UNIVERSIDAD POLITÉCNICA DE MADRID

Reconocimiento de cartas de póker mediante visión artificial

Carlos Morillo Lozano – M15329

11/01/2016

Contenido

1. Reconocimiento de cartas de póker

El problema que se plantea consiste en la determinación del valor y el palo de cualquier carta de una baraja de póker. De este modo se tiene una muestra de cuatro palos: rombos, corazones, picas y tréboles; con diez cartas y tres figuras por cada palo. Además la baraja también consta de tres cartas iguales adicionales que corresponden con el joker. En total se deberán de poder reconocer 55 cartas diferentes.

2. Planteamiento de la solución

2.1. Baraja elegida

Antes de nada es necesario conocer el problema exacto que se va a tener, para ello hay que seleccionar una baraja de muestra que sirva para comprobar la efectividad del algoritmo.

La baraja elegida para el desarrollo del trabajo ha sido la que se ve en la siguiente figura. Presenta la particularidad de que los colores típicamente conocidos están invertidos, sin embargo esto ni complica ni facilita el desarrollo del algoritmo. Algo que si lo complica es la ausencia de colores en los palos, pero para salvar esa dificultad basta con hacer más robusta la identificación de la figura.

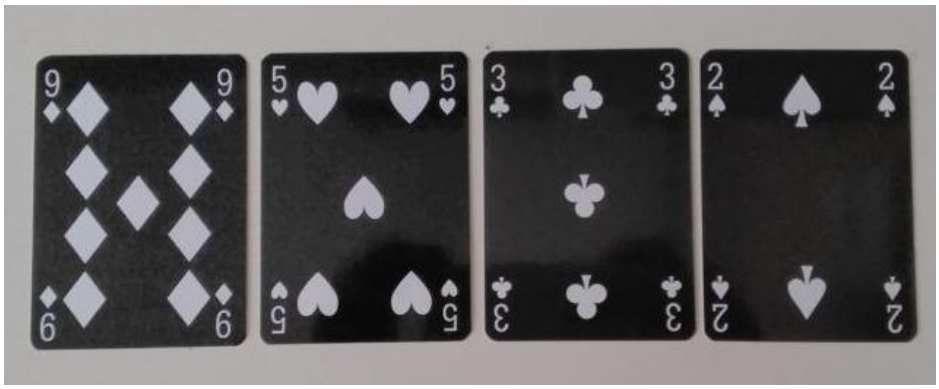


Ilustración 1: Muestra de la baraja

2.2. Disposición de la escena

Dado que las cartas son de color negro, se necesitará un fondo blanco para aumentar el contraste entre ambos objetos y así poder reconocer mejor el naipe. La cámara debe estar situada en un punto que otorgue una vista cenital de la escena, por ello se ha elaborado una estructura que la ubica a una altura de 41 cm.

Dicha estructura ha sido elaborada a partir de un mástil metálico y piezas impresas en 3D para la correcta colocación del mástil y de la cámara. El diseño se ha pensado para ser desmontable y por ello que tenga facilidad de transporte.

