UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Aeroespacial y Diseño Industrial

RECONOCIMIENTO DE MANOS DE PÓKER

Trabajo de Visión Artificial
Grado en Ingeniería Electrónica Industrial y Automática

AUTORES:
Lucas Iglesias Toboso
Carles Bataller Sebastiá

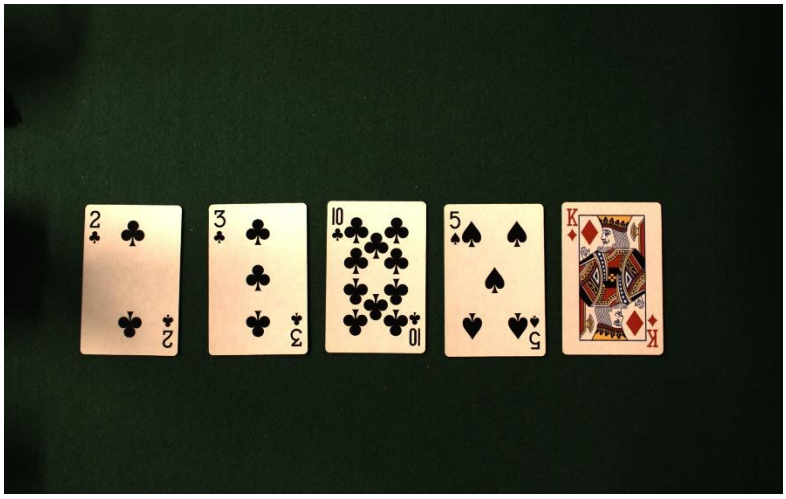
CURSO ACADÉMICO: 2024-2025

ÍNDICE

Contenido

Obtención de las imágenes

Las imágenes las obtuvimos usando la cámara del laboratorio de la UPV, teniendo a nuestra disposición dos focos para iluminar correctamente las cartas. En nuestro caso, no utilizamos esta iluminación, pues se reflejaba en las cartas causando dificultad en el análisis y postprocesado de las imágenes. Además, usamos un tapete verde de fondo, aparte de para simular un ambiente real en una casa de póker, se usó para facilitar la distinción de las cartas con el fondo blanco. Se hicieron 40 imágenes, 20 para programar el algoritmo y 20 de prueba, con las que se hicieron el análisis de los datos obtenidos. A continuación, una de las imágenes que tomamos:

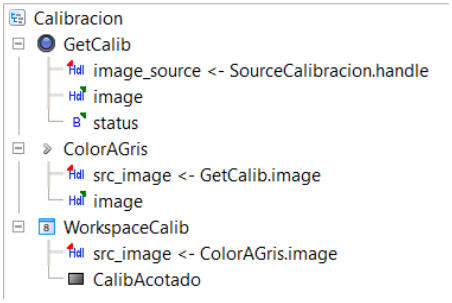


Análisis de las imágenes

Vamos a explicar el programa por orden de ejecución:

- Calibración

Lo primero que se hace es una llamada a la subrutina de calibración, donde se obtiene la imagen de calibración y se transforma a escala de grises. Finalmente, se acota el patrón de calibración y se añade a la variable llamada "*PatronCalib*", para su posterior uso en la calibración del workspace de análisis de cartas. Aquí una foto del código interno de la subrutina:



En este paso calibramos la imagen de la cámara para obtener un post procesado consistente, eliminar los posibles errores de distorsión y mejorar la precisión.

- Inicialización de variables

A continuación, se ejecuta la llamada a la inicialización de las variables, donde se encuentra un script simple donde se inicializan las variables tipo *string* de los números a *'0'* y el de los palos a *'Ceros'*. También se ha inicializado una variable llamada *"Saltodelinea"* a cero, que se usará en el proceso de escritura de los resultados. Aquí una imagen del código del script:

