# Bot Póker Online

Pau Cebrian-Trunas

**Resumen–** A través de este documento se propone una aplicación que permite substituir a un jugador humano en servicios de póker online. Para realizar dicha función, hago uso de herramientas basadas en el procesamiento de imágenes, para la extracción de la información de las mesas de juego, y un sistema experto basado en casos para la toma de decisiones. En este artículo se explica el desarrollo de esta aplicación dividida en cuatro elementos principales, la extracción de información, la toma de decisiones, la interacción entre sus componentes y el paralelismo empleado.

**Palabras clave–** Bot, Filtros de Color, OCR, Póker Texas Hold'em, Reconocimiento, Rendimiento, Sistema Experto.

**Abstract–** Through this document is proposed an application that allows to replace human players in online poker services. To accomplish that function, I apply computer vision based tools to extract the information of the gaming tables, and a case-based reasoning expert system for the decision making process. This article describes the development of this application divided into four main elements, the information extraction, the decision making, the interaction between components and the implemented parallelism.

**Keywords–** Bot, Color Filters, Expert System, OCR, Recognition, Texas Hold'em Poker, Throwput.

## 1 Introducción

Este proyecto consiste en dar solución al problema de simular a un usuario humano en juegos de póker online. La razón del tema viene dada por el gran reto que suponen los juegos dinámicos de información incompleta [1] para la inteligencia artificial. El póker online y en especial su versión de juego Texas hold'em, resulta un área perfecta sobre la que aplicar técnicas de esta rama de la ingeniería informática, ya sea utilizando detectores de objetos, con el fin de extraer la información representada en la mesa de juego, o sistemas expertos, para obtener beneficios según el estado de la partida.

El póker es un juego en el que cada partida consta de cuatro turnos en los que pueden realizarse apuestas. En el primer turno, llamado *pre-flop*, se le entregan 2 cartas a cada participante, ningún jugador conoce el valor de las cartas del oponente en este momento, además, dos de los jugadores deben realizar apuestas mínimas, son los llamados small blind y big blind. En el segundo turno, *flop*, se muestran 3 cartas en el centro de la mesa, en el tercer turno, *turn*, se añade otra carta mas al centro de la mesa, y en el útilimo turno, *river*, se añade una quinta carta a la mesa. La finalidad de estas cartas comunitarias es la de formar figuras (combinaciones) con las cartas privadas de los jugadores. El bote de una partida lo gana el jugador que forme la figura de mayor valor. A lo largo de los turnos se realizan apuestas para forzar la salida de otros jugadores o conseguir que añadan fichas al bote de la partida. El objetivo final del juego es acumular el máximo número de fichas a lo largo de estas partidas.

En este artículo se utilizará terminología asociada al póker y se supone que el lector conoce en profundidad las reglas del juego, para más información consultar las reglas oficiales en este enlace.

Con este proyecto pretendo aportar una solución simplista y efectiva a este problema, la cual se puede dividir en dos apartados diferenciados, uno relacionado con la inteligencia artificial y otro apartado relacionado con la aplicación de estas soluciones de inteligencia artificial en un entorno real. Para el primer apartado, relacionado con la inteligencia artificial, se ha tenido que lidiar con la extracción de información de la mesa y la toma de decisiones. Para el apartado de la aplicación de las soluciones anteriores en un entorno real, se ha tenido que trabajar

• E-mail de contacto: pau.cebrian@e-campus.uab.cat
• Mención realizada: Computación
• Trabajo tutorizado por: Aura Hernàndez-Sabaté (Ciencias de la Computación)
• Curso 2015/16

con la optimización del programa para conseguir unos resultados dentro de un tiempo de reacción aceptable, y la entrada/salida de datos del sistema.

Así, la organización del proyecto queda de la siguiente manera:

- Soluciones basadas en inteligencia artificial:
  - Captura de información de las mesas de juego.
  - Sistema experto para la toma de decisiones
- Aplicación a un entorno real:
  - Paralelización y optimización del programa.
  - Integración de elementos del sistema.


Fig. 1: Iteraciones Metodología Ágil

## 2 Estado del Arte

De forma similar a como pasó con el ajedrez, existen bots de póker y grupos de investigación centrados en conseguir un sistema capaz de derrotar a cualquier jugador profesional. Alguno de los nombres de referencia de estos bots són Cepheus o Polaris [2], ambos de la Universidad de Alberta, o Claudico [3], el cual necesitó hacer uso del supercomputador Blacklight para sus cálculos. No obstante, los éxitos en estos proyectos han sido más bien discretos.

Como añadido, actualmente se realizan competiciones anuales entre los mejores bots de póker y se han introducido como atracción en alguno de los campeonatos mundiales de póker como un jugador más.

Desafortunadamente, al contrario que con el caso del ajedrez, aún no se ha encontrado una solución a este problema y existen muy pocas aproximaciones publicadas [4]. Debido a esto, apenas se ha encontrado literatura en la que basarse para realizar este proyecto.