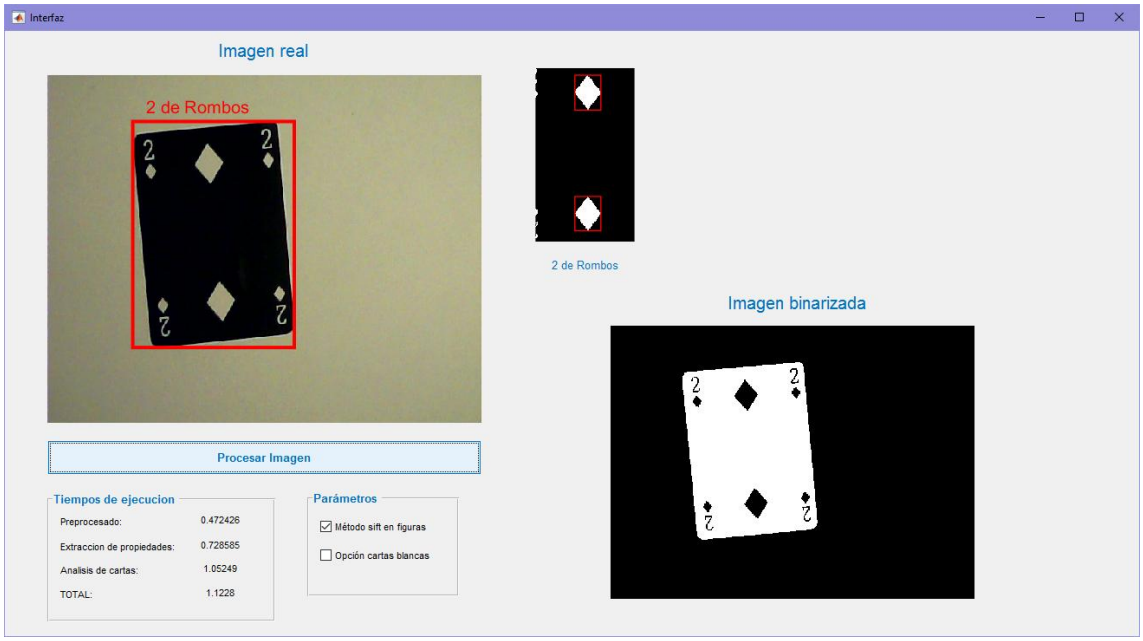
También se ha elaborado un foco mediante una matriz de Leds de alta potencia con un consumo de 12 W. La iluminación proporcionada por el foco no se realizará de forma directa si no que la luz iluminará la escena a través de su reflexión en unos paneles de color blanco. Esto se realiza para lograr una iluminación uniforme en la escena y evitar claroscuros.

2.3. Interfaz de control

El control de todo el programa se realiza a través de una interfaz muy sencilla que permite la visualización de las imágenes captadas por la cámara en todo momento. Mediante el botón se realiza la captura de la imagen y se muestran los resultados en distintas imágenes que permiten visualizar parte del proceso de análisis.

Los parámetros permiten realizar dos ajustes, el primero activa el uso del método sift para la detección del palo en las figuras. El segundo permite activar un modo de reconocimiento de cartas blancas, es decir, cartas comunes que son el invertido de la baraja que se ha empleado aquí.

Por último, hay un apartado para poder visualizar los tiempos de ejecución de las distintas fases del programa.



