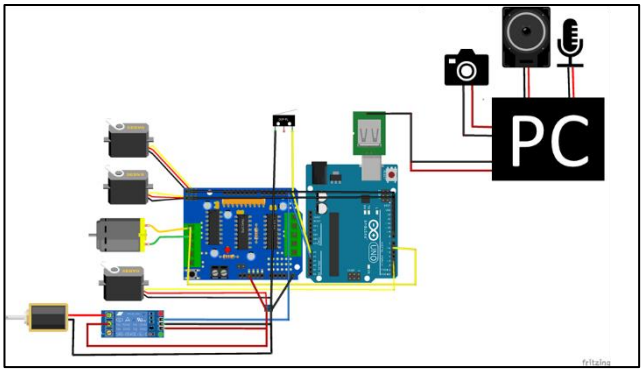


Fig. 4: Arquitectura hardware del robot

4.1.3 Cinemática inversa

Los ángulos de rotación de los servomotores del robot

se pueden calcular utilizando distintos métodos. El primero es el método geométrico que aplica el teorema del coseno. Los datos que conocemos son las medidas del robot y los puntos en coordenadas X e Y a las que queremos acceder. La elevación o eje Z también puede ser calculado, pero al utilizar un final de carrera no es necesario.

Para calcular el primer ángulo, se tiene en cuenta que el robot, al alcanzar la posición deseada, generará un triangulo rectángulo si se conecta el punto (x,y) deseado con la base del primer eje. De este modo, podemos calcular la hipotenusa de estre triangulo la cual representa la distancia desde el primer eje del robot hasta el punto deseado y, en este punto, podemos calcular el ángulo como la suma del ángulo interno del triangulo más el externo, los cuales equivaldrían a Beta y Alfa en la figura 5 respectivamente. Aplicando el teorema del coseno, se obtiene el ángulo Beta y aplicando la arcotangente se obtiene Alfa. Sumando estos dos resultados, obtenemos el primer ángulo deseado.

El segundo ángulo se obtiene de manera más inmediata, pues tras haber calculado la hipotenusa, únicamente se ha de aplicar el teorema del coseno para calcular el ángulo del segundo eje. Es importante tener en cuenta que las variables dependiendo del orden de las variables del teorema, ya que con los mismos datos se puede calcular un ángulo u otro.

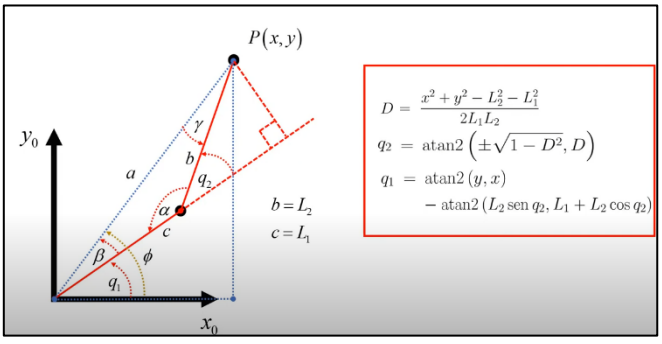


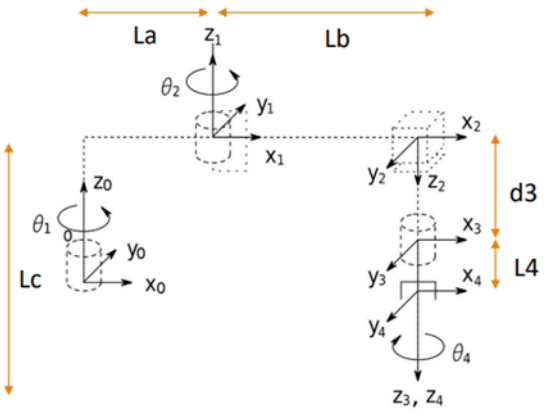
Fig 5. Representación de la cinemática inversa de un robot SCARA con metodología geométrica.

El último eje el cual está situado junto al manipulador sirve para corregir los ángulos en el que el robot recoge las piezas. De esta manera, aplicando una corrección al inicio y un giro al final del desplazamiento, se logra distribuir un objeto en una posición determinada con una orientación en concreto. Para este proyecto es necesario este eje para colocar cartas, mientras que las fichas, al ser circulares, no necesitan el uso de este eje.

