UNIVERSIDAD POLITECNICA DE VALENCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Desarrollo de aplicaciones integrando robótica y visión en un robot industrial KUKA para demostrar sus capacidades

PROYECTO FINAL DE CARRERA

Pablo Muñoz Rodríguez

Directores:
Martín Mellado Arteche
Carlos Ricolfe Viala

Tutor Empresa:
Víctor Peiró Torres

SEPTIEMBRE 2011

Contenido

1 Introducción

Este documento describe el trabajo realizado en el Proyecto Final de Carrera con titulo *Desarrollo de aplicaciones integrando robótica y visión en un robot industrial KUKA para demostrar sus capacidades* realizado por Pablo Muñoz Rodríguez, cursado en la Escuela Técnica Superior de Informática de la Universidad Politécnica de Valencia.

Motivación y entorno

El proyecto trata sobre el desarrollo de una aplicación práctica para demostrar las capacidades del robot KR 5 SCARA R550 Z320 de KUKA. Se pretende que la aplicación pueda ser usadas por la empresa en ferias y eventos de esta clase para dar a conocer las características del robot y que los posibles clientes puedan verlo en funcionamiento.

Figura 1: KR 5 SCARA R550

Para el desarrollo del proyecto, la empresa ha puesto a disposición un robot de cara al diseño de las demos, donde se vean sus características y puntos fuertes en funcionamiento. Se integrarán las tecnologías de visión y robótica, ya que los movimientos que efectúe el robot dependerán de la situación de los objetos de su entorno, situación que captamos mediante una webcam y, tras ser procesada adecuadamente, se emiten las órdenes de movimiento correspondientes al robot.

El proyecto se ha desarrollado en las instalaciones de la empresa IMASD, una ingeniería especializada en automatización industrial que trabaja con KUKA.

Objetivos

El objetivo principal del proyecto es el desarrollo de una aplicación de demostración de funcionamiento del robot. En esta aplicación el robot actúa como croupier en un juego similar a la ruleta.

La demo está compuesta por varias aplicaciones trabajando paralelamente e intercambiando información entre ellas. La aplicación principal ha sido desarrollada utilizando el lenguaje de programación C++ y con ayuda de la librería OpenCV para trabajar con todo lo relacionado con la parte de visión.

Inicialmente se pensó que las aplicaciones se ejecutarían en un PC conectado al robot, pero más adelante se decidió aprovechar el mismo PC que hace de unidad de control del robot para contener las aplicaciones de demostración y así facilitar su posterior uso en ferias.

Definiciones y siglas

- **Automatización Industrial:** es el uso de sistemas o elementos computarizados y electromecánicos para controlar maquinarias y/o procesos industriales sustituyendo a operadores humanos.
- **Croupier:** es un empleado de los casinos cuya función consiste en controlar las apuestas del juego. En cada juego se atendrá exclusivamente a las reglas que para el mismo aplique el casino, por lo que su trabajo es mecánico, sin ninguna capacidad de iniciativa.
- **Ethernet:** es un estándar de comunicación de datos para redes de área local (LANs). Permite el intercambio de información entre los dispositivos conectados a la red.
- **KRL**: KUKA Robot Language. Lenguaje utilizado para escribir los programas que ejecuta la unidad de control de los robots de KUKA.
- **KUKA**: fabricante de robots industriales y de soluciones para automatización de fábricas con robots.
- **OpenCV**: Open Source Computer Vision es una librería de funciones para la programación con visión por computador en tiempo real. Se encuentra en continuo desarrollo y es de uso gratuito.
- **Pixel**: es la menor unidad homogénea en color que forma parte de una imagen digital. Son los puntos de color que componen una imagen.
- **Robot**: es un agente mecánico inteligente que puede realizar tareas por sí solo, o con orientación.
- **SCARA**: Selective Compliant Assembly Robot Arm.
- **XML**: Extensible Markup Language. Es un conjunto de reglas para la codificación de documentos en forma legible por una máquina.

2 Equipamiento

A continuación analizaremos las diferentes herramientas que hemos utilizado para la realización del proyecto.

Hardware

Desde el punto de vista físico disponemos de un robot SCARA de KUKA con su unidad de control, una ventosa que se le ha montado como herramienta y una webcam acoplada al brazo del robot.

Figura 2: Robot, unidad de control y panel de control del robot

El robot consiste en un brazo articulado con tres ejes angulares (dos rotacionales y uno traslacional) y uno prismático. Tiene un peso de unos 20 kg y puede trabajar con cargas de hasta 5 kg.

Totalmente desplegado el robot tiene unas dimensiones aproximadas de 82 cms de alto por 76 cms de largo. Su eje rotacional principal tiene un campo de trabajo de 310º aunque, gracias a los otros ejes, la herramienta alcanza un campo de trabajo de 360º. Su alcance es de 55 cms con una exactitud de posicionamiento de menos de 0,02 mm.

Teniendo en cuenta que el robot se encuentra sobre una mesa y ésta junto a una pared, para poder utilizarlo de forma segura nos hemos visto obligados a limitar su espacio de trabajo. De esta forma impedimos que pueda golpearse contra la pared mientras realice algún movimiento, o simplemente impedimos que pueda moverse a una zona en la que no nos interese que pueda posicionarse.

Al hacer uso de las mecánicas SCARA más rápidas del mundo, el robot es muy rápido, con una velocidad máxima de 7,1 m/s.

El robot se programa desde el panel de control de KUKA que va conectado directamente a su unidad de control. Desde este panel de control se pueden cargar y editar programas y también crearlos desde cero, introduciendo las órdenes línea a línea. Sin embargo para nuestro proyecto se ha optado por utilizar un PC externo para hacer los programas y posteriormente llevar los ficheros ejecutables generados a la unidad de control del robot y lanzarlos desde ahí.

La herramienta seleccionada para el proyecto ha sido una ventosa de succión por aire comprimido.

Para el correcto funcionamiento de las aplicaciones se simulará una conexión Ethernet de la unidad de control del robot consigo misma y se enviarán al robot órdenes por medio de cadenas XML, que es la forma en que el robot se comunica con los dispositivos que le conectamos vía Ethernet.

Software

Se han desarrollado varias aplicaciones en el lenguaje de programación C++ para lo cual se ha utilizado el MS Visual Studio 2008. También se ha hecho uso de la librería grafica OpenCV para trabajar con todo lo relacionado con la parte de visión. Usamos esta librería porque dispone de interfaces en C++, que es el lenguaje elegido para esta parte al ser la velocidad de proceso de las imágenes y la respuesta generada tras el análisis de éstas un factor crítico en la aplicación. Además, se ha escrito un programa en el lenguaje KRL, que es el que contiene las órdenes de movimiento del robot y para ello hemos utilizado el KRC Editor, programa facilitado por los técnicos de KUKA.

El robot puede funcionar en modo manual, en el cual elegimos qué eje queremos mover en cada momento o también podemos elegir moverlo en base a coordenadas del mundo real, o en modo automático. En este ultimo modo, que será el que utilizaremos nosotros, se selecciona un programa y se lanza su ejecución. El programa debe estar en la unidad de control y además debe estar escrito KRL, un lenguaje de programación basado en PASCAL que se ha ampliado con las órdenes propias de movimiento de los robots de KUKA.

Como ya hemos dicho antes, el robot se comunica con otros dispositivos mediante cadenas XML, así que el método de funcionamiento que vamos a utilizar será el siguiente:

Las aplicaciones escritas en C++, que hacen uso de openCV, obtienen las imágenes de la webcam, las procesan y calculan coordenadas de puntos importantes para el funcionamiento de aplicación, puntos a los cuales deberá moverse el robot. Entonces se emiten las cadenas XML que contienen estos puntos y el programa escrito en KRL las recibe y genera las órdenes de movimiento adecuadas.

Es importante destacar que el lenguaje KRL trabaja directamente con el hardware del robot, haciendo muy sencillo para el programador poder controlar el movimiento de éste, ya que solo hay que especificar el punto destino al cual queremos que se desplace y el tipo de trayectoria a seguir para alcanzarlo (lineal, punto a punto o arco circular) y del resto se encarga el programa. Así mismo, la forma en que se tratan los puntos de destino en KRL hace que no existan ambigüedades a la hora de alcanzarlos, ya que además de las coordenadas del punto también especifica la posición de cada uno de los ejes del robot en ese punto.

Las características de velocidad y fiabilidad de los programas escritos en C unido a la robustez de los programas en KRL hacen que el sistema completo que forma nuestra aplicación aproveche al máximo las capacidades del robot.

3 Especificación de requisitos

Funciones del producto

Como ya se ha comentado anteriormente, la aplicación pretende servir como demostración de capacidades del robot y para ello éste funcionará como croupier del juego de la ruleta adaptado para el proyecto.

Básicamente servirá para ver en funcionamiento al robot manipulando pequeños objetos, por lo que servirá como demostración de precisión de su herramienta. También se combinaran varios elementos desarrollados en lenguajes de programación con características diferentes, por lo que habrá que pensar, programar y utilizar algún mecanismo de comunicación entre ellos.

Características del usuario

El sistema está pensado para servir de demostración en ferias del robot de KUKA, así que distinguimos entre usuario operador del robot y usuario visitante.

- El operador del robot podrá poner en marcha el sistema, ejecutando las diferentes aplicaciones que lo forman y jugar una partida al juego de la ruleta con el robot. Generalmente será un empleado de la propia empresa, aunque no es necesario que tenga conocimientos técnicos sobre el funcionamiento del robot.

- El usuario visitante no maneja el robot directamente, pero si puede participar en el juego haciendo apuestas a algún numero o color. Cualquier persona puede jugar una partida con el robot, ya que solo tendrá que elegir un número o un color.

Restricciones

Para poder ejecutar nuestras aplicaciones en el robot se debe acceder a su disco duro bajo la interfaz gráfica de su sistema operativo Windows XP. Esto sólo es posible configurando el usuario del robot como Experto y para hacer esto se nos pedirá la contraseña correspondiente.

Además, si las aplicaciones no se encuentran en el mismo robot sino en un PC externo, éste deberá estar conectado al robot mediante una red Ethernet configurada adecuadamente, tal y como se explica en el capitulo correspondiente.

Requisitos de contenido

La aplicación es una demostración del funcionamiento del robot, por lo tanto no tiene ningún contenido que puedan consultar los usuarios.

Requisitos de interfaz

La interfaz visual de la aplicación es prácticamente inexistente ya que la partida se desarrolla sobre una mesa de juego y posiblemente ni siquiera haya una pantalla.

Interfaz hardware:

El panel de control de KUKA, que va conectado directamente a la unidad de control del robot, será manejado únicamente por el operador del robot. Además, si la aplicación se ejecuta desde un ordenador externo, éste también será manejado por el operador del robot.

El usuario visitante en ningún caso puede acceder a la consola de control del robot ni interactuar con él, aparte de realizando sus apuestas durante la propia partida. No hay ninguna otra forma de acceder al sistema.

Interfaz software:

La aplicación de croupier está formada por varios programas. Los que se encargan de gestionar la comunicación, la parte de visión y la escritura en ficheros XML funcionan en cualquier plataforma que soporte Windows. El programa en KRL que se encarga del movimiento del robot está diseñado para funcionar en una unidad de control KR C2 de KUKA.