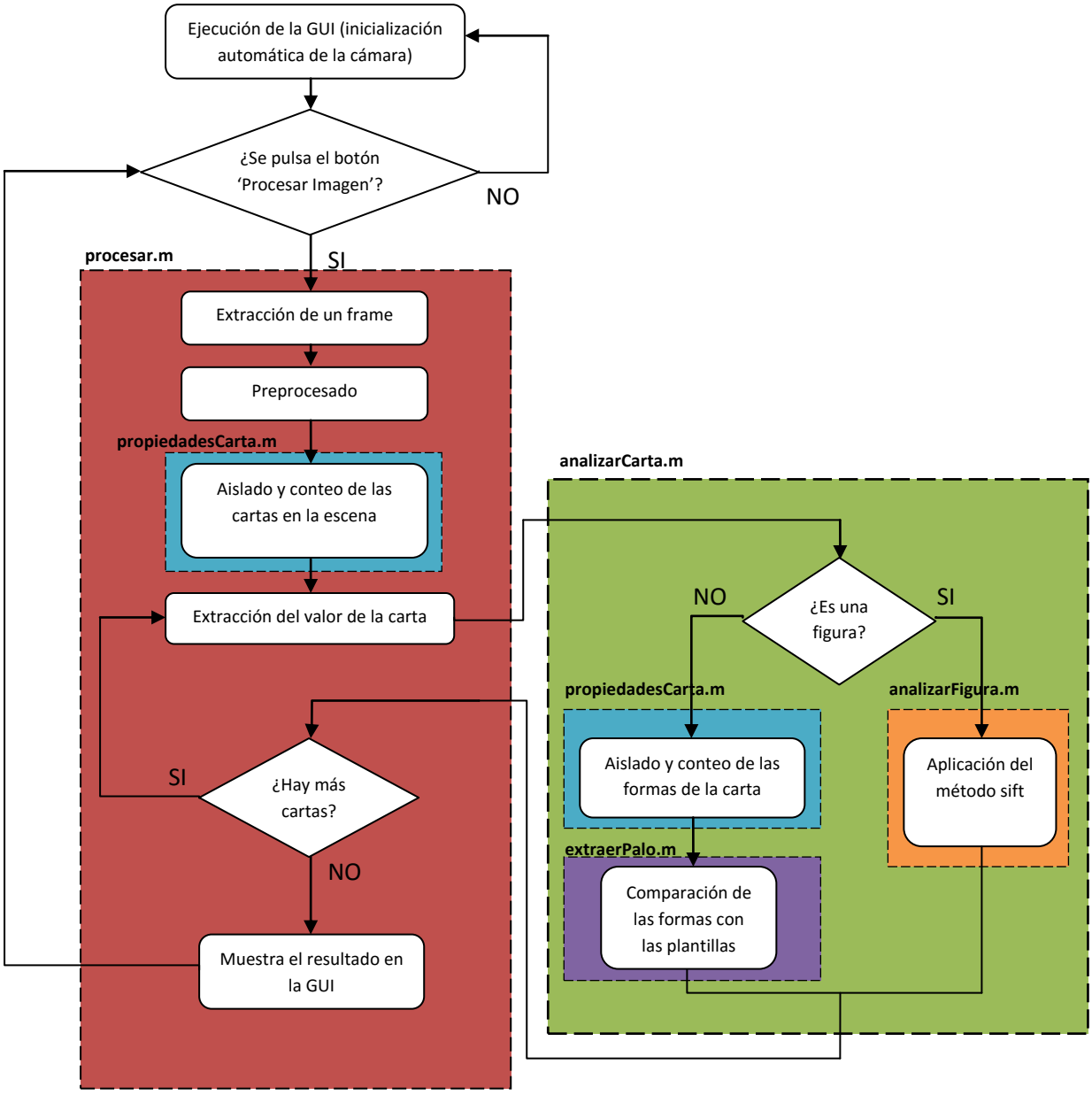
2.4. Planteamiento de la solución

Para la detección de todas las cartas se han empleado métodos diferentes que fuesen otorgando información sobre la escena que facilitase el reconocimiento. Hay que destacar la diferencia entre los métodos de reconocimiento de las cartas con figuras o el joker del resto de cartas con valor numérico.

2.4.1. Ciclo de ejecución

El ciclo de ejecución de las funciones más relevantes para el desarrollo del programa se puede ver a continuación. Si se desea tener más información sobre cada función en particular se puede consultar el anexo ubicado al final de este documento.

Ciclo de ejecución simplificado con las funciones más representativas es:



2.4.2. Explicación de la ejecución

Para comprender mejor las tareas realizadas en casa fase, es conveniente explicar brevemente algunos de los pasos realizados durante la ejecución con algún ejemplo. Dada la siguiente imagen inicial se procederá a su procesado:

Ilustración 2: Imagen real

- **Preprocesado**: durante la etapa de Preprocesado se llama a una función encargada de transformar la imagen inicial con color que capta la cámara a una imagen que nos facilite el trabajo en las sucesivas iteraciones, por ello en dicha función se binariza la imagen y se filtra de manera adecuada para eliminar el ruido provocado por fallos de captación y para suavizar las formas de la imagen.

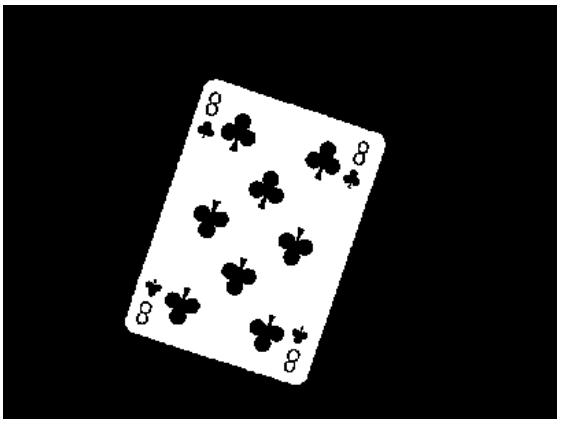


Ilustración 3: Imagen preprocesada

- **Extracción de propiedades**: en esta etapa se ejecuta la función *propiedadesCarta.m* y permite realizar el conteo del numero de cartas en la imagen mediante la función de Matlab *regionprops*. Además también permite la extracción de numerosas propiedades como el área que abarca cada carta o su inclinación. Para evitar tener en cuenta de propiedades de zonas no deseadas, se eliminan previamente los contenidos que hay en las cartas como se ve en la imagen.