Es importante tener en cuenta que durante el proceso de cálculo de ángulos se debe realizar la conversión de radianes a grados.

El segundo método es el de Denavit-Hartenberg (D-H)[6]. Este requiere que calculemos parámetros en base a las posiciones y ángulos entre un eje y el subsiguiente. Exactamente, es necesario calcular las distancias, alturas y rotaciones que hay de un eje a otro, también teniendo en cuenta el tipo de eje que es. Al aplicar el método, se obtiene la matriz D-H que junto a un resolvedor de matrices se puede obtener el resultado de todos los ángulos, incluida también la altura del eje prismático.

En la figura 6 se muestra la tabla de D-H calculada en base a la estructura del robot, que ha permitido obtener la matriz de D-H de la figura 7, la cual ha sido utilizada en este proyecto. Esta, junto a la función *nsolve* de la librería *Sympy*[7] de Python, calcula los ángulos deseados en cuestión de milésimas de segundo.



	θ	d	a	α
1	Θ1	Lc	La	0
2	Θ2	0	Lb	180
3	0	d3	0	0
4	Θ4	L4	0	180

Fig. 6: Diagrama y tabla de Denavit-Hartenberg (D-H) de un SCARA con 4 ejes de libertad.

$1.0\cos(\theta_1+\theta_2-\theta_4)$	$-1.0\sin(\theta_1+\theta_2-\theta_4)$	0	$la\cos(\theta_1)+lb\cos(\theta_1+\theta_2)$
$1.0\sin(\theta_1+\theta_2-\theta_4)$	$1.0\cos(\theta_1+\theta_2-\theta_4)$	0	$la\sin(\theta_1)+lb\sin(\theta_1+\theta_2)$
0	0	1.0	$-1.0d_3-1.0l_4+1.0lc$
0	0	0	1

Fig. 7: Matriz de Denavit-Hartenberg (D-H) de un SCARA con 4 ejes de libertad.

Ambos métodos son efectivos y suficientes para el cálculo de los ángulos, por tanto, el método a aplicar es meramente arbitrario.

4.1.4 Manipulador

El manipulador del robot está compuesto por un solenoide y una ventosa que se alzan o descienden mediante un eje prismático.

El solenoide es el que permite agarrar las fichas junto con la ventosa genera una zona sin circulación de aire y, al entrar en contacto con un elemento, se genera una zona de vacío la cual realiza que el objeto se adhiera a la ventosa. Este, al recibir una corriente de 5 voltios, tapa el conducto por el que está conectado la ventosa, generando así la pequeña cámara de vacío, mientras que, al recibir 0

voltios, permite que el aire circule, soltando así cualquier elemento que estuviera adherido a la ventosa. De este modo, el robot es capaz de desplazar elementos de la zona de juego sin problema.

4.2 Software

Los módulos de software del robot se dividen en 3 grandes secciones: La detección de cartas, la detección de ficha y el módulo de comandos de voz.

A diferencia de los módulos hardware, estos no tienen relación entre ellos, por tanto, el funcionamiento de uno no depende del correcto funcionamiento de otro.

4.2.1 Detección de cartas

La primera función necesaria con visión por computador es la de detección de cartas. En esencia, el módulo detecta todas las cartas con el valor visible para que sean procesados con un algoritmo de combinatoria y determinar quien es el ganador de la partida.

Para ello se ha entrenado una red neuronal conocida como YOLOv5 (You Only Look Once)[8] la cual destaca por detectar elementos con una precisión muy elevada y una tasa de fallo muy baja. Se debe generar, de manera previa, una base de datos de imágenes o *dataset* para mostrar qué elementos nos interesa clasificar, que en este caso son todas las posibles cartas de una baraja de póker, la cual consta de 4 palos y 13 números por palo, un total de 52 cartas únicas.

4.2.1.1 Creación del dataset

El *dataset* es el elemento más importante a la hora de entrenar la red neuronal, pues mostraremos qué elementos queremos categorizar y detectar. Además, generando un *dataset* variado y amplio, se mejora la precisión de la red y también ampliamos las situaciones en las que puede detectar cada elemento. El proceso mostrado que aparece en la figura 8 será detallado a continuación.