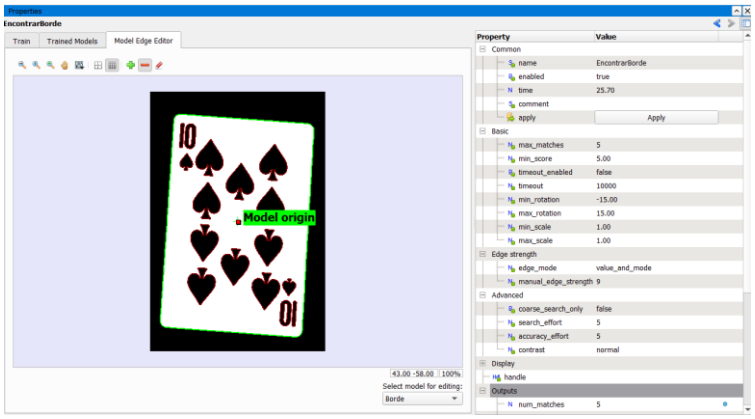
```
ve.Numero1.value=0;
ve.Numero2.value=0;
ve.Numero3.value=0;
ve.Numero4.value=0;
ve.Numero5.value=0;

ve.Palo1.value='Cero';
ve.Palo2.value='Cero';
ve.Palo3.value='Cero';
ve.Palo4.value='Cero';
ve.Palo5.value='Cero';

ve.Saltodelinea.value="0";
```

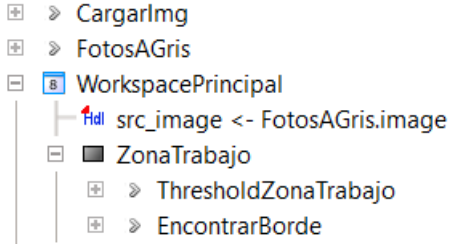
- Obtención, procesado de imágenes y reconocimiento

Una vez ejecutado el código anterior, se obtiene una de las imágenes de la batería de evaluación y se pasa a escala de grises. Obtenida la imagen a tratar, se añade el patrón de calibración en el *"workspace"* y se delimita la zona de trabajo (en nuestro caso es la imagen entera) y se ejecuta un *"threshold"* de toda la imagen para facilitarnos el análisis. A continuación, se usa el algoritmo *"search_edge"* para encontrar todas las cartas, pues buscamos una carta sin bordes internos, de esta forma el algoritmo busca un rectángulo blanco. Tal y como se indica en la siguiente foto:



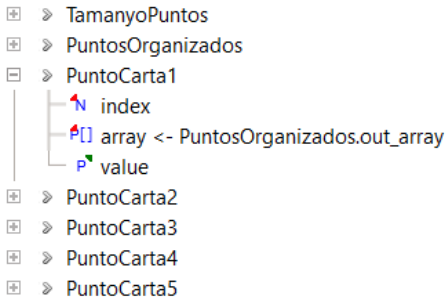
Como se puede observar, se han eliminado los bordes internos de la carta.

Esta función se encuentra dentro del *workspace* principal:



Una vez encontradas las cinco cartas, se usa la función *"array_size"*, que en nuestro caso se llama *"TamanyoPuntos"*, para comprobar que efectivamente se han encontrado 5 cartas. Después se

usa un organizador de vectores, que en nuestro caso se llama *"PuntosOrganizados".* Usando este comando se organiza el vector de los centroides de las cartas obtenidos con anterioridad de forma ascendente en la coordenada x, de esta manera sabemos qué centroide pertenece a cada carta, siendo el primero (con la coordenada *x* más pequeña) y el quinto (con la coordenada *x* más grande). Una vez organizados los centroides los asignamos a unas variables de tipo punto, que en nuestro caso se llaman *"PuntoCarta1"*, *"PuntoCarta2"* y así sucesivamente tal y como se puede ver en la siguiente imagen:



Una vez obtenido y clasificado cada centroide por separado, se hace el siguiente procedimiento: Se añade un *"aligment"* con el nombre *"AligmentCartaX"* con el valor de posición del centroide correspondiente. Después se crea una *"roi"* (que usará ese *"AligmentCartaX"*) que contiene un *"threshold" ("ThresholdCartaX")* y dos *"search_edges" ("EncontrarPaloX"* y *"EncontrarNumeroX")* orientados en la esquina superior izquierda de la carta (pues allí se encuentran los números y los palos de las cartas), uno para obtener el valor numérico y otro para obtener el palo. Cada uno se ha entrenado con todos los palos y números posibles respectivamente. Este proceso se repite cuatro veces para obtener una *"roi"* por cada carta tal que:

En este paso hemos hecho que el programa localice la zona a identificar, llamada "*roi"*, esta zona cambia entre fotos, pues las cartas no están posicionadas siempre en el mismo sitio, es por eso por lo que usamos un *"aligment"* que es una función que hace que la *"roi"* se mueva entre fotos siguiendo la posición de los centroides obtenidos anteriormente

Los algoritmos *"EncontrarPaloX"* y *"EncontrarNumeroX"* tienen como salida la *string "patter_name"* que la guardaremos en las variables inicializadas al principio del programa, estas variables llevan por nombre *"PaloX"* y *"NumeroX"* dependiendo de a que carta hacen referencia.