Interfaz de comunicaciones:

Los interfaces de comunicación son el estándar TCP/IP sobre los que funciona Ethernet. Este estándar es el que se encarga de la transmisión de cadenas XML entre las diferentes aplicaciones que componen el sistema.

Requisitos funcionales

Usuario visitante

- Apostar a un número de la ruleta (o a un color).

Usuario operador del robot

Tendrá las mismas funciones que el usuario visitante (nada impide que pueda jugar con el robot) más las que se citan a continuación.

- Lanzar las diferentes aplicaciones para su ejecución.

Usuario administrador

Tendrá las mismas funciones que el usuario operador del robot más las que se citan a continuación.

- Editar y borrar programas.

4 Análisis

En este capítulo se describe el funcionamiento y los contenidos de la aplicación. Para realizar el análisis del sistema se usará el modelo UML (Unified Modeling Language) que es uno de los lenguajes de modelado que más se utiliza en la actualidad. Se usa para visualizar, especificar, construir y documentar un sistema software. Es un lenguaje para especificar y no para describir métodos, por lo tanto no se mostrarán detalles de implementación, si no que corresponde a un nivel más abstracto para que cualquier usuario pueda comprenderlo.

Hay muchos tipos de diagramas en UML, pero sólo usaremos 2 de ellos:

- Diagrama de casos de uso: representa las distintas funcionalidades.
- Diagrama de clases: representa los contenidos de información.

Casos de uso

Este modelo es una de los que existen para representar las funciones o procesos de una aplicación. En él se trata de describir las posibles acciones en casos que puede llevar a cabo el usuario en el contexto de la aplicación.

Los diagramas de casos de uso basados en UML permiten una descripción rápida e intuitiva que pueda ser fácilmente evaluada por cualquier usuario. Sirven para especificar la comunicación de un sistema mediante su interacción con los usuarios.

El diagrama de casos de uso se representa mediante un cuadrado, en el cual los casos de uso están en el interior y los actores fuera. El diagrama describe la interacción entre el actor y el sistema.

En la siguiente figura se muestra la relación entre los actores de la aplicación.

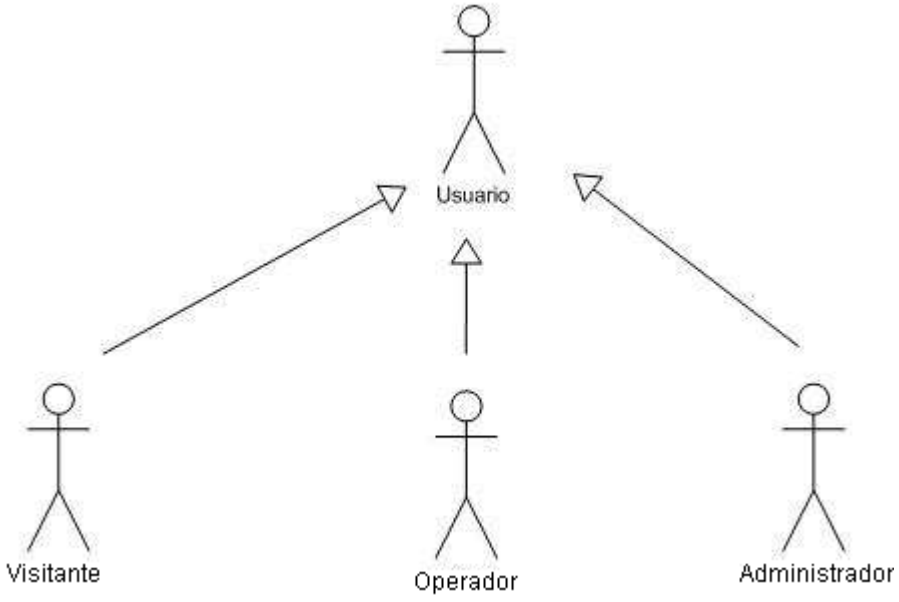


Figura 3: Diagrama de casos de uso de los actores del sistema

Los actores Visitante, Operador y Administrador heredan de Usuario. En las figuras siguientes se muestran los diagramas de uso para cada usuario.

Usuario: la entidad usuario puede participar en el juego con el robot realizando apuestas. Es el actor que generaliza a todos los demás, que posteriormente se detallarán. El caso de uso se encuentra en la figura 4.

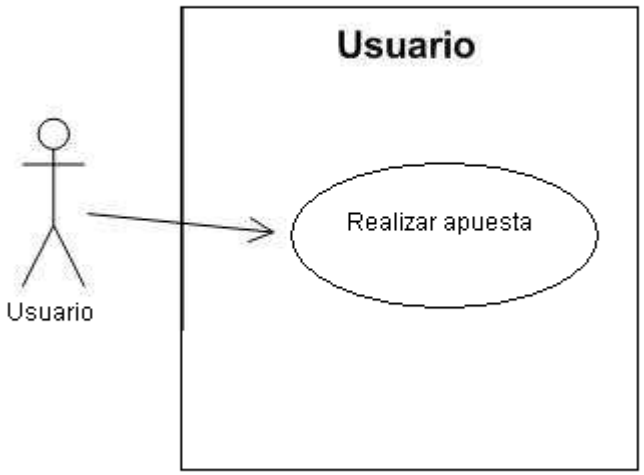


Figura 4: Diagrama de caso de uso del usuario

Visitante: hereda las funciones del actor usuario. Es un visitante de la feria en la que esté expuesto el robot. Su caso de uso particular no aporta nuevas opciones sobre las del usuario genérico.

Operador: puede realizar las mismas funciones que el usuario y su caso de uso se muestra en la figura 5. Controla la ejecución de la aplicación en el robot y previamente se ha autentificado como Experto en el sistema del robot. Este usuario posee el control de la partida, lo que significa que puede:

- Iniciar la partida, cuando se hayan cumplido las condiciones para el inicio del juego será el encargado de ejecutar la aplicación.
- Controlar las apuestas, colocando las fichas en el tablero de apuestas según los números que hayan elegido los jugadores.

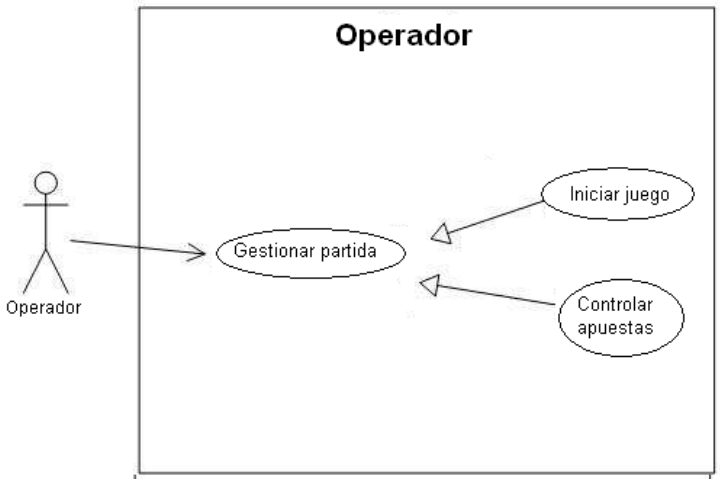


Figura 5: Diagrama de caso de uso del operador

Administrador: el administrador del sistema es el encargado de realizar el mantenimiento de los programas del robot. Tiene todos los derechos posibles en su sistema, es decir, que puede consultar, crear, editar o eliminar programas. Se muestra su caso de uso en la figura 6.

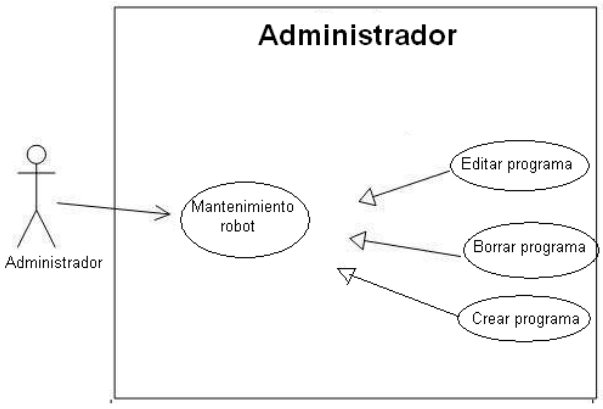


Figura 6: Diagrama de caso de uso del administrador

Diagrama de clases

El diagrama de clases permite representar las principales entidades que intervienen en la aplicación, mostrando sus clases y sus relaciones. Dichas entidades o clases deben hacer referencia a elementos característicos, como métodos, atributos y las relaciones entre clases con su cardinalidad.

En el diagrama de clases la entidad principal es el robot. De esta entidad surgen los usuarios y la aplicación de la ruleta. Esta última está dividida en 5 partes (reconocimiento de imagen, análisis de las apuestas, programas, calculo de puntos destino y generación de las ordenes de movimiento).

De la parte de los usuarios, como ya se ha mencionado anteriormente, hay 3 tipos (visitante, operador y administrador). Todo usuario podrá apostar en una partida con el robot. El usuario operador podrá iniciar la aplicación y el administrador podrá gestionar los diferentes programas que contiene el robot y que forman la aplicación.

En la figura 7 se muestra el diagrama de clases de la aplicación, pero no se han especificado los métodos, ni los atributos para una mayor claridad.

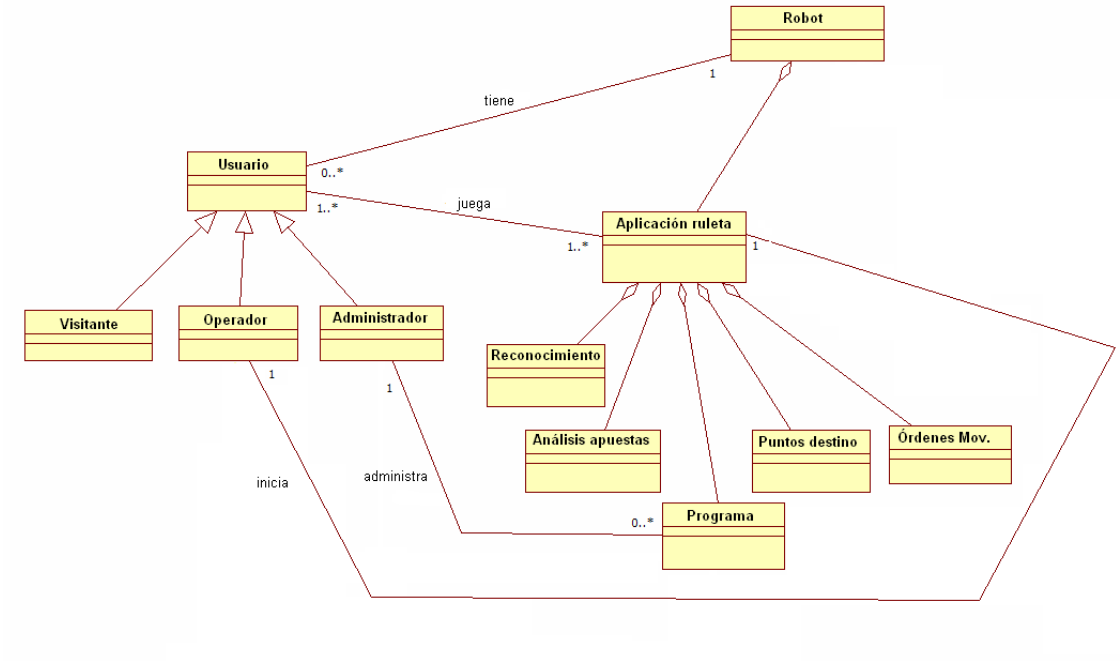


Figura 7: Diagrama de clases

5 Puesta a punto

En este capítulo vamos a explicar brevemente en qué consiste el juego de la ruleta de los casinos y de qué forma hemos simulado el tablero para que se pueda jugar con el robot. También veremos algunos conceptos básicos de visión por computador que nos ayudaran a comprender mejor el siguiente punto de la memoria, en el que ya se explican las diferentes aplicaciones que hemos desarrollado y de las que se compone nuestro sistema.