Las recomendaciones básicas son generar un *dataset* que contenga imágenes de mismos elementos des de distintos ángulos, distinta luminosidad y distinto tamaño, para que en situaciones desfavorables o variadas el algoritmo sea capaz de detectar los elementos. En este proyecto, la cámara está colocada en un soporte vertical para obtener una vista zenital de la zona de juego, por lo que no es necesario generar imágenes des de distintos ángulos, pero continúa siendo recomendable.

Para facilitar esta tarea, se ha utilizado el programa del usuario Geaxgx en GitHub llamada playing-card-detection[9] el cual facilita la tarea de generación de un *dataset* variado. El programa utiliza vídeos grabados por el usuario para generar imágenes de las cartas en cada fotograma o *frame*, por tanto, si durante el video se varía la luminosidad y el ángulo de la cámara, la aplicación podrá generar una gran cantidad de imágenes variadas para cada una de las cartas. Además, detecta en qué *frames* la carta es claramente visible, por lo que la calidad de la cámara, la intensidad de luminosidad, los reflejos que puedan aparecer en la carta y una buena saturación son cruciales, pues obtener casos extremos demasiado una carta puede provocar que no se genere ningúna imagen de esta.

Una vez extraídas las imágenes por cada carta, el programa las recorta y ajusta para obtener únicamente la carta como imagen, extrayendo el fondo. En este punto, un algoritmo detectará qué imágenes aportan más información que otras en base al área mínima que muestra la información del número y el palo en las esquinas de la carta. Este paso es importante porque, dependiendo del ángulo, puede hace creer a la red neuronal que existen otros elementos a parte de los símbolos en las esquinas que determinen la clase de una carta. Por ello se eligen las imágenes que aporten más información y que faciliten la tarea de entrenamiento.

Tras haber realizado la selección de imágenes con mayor valor, solo falta generar las própias imágenes del *dataset*. La idea de haber realizado todo este proceso previo es para que se puedan generar combinaciones entre varias cartas y varias intensidades de cada carta, lo que permitirá a la red neuronal adaptarse a cualquier tipo de situación en un futuro. Lo único que falta por modificar es el fondo de la imagen, los cuales son extraídos de una conocida base de datos llamada Describable Textures Dataset (DTD)[10]. Este *dataset* contiene una gran variedad de fondos con distintas formas y colores para aumentar la variedad de casos en las que puede haber un elemento. Utilizando fondos tan variados, permitimos a la red neuronal adaptarse a cualquier situación y contexto, pudiendo realizar una detección de elementos independientemente del fondo que haya.

Finalmente, solo queda juntar estos elementos para generar nuestra propia base de datos que utilizará la red neuronal para entrenar. El programa permite generar la cantidad que se desee de imágenes y se puede elegir entre imágenes que contengan combinaciones de dos o tres cartas. Se recomienda utilizar sobre todo combinaciones de tres cartas, pero variar entre imágenes de dos o tres cartas es la situación ideal.

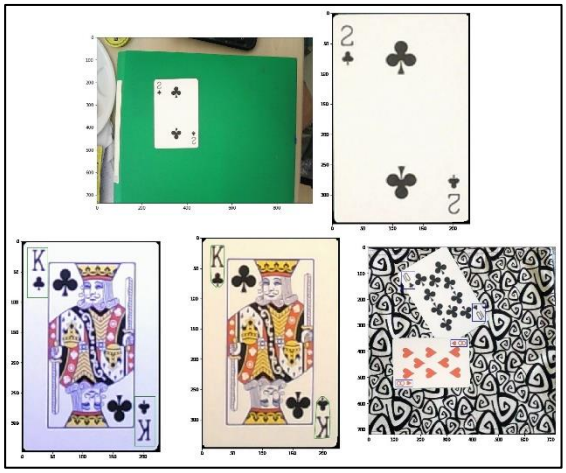


Fig. 8: Proceso de generación de imágenes del *dataset*.

En este proyecto se ha generado un *dataset* de 6000 imágenes, las cuales la mitad están compuestas de 2 cartas y la otra mitad de 3.