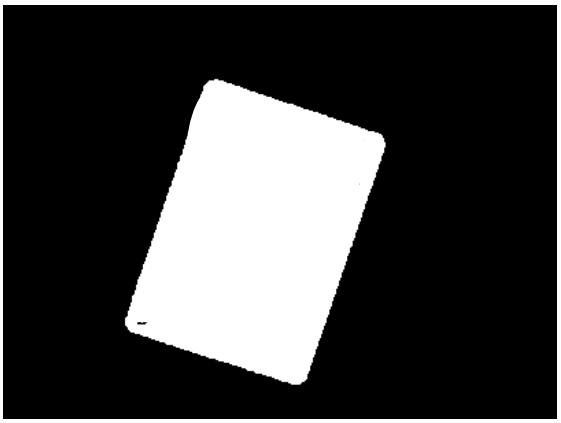


Ilustración 4: Imagen para extracción de propiedades

- **Análisis de la carta**: en este punto ya se sabe cuántas cartas hay y su posición, de manera que el siguiente paso a realizar será analizar cada carta individualmente para poder extraer la información. Antes de comprobar si se trata de una figura o de una carta normal, cada naipe es orientado y colocado en posición vertical mediante la función *imrotate*. Una vez orientada se recorta la carta para aislarla y poder estudiarla por separado, se vuelve a ejecutar *propiedadesCarta.m* que en esta ocasión dará la información sobre las formas contenidas en el naipe y por comparación en el área del rectángulo de dichos resultados se determinará si la carta es una figura o no lo es.

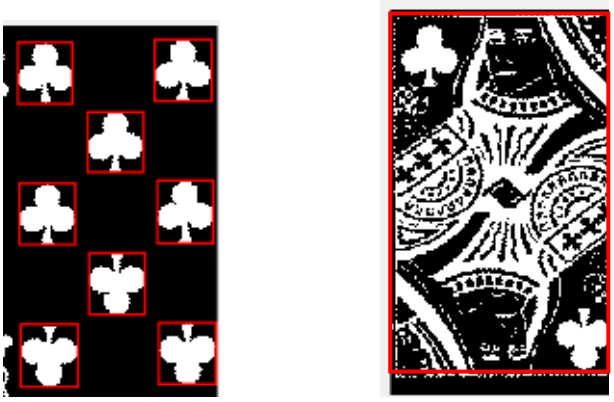


Ilustración 5: Carta normal y carta con figura

2.4.3. Cartas con valor numérico

Cuando se detecta que la carta es de tipo 'normal' en la función *analizarCarta.m* ya se tiene detectado la cantidad de formas en el interior de la carta, pero como la imagen cortada no suele ser exacta de manera que elimine las no deseadas, hay que hacer un filtrado por tamaño, eliminando elementos pequeños como el que se ve en el rectángulo amarillo de la figura y quedándose con los recuadrados en rojo.

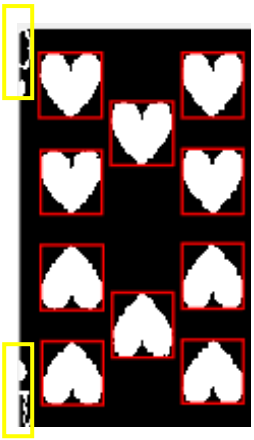


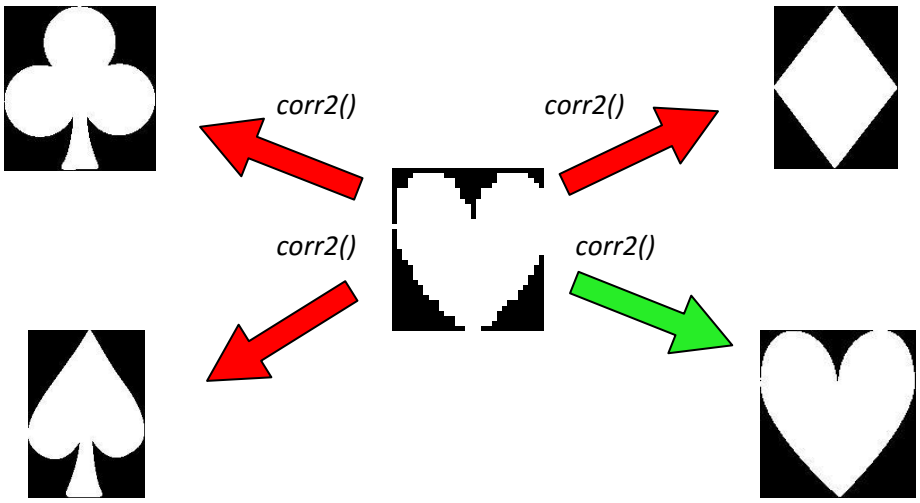
Ilustración 6: Imagen de la carta recortada