Introducción al juego de la ruleta

A continuación explicaremos el juego de la ruleta americana para así entender qué debe hacer la aplicación de croupier. Tras una breve introducción, hablaremos un poco del tablero de juego y de los requisitos que deben cumplirse para que la aplicación funcione correctamente.

La ruleta es un juego de azar típico de los casinos. Consta de una ruleta o plato, una bola y el tapete de las apuestas. El plato de la ruleta tiene números que van desde el 0 al 36 dispuestos siguiendo unas normas particulares. La ruleta americana además tiene una casilla extra, el 00. En la ruleta francesa o europea solo hay una casilla con el 0. Los jugadores pueden apostar a cualquier número de la ruleta excepto al 0 y 00, reservados para la casa.

Las apuestas se realizan en un tapete con números, sobre el que se colocan fichas que son intercambiables por dinero. Sobre el plato de la ruleta en movimiento se lanza una bola que gira en sentido contrario al sentido de la ruleta. Al perder velocidad, la bola cae y entra en una de las casillas correspondiente a un número, decidiendo así cuál es el ganador.

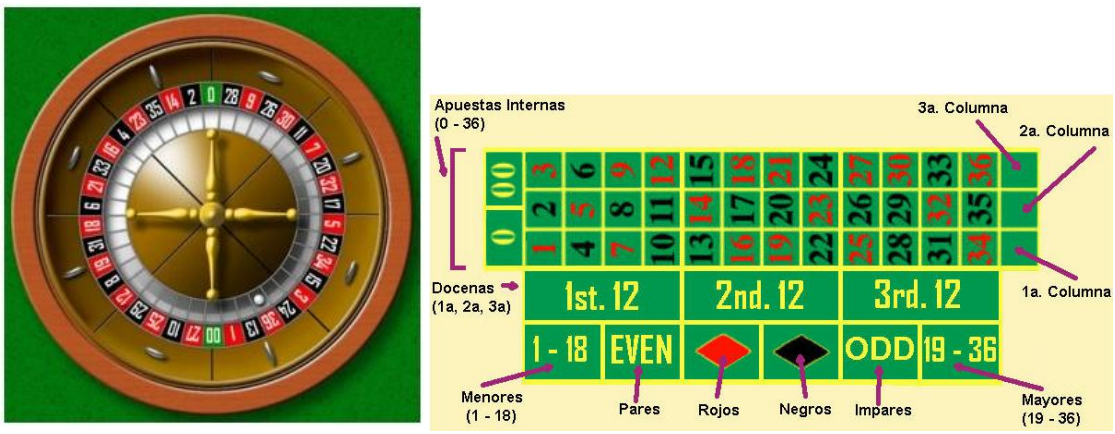


Figura 8: Ruleta americana y tapete de apuestas

El croupier es la persona que está al cargo de la mesa de juego y su trabajo consiste en hacer girar la ruleta, lanzar la bola, y repartir las fichas de las apuestas según se desarrolle el juego.

Hay muchas formas de apostar en el juego. Se puede apostar a un número concreto, a una pareja, tríos, y también a columnas, docenas, rojo o negro, par o impar…. Evidentemente cuanto más simple es la apuesta menor es la probabilidad de ganar y también el premio en caso de hacerlo es mayor.

Por ejemplo en caso de apostar por un número concreto, pongamos el 5, la probabilidad de ganar es de 1/36 = 0.028 y si lo hacemos obtenemos la cantidad apostada multiplicada por 35. En cambio si apostamos al color negro, la probabilidad de ganar será de 17/36 = 0.47 y en caso de ganar obtendríamos únicamente el doble de la cantidad apostada.

Para más información consultar el anexo I, correspondiente a las reglas del juego de la ruleta.

El tablero de juego

En el proyecto hemos tenido que adaptar el juego original para que el robot pueda funcionar como croupier. Para empezar, en el espacio de trabajo del robot hay dos zonas claramente diferenciadas: el tablero de juego y la zona de apuestas:

- El tablero de juego consiste en una superficie cuadrada con múltiples huecos o casillas (5x6), en los cuales caerá la bola. En cada casilla hay un número, del 0 al 29 y un color, rojo o negro, excepto la casilla correspondiente al número 0 que será de color verde. Los números se han distribuido en un orden que sigue las reglas del juego de la ruleta original con algunas modificaciones, ya que nuestra ruleta no es redonda y en lugar de tener 2 vecinos cada número tiene 4. Además nuestra ruleta solo tiene 30 números en lugar de los 38 de la original:

    - Los números rojos y negros deben estar alternados.
    - Los números negros son pares.
    - Partiendo el tablero por la mitad, con una línea imaginaria entre el 0 y el 9, en cada una de las mitades debe haber la misma cantidad de números que pertenezcan a cada una de las 2 quincenas (15 números).

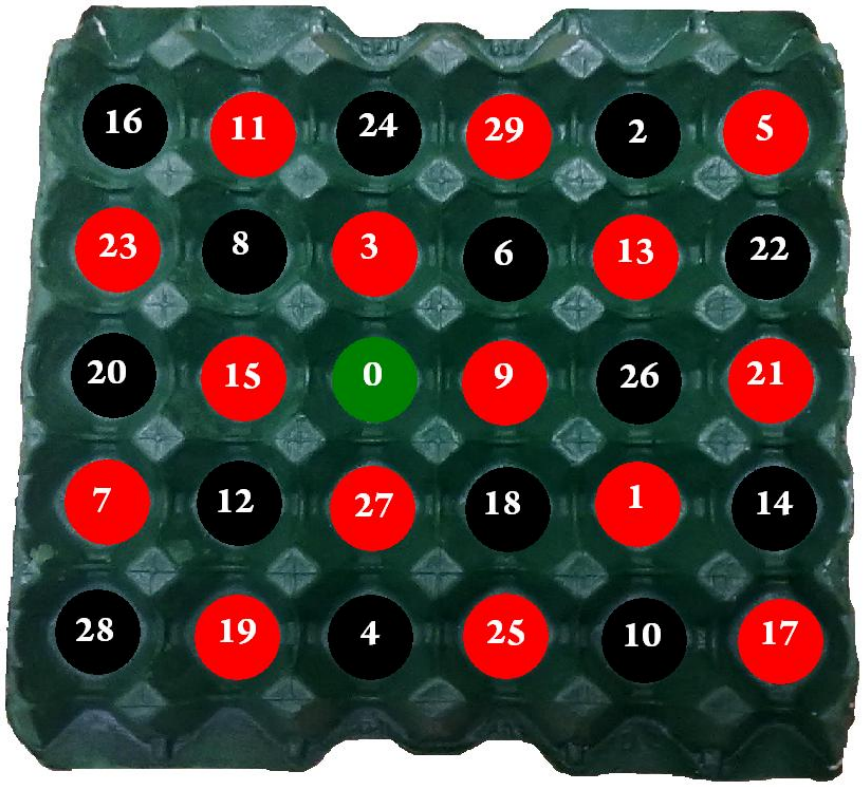


Figura 9: Tablero de juego

- Además del tablero de juego también disponemos de un tapete con casillas numeradas correspondientes a los diferentes números de la ruleta y sobre el cual los jugadores hacen sus apuestas.

- La bola que utilizaremos es una pelota de ping-pong que, debido a su tamaño, rigidez y peso resulta ideal para nuestro propósito, ya que puede ser fácilmente succionada y mantenida por la ventosa del robot y además, encaja perfectamente en las casillas del tablero de juego.

Anteriormente hemos dicho que hay múltiples apuestas posibles en el juego, pero por comodidad para nuestro proyecto vamos a limitarlas a números individuales o colores.

Para que la aplicación de croupier funcione correctamente, en el momento de iniciar la partida la bola debe estar en una posición inicial que hemos definido en el robot y las fichas deben estar colocadas en el tapete de apuestas conforme a las apuestas que hayan hecho los jugadores. Cuando la demo termina, la bola está en alguna casilla (aleatoria) del tablero de juego y las fichas del tablero de apuestas se encuentran en el tablero, en la zona correspondiente a un jugador o en la zona correspondiente a la banca, según cual haya sido el resultado de la partida.

Todas las simplificaciones que hemos hecho en nuestro juego, que concretamente son las que afectan al tipo de apuestas q podemos hacer o las reglas q sigue la numeración de las casillas del tablero se deben a que el objetivo de la aplicación de croupier es el de demostrar las capacidades del robot en cuanto a

velocidad, precisión,... y no se pretende desarrollar una aplicación de juego profesional que sea utilizada por ejemplo en un casino con una ruleta real.

Visión artificial y reconocimiento de objetos en OpenCV

La visión artificial, también conocida como visión por computador (del inglés *computer vision*), es un subcampo de la inteligencia artificial. El propósito de la visión artificial es programar un computador para que "entienda" una escena o las características de una imagen.

Para el proyecto nos vamos a centrar en la detección, localización y reconocimiento de objetos en imágenes y, en concreto, las imágenes serán las obtenidas por medio de una webcam montada en el brazo del robot y los objetos que buscaremos serán la bola y las fichas de las apuestas.

Una imagen digital no es más que una matriz, o array bidimensional, de números. En nuestro caso, esa imagen se ha obtenido por medio de un dispositivo de conversión analógica-digital, la webcam.

A las imágenes digitales se las suele caracterizar por su altura y anchura (en pixels) y por su profundidad de color (en bits por pixel), que determina el número de colores distintos que se pueden almacenar en cada pixel, y por lo tanto, en gran medida, la calidad del color de la imagen. Un pixel es cada uno de los puntos que forma una imagen.

En las imágenes digitales que utilizamos en la aplicación se está utilizando una profundidad de color de 32 bits. Esto quiere decir que estamos usando un octeto (2^8 = 256 valores) para representar cada uno de los colores primarios, rojo, verde y azul (256x256x256) y un octeto extra para el denominado canal alfa, q contiene información acerca de la transparencia de la imagen.

Sobre estos números que representan el valor del color en cada pixel podemos aplicar las mismas operaciones que sobre cualquier número y obtendremos así algunos resultados interesantes. Cada una de estas operaciones tendrá un significado, utilidad y aplicaciones específicos.

Para continuar estudiando las imágenes digitales, es necesario conocer lo que es el histograma de una imagen: representa las frecuencias de los diferentes valores de gris en la imagen. El histograma nos ayuda a decidir cuál es el procesamiento más adecuado para **mejorar la calidad** de una imagen

- Tanto **cualitativamente** (qué operación aplicar),

- Como **cuantitativamente** (en qué cantidad).