Para poner un ejemplo, siendo la carta 2 un tres de tréboles, obtendremos que las variables *"Palo2"* y *"Numero2"* tienen como valores *"Trebol"* y *"Tres"* respectivamente, además estos valores se retransmitirán en el apartado *"Watch"* por si queremos corroborar que el análisis se ha hecho de forma correcta.

- Conexión servidor TCP y envío de datos obtenidos

A continuación, se define una variable servidor de tipo *TCP* llamada *"ServerComunicacion"* para poder crear una conexión entre el fichero de Matlab y el programa desarrollado en Sherlock.

Una vez analizadas las cinco cartas y guardado sus respectivos valores en variables, se procede a hacer una llamada a la subrutina *"ComunicacionEnviar"*, que como su nombre indica, se encuentran las funciones *"send_ascii_line"*, donde se envía a Matlab los valores numéricos y los

palos obtenidos de las cartas para su análisis. En esta subrutina tenemos diez funciones, cinco para enviar los palos y cinco para enviar los números, estas se llaman *"EnviarPaloX"* y *"EnviarNumeroX"*.

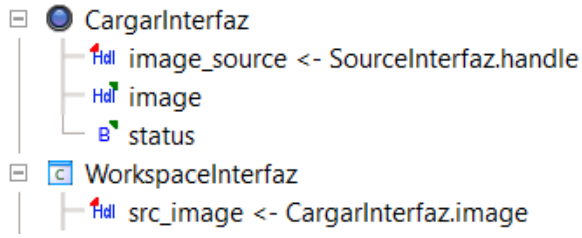
Además, en esta subrutina se cuenta con un script llamado *"ComprobacionTamanyo"* donde se corrobora que la variable *"TamanyoPuntos"* contiene efectivamente cinco puntos, si no fuese el caso, se envía un 0 en la variable *"Numero5"*, básicamente es para conocer que en la imagen no hay cinco imágenes, esto se explicará con más profundidad en la parte del Matlab (aunque en las imágenes que analizaremos no hay ninguna imagen con 4 o menos cartas, este proceso se hace simplemente para añadirle robustez al programa).

- Obtención con TCP de resultado de mano tras análisis en Matlab

Después de enviar los datos, se llama a la subrutina de obtención de datos, esta vez será Matlab quien enviará una línea *ascii*, el Sherlock la obtiene usando el comando *"read_ascii_line"*, en nuestro caso *"RecibirValores"*.

- Interfaz de resultados

Una vez obtenido el valor del análisis, se obtiene la imagen de la interfaz que usaremos y se crea un *workspace* donde visualizar la interfaz:

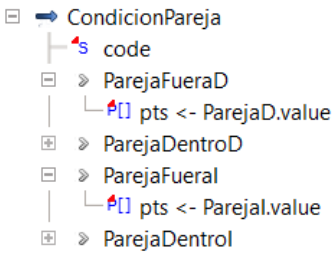


Siendo esta la interfaz:

Dentro del *workspace* se encuentra una condición por cada tipo de mano posible, donde se evalúa el valor del análisis obtenido. Si el valor es el correcto, entonces se dibujan cuatro flechas, dos externas de color negro y dos internas de color blanco más pequeñas, un par de flechas apuntando por la parte diestra y otro par por la parte zurda al texto de la mano correspondiente.