4.2.1.2 Entrenamiento de la red neuronal

A la hora de entrenar una red YOLO son necesarios los siguientes parámetros: el número de iteraciones o *epochs*, el número de subiteraciones o *batchs*, la resolución de las imágenes, los pesos o *weights* iniciales de los nodos y el *dataset* a utilizar para entrenar la red neuronal.

Se ha utilizado la herramienta gratuita Google Colaboratory[11] que permite realizar todo el entrenamiento desde un computador remoto, permitiendo utilizar GPU dedicada a la ejecución.

De este modo, el único paso necesario es determinar los parámetros mencionados anteriormente para comenzar el entrenamiento. El creador de la red neuronal recomiendo seguir una serie de cálculos para determinar estos valores[8], y tras varias pruebas estos han sido los valores finales:

Epochs: 150
Batchs: 16
Resolución: 416
Weights: yolov5s.pt

Dichos parámetros sumado a una ejecución de 5 horas, 58 minutos y 24 segundos han resultado en la obtención de los pesos utilizados para este proyecto, los cuales permiten realizar una detección precisa de las cartas.

4.2.1.3 Programa detector

YOLOv5 dispone de un archivo de tipo Python que permite detectar por cámara a tiempo real o por una imagen individual los elementos de las clases para la cual ha sido entrenada la red. Visto que esta clase de algoritmos pueden generar un mínimo error, se realiza una grabación en vivo de 3 segundos a 30 *frames* por segundo (FPS) para captar las imágenes. Después, se calcula el número de veces que aparece cada tipo de carta y se limpian los errores que hayan podido aparecer, obteniendo así las cartas que realmente aparecen en el vídeo. La figura 9 muestra un ejemplo de esta limpieza.

Fig. 9: Ejemplo de detección de cartas con luminosidad especial.

4.2.2 Detección de fichas

A diferencia de la detección de cartas, se aprovecha el hecho de que las fichas sean de colores específicos en función de su valor (por ejemplo, la ficha roja equivale a 10€). De este modo, no es necesario realizar una detección tan específica como con las cartas.

Así pues, las cualidades de las fichas aprovechadas son el color y la forma. La librería OpenCV[12] de Python la cual permite trabajar con imágenes, colores y formas, lo cual es ideal para este caso.

La figura 10 representa el protocolo utilizado para la detección de color: En primer lugar, dada una imagen, se detectan todas las formas circulares que haya en la imagen. Seguidamente, se recorta y redimensiona una imagen por cada una de las fichas que aparezcan. Finalmente, se detecta el color[13] que abunda más en la imagen.

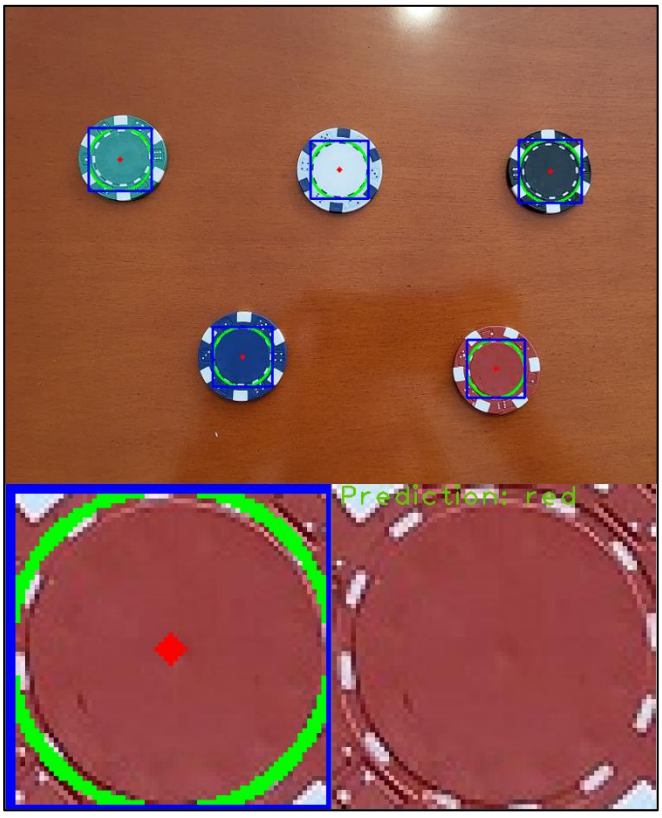


Fig. 10: Proceso de detección del color de fichas.