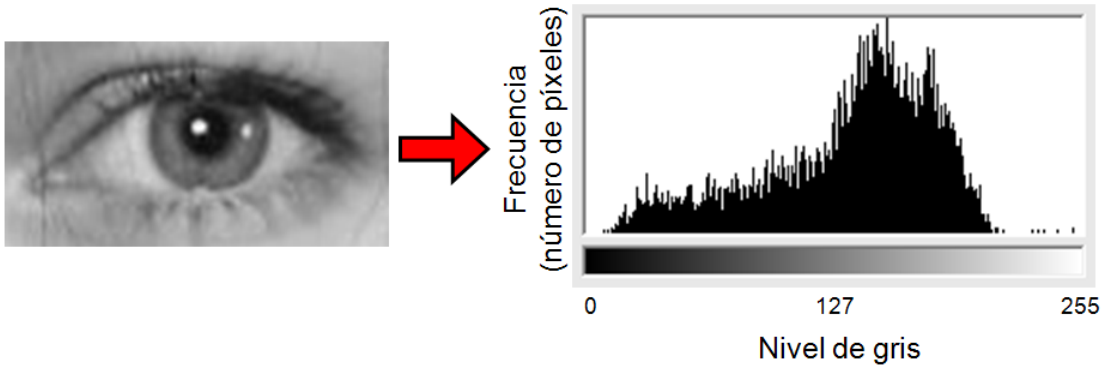


Figura 10: Histograma de una imagen digital

Una buena imagen debería producir un histograma más o menos uniforme y repartido en todo el rango de valores: Si los valores del histograma son muy bajos, la imagen asociada será muy oscura y si, por el contrario, el histograma de una imagen tiene la mayoría de sus valores muy altos, ésta será demasiado clara. Así mismo, si los valores del histograma se encuentran concentrados alrededor de un rango de valores de gris, la imagen tendrá poco contraste.

Las imágenes en color, como las que vamos a utilizar, poseen histogramas de color, que son similares a los anteriores pero en lugar de una escala de grises tienen tres histogramas, uno para cada uno de los canales de los colores básicos.

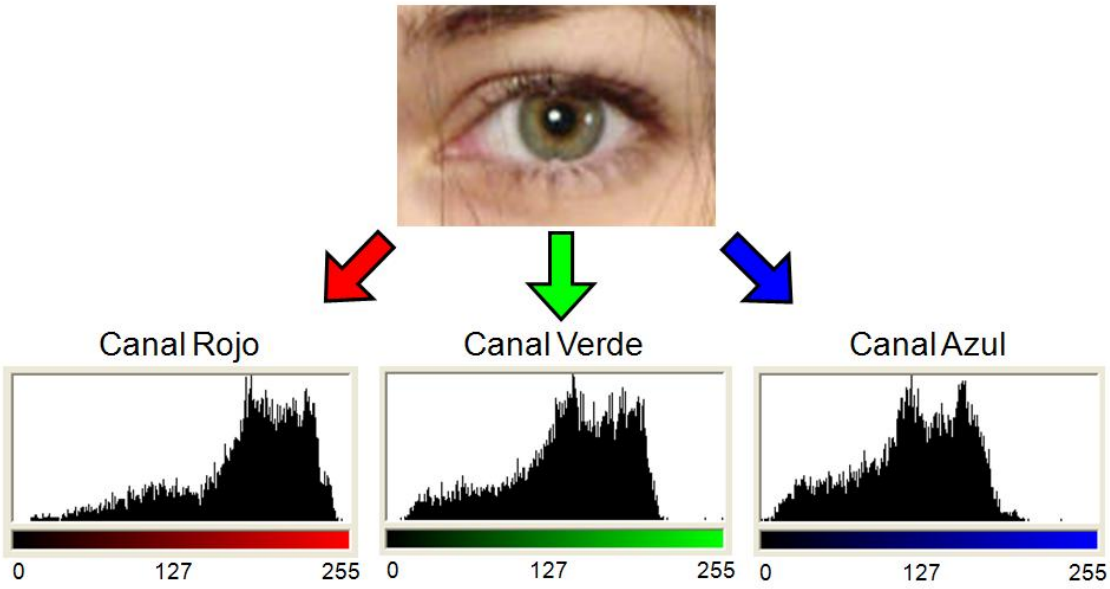


Figura 11: Histogramas de una imagen digital en color

Antes hemos dicho que se puede operar sobre los pixeles de una imagen y que el histograma nos ayuda a decidir cuál es el operador más conveniente para nuestra imagen. Ahora veremos cómo hacerlo.

Volviendo al supuesto de una imagen cuyo histograma presentara valores bajos, es decir, una imagen con tonos demasiado oscuros, la mejor operación a aplicar para mejorar la calidad de la imagen sería la que ayudara a repartir los valores del histograma de la imagen por todo el rango de la escala.

Las operaciones elementales con píxeles se aplican de manera individual a cada uno de los píxeles de la imagen.

La suma de cierto valor “a” a todos los píxeles de una imagen provoca que el histograma de dicha imagen se desplace a la derecha y además la imagen aumenta su brillo en la cantidad indicada en “a”. En imágenes en color, la suma se realiza sobre los tres canales (R, G y B) y con el mismo valor en los tres.

Al restar una cantidad “a” a los píxeles de una imagen conseguimos el efecto contrario al que acabamos de ver: el histograma se desplaza a la izquierda y la imagen decrementa su brillo en la cantidad indicada en “a”.

Al multiplicar un valor “b” a los píxeles de una imagen conseguimos aumentar la intensidad de la imagen en “b” y su histograma se “estira” hacia la derecha. Hay que tener cuidado de no saturar la imagen, ya que perderíamos información.

La operación contraria a la multiplicación por una constante es la división (o multiplicar por 1/b) y en este caso el resultado que obtendremos será que el histograma de la imagen se “encoge”.

Por último, la umbralización de imágenes permite convertir la imagen en binaria, o recortar cierto rango de valores. Para conseguir esto se aplica una función de transformación al histograma y según la función que utilicemos podemos binarizar la imagen eligiendo un valor como frontera, cortar un rango de valores y mantener el resto o incluso seleccionar un determinado rango de valores del histograma.

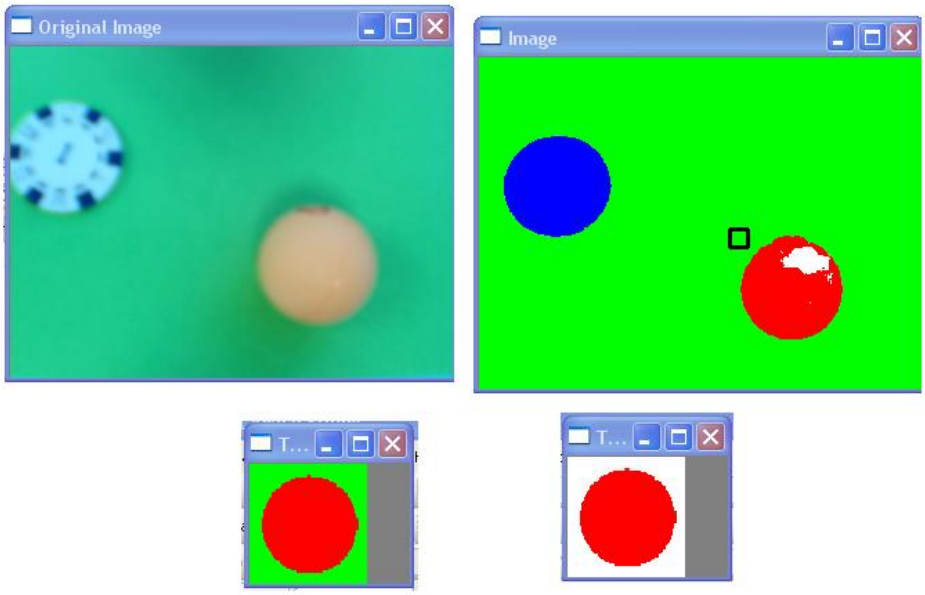


Figura 12: Imagen obtenida de webcam, imagen umbralizada y patrones

Existen operaciones más complejas sobre el histograma de una imagen digital, pero para nuestro proyecto nos basta con las que hemos explicado.

En OpenCV existen múltiples operaciones de procesamiento global. Éstas pueden ser unarias o binarias según si reciben una o dos imágenes de entrada para producir una de salida. Hay que tener en cuenta que muchas de las operaciones trabajan con ROI, COI y mask**:** no operan sobre toda la imagen sino sólo sobre una parte concreta (rectángulo, canal o máscara de interés).

Nosotros vamos a utilizar operaciones unarias y generalmente las aplicaremos a determinadas regiones de interés de la imagen. La más importante de todas es la función cvMatchTemplate, que recibe una imagen y un patrón (otra imagen con algún objeto) y busca coincidencias (matchings) del patrón sobre la primera imagen. Se utilizan seis métodos distintos para hacer esta comparación, empezando con el cuadrado de la diferencia entre el patrón y la imagen. El resultado de esta operación es una imagen del mismo tamaño que la imagen sobra la que hemos buscado el patrón que contiene un 0 en los puntos donde hay coincidencia y valores mayores en los puntos donde no hay una buena coincidencia.

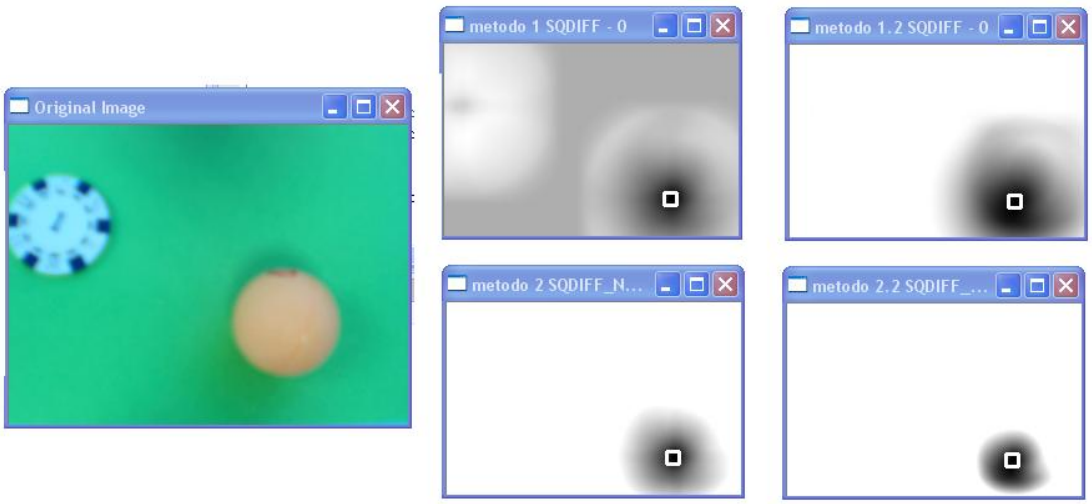


Figura 13: Ejemplo de uso de cvMatchTemplate

La figura 7 es el resultado de comparar el patrón de una pelota de ping-pong con la imagen en color de la izquierda. Se ha mostrado el resultado de la comparación y se puede observar que se han generado nuevas imágenes que presentan regiones oscuras en la zona que corresponde a la posición de la pelota, y el resto de la imagen es blanca o gris clara.