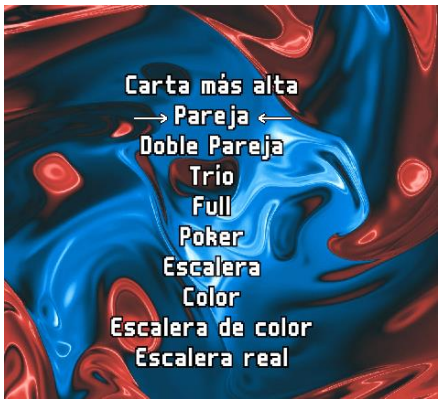
Para dibujar estas flechas se ha usado el comando *"draw_polyline"*, que requiere de una entrada de vector de puntos y esto es lo más laborioso, pues se han de añadir un par de vectores de

puntos a mano siguiendo el siguiente orden, vértice superior-> vértice central-> vértice trasero-> vértice central-> vértice inferior, para que la polilínea se dibuje siguiendo la forma de la flecha en las coordenadas deseadas. Particularizando el caso de la Pareja:



Siendo *"ParejaFuera"* las flechas negras y *"ParejaDentro"* las flechas blancas, y las terminaciones *"D"* e *"I"* para indicar derecha o izquierda. Como se puede observar en la imagen, se usan dos variables de puntos, uno para indicar la posición a la derecha y otro a la izquierda. Las demás manos siguen la misma nomenclatura.

Siguiendo en el caso de la pareja, cuando la mano analizada sea una pareja, se dibuja en la interfaz las flechas explicadas con anterioridad:

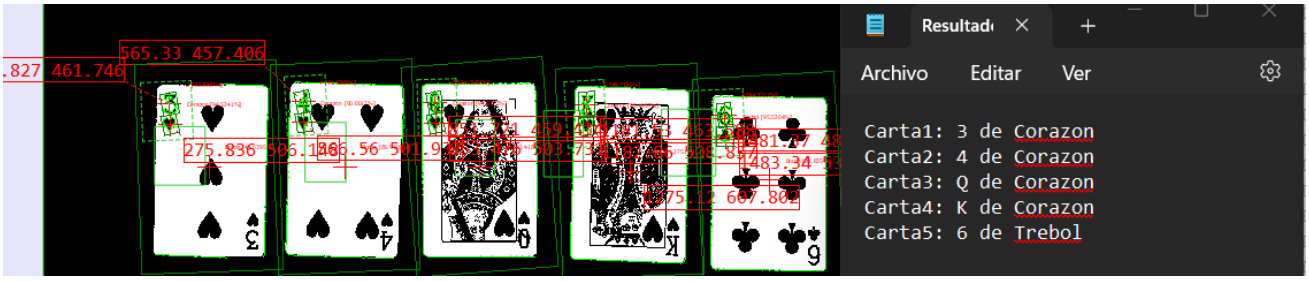


Las condiciones numéricas que indican que mano se ha obtenido se explicará en la parte del Matlab.

- Escritura de valores y palos en fichero

Procedemos, a través de las herramientas de escritura que nos proporciona Sherlock, a reflejar en un archivo de texto llamado Resultado.txt los diferentes valores y palos presentes en cada mano detectada.

Primero usamos un script para asignar **en javascipt** el valor /n que posteriormente usaremos como salto de línea, y a continuación llamaremos a una subrutina encargada de la escritura.



Código en MATLAB

- Conexión TCP con Sherlock y recepción de datos de entrada

```
%conexion

client = tcpclient('127.0.0.1', 30);

while(true)
    while client.NumBytesAvailable == 0
        pause(0.1);
    end
    valores = [];
    palos = {};

    for i=1:10 %lectura 10 veces 5 palos 5 valores

    response = readline(client);
    disp(['Respuesta: ', response]); %borrar

    if ismember(response, ["Pica", "Corazon", "Diamante", "Trebol"])

        if iscell(response)
        response = response{1};
        end

            palos{end + 1} = char(response);
        elseif ismember(response, ["J", "Q", "K", "A"]) || all(isstrprop(response, 'digit'))
            if response == "J"
                valores(end + 1) = 11;
            elseif response == "Q"
                valores(end + 1) = 12;
            elseif response == "K"
                valores(end + 1) = 13;
            elseif response == "A"
                valores(end + 1) = 14;
            else
                valores(end + 1) = str2double(response);
            end
    else
        disp('Entrada no valida');
    end
    end


    for i = 1:length(palos)
        if iscell(palos{i})
        palos{i} = palos{i}{1};
        end
    end
```

En primer lugar, es necesario implementar en Matlab las instrucciones que nos permitirán una correcta comunicación TCP con la parte de nuestro proyecto en Sherlock encargada de reconocer los palos y valores de las imágenes de las manos.