Para mejorar la precisión del algoritmo, se delimita el tamaño máximo de la circunferencia detectada para escoger el ciruclo interno de las fichas. De este modo, a la hora de recortar las imágenes, logramos evitar más píxeles de la zona de juego e incrementar la cantidad de píxeles del color adecuado, incrementando así la tasa de acierto.

4.2.3 Seguimiento por voz

Gracias a las interfaces de programación de aplicaciones o APIs, se ha implementado una librería que permite generar texto en función de un audio de entrada (*speech to text*) y viceversa (*text to speech*). En este proyecto ambas APIs son de Google[14][15].

El algoritmo actual dispone de varios diccionarios de palabras para cada tipo de acción, evitando así que los jugadores están limitados a una palabra por acción. Como muestra en el diagrama de flujo de la figura 11, el programa queda a la espera hasta que detecta una palabra o frase de un jugador mediante la API de Speech to text. En cuanto lo hace, aplica la función *get_close_matches*() de la librería *difflib*[16] para corregir pequeños errores de detección o pronunciación a causa del ruido que se genera por la calidad del audio. Seguidamente, detecta si este input pertenece a alguno de los diccionarios de acciones. En caso de no encontrar el input en ninguno de los diccionarios, informa mediante la API de text to speech que no se ha entendido el audio y pide al jugador que vuelva a realizar a generar un input por voz y vuelve a quedar a la espera. Por otro lado, si un diccionario contiene el input entonces realiza una acción en base a este. Mientras la acción se realiza, el módulo de voz se desactiva para no generar ningún tipo de error. En cuanto finaliza la acción, detecta si hay más jugadores pendientes por jugar y, en caso de haber, repite el ciclo.

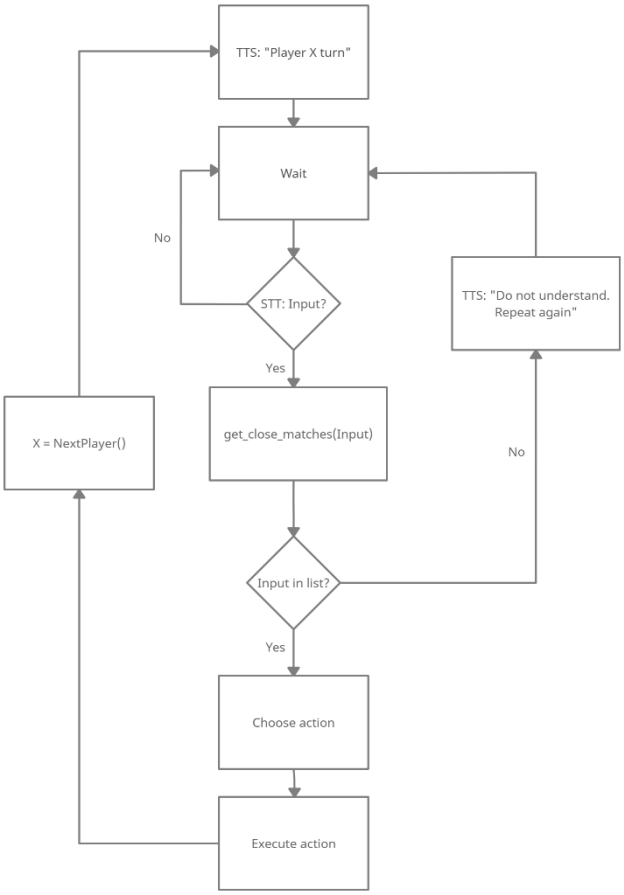


Fig. 11: Diagrama de flujo del módulo de voz.

Para decidir qué palabras compondrán los diccionarios, se ha realizado una encuesta a empleados de una empresa de casinos sobre cuales son los términos más utilizados para cada una de las posibles acciones de póker.

4.3 Módulo de control

El objetivo del módulo de control del proyecto es el de unificar todos los demás módulos descritos previamente para sincronizar los procesos. Este se podría categorizar como el archivo principal o *main* del proyecto.