En nuestro proyecto utilizamos esta función para, a partir del análisis de las imágenes resultantes de la comparación, obtener las coordenadas en las que se encuentra el objeto que estamos buscando, ya sea este una bola, una ficha de apuestas o un robot móvil y poder calcular la posición de destino del próximo movimiento del robot.

6 Desarrollo

Para la realización del proyecto se han desarrollado aplicaciones en C++ que se encargan de la parte de visión y calculan las coordenadas donde deberá moverse el robot en cada momento, un programa en KRL que recibe coordenadas de las aplicaciones anteriores y emite las ordenes de movimiento correspondientes, y una aplicación en C# que se encarga de la comunicación entre las aplicaciones que generan las coordenadas y el programa en KRL mediante el envío de cadenas XML.

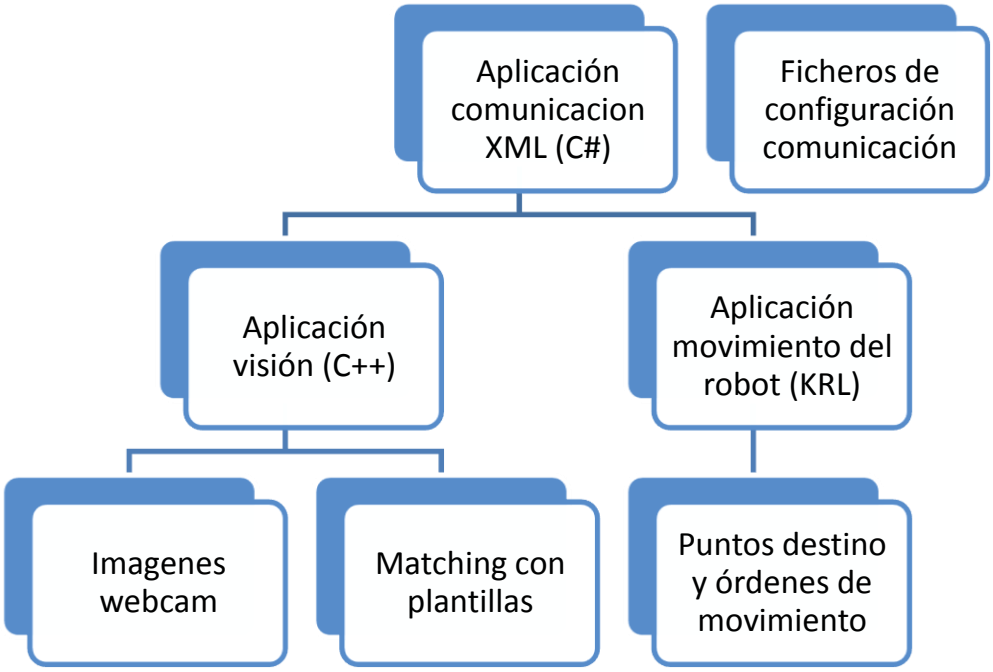


Figura 14: Diagrama de comunicación

La estructura del capítulo irá dividida según estas partes.

Ficheros de configuración de la comunicación

Para empezar vamos a explicar cómo funciona la comunicación XML con el robot. Éste cuenta con varios ficheros de configuración, que hay que modificar adecuadamente, para que se pueda realizar la comunicación.

En primer lugar, en el fichero C:\KRC\ROBOTER\INIT\XmlApiConfig.XML (escrito siguiendo la sintaxis XML) hay que definir un nuevo canal de comunicación y asignarle la dirección IP del dispositivo con el que vayamos a comunicarnos a través de este canal. Tiene especial importancia el nombre que le demos al canal, ya que lo usaremos posteriormente para continuar con la configuración de la comunicación.

```
<?xml version="1.0"?>
<!--  NAME                                          -->
<!--    C:\Krc\Roboter\Init\XmlApiConfig.XML        -->
<!--                                                -->
<XmlApiConfig>
  <!--  -->
  <!--  -->
  <XmlApiParam InitOnce="false"/>
  <!--  -->
  <!--  -->
  <Channel SensorName="StackCam" SensorType="dvt600">
     <TCP_IP IP="192.168.0.29" Port="5001" Route="true" Map-
Port="5001"/>
  </Channel>
</XmlApiConfig>
```

Figura 15: Ejemplo de fichero XmlApiConfig.XML

El siguiente paso es escribir un fichero en el directorio del robot C:\KRC\ROBOTER\INIT cuyo nombre tiene que ser exactamente el mismo que hayamos puesto al canal de comunicación y en el cual definiremos la estructura que queremos recibir.

Por ejemplo, en nuestro proyecto el canal que utilizamos se llama Pablo, así que tenemos un fichero llamado Pablo.xml con la estructura de datos que se envía en la comunicación con el robot. Dicha estructura contiene el nombre de cada uno de los campos y su tipo (real, entero, booleano, …) en el orden en que se van a recibir.

Los datos que se envían son siempre los mismos y cada vez que nos comunicamos con el robot se envían todos de nuevo. Por este motivo, conviene que la información que necesitemos enviar sea la mínima posible. En nuestro caso se envían 3 datos reales con las coordenadas (X, Y, Z) del próximo punto al cual deberá moverse el robot y un booleano que indicará si la ventosa debe estar activada o desactivada.

Si quisiéramos recibir datos que envíe el robot, deberíamos tener otro fichero con nombre Pablo+.xml escrito también en XML y que contendría la estructura de datos que enviaría el robot. Para nuestro proyecto no es necesario recibir ningún dato del robot, solo con enviarle datos nos basta, así que no comentaremos nada más sobre la recepción de datos del robot.

A continuación vemos un ejemplo de lo que podría ser el contenido de uno de estos ficheros:

```
<Elements>
  <Element Tag="ExData" Type="STRUCTTAG" Stacksize="5" />
  <Element Tag="ExData.TString" Type="STRING" Size="80" Stacksize="5"
/>
  <Element Tag="ExData.Position" Type="STRUCTTAG" Stacksize="5" />
  <Element Tag="ExData.Position.XPos" Type="REAL" Stacksize="5" />
  <Element Tag="ExData.Position.YPos" Type="REAL" Stacksize="5" />
  <Element Tag="ExData.Position.ZPos" Type="REAL" Stacksize="5" />
  <Element Tag="ExData.Temperature" Type="STRUCTTAG" Stacksize="5" />
  <Element Tag="ExData.Temperature.Cpu" Type="REAL" Stacksize="5" />
  <Element Tag="ExData.Temperature.Fan" Type="REAL" Stacksize="5" />
  <Element Tag="ExData.Ints" Type="STRUCTTAG" Stacksize="5" />
  <Element Tag="ExData.Ints.AState" Type="INTEGER" Stacksize="5" />
  <Element Tag="ExData.Ints.BState" Type="INTEGER" Stacksize="5" />
  <Element Tag="ExData.Boolean" Type="STRUCTTAG" Stacksize="5" />
  <Element Tag="ExData.Boolean.CState" Type="BOOLEAN" Stacksize="5" /
>
  <Element Tag="ExData.Frames" Type="STRUCTTAG" Stacksize="5" OnTag-
SetPort="1" />
  <Element Tag="ExData.Frames.XFrame" Type="FRAME" Stacksize="10" />
</Elements
```

Figura 16: Ejemplo de fichero SensorName.XML

Aplicación de comunicación XML

A continuación vamos a explicar la aplicación escrita en C# que se encarga del envío de cadenas XML al sistema del robot. Esta aplicación, aunque sencilla de programar, es fundamental para el funcionamiento de todo el sistema.

Figura 17: Aplicación de comunicación XML esperando a un cliente

Como ya hemos explicado anteriormente, el robot se comunica con los dispositivos que le conectemos a través del envío/recepción de cadenas XML que

contienen los datos que deseamos enviar/recibir. Por ello y dado que en nuestro proyecto intervienen varios dispositivos así como diferentes aplicaciones, es necesaria una comunicación entre todas las partes del proceso.

En la comunicación XML entre el robot y cualquier otro dispositivo/aplicación se envían siempre los mismos datos y, para simplificar al máximo el proceso de comunicación, de cara a obtener el menor tiempo de respuesta posible por parte del robot, hemos optado por escoger estos datos cuidadosamente para que la cantidad de información transmitida sea la menos posible.

La aplicación encargada de transmitir esta información consiste en un proyecto de MS Visual Studio con un programa escrito en C# y un fichero auxiliar (ExternalData.xml) escrito en XML con la información que se va a transmitir, en nuestro caso las variables que comentábamos anteriormente.

El funcionamiento de la aplicación es sencillo: se crea un socket en el puerto q le especificamos, y que previamente ha sido configurado en los ficheros del robot que han sido explicados en el apartado anterior, y se queda a la espera de que se conecte algún cliente, en este caso el robot. Cuando esto sucede, entra en un bucle de envío de datos permanente hasta que se termina la ejecución de la aplicación. En este bucle se lee el fichero ExternalData.xml y se transmite a través del socket.

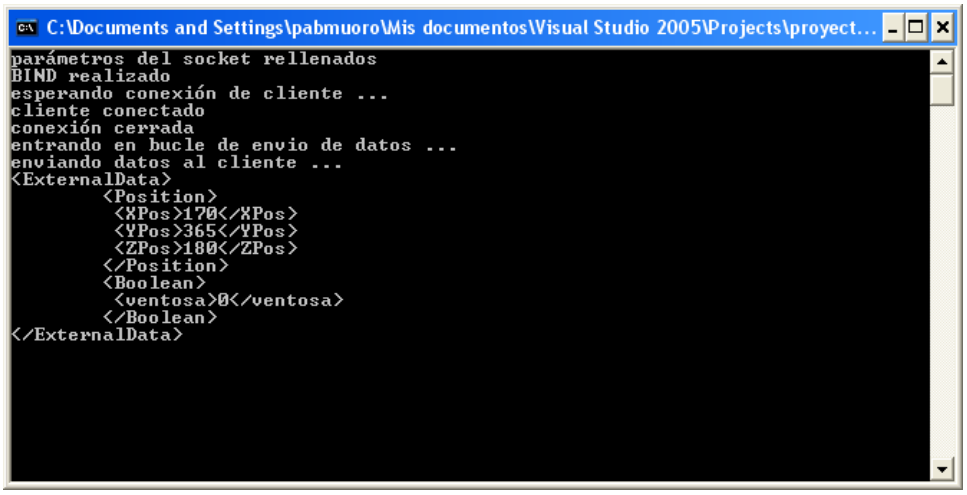


Figura 18: Aplicación de comunicación XML transmitiendo datos

Aplicación de movimiento del robot

Esta es la aplicación que se encarga de generar las ordenes de movimiento del robot. Está escrita en lenguaje KRL que es el q utiliza KUKA en sus sistemas y de todo el proyecto, es la única aplicación que mueve el robot directamente.

En KRL los programas se componen de 2 ficheros: uno con extensión src, que contiene las instrucciones del programa y otro con extensión dat, que contiene la declaración de todas las variables que se utilizan en el programa ya sean datos reales, cadenas de caracteres o incluso los puntos a los cuales moveremos el robot.