Como se muestra en el código, tras la conexión lo primero será leer los sucesivos valores de las cartas que conforman la mano de póker y, debido a la forma de nuestro programa en Sherlock de transmitir los valores asociados a figuras y el As, a continuación, haremos una sustitución de estos valores para obtener una codificación completamente numérica, la cual nos facilitará posteriormente la búsqueda de los posibles tipos de mano. Los 5 valores se asignaran a las

correspondientes posiciones dentro del array valores=[]. Cabe destacar que se ha tenido en cuenta la dualidad que presenta el As a la hora de conformar escaleras, programando el código de tal forma que esta dualidad es evaluada partiendo de un valor inicial del As como 14.

Una vez leídos y asignados los valores en valores=[], procedemos a leer y asignar en palos={} los palos de las 5 cartas de la mano. Debido a que se formaba un array anidado, hemos implementado el código de la forma mostrada con el objetivo de desanidar palos, y que de esta forma palos sea un string simple.

- Evaluación de mano

Una vez tenemos los valores y palos de las 5 cartas asignados, estamos en condiciones de evaluar las diferentes manos de póker posibles. En primer lugar, procedemos a inicializar a 0 los diferentes tipos de mano posibles de la siguiente forma:

```
%posibles resultados tras evaluacion de manos
escreal=0; esccolor=0; esc=0; color=0; poker=0; full=0; trio=0; doblepar=0;
pareja=0; caralt=0; cincocartas=0; nulo=0;
```

Mediante diversas funciones propias elaboradas en Matlab en nuestro código y el valor que estas devuelven según su codificación interna, asignaremos a 1 (TRUE) los posibles resultados tras la evaluación de manos. Una vez ya se ha evaluado la mano con nuestras funciones y las pertinentes variables asociadas a los posibles resultados se han igualado a 1, realizaremos, como se muestra más adelante, una codificación adicional para asignar a cada posibilidad un dígito único de identificación que será enviado a Sherlock.

Como primer paso del estudio de la mano, establecemos que el análisis de mano solo se efectúe si se han asignado 5 valores y 5 palos en valores=[] y palos={} respectivamente de la siguiente forma:

```
if length(palos) == 5 && length(valores) == 5
```

En caso de no ser así se ha programado que indique en la consola de MATLAB que se ha producido un error en el proceso de lectura de datos y asignación:

```
else
disp('No se han leido correctamente los palos y valores') %nulo por no haber 5 valores y 5 palos
end
```

En el caso de que si haya 5 palos y valores, se continua con el estudio de mano evaluando a través de nuestra función **evaluarcincocartas** que así sea:

```
evaluarcincocartas=Evaluarcincocartas(valores); %Evaluamos que hay 5 cartas
if(evaluarcincocartas==1)
```

En caso de que no sea así ya estará definido el tipo de mano analizada, siendo su tipo marcado como nulo y se expresará a través de la consola de Matlab el siguiente mensaje:

```
else

    nulo=1;
    disp('No se han detectado 5 cartas, pruebe otra vez');

end
```

Si se han detectado 5 cartas, y siendo así una mano valida, continuaremos con el análisis.

- Evaluación de manos relacionadas con palos y valores

```
evaluarmismopalo=Evaluarmismopalo(palos); %

if(evaluarmismopalo==1)

    evaluarescalera=Evaluarescalera(valores);

    if(evaluarescalera==1)

        evaluarrealocolor=Evaluarrealocolor(valores);

        if(evaluarrealocolor==1)
            escreal=1;
            disp('Escalera real');
        else
            esccolor=1;
            disp('Escalera de color');
        end

    else

        color=1;
        disp('Color');

    end

else
```

Evaluamos gracias a nuestra función **evaluarmismopalo** si se cumple esta condición. Es importante analizar esta condición al principio del estudio de las manos validas, porque de esta manera simplifica enormemente el proceso de análisis, limitando los valores posibles a un número reducido de tipos de mano, las que tienen en cuenta el palo de las cartas. Estas son: color, escalera de color, escalera real. La funcion **evaluarmismopalo** devuelve 1 o 0 en función de si se cumple esta condición.

En caso de que **evaluarmismopalo** devuelva 1, evaluamos si la mano es (como mínimo) una escalera simple empleando nuestra función **evaluarescalera**, que devuelve 1 de ser así. De esta manera Sabremos que hay como mínimo la condición básica de escalera, pero posteriormente deberemos evaluar si, además de escalera, la mano presenta atributos propios de una escalera de color o escalera real.