Combinando varios de los archivos de Python generados, el robot es capaz de:

- Detectar la posición de las fichas y mover el manipulador hacia ellas.
- Anunciar el ganador en base a las cartas reveladas.
- Repartir cartas en función del número de jugadores.
- Adaptar los módulos independientemente del

número de jugadores que abandonen la partida.

Por último, también implementa módulos de seguimiento de la partida. En base a las acciones que los jugadores declaran y que el módulo de voz detecta, el algoritmo principal determina el siguiente jugador a jugar.

5 Resultados

En esta sección, se muestran los resultados de los módulos de manera individual

Los módulos desarrollados son efectivos de forma individual, generando resultados positivos de manera general y resultados extremadamente positivos en condiciones ideales.

Pese a ello, la combinación de estos módulos genera resultados inferiores a los esperados debido a la acumulación de error de estos, destacando en particular los módulos de cinemática inversa.

5.1 Hardware

El robot es capaz de desplazarse a todos los puntos posibles dentro del área de juego y, obviamente, dentro de su rango. Los cálculos son correctos y, de forma teórica, se aplican ángulos precisos para desplazar el brazo dentro del área de juego. Pese a ello, debido a la velocidad de rotación de los servomotores y a la imprecisión que estos puedan tener, el manipulador no se sitúa exactamente encima de la coordenada deseada.

Además, debido a la arquitectura del robot, este tiene una leve inclinación en la base del brazo, lo que provoca que en el extremo o en la posición donde se encuentra el manipulador el error aumente hasta provocar una inclinación de magnitud considerable, lo que provoca que exista un cierto ángulo en la posición del solenoide y que este tenga dificultades a la hora de manipular objetos.

No obstante, el robot accede a las zonas deseadas y es capaz de manipular los objetos que hay en la zona de juego, aún que a causa de los errores acumulados le es complicado manipularlos, pero los manipula.

5.2 Software

El resultado de los módulos software son muy satisfactorios en la gran mayoría de casos. Los casos en los que los módulos erran son sobre todo por problemas de la calidad de tanto la cámara como el micrófono.

5.2.1 Detección de cartas

La red neuronal tiene unos valores de *precision* y *recall* muy elevados, como se aprecian en las figuras 12 y 13 respectivamente.

La detección no es impecable, pero, gracias al algoritmo de limpieza, se logra acertar todas las ejecuciones en las cuales las cartas aparecen en la grabación. Los únicos casos en los que el algoritmo erra son en aquellos donde las cartas están muy solapadas o la iluminación es excepcionalmente baja.

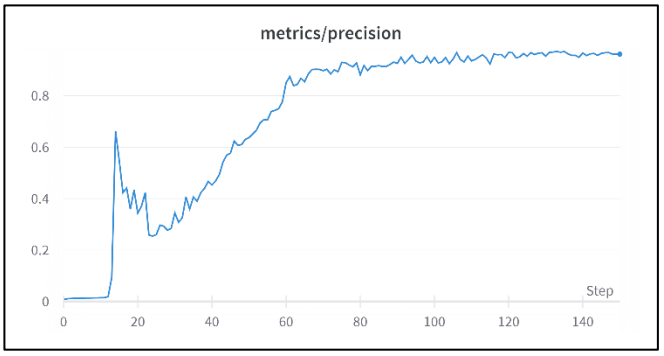


Fig. 12: *Precision* de la red neuronal.

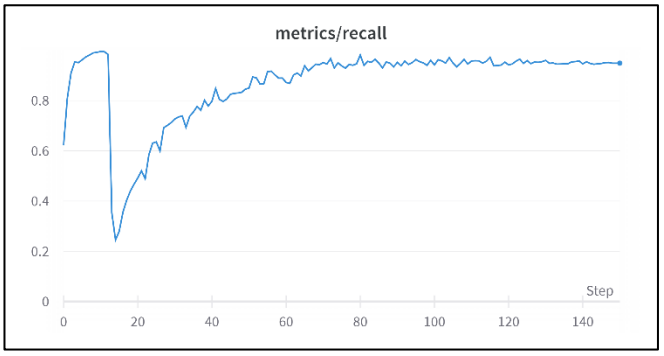


Fig. 13: *Recall* de la red neuronal.

5.2.2 Detección de fichas

El algoritmo de detección de fichas funciona excepcionalmente en aquellas con un color muy distinto a las demás. En este caso, la detección de los colores rojo, azul y blanco tiene una tasa de acierto superior al 95%.