En los ficheros src el código se organiza mediante bloques, llamados FOLDs, que contienen una o varias instrucciones que guardan algún tipo de relación entre si. Por ejemplo, tenemos un FOLD al principio del programa que contiene la inicialización de diferentes variables, más adelante hay un FOLD que se encarga de definir la estructura que utilizaremos para la comunicación, otro FOLD para abrir el canal de comunicación, etc.

Para mover el robot, disponemos de tres órdenes de movimiento diferentes: directamente al punto por el camino más corto posible (PTP), en línea recta hacia el punto (LIN), y describiendo un arco circular hasta el punto (CIRC). Para nuestro proyecto hemos utilizado la primera de estas órdenes, pero en todas ellas el punto de destino debe proporcionarse por medio de una variable de tipo posición. En el KRL, existen dos formas diferentes de especificar una posición: puede ser de tipo POS, que contiene solamente la posición del robot, o de tipo E6POS, que contiene además la posición de 6 ejes externos. Esta última es la utilizada por defecto en los programas pero vamos a utilizar las de tipo POS siempre que podamos por simplificar, ya que podemos utilizar indistintamente una variable POS o una E6POS en las órdenes de movimiento sin ningún problema.

Una orden de movimiento PTP incluye el punto de destino (variable POS o E6POS), la velocidad a la que debe moverse el robot (en porcentaje) y datos sobre la herramienta que hay montada en el robot. Aparte de las órdenes de movimiento, el lenguaje KRL admite bucles FOR y WHILE, similares a los aceptadas por lenguajes de programación como C.

```
1  DEF pablodemo( )
2  INI
3
4  Initial variables
5  Build Dataset to send and receive
6
7  DECLare sensorname and open channel pablo
8  WHILE cstate == FALSE
9  ptp $pos_act
10
11 PTP P2  Vel= 100 % PDAT4 Tool[0] Base[0]
12 Holdon
13 Send Data by using easy send commands
14 ; Next commands get the data from the stacks and filled the KRC variables
15 ; Open the FOLD and take a look at: get frame...
16 Holdon
17 Get Data from stack, only if there are new values
18 Check position
19 Send data by using predefined XML document
20 Holdon
21 ENDWHILE
22 ;se cierra el canal
23 At the end the channel have to be closed
24
25 END
```

Figura 19: Estructura de un programa en KRL (con FOLDs cerrados)

Nuestro programa tiene un bloque inicial de inicialización de variables, la estructura necesaria para enviar y recibir información de la aplicación de comunicación XML, un bloque en el cual se declara y abre el canal de comunicaciones que hemos configurado en el fichero XmlApiConfig.XML y una orden de movimiento para posicionar la herramienta en el punto inicial.

En este punto hay que iniciar el proceso de comunicación: el programa envía una pequeña estructura que actuará como una señal de handshaking con el sistema externo (en nuestro caso, la aplicación de comunicación) y desde este momento estará listo para recibir cadenas XML.

Para empezar, la bola está siempre en una posición determinada y el robot se mueve a ese punto inicial y la recoge. A continuación se mueve sobre la zona de juego y aleatoriamente suelta la bola, que caerá en alguna casilla. Tras esto y por medio de la webcam se toma una imagen del tablero y se analiza el punto donde ha caído la bola para saber cuál ha sido el número ganador. A continuación el robot se mueve a la zona de las apuestas y analiza en dónde se encuentran las fichas de las apuestas. Con esta información y conociendo cuál ha sido el número ganador, el robot se mueve por aquellas casillas del tablero de apuestas que contengan fichas y las reparte según corresponda, a los jugadores o a la casa. Para más información sobre el reparto de apuestas consultar el anexo I, correspondiente a las reglas del juego de la ruleta.

Tras el reparto de las fichas al ganador o ganadores, el robot se mueve a una posición de reposo y se termina el programa.

Todas las posiciones a las cuales debe moverse el robot, excepto la inicial y la final que serán fijas, dependen de la situación de la partida y ésta será captada por la webcam. Las imágenes son procesadas por el programa encargado de la parte de visión y decidirá en cada caso cual debe ser la próxima posición a la cual deberá moverse el robot. Por este motivo, excepto las dos posiciones que hemos dicho todas las demás serán transmitidas en forma de coordenadas mediante cadenas XML al programa en KRL para que modifique los puntos de destino correspondientes y posteriormente el robot se mueva a ellos.

Las posiciones inicial y final serán variables de tipo E6POS mientras que todas las demás serán de tipo POS, ya que deseamos que la información transmitida en el proceso de comunicación sea la mínima posible y las variables de tipo POS solo necesitan 3 datos reales para definir una posición. En cambio si utilizáramos variables de tipo E6POS en todo el programa, además de los 3 números reales necesarios para definir las coordenadas del punto también tendríamos que transmitir las 6 posiciones de los ejes externos y el valor de las variables S y T, que son utilizadas para eliminar ambigüedades en el posicionamiento del robot.

Aplicación de visión

Por último, tenemos la aplicación escrita en C++, que hace uso de la librería OpenCV y que se encarga de gestionar toda la parte de visión del proyecto. Se ha escogido este lenguaje de programación por producir un código muy rápido, aspecto fundamental para el proyecto.

Esta aplicación empieza su ejecución partiendo de una situación en la cual la aplicación KRL que acabamos de explicar ha soltado la bola en algún punto aleatorio del tablero de juego. Su tarea está dividida en varias partes y cada una de ellas proporciona información necesaria para la siguiente:

- Desde la posición inicial, obtener imagen:

  El robot se encuentra con su herramienta en la posición de inicio y la webcam enfoca el tablero de juego. Por medio de las funciones correspondientes de la librería de OpenCV se activa la webcam y se toma una foto del tablero.

- Matching con la plantilla de la bola:

  Con la foto obtenida del proceso anterior y con ayuda de una plantilla de la bola, se hace un matcheado de ambas imágenes y obtenemos la posición del tablero en donde ha caido la pelota.

- Identificar la casilla ganadora:

  Cada posición del tablero corresponde con un número de la ruleta. Sabiendo cuál ha sido la casilla donde ha caido la pelota es fácil determinar el número ganador de la partida actual.

- Mover el robot a apuestas:

  El robot se mueve a una posición auxiliar desde la cual la webcam enfoca el tablero con las apuestas. Ésta se activa de nuevo y se obtiene una imagen con la situación de las apuestas. En este punto ya se conoce cuál ha sido el numero ganador y analizando la imagen de las apuestas podemos concluir cómo ha acabado la partida.

- Reparto de premios:

  Se calculan las coordenadas de las diferentes casillas del tablero de apuestas con fichas y se escriben en el fichero XML. Se van transmitiendo al robot y se mueve de casilla en casilla repartiendo los premios entre los diferentes jugadores, según corresponda al resultado de la partida.

Cuando todas las fichas han sido repartidas, el robot vuelve a una posición fija de reposo, listo para que se juegue otra partida, y se termina el programa.

7 Evaluación

Este capítulo describe cómo responde el sistema ante una partida de prueba.

Figura 20: Situación inicial y apuestas

Inicialmente la bola está en su posición de partida y las fichas de apuestas se han distribuido tal y como puede observarse en la figura 20:

- Un jugador ha apostado por el 20 negro.
- Un jugador ha apostado por el 7 rojo.
- Un jugador ha apostado por el color rojo.

Con el robot en modo de operación automático, se conectan los accionamientos y el operador del robot selecciona el programa juegoRuleta.src (todo desde el panel de control del robot KCP). Comprobamos que la válvula del compresor de aire está abierta.

A continuación se inician las aplicaciones de PC comunicacionXML.exe y proyecto.exe, que se encargan de la comunicación del sistema y de la parte de visión respectivamente.

Desde el KCP se pulsa la tecla verde de arranque de programa y el robot se pone en movimiento:

En primer lugar se desplaza sobre la posición inicial de la bola, la ventosa desciende lentamente hasta tocarla y se activa la succión por aire comprimido.

Figura 21: Detalle del proceso de succión de la bola

La ventosa se eleva sujetando la bola e inicia un recorrido sobre el tablero de juego.

En cierto instante, la succión de la ventosa se detiene dejando caer la bola sobre el tablero. En nuestra prueba la bola ha caído en la casilla del 20 negro. El robot se mueve a una posición desde la cual la webcam tiene todo el tablero de juego en su campo de visión y reconoce cuál es la posición de la bola para así determinar cuál ha sido el número ganador.

Figura 22: El robot mueve la bola sobre el tablero de juego hasta que la suelta aleatoriamente

El robot se mueve ahora encima del tablero de apuestas y reconoce cómo están distribuidas éstas. Se coloca sobre la ficha de apuestas que hay en el número 7 y la coge con la ventosa de forma similar a como lo hizo con la bola al principio. Mientras sujeta la ficha, se mueve a una zona del tablero reservada para la banca y la deposita allí.

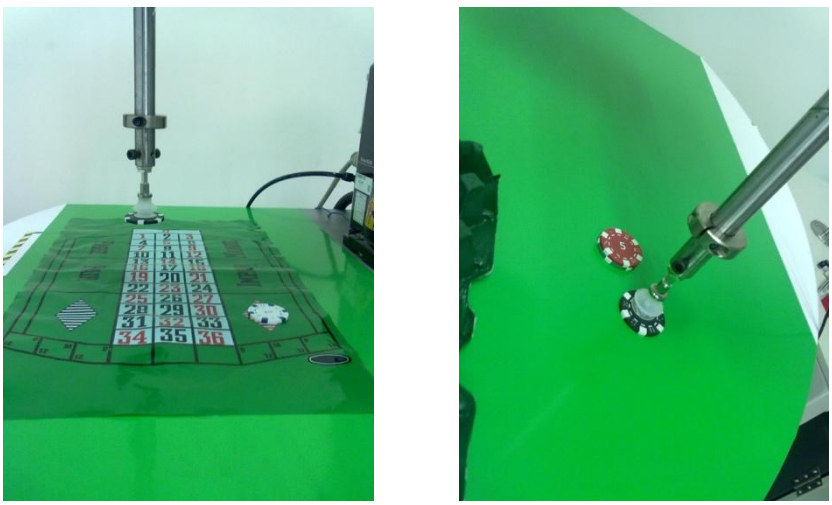


Figura 23: El robot coge una ficha y la deposita en la zona de la banca

Tras esta operación, el robot vuelve a la zona de apuestas y en esta ocasión se coloca sobre la ficha del 20 negro. Como esta ficha pertenece al jugador que ha ganado la partida, la recoge y la lleva a la zona de la mesa perteneciente a este jugador. A continuación el robot vuelve a la zona perteneciente a la banca y se coloca sobre la pila de fichas destinadas a premios. Recoge una y la lleva también a la zona del jugador ganador.

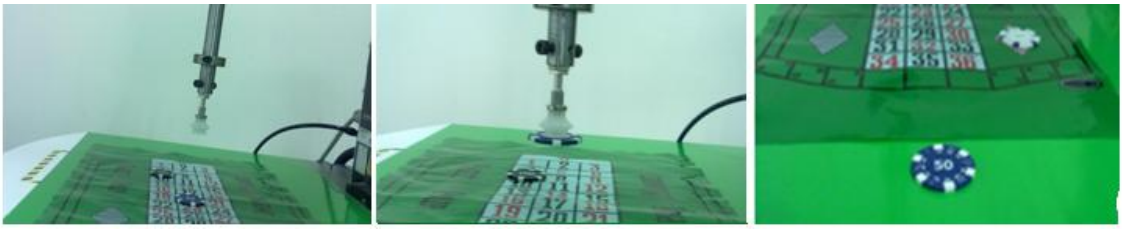


Figura 24: El robot coge una ficha y la deposita en la zona del jugador ganador

El robot se mueve de nuevo a la zona de apuestas y se coloca sobre la ficha de la apuesta al color rojo, la recoge y la lleva a la zona de la banca.