Una vez se han detectado y aislado todas las formas del interior de la carta, hay que determinar tanto el valor de la misma como el palo al que pertenece, esto se realiza en dos fases:

- **Valor del naipe**: la variable *propiedades_intrinsecas* es vector de información sobre cada una de las formas de la figura recuadradas en rojo que se obtuvo al llamar a la función *propiedadesCarta.m*. El número de filas de ese vector es igual al número de formas que hay y por lo tanto, es igual al valor del naipe. Por lo tanto se puede averiguar con la simple instrucción:

  valor_ naipe= size(propiedades_intrinsecas,1);

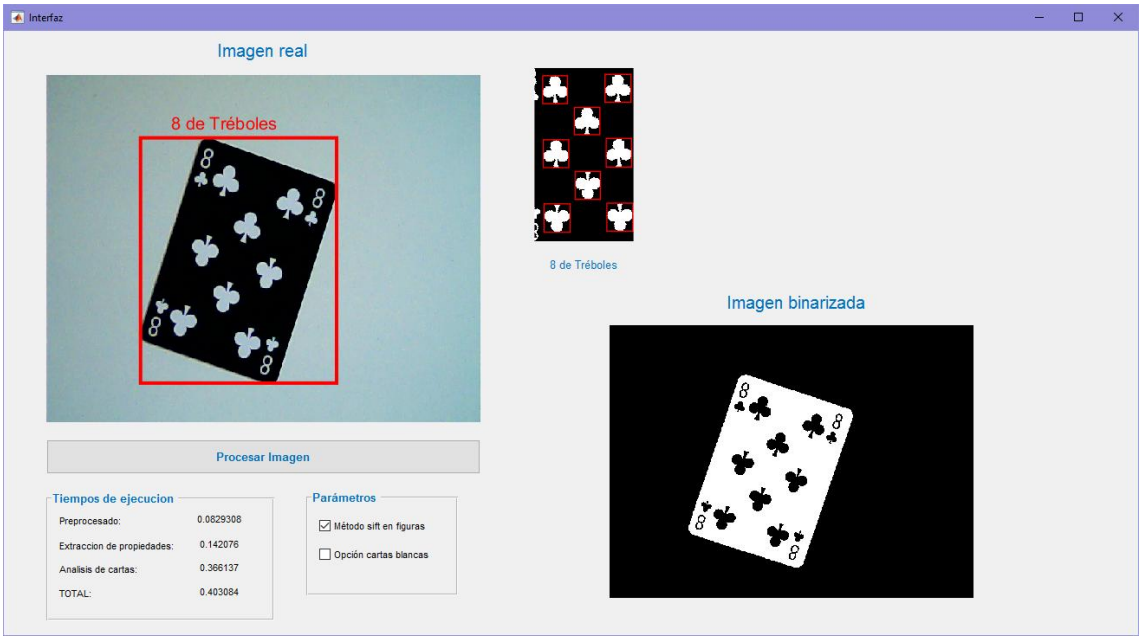
- **Palo del naipe**: en este caso hay que realizar algunas operaciones más y se utiliza una nueva función llamada *extraerPalo.m*. En ella se aísla una de las formas que se tienen de la imagen mediante *imcrop.* La imagen resultante será la forma del palo totalmente aislada del resto. Dado que solamente hay cuatro posibilidades no tiene sentido elaborar un algoritmo de machine learning. Para resolverlo se han tomado cuatro plantillas de cada forma y se comparan mediante la funcion *corr2* con la figura real. Aquella plantilla que mayor índice de correlación presente será la que indique el palo de la carta.