Para este análisis de los posibles subtipos de escalera, y teniendo en cuenta que ya contamos con la condición de mismo o color y escalera, empleamos la función propia **realocolor**. Esta función será responsable de analizar si dentro de los sucesivos valores de la escalera hay presentes un As (14, no posible dualidad) y un Rey (trece) simultáneamente, lo que será inequívocamente prueba de que se trata de una escalera real (escalera acabada en As y del mismo color). En caso de ser así **realocolor** devolverá 1, y asignaremos el resultado de la mano escreal=1, y de no ser así, devolverá 0, lo que hará que se trate de una escalera de color (pero no real) y asignaremos el resultado esccolor=1. Cabe resaltar que la colocación de la función **realcolor** sea llamada si previamente **evaluarmismopalo** y **evaluarescalera** son afirmativas, ya que realcolor se limita a evaluar la mano es escalera real o color sin evaluar previamente que se trate de una escalera ni que los 5 palos coincidan. Por ello, la jerarquía y orden de llamada de funciones es de vital importancia.

En caso de que **evaluarescalera** devuelva 0 (y ya **evaluarmismopalo** haya devuelto 1), la mano se tratará indudablemente de color, y asignaremos el tipo de mano color=1.

- Evaluación de manos relacionadas con únicamente con valores

```
else

    evaluarescalera=Evaluarescalera(valores);

    if(evaluarescalera==1)
        esc=1;
        disp('Escalera');
    else

        evaluariguales=Evaluariguales(valores);

        switch evaluariguales
            case 1
                poker=1;
                disp('Poker');
            case 2
                full=1;
                disp('Full');
            case 3
                trio=1;
                disp('Trio');
            case 4
                pareja=1;
                disp('Pareja');
            case 5
                caralt=1;
                disp('Carta alta');
            case 6
                doblepar=1;
                disp('Doble pareja');
        end

    end

end
```

En caso de que **evaluarmismopalo** nos devuelva 0, serán descartados los tipos de manos antes mencionados (color, escalera de color y escalera real), los cuales parten de la condición de 5 palos coincidentes y se comenzará a hacer un análisis de los posibles resultados relacionados plenamente con los valores de las cartas (escalera, poker, full, trio, pareja, doble pareja y carta alta).

Comenzaremos haciendo uso de la función **evaluarescalera** antes empleada, solo que ahora es llamada si se da la condición previa de la no presencia de 5 palos coincidentes. De tratarse de una escalera, **evaluarescalera** devolverá 1, y asignaremos el resultado de mano esc=1. De no ser así, **evaluarescalera** devolverá 0 y continuaremos con el análisis.

Como el resto de posibles resultados de mano que parten de la condición de no haber 5 palos coincidentes se basan todos en la multiplicidad de los valores presentes, se simplifica el proceso de estudio del resto de posibles manos haciendo uso de nuestra función propia **evaluariguales**, que tendrá 5 posibles valores de respuesta, cada uno asignado a un resultado de mano (explicado más adelante en la explicación de funciones propias), Según el valor devuelto por **evaluariguales**, asignaremos iguales a cada tipo de mano poker, full, trio, pareja, caralt o doblepar a 1 según cada caso.

- Funciones propias empleadas en el análisis

Durante el proceso de estudio de la mano hemos empleado diferentes funciones propias que nos ayudan a determinar el tipo de mano según las características de las cartas. A continuación, procedemos a explicarlas:

- Evaluarcincocartas

La función **Evaluarcincocartas** devuelve 0 si algún valor del string de valores es 0 (nulo) y por lo tanto no hay 5 cartas. En caso de que haya 5 valores válidos, **Evaluarcincocartas** delvuelve 1.

```
function comprobacion=Evaluarcincocartas(valores)
    if(any(valores==0))
        comprobacion=0;
    else
        comprobacion=1;
    end
end
```

- Evaluarmismopalo

La función **Evaluarmismopalo** compara todos los valores del string de palos con el primer palo. En caso de que todos coincidan **Evaluarmismopalo** devuelve 1, y en caso de no ser así devuelve 0.

```
function color=Evaluarmismopalo(palos)
    color=all(strcmp(palos, palos{1}));
end
```