Sin embargo, el algoritmo tiene una tasa de acierto del 80% para el color negro y del 75% del color verde. Esto es debido a la tonalidad de las fichas, pues en los casos de error, confunde la ficha negra con una azul y la verde con el azul también. En gran parte, este error es debido a la iluminación de la sala sumado a la tonalidad de la ficha, pero ajustando los rangos de color de cada ficha, se ha logrado incrementar la tasa de acierto en un 10%.

5.2.3 Detección de voz

El módulo de detección de voz acierta un 95% de los casos utilizando un micrófono de alta calidad a una distancia próxima al jugador. No obstante, si el micrófono se aleja y el ruido generado por otras personas es elevado, entonces la captación de palabras se reduce en gran cantidad, provocando que el jugador deba alzar más la voz y repetir el comando una media de 3 veces.

6 Conclusiones

Gracias a este proyecto se ha podido descubrir el gran potencial de la inteligencia artificial y la robótica. La visión por computador es un campo que dispone de un gran abanico de posibilidades. La red neuronal YOLO es un claro ejemplo de ello, permitiendo de manera sencilla detectar cualquier objeto al cual se haya entrenado de manera previa.

Como un primer acercamiento a estos campos, se puede apreciar que la exactitud y la precisión son requisitos vitales para obtener unos resultados satisfactorios, pero a su vez se aprende que el camino más complicado en proyectos de este tipo no es el de generar un modelo funcional, sino el de generar un modelo eficiente.

Esta primera versión del robot muestra funcionalidad con una falta de eficiencia y, en caso de incrementar esta última, podría llegar a convertirse en un producto con potencial de comercialización. No obstante, para lograr ese objetivo, sería necesaria la implantación de módulos adicionales, de los cuales algunos están propuestos en la sección de trabajos futuros.

7 Trabajos futuros

Con el objetivo de aumentar la calidad del proyecto, es necesaria mejorar e implantar módulos que permiten alcanzar la excelencia en este campo.

Al tratarse de un crupier de un juego de casino, sería necesario la implantación de módulos de seguridad para no amañar algunos resultados o *outputs* de los módulos ya implementados. Lo ideal sería implementar funciones o sistemas para preveer falsificaciones de cartas o robos de cartas preestablecidos, un sistema de reconocimiento de voz para evitar que un jugador pueda determinar la acción de otro, seguimiento de los movimientos de los jugadores para evitar casos sospechosos y un último sistema para evitar que los jugadores puedan interactuar con elementos que no les pertenezcan.

En cuanto a mejoras, con el uso de componentes de mayor calidad, se podría mejorar la ejecución de los módulos de cinemática inversa o los de inteligencia artificial: Utilizar un micrófono de mayor calidad, motores más precisos o una cámara con mejor resolución puede provocar un incremento en la tasa de aciertos de los algoritmos, disminuyendo el criterio de requisitos para obtener un entorno ideal.

Finalmente, en colaboración con la compañía Stäubli de Sant Cugat del Vallés (Barcelona), se planea realizar el implemento de los módulos desarrollados en este proyecto a un robot SCARA de carácter industrial para implementar los módulos desarrollados en este proyecto. Con ello, se podría lograr generar un crupier robótico con una precisión mucho más elevada que con el robot manufacturado.

Agradecimientos

Me gustaría agradecer a mi tutor Carlos García por el constante apoyo y seguimiento que ha realizado en el proyecto, permitiéndome además trabajar en un entorno cómodo y con componentes de alta calidad. Agradecer también a mis amigos por su constante interés en el proyecto, motivándome a seguir desarrollandolo.

Mencionar también a mis compañeros de trabajo de Cirsa por el apoyo e interés, sobre todo a Abel Martinez el cual me ha brindado material para poder entrenar los módulos de visión por computador.

Agradezco a mis padres y a mis abuelos por su preocupación, apoyo y amor constante en la duración del proyecto, en especial a mi madre Sylvie.

Doy las gracias también a Aimara Ventoso por dedicar parte de su tiempo en revisar el informe y proponer cambios para mejorar la calidad de este.