El método de correlación presenta un inconveniente, las imágenes o matrices que se introducen como argumento deben ser de las mismas dimensiones, por lo que es necesario realizar un pequeño escalado de la imagen real antes de poder compararla con el resto de imágenes. Se realiza en dos fases, en primer lugar se ajustan las filas mediante *imresize* y en segundo lugar, en caso de que las columnas no coincidan, se añaden a la imagen real o a la de plantilla columnas de ceros hasta igualar tamaños. Tras el resize que ajusta las filas, las columnas quedan muy igualadas y como mucho hay que añadir tres, lo que no modifica la correlación con la plantilla correcta y a la vez la empeora con las incorrectas.

Por último, ya que se tiene toda la información sobre la carta, solo queda ordenarla en una misma estructura. Todos los valores retornan a la función de *procesar.m* donde quedan almmacenados en un vector de cartas en las que cada componente del vector es una estructura con las variables 'valor' y 'palo'. La dimensión de dicho vector es dinámica en funcion de las cartas que haya en la escena.

El ejemplo antes propuesto de la carta del 8 de tréboles finalizaría de la siguiente forma:



2.4.4. Figuras y joker

Ante la gran diferencia que presentan este tipo de cartas con las anteriores es imposible determinar su valor y palo con los mismos métodos empleados para los naipes 'normales'. Para ello se han desarrollado diversos métodos alternativos. Cuando se detecta en la función *analizarCarta.m* que la carta analizada es una figura, automáticamente se lanza la función *analizarFigura.m* que realizará todas las operaciones siguientes.

En primer lugar cabe destacar que las figuras son doce cartas, cuatro reyes, cuatro reinas y cuatro sotas una de cada palo. En primer lugar se podría pensar en realizar una lectura del palo ya que es una imagen clara en la carta. Para determinar el valor (J,Q o K) sería necesario

distinguir los reyes, las reinas y las sotas del resto de figuras, sin embargo, surge un gran problema; cada una de las doce cartas es diferente entre sí y los patrones para la diferenciación son prácticamente nulos y los pocos que existen se pierden la mayoría de las veces en la captura de la imagen debido a su pequeño tamaño. La solución planteada para este caso es la utilización del método de localización de puntos característicos sift.

- **Palo de la figura (sin sift):** dado que el palo de las figuras se puede calcular gracias a aislar la zona donde aparece la forma del palo. Esto se realiza de la misma forma que en las cartas 'normales'. Tras conocer y aislar esa forma se ejecuta la función ya conocida *extraerPalo.m* que se encargaba de analizar la correlación del palo con las plantillas guardadas.
- **Palo y valor de la figura mediante sift:** como se ha dicho, el valor J,Q o K de la figura se va a realizar mediante correlación de puntos obtenidos con sift. Este método va a devolver la información de la carta completa, por lo que el palo también se determina en este proceso.
  En primer lugar hay que explicar en qué consiste el método sift. El significado de sus siglas es Scale-invariant feature transform y localiza puntos en la imagen que son de interés en función de unos parámetros considerados invariantes a la iluminación, contrastes, rotaciones o traslaciones.
  La aplicación que tendrá al algoritmo desarrollado será la de extraer los puntos de interés de la figura real captada y compararlo con los puntos de interés de todas las figuras guardadas previamente en una base de datos mediante la función *match.m*. Aquella que más puntos logre casar será la figura correcta y dará el valor y el palo de la carta. Hay que destacar que para la comparación las imágenes no se preprocesan para no perder información.

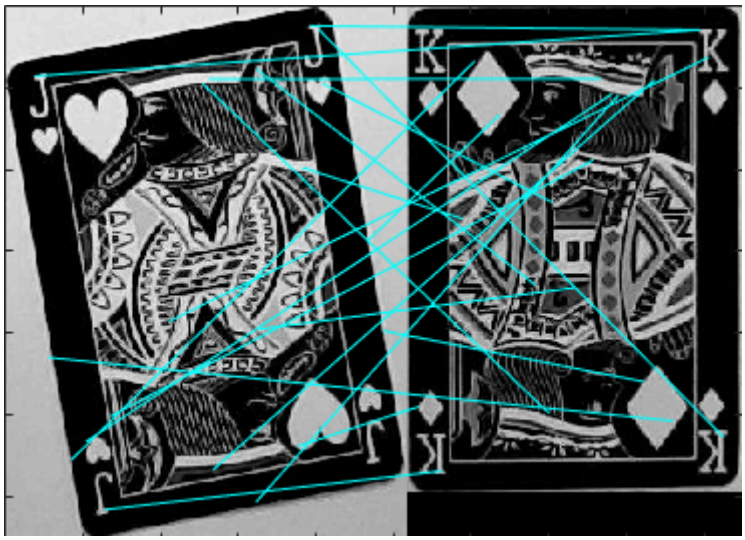


Ilustración 7: Unión de puntos en figuras diferentes (Puntos casados=20)
Izquierda: imagen real / Derecha: plantilla