- Evaluarescalera

La función **Evaluarescalera** realiza una doble comprobación de si hay 5 valores consecutivos, una comprobación de 5 valores consecutivos considerando el As como 14 (valor de entrada por defecto), y posteriormente, en caso de que haya habido un As (14), es sustituido por 1 y se vuelve a estudiar la posible existencia de escalera con esta nueva condición. Si con alguna de las dos condiciones producidas por la dualidad del As se detecta escalera Escalonconascatorce o Escalonconasuno, según corresponda, se igualan a 1. Si tras ambos análisis Escalonconascatorce o Escalonconasuno son 1, **Evaluarescalera** devolverá 1, y de no ser así devolverá 0.

```
function escalera=Evaluarescalera(valores)

    %A la hora de evaluar escalera, se debera tener en cuenta la dualidad
    %del as 1-14 por lo que realizamos el analisis con el 14 de entrada y
    %si hay un 14 lo sustituimos por 1 y volvemos a comprobar. En caso de
    %detectar escalera con uno de los analisis habra escalera

    Escalonconascatorce=0;
    Escalonconasuno=0;

    minimo=min(valores);
    escaleraEsperada=minimo:minimo+4;
    Escalonconascatorce=all(ismember(escaleraEsperada,valores));

    if any(valores == 14)
         valores(valores == 14) = 1;
         minimo=min(valores);
         escaleraEsperada=minimo:minimo+4;
         Escalonconasuno=all(ismember(escaleraEsperada,valores));
    end

    if ((Escalonconasuno||Escalonconascatorce)==1)
        escalera=1; %como minimo sera escalera
    else
        escalera=0; %no hay escalera
    end
end
```

- realocolor

La función **realocolor** parte, en base a su posición en el código, de haber evaluado previamente la presencia de escalera y de 5 palos coincidentes, por lo que simplemente evalua la presencia de un As (14, no posible dualidad) y un Rey (13). De ser así **realocolor** devuelve 1, y si no es caso devuelve 0.

```
function realocolor=Evaluarrealocolor(valores)

    reyyas=ismember(13, valores)&&ismember(14, valores); %comprobamos si en los valores hay 13 (rey) y 14 (as).

    %En este caso, no hace falta tener en cuenta la dualidad 1-14 del as

    if (reyyas == 1)
        realocolor = 1; %Escalera real (caso especifico de esc de color con rey y as)
    else
        realocolor = 0; %Escalera de color (cualquier caso de esc de color que no sea real)
    end
end
```

- Evaluariguales

La función **Evaluariguales** nos sirve para evaluar conjuntamente todas las manos que no parten de la condición de color menos la escalera. La función **EvaluarIguales** parte de que todos esos posibles tipos de mano se basan en el principio de multiplicidad de valores, y de que el comando de MATLAB any analiza la multiplicidad de forma estricta y exacta. Por lo tanto, según las condiciones de multiplicidad propias de cada tipo de mano, la función **Evaluariguales** devolverá un valor del 1 al 5, valor que será asignado a iguales en la evaluación.

```
function iguales = Evaluariguales(valores)
    % Vemos cuantas veces se repite cada valor de cartas
    valoresUnicos = unique(valores); % Encuentra valores únicos
    conteos = arrayfun(@(x) sum(valores == x), valoresUnicos); % Contamos el numero de veces

    if any(conteos == 4)
        iguales = 1; % Poker
    elseif any(conteos == 3) && any(conteos == 2)
        iguales = 2; % Full House
    elseif any(conteos == 3)
        iguales = 3; % Trío
    elseif sum(conteos == 2) == 2
        iguales = 6; % Doble Pareja
    elseif any(conteos == 2)
        iguales = 4; % Pareja
    else
        iguales = 5; % Carta Alta
    end
end
```

- Codificación final y envío de resultados mediante conexión TCP a Sherlock.

Una vez ya evaluado el tipo de mano procederemos a enviar el resultado a Sherlock para que Sherlock actúe en consecuencia.