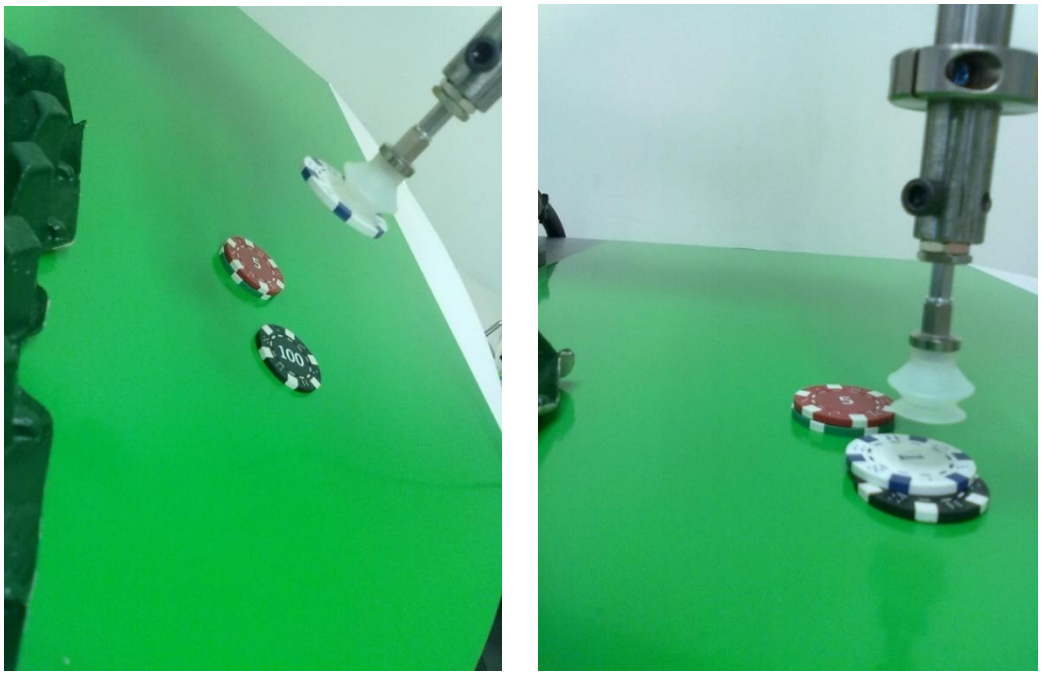


Figura 25: El robot coge la última ficha y la deposita en la zona de la banca

Por último, el robot se coloca en una posición final que hemos definido sobre el tablero de juego y se termina el programa.

Para jugar una nueva partida hay que poner la bola en su posición de inicio, distribuir las fichas de apuestas por el tablero y resetear el programa KRL desde el panel de control del robot.

8 Conclusiones

La realización de este proyecto me ha permitido aprender a trabajar desde cero con un robot industrial. Desde el principio del proyecto estuve en contacto con los técnicos de KUKA que nos instruían en el manejo del robot y también me ayudaban a resolver los problemas que se me iban presentando.

Además me ha servido para aprender sobre visión artificial ya que en mi especialidad no he cursado asignaturas con esa temática y, por ese lado, el proyecto me ha ayudado a completar mis conocimientos.

Al principio esperaba que en KUKA me facilitaran una librería o algo similar para trabajar con el robot sobre C, pero en cambio he aprendido el lenguaje de programación que utilizan en sus sistemas para mover los robots: el KRL. Al estar basado en PASCAL y no tener yo ninguna experiencia con este lenguaje fue bastante complicado al principio, pero tras leer los manuales de programación y de operación resultó ser más sencillo de lo que parecía.

La parte que más problemas me dio y lo que más me costó, pese a las múltiples explicaciones de los manuales, fue a la hora de comunicar el robot con el PC. La aplicación para comunicarse con el robot que nos facilitaron los técnicos de KUKA estaba escrita en C#, del cual apenas tenía conocimientos, así que cambié lo mínimo y necesario para que funcionase la comunicación entre robot y PC y supongo que esa parte del proyecto se podría mejorar si hubiera tenido más experiencia con este lenguaje de programación.

Como conclusión puedo decir que este proyecto me ha servido para ampliar mis conocimientos en robótica y visión y haber podido experimentar el funcionamiento de un robot industrial.

Referencias

Aplicaciones con SCARA

En la Figura 20 se muestra un robot con la configuración SCARA (Selective Compliance Assembly Robot Arm). Este robot fue creado por un grupo de industrias electrónicas japonesas, en colaboración con dos universidades, para insertar componentes de forma vertical. Como se puede observar, esta configuración está formada por dos articulaciones de rotación con respecto a dos ejes paralelos entre sí y perpendiculares al plano de trabajo, y una de desplazamiento en una dirección paralela a la de los ejes de rotación.

Figura 26: Configuración SCARA

El robot SCARA es muy utilizado en todo tipo de aplicaciones industriales, especialmente en aquellas que se realizan en un plano. A continuación se enumeran las aplicaciones más habituales en las que se utilizan estos robots:

- Dispensar o distribuir.
- Soldaduras.
- Coger y colocar.
- Guiado.
- Montaje de componentes.
- Atornillar.

En algunas de estas aplicaciones es necesario el guiado por visión. Para ello algunos de los robots incluyen un software para el procesamiento de imágenes, que se ejecutará en un PC. Este tipo de herramientas mejoran mucho la eficiencia de las aplicaciones en comparación con los métodos tradicionales. Como ejemplo de este tipo de aplicaciones se van mostrar cuatro escenarios en los que se utiliza una cámara junto con el robot para el desarrollo del trabajo.

En la aplicación de la Figura 21 (a) hay una cámara fija apuntando hacia abajo. El robot coge las piezas y las va colocando en su lugar correspondiente en una paleta. En la Figura 21 (b) tenemos una cámara fija enfocando hacia arriba con el objetivo de comprobar la correcta posición y orientación de la pieza antes de posicionarla. En la Figura 21 (c) la cámara está sujeta al robot de forma que se puede comprobar el lugar donde se va a insertar la pieza. Por último, en la Figura 21 (d) el robot coge y mueve las piezas en una cinta transportado en continuo movimiento.

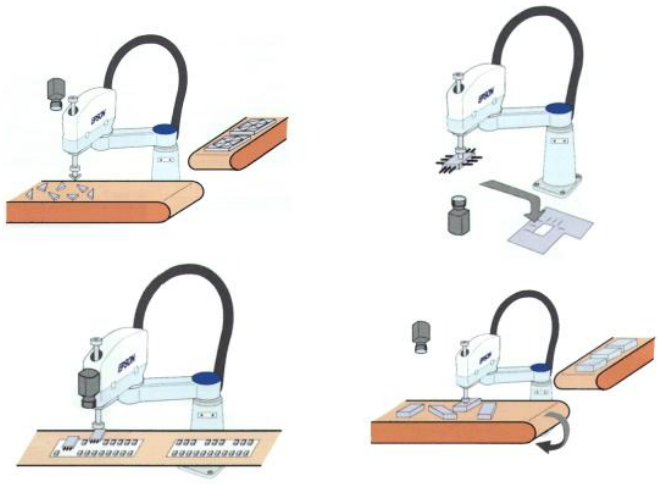


Figura 27: Aplicaciones del robot SCARA

Robots croupiers

Se han visto robots ocupando puestos de cocinero, de niñera o de músico, pero investigando un poco he encontrado varios proyectos en los que se usa un robot como croupier de algún juego de azar.

En primer lugar tenemos el robot de la figura 22, que funciona como croupier para repartir cartas en una partida de póker y es el proyecto de unos estudiantes norteamericanos.

Construido con un microprocesador PIC programando en C, es capaz de reprogramarse para distintos juegos de cartas. El repartidor robótico detecta la posición de los jugadores en la mesa tomando como referencia un reflector de bicicleta situado frente a ellos. La distancia a la que debe arrojar la carta se calcula mediante el sensor ultrasónico que tiene incorporado. Es bastante preciso, pero su principal problema es que todavía resulta demasiado lento, lo que puede llegar a exasperar a los jugadores. Mejorando un poco su agilidad y su velocidad no sería extraño verlos en los mejores casinos de Las Vegas dentro de poco, si es que se cumplen las previsiones de creación de empleo robótico en Japón para el futuro.

Figura 28: Robot croupier que reparte cartas de poker

El equipo formado por Will Berriel y Andew Klose planeó implementar un repartidor de cartas robótico para el juego del blackjack usando el sistema de brazo robotizado Rascal Robix junto con una webcám digital, según se muestra en la figura 23. Sus movimientos para el reparto inicial están predeterminados, pero deberá reaccionar a los gestos de los jugadores para poder completar el juego. La parte de mayor dificultad de este proyecto fue el reconocimiento de las cartas ya repartidas, para saber si un jugador había ganado o perdido.

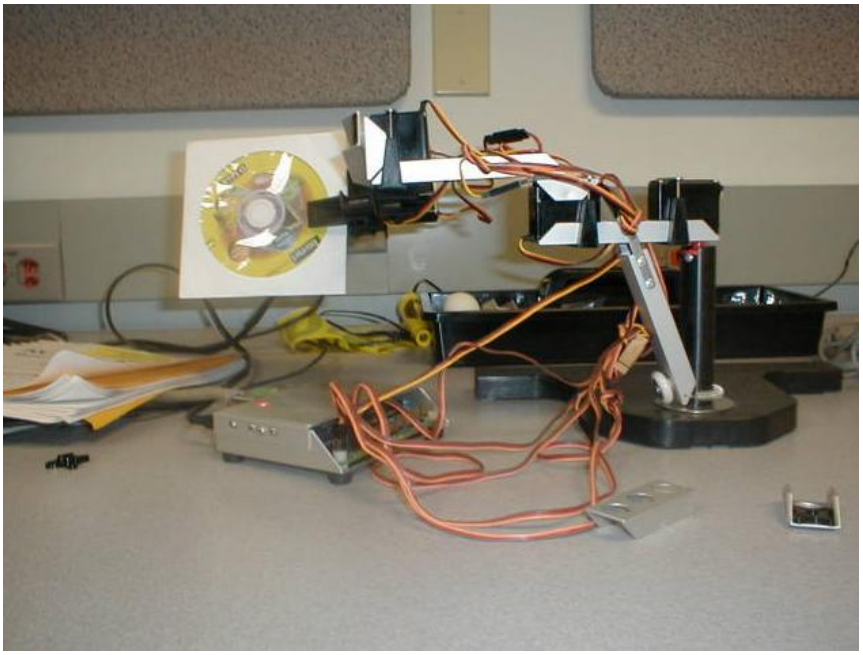


Figura 29: Robot repartidor de blackjack

Bibliografía

Para la documentación sobre la parte de visión y de reconocimiento de objetos en imágenes digitales he consultado múltiples páginas web sobre el tema, así como diversos tutoriales sobre OpenCV.