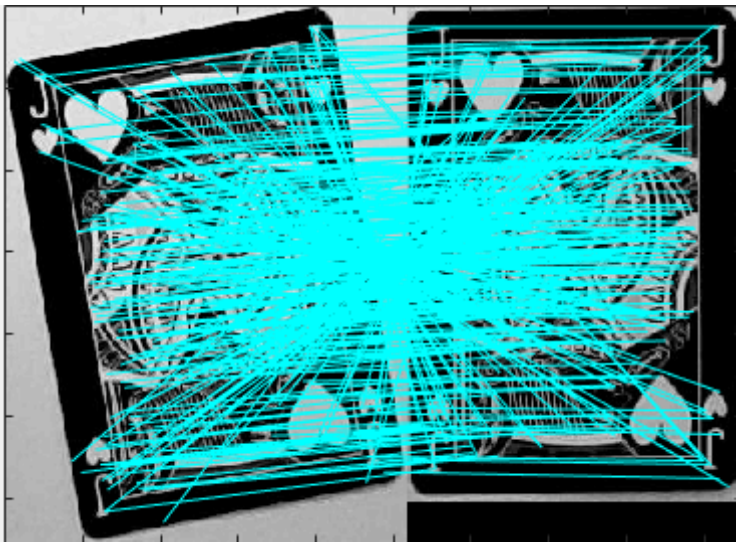


Ilustración 8: Unión de puntos en figuras iguales (Puntos casados=444)
Izquierda: imagen real / Derecha: plantilla

Cabe destacar que para el cálculo de la relación entre imágenes sólo se hace sift de la imagen real dado que para disminuir tiempo de procesamiento los puntos característicos de las plantillas ya están almacenados previamente. Como puede verse la rotación de la imagen es indiferente gracias a sift.

Se ha hablado de dos métodos para la detección del palo y por ello se puede seleccionar a través de una checkbox de la interfaz la posibilidad de calcular el palo mediante un método u otro en función de lo deseado. La diferencia es que el valor del 'palo' en el vector de cartas vendrá de hacer sift o de hacer correlación. Por el contrario el 'valor' sólo se podrá determinar con sift.

3. Anexo - Funciones

A continuación se explican las funciones empleadas en el programa para su mejor comprensión. Las explicaciones están sacadas de los comentarios que se encuentran en la cabecera de cada uno de los archivos .m al que pertenecen.

3.1. grabar.m

Esta función se encarga de recoger el frame de video que se va a procesar y de llamar a la función *procesar.m* donde se realizará dicha operación.

```
<Valores_naipes>=grabar(<camara>)
```

3.2. procesar.m

Es la función principal que se encarga de llamar a todas las funciones auxiliares necesarias para lograr los requisitos de detección. Se introduce el fotograma que se ha captado de la cámara y devuelve los valores de las cartas.

```
<Valores_naipes>=procesar(<frame_captado>)
```

Pasos:

1- preprocesamiento de la imagen a color
2- extracción de las propiedades (conocer el numero de cartas)
3- análisis de cada carta individualmente
4- muestra por pantalla el resultado de los cálculos

3.3. preprocesar.m

Se encarga de realizar una conversión previa de la imagen captada para adaptarla a facilitar las necesidades del resto de funciones y procesos.

```
<frame_preprocesado>=preprocesar(<frame_captado>)
```

Explicación de cada paso:

1- Convertir de RGB a Gray --> im_gray=rgb2gray(im_inicial);
2- Aumentar la luminosidad de la imagen --> im_eq= imadjust(im_gray,[0;1],[0;1],0.85);
3- Convertir de Gray a binario --> im_bin=im2bw(im_gray);
4- Se le aplica un filtro de apertura y cierre para eliminar pixeles aislados no deseados -->

```
SE=strel('square',1);
im_open=imopen(im_bin,SE);
im_close=imclose(im_open,SE);
```