Finalmente, agradecer a Safa El Hmidi por haber vivido conmigo tanto las situaciones más estresantes y complicadas del desarrollo como las más gratificantes, además de haber confiado en mí en todo momento. Le dedico este trabajo.

Bibliografia

[1] «SCARA Robot | How To Build Your Own Arduino Based Robot», *How To Mechatronics*, 2 de octubre de 2020. https://howtomechatronics.com/projects/scara-robot-how-to-build-your-own-arduino-based-robot/ (accedido 1 de marzo de 2022).

[2] J. J. Craig, *Introduction to Robotics: Mechanics and Control*. Pearson Educación, 2005.

[3] «Texas hold 'em», *Wikipedia, la enciclopedia libre*. 14 de enero de 2022. Accedido: 1 de marzo de 2022. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=Texas_hold_%27em&oldid=140943257

[4] «Combinaciones de poker | 888 Poker». http://prod-blog-www.888poker.es/blog/combinaciones-de-poker (accedido 12 de junio de 2022).

[5] EmilCarvajal, *[ROBOTICA] Deal-AI*. 2021. Accedido: 1 de marzo de 2022. [En línea]. Disponible en: https://github.com/EmilCarvajal/Deal-AI

[6] «chap3-forward-kinematics.pdf». Accedido: 12 de junio de 2022. [En línea]. Disponible en: https://users.cs.duke.edu/~brd/Teaching/Bio/asmb/current/Papers/chap3-forward-kinematics.pdf

[7] «Solvers — SymPy 1.10.1 documentation». https://docs.sympy.org/latest/modules/solvers/solvers.html (accedido 12 de junio de 2022).

[8] *ultralytics/yolov5*. Ultralytics, 2022. Accedido: 1 de marzo de 2022. [En línea]. Disponible en: https://github.com/ultralytics/yolov5

[9] geaxgx, *playing-card-detection*. 2022. Accedido: 21 de mayo de 2022. [En línea]. Disponible en: https://github.com/geaxgx/playing-card-detection

[10] «Describable Textures Dataset». https://www.robots.ox.ac.uk/~vgg/data/dtd/ (accedido 21 de mayo de 2022).

[11] «Google Colaboratory». https://colab.research.google.com/?hl=es (accedido 21 de mayo de 2022).

[12] «Home - OpenCV». https://opencv.org/ (accedido 15 de junio de 2022).

[13] «color_recognition/src at master · ahmetozlu/color_recognition», *GitHub*. https://github.com/ahmetozlu/color_recognition (accedido 15 de junio de 2022).

[14] «Speech-to-Text: reconocimiento de voz automático | Cloud Speech-to-Text», *Google Cloud*. https://cloud.google.com/speech-to-text?hl=es (accedido 1 de marzo de 2022).

[15] «Text-to-Speech: conversión de texto a voz natural | Cloud Text-to-Speech», *Google Cloud*. https://cloud.google.com/text-to-speech?hl=es (accedido 1 de marzo de 2022).

[16] «difflib — Helpers for computing deltas — Python 3.10.5 documentation». https://docs.python.org/3/library/difflib.html#difflib.get_close_matches (accedido 15 de junio de 2022).

APÉNDICES

A1. MODALIDAD TEXAS HOLDEM

La modalidad Texas Holdem se diferencia al resto por repartir dos cartas a cada jugador las cuales no pueden ser intercambiadas y donde se juega un total de tres rondas, en las cuales se revelan cartas en orden de 3, 1 y 1 cartas en cada una de las rondas. Dentro de una ronda, cada uno de los jugadores puede realizar una acción entre pasar el turno, aumentar la apuesta o retirarse. En caso de que un jugador aumente la apuesta, todos los demás jugadores deberán realizar una nueva acción en la misma ronda. En caso de retirarse o abandonar, la partida prosigue con un jugador menos y, en caso de haber un único jugador restante, la partida finaliza con el jugador como ganador. Si más de un jugador avanza hasta la ronda final, cada jugador revela sus 2 cartas y, junto a las 5 cartas compartidas mostradas por el crupier, aquel que tenga la combinación más alta[4] gana la partida. Finalmente, cabe destacar que al inicio de cada partida se determina un jugador con la ciega pequeña y ciega grande para que exista una apuesta inicial antes incluso de ver las cartas.