- Codificación

Previamente al envío, según los resultados de la evaluación de la mano, procederemos a elaborar una codificación, cuyo resultado será asignado a la variable envioasherlock tal y como se muestra a continuación:

```
if(escreal==1)
    envioasherlock=1; %escalera real
end

if(esccolor==1)
    envioasherlock=2; %escalera de color
end

if(esc==1&&esccolor==0&&escreal==0)
    envioasherlock=3; %solo escalera
end

if(color==1)
    envioasherlock=4; %color
end

if(full==1)
    envioasherlock=5; %full
end

if(trio==1)
    envioasherlock=6; %trio
end

if(pareja==1)
    envioasherlock=7; %pareja
end

if(doblepar==1)
    envioasherlock=8; %doblepar
end

if(poker==1)
    envioasherlock=9; %poker
end

if(caralt==1)
    envioasherlock=10; %carta alta
end

if(nulo==1)
    envioasherlock=11; %nulo (algun valor=0)
end
```

En total, hay 11 tipos de respuestas que puede obtener Sherlock

- Envío del valor codificado en envioasherlock

Procedemos a enviar mediante a conexión TCP el valor de envioasherlock a Sherlock:

```
%envio a sherlock de la codificacion de resultado

writeline(client, num2str(envioasherlock));
disp(['Valor enviado a Sherlock: ', num2str(envioasherlock)]);
```

Análisis Final

- Análisis de los resultados

El análisis que hemos hecho ha sido al nivel más bajo posible, es decir, hemos analizado si el programa ha interpretado correctamente los números y los palos de las cartas. Se debe tener en cuenta que hemos usado los valores de las fotos de evaluación para crear la matriz de confusión. Las matrices resultantes son las siguientes:

Palos		Clase Verdadera						
		Corazon	Pica	Diamante	Trebol	Suma		Precision
Clase Detectada	Corazon	**24**	0	0	0	24		1,00
	Pica	**1**	**23**	**6**	0	30		0,77
	Diamante	0	0	**25**	0	25		1,00
	Trebol	0	0	0	**21**	21		1,00
Suma		25	23	31	21	100		**Media Precision**
Recall		0,96	1,00	0,81	1,00	**Media Recall**	**94,16%**	**94,17%**
F1		0,98	0,87	0,89	1,00	**Media F1**	**93,51%**	

Números		Clase Verdadera														
		A	2	3	4	5	6	7	8	9	10	J	Q	K	Suma	Precision
Clase Detectada	A	**3**	0	0	0	0	0	0	0	0	0	0	0	0	3	1,00
	2	0	**3**	0	0	0	0	0	0	0	0	0	0	0	3	1,00
	3	0	0	**9**	0	**1**	0	0	0	0	0	0	0	0	10	0,90
	4	0	0	0	**5**	0	0	0	0	0	0	0	0	0	5	1,00
	5	0	0	0	0	**8**	0	0	0	0	0	0	0	0	8	1,00
	6	0	0	0	0	**1**	**12**	0	0	0	0	0	0	0	13	0,92
	7	0	0	0	0	0	0	**11**	0	0	0	0	0	0	11	1,00
	8	0	0	0	0	0	0	0	**8**	0	0	0	0	0	8	1,00
	9	0	0	0	0	0	0	0	0	**3**	0	0	0	0	3	1,00
	10	0	0	0	0	0	0	0	0	0	**10**	**1**	0	0	11	0,91
	J	0	0	0	0	0	0	0	0	**1**	0	**10**	0	0	11	0,91
	Q	0	0	0	0	0	0	0	0	0	0	0	**7**	0	7	1,00
	K	0	0	0	0	0	0	0	0	0	0	0	0	**7**	7	1,00
Suma		3	3	9	5	10	12	11	8	4	10	11	7	7	100	**Media Precision**
Recall		1,00	1,00	1,00	1,00	0,80	1,00	1,00	1,00	0,75	1,00	0,91	1,00	1,00	**Media Recall**	**95,84%**	**97,24%**
F1		1	1	0,9	1	0,9	1	1	1	0,86	0,95	0,9	1	1	**Media F1**	**96,27%**	

Como conclusión se puede observar como el palo peor detectado es el diamante, y la carta peor detectada es el 5, aunque también hay errores en el 9 y J, por lo tanto, si quisiéramos mejorar el diseño se podrían rediseñar los modelos de análisis de estos números y del palo de diamantes.

Sin embargo, la media de precisión, recall y F1 son muy elevadas siendo superiores al 93%, por lo tanto, podemos decir que nuestro sistema es confiable.

Video Explicativo

- Video explicativo

Aquí el enlace del vídeo explicativo subido a Youtube:
https://www.youtube.com/watch?v=ydgz4GcLAl8