5- Se eliminan huecos de menor diametro al indicado -->

```
im_close_menor=bwareopen(im_close,5);
```

3.4. propiedadesCartas.m

Coge la imagen preprocesada para contar las cartas que hay en total en la mesa y determina las propiedades que encuentra en cada una de ellas como área, perímetro, orientación…

```
<propiedades> = propiedadesCartas(<frame_preprocesado>)
```

Conversión de la imagen paso a paso:

1- Rellenar las cartas completamente --> im_fill=imfill(im_bin,'holes')
2- Contar el numero de cartas que hay --> [im_label,N]=bwlabel(im_fill,4);
3- Obtener las propiedades de cada carta individualmente -->

```
propiedades=regionprops(im_label,'Boundingbox','Centroid','Orientation');
```

3.5. analizarCarta.m

Se encarga de analizar cada carta individualmente para dar el valor y el palo del naipe.

```
<Valor_de_un_naipe>=analizarCarta(<frame_preprocesado>,<propiedades>)
```

Explicación del proceso:

1- Orienta la carta
2- Recorta la carta
3- Se invierten los colores
4- Se extraen las propiedades de las figuras de la carta
5- Se eliminan de la lista las figuras de áreas reducidas
6- Comprueba si se trata de una figura
7- Se comparan las formas para obtener las figuras y se obtienen los valores de la carta

3.6. analizarFigura.m

Se encarga de analizar cada FIGURA individualmente para dar el valor y el palo del naipe. Recibe como argumento la imagen tomada por la cámara pero con el naipe orientado.

```
<Valor_de_un_naipe>=analizarFigura(<frame_orientado>,…
 <propiedades_orientadas>,<propiedades>)
```

Etapas:

- Modo de detección del palo CON sift:

1- Correlación mediante sift

- Modo de detección del palo SIN sift:

2- Se recorta la parte de la carta con la información del palo
3- Se elimina todo el ruido posible para aislar el palo del resto de la carta
4- Se extraen las propiedades del palo
5- Se elimina todo menos la imagen del palo en sí
6- Asignación del palo correspondiente

3.7. extraerPalo.m

Se encarga de analizar la correlación de cada símbolo de las cartas y asignarle a un palo correspondiente

```
<palo>=extraerPalo(<carta_recortada>,<propiedades_intrinsecas>)
```

Explicación del proceso:

1- Se extrae de la imagen real una imagen del símbolo
2- Se guardan en memoria las plantillas con las que se va a comparar la imagen
3- Se añaden las columnas necesarias a la imagen para que sea del mismo tamaño que la real
4- Se comparan ambas imágenes mediante el método de correlación
5- En función del resultado se devuelve un palo correspondiente

3.8. extraerFigura.m

Se encarga de comparar mediante sift todas las figuras con plantillas previamente relacionadas de cada figura de la baraja.

```
[<valor>,<palo>]=extraerFigura(<frame_captado>,<propiedades>)
```

Etapas

1- Se recorta la zona de la carta
2- Se guarda la carta en memoria para el posterior procesado de sift
3- Se almacenan los descriptores y las localizaciones mediante sift
4- Se cargan los descriptores y localizadores de todas las plantillas
5- Se comparan las plantillas con la muestra uniendo los descriptores de cada imagen
6- Se asigna un valor a la carta en función del grado de correlación entre los diferentes análisis

3.9. match.m

Se encarga de realizar la unión de los puntos característicos de las plantillas de las figuras y de la carta que se está analizando en ese momento. Devuelve la cantidad de puntos unidos por el algoritmo.

```
<puntos_unidos>= match(0.8,<descriptor1>,<descriptor2>,<loc1>,<loc2>)
```

El valor de 0.8 es el umbral que determinará la precisión con la que unirá los puntos de ambas imágenes. Los descriptores son las características de cada punto en la imagen y los valores de loc son las localizaciones de esos descriptores en cada imagen.

3.10. Interfaz.m

Únicamente se encarga de inicializar la GUI y de gestionar el uso de los botones y checkboxes. Además también permite la visualización de todos los resultados en pantalla de forma cómoda.