Me ha sido especialmente útil el libro *Learning OpenCV. Computer Vision with the OpenCV Library* de O'Reilly.

Para la parte de robótica, fundamentalmente he consultado los manuales que me facilitaba KUKA, que son:

- *KR C2 / KR C3 Instrucciones KUKA System Software (KSS) (manual de la unidad de control del robot)*
- *KR C2 / KR C3 Programación por el usuario KUKA System Software (KSS) (Procesamiento e instrucciones de programas)*
- *KUKA.Ethernet KRL XML 1.1 For KUKA System Software (KSS) 5.x, 7.0 (configuración y programación de la comunicación con el robot)*
- *KR 5 scara R350, R550 Instrucciones de servicio del robot*
- *KUKA Robot Group Controller KR C2 sr Instrucciones de servicio de la unidad de control*
- *KUKA Control Panel (KCP) Manual del panel de control de KUKA*

Anexo I:
reglas del juego de la ruleta

El objetivo del juego de la ruleta es determinar en qué número y/o en qué color caerá la bola que lanza el croupier. Los jugadores apuestan contra el croupier, denominado también "Banca" o "Casa".

Como primer paso se realizan las apuestas de los jugadores esperando un tiempo aproximado no mayor a los 2 minutos, luego el croupier coloca la bola en la orilla de la ruleta y la impulsa con gran fuerza para que comience a rodar por alrededor de la misma manteniéndose allí.

Una vez que la rueda decrece en velocidad la bola caerá por efecto de inercia en cualquier casilla que posee un color y un número. No se parará la ruleta hasta que la bola por fin quede fija en alguna casilla.

Todos los participantes deberán enfrentarse a "La Casa" como único medio para ganar el tablero de paño.

Se da por finalizada la jugada una vez que la bola se detiene en alguna casilla de la ruleta, de esta forma dejando en mesa las apuestas con premio y llevándose el croupier todas las apuestas perdidas hacia su banca.

En la ruleta existen apuestas mínimas para cada jugador y mesa, por lo tanto antes de jugar debe consultar al croupier cuál es la apuesta mínima y si tiene "color", que serán las fichas que a usted le tocarán.

Hay dos tipos de apuestas en la Ruleta: la Interna (Inside Bet), y la Externa (Outside Bet).

En la ruleta existen diferentes tipos de apuestas que se dividen por el uso del paño. Aquellas que se encuentran en la parte central del Paño son las denominadas Apuestas Internas, y éstas son:

1. La Recta, Sencilla ó tambien llamado Pleno.
2. La Dividida, Dos Números, Caballo ó Semipleno.
3. La de Calle, ó de Tres Números, ó Transversal.
4. La de Esquina, ó Cuadro.
5. La de Cinco Números. (0,00,1,2,3)
6. La de Línea ó Seisenas.

Luego también, existen las apuestas que se rigen por utilizar el contorno del tablero, denominadas Apuestas Externas, en estas podemos encontrar también diferentes tipos como por ejemplo:

1. Las de suertes sencillas:
   a. La de Color Rojo ó Negro.
   b. La de Número Bajo ó Número Alto
   c. La de Par ó Impar.

2. La de Columna.
3. La de Docena.
4. La del Cero y el Doble Cero.

Todas las apuestas se pueden combinar, de esta forma usted podrá realizar una apuesta interna y externa sin problema alguno, deberá consultar asimismo cual es el límite de las apuestas externas, ya que al ser de probabilidades mayores de ganancia tienen un máximo de apuesta por jugador.

Las ganancias de las Apuestas Internas se contabilizan de la siguiente forma:

- Pleno, Recta o Sencilla: Es una de las apuestas más fuertes de la Ruleta, usted deberá seleccionar cualquier número del tablero con su color y colocar la ficha en su centro. De ser correspondido con el número ganará cada 1 unidad 35 veces lo que apostó. Por ejemplo: Si apostamos 2 euros a un número que sale en la ruleta, estaremos ganando 70€ (35 x 2). También nos permitirán re apostar con la ficha que nos hizo ganar, es decir, no la perdemos sino que también vuelve a nosotros, podremos elegir dejarla en el mismo número que recién nos dio fortuna o moverla a algún otro.

- Dividida, Dos Números o Semipleno: Sucede cuando se coloca una ficha en medio de dos números. Si salen favorecidos cualquiera de los dos números estaremos ganando 17 veces nuestra apuesta. Por ejemplo: Si apostamos 2 euros a dos números y alguno de ellos sale en la ruleta, estaremos ganando 34€ (17 x 2). También nos permitirán re apostar con la ficha que nos hizo ganar, es decir, no la perdemos sino que también vuelve a nosotros, podremos elegir dejarla en el mismo número que recién nos dio fortuna o moverla a algún otro.

- Calle o Tres Números: En este caso la calle le permitirá jugar con 3 números de una fila con una sola ficha de apuesta. Lo único que deberá hacer es indicar en la línea exterior del tablero la fila que le interese en el espacio de los tres números que haya elegido (en todas las ruletas la calle se debe colocar en la línea contraria a donde se encuentra la Ruleta). Si usted es el ganador, entonces la banca le dará 11 veces lo que usted aposto. Por ejemplo: Si apostamos 2 euros a una fila de números y resulta premiada, estaremos ganando 22€ (11 x 2). También nos permitirán re apostar con la ficha que nos hizo ganar, es decir, no la perdemos sino que también vuelve a nosotros, podremos elegir dejarla en el mismo número que recién nos dio fortuna o moverla a algún otro.

- Cuadros o la "Apuesta de Esquina": Es una de las apuestas más utilizadas por su gran probabilidad de éxito sumado a su no tan mala ganancia. Para jugar un "cuadro" deberá colocar una ficha en el centro de 4 números. De esta forma estará apostando a cualquiera de ellos cuatro, en este caso la ganancia será de 8 veces lo apostado. Por ejemplo: Si apostamos 2 euros a un cuadro de cuatro números y sale alguno de esos cuatro premiado, estaremos ganando 16€ (8 x 2). También nos permitirán re apostar con la ficha que nos hizo ganar, es decir, no la perdemos sino que también vuelve a nosotros, podremos elegir dejarla en el mismo número que recién nos dio fortuna o moverla a algún otro.

- Apuesta de Cinco Números: Éste caso se da por única vez en los números, "0" - "00" - "1" - "2" y "3". Como pueden ver por su disposición en el tablero nos permitirá colocar una apuesta como si se tratara de una "calle" en la linea exterior del "0" y el "1", automáticamente estará apostando a los 5 números que recién fueron mencionados. La ganancia será de 6 veces lo apostado. Por ejemplo: Si apostamos 2 euros a los cinco números y sale alguno de esos cinco premiado, estaremos ganando 12€ (6 x 2). También nos permitirán re apostar con la ficha que nos hizo ganar, es decir, no la perdemos sino que también vuelve a nosotros, podremos elegir dejarla en el mismo número que recién nos dio fortuna o moverla a algún otro.

- Apuesta de Línea o Seisenas: Esta es la jugada de menos ganancia retroactiva en las apuestas internas, ya que si salimos favorecidos sólo tendremos 6 veces lo apostado. Para realizarla se deberá colocar la ficha entre los últimos números de las líneas exteriores de dos filas (le permitirá apostar a 6 números en una jugada). Por ejemplo: Si apostamos 2 euros a una Seisena y alguno de esos seis números es premiado, estaremos ganando 12€ (6 x 2). También nos permitirán re apostar con la ficha que nos hizo ganar, es decir, no la perdemos sino que también vuelve a nosotros, podremos elegir dejarla en el mismo número que recién nos dio fortuna o moverla a algún otro.

Entre las apuestas Externas podremos encontrarnos con:

- Apuestas 1 a 1 o "al Par": Son aquellas que tienen como ganancia lo que apostamos. Si apostamos 2 euros, ganaremos 2 euros (sin contar la ficha apostada que quedará en nuestro poder). Cabe destacar que al ser probabilidades bajas de error, tienen apuestas mínima, consulte al croupie" cuál es la cantidad mínima a apostar de apuestas externas.
- Apuesta de Color: Son aquellas apuestas que tienen 50% de probabilidad de error, sólo debemos elegir entre el Rojo y el Negro. Colocamos nuestra ficha y, en caso de salir el color que elegimos, ganaremos lo que apostamos.

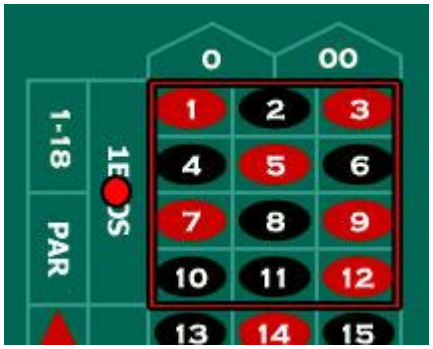
- Número Alto o Bajo: Son aquellas apuestas en la cual debemos de elegir entre un número alto (19-36) o un número bajo (1-18), colocando nuestra ficha en el casillero que corresponda. Pierde si sale un número contrario a lo que apostó o también si sale "0" ó "00". Por ejemplo: Apostamos a un número alto (19-36), y en la jugada sale el número 24 estaremos ganando lo que apostamos.

- Par o Impar: Como su nombre indica podremos elegir entre un número par de la ruleta o un número impar. Si sale el seleccionado ganaremos lo que apostamos y pierde si sale un número contrario a lo que apostó o también si sale "0" ó "00".

- Apuesta de Columna: Esta apuesta le permitirá elegir entre las 3 columnas de número que hay en el tablero, al haber un 33,3% de probabilidades de éxito la jugada se paga 2 veces lo apostado. Por ejemplo: Si apostamos 2 euros a la primer columna y sale el número 16 habremos ganado, por lo tanto cobraremos 4€ (2 x 2). También nos permitirán re apostar con la ficha que nos hizo ganar, es decir, no la perdemos sino que también vuelve a nosotros, podremos elegir dejarla en el mismo número que recién nos dio fortuna o moverla a algún otro.

- Apuesta de Docena: En este caso se seleccionará como una columna pero la docena que prefiera, al haber un 33,3% de probabilidades de éxito la jugada se paga 2 veces lo apostado. Las docenas son las siguientes:
    - Primer Docena: 1 al 12
    - Segunda Docena: 13 al 24
    - Tercer Docena: 25 al 36

  Ejemplo: Si apostamos 2 euros a la primer decena y sale el número 9 habremos ganado, por lo tanto cobraremos 4€ (2 x 2). También nos permitirán re apostar con la ficha que nos hizo ganar, es decir, no la perdemos sino que también vuelve a nosotros, podremos elegir dejarla en el mismo número que recién nos dio fortuna o moverla a algún otro.



- Cero y Doble Cero: Al igual que cualquier número pleno si apostamos una ficha al "0" o al "00" estaremos ganando 35 veces lo apostado. Ejemplo: Si

apostamos 2 euros al "0" y éste sale, estaremos ganando 70€ (35 x 2). También nos permitirán re apostar con la ficha que nos hizo ganar, es decir, no la perdemos sino que también vuelve a nosotros, podremos elegir dejarla en el mismo número que recién nos dio fortuna o moverla a algún otro.