Las soluciones existentes para aportar ventajas a los jugadores están centradas en tablas de acciones recomendadas que ha generado la comunidad. Estas tablas se han creado en función de las probabilidades de las cartas, los beneficios que ofrece la posición de la mesa en la que se encuentra el jugador dentro de la partida, y teniendo en cuenta las acciones realizadas por los oponentes anteriormente.

## 3 Metodología

Para desarrollar el proyecto se ha utilizado una metodología de desarrollo ágil del software, marcando una lista de prioridades y adaptando cada uno de los módulos del proyecto para conseguir un software funcional dentro de los límites de tiempo establecido. Se muestra un diagrama de esta metodología en la figura 1.

Para cada uno de los módulos planeados se ha realizado una fase de investigación previa de soluciones existentes, con el correspondiente análisis de cuál de ellas se adapta mejor al proyecto y una fase breve de testing de esta solución escogida, de esa forma se ha podido comprobar la aplicación real de la cada solución. Posteriormente se han realizado diseños de los módulos en cuestión para encajar dentro del sistema del bot, una fase de implementación del código y una última fase de pruebas de validación para ver cuán efectiva es la solución implementada, con sus respectivas iteraciones para la mejora de los resultados obtenidos.

El incluir una fase de testing previa dentro de la fase de descubrimiento ha permitido desestimar muchas soluciones, que en un principio parecían buenas candidatas, pero cuyos resultados reales se alejaban de los esperados, ya fuera por cuestiones de rendimiento o validación.

En el apéndice del artículo se adjunta el diagrama de Gantt 13 que muestra cada una de las fases del desarrollo y sus dependencias temporales.

A continuación se muestra en detalle las soluciones encontradas para cada uno de los módulos principales del proyecto.

### 3.1. Captura de Información

Este primer módulo del programa se encarga de extraer toda la información necesaria del estado de la mesa para la posterior toma de decisiones. Esto implica saber cuál es el bote acumulado, con que cartas se está jugando y los datos sobre los oponentes que están participando, es decir, sus posiciones respecto al dealer, saber si están jugando u observando, y la cantidad de fichas que posee cada uno.

En la figura 2 se puede observar un ejemplo de las mesas de juego sobre las que actuará el bot desarrollado.

Cabe destacar que la correcta captura de cierta información es crítica para este proyecto, ya que, si los datos sobre los que se basa la decisión no son correctos, es altamente improbable que la acción escogida sea la mejor opción para el contexto real del juego. Por ese motivo, es en este módulo donde se han centrado la mayoría de esfuerzos y recursos del proyecto, tanto en tiempo dedicado al desarrollo como a la capacidad de hardware implicada


Fig. 2: Ejemplo de mesa de juego

en la obtención de resultados.

La solución implementada para la extracción de información de la mesa de juego se ha conseguido utilizando segmentación de imágenes con un espacio de color HSV [5]. Con esto se consigue detectar las regiones de interés de la imagen, como por ejemplo las relacionadas con la posición de los jugadores activos. De esta forma se pueden aislar los elementos necesarios del resto de la imagen.

Los datos obtenidos de la segmentación se utilizan en combinación con detectores OCR, para extraer toda la información relacionada con cadenas de caracteres, es decir, valores de las cartas, bote y fichas de los jugadores.

Respecto al proceso de segmentación de imágenes, se ha utilizado el espacio de colores HSV, es debido a que al definir el pixel mediante tono, saturación y valor, el pixel queda mucho mejor representado que usando el espacio RGB. Este cambio de espacio de color supone grandes ventajas al aislar los elementos cuyos colores se alejen de los primarios rojo, verde o azul. No obstante, debido a que las imágenes se almacenan en un espacio RBG, se debe aplicar la correspondiente transformación a HSV en cada imagen, lo que implica un coste computacional relativamente elevado. En el apartado de paralelismo y optimización se explicará cómo se ha lidiado con esto.

Se puede observar parte del proceso de segmentación, aplicado a la detección de los jugadores activos, en la figura 3.

Inicialmente se hicieron pruebas para la captura de información con LBP y HOG [6], pero ya con las primeras ejecuciones se hizo evidente la complejidad innecesaria que esto suponía, sobre todo teniendo en cuenta el problema tan simple con el que nos encontramos, es decir, separar elementos fácilmente identificables mediante colores, sin ningún tipo de variación de forma, escala, rotación ni posiciones en la mayoría de los casos. Los cálculos para obtener el LBP y el HOG resultaron ser considerablemente lentos, y la comprobación posterior con ventana deslizante, no generaba ningún resultado que sirviera para identificar los elementos de forma inequívoca.

Por otro lado, la generación de candidatos para el entrenamiento de HOG habría supuesto un incremento de faena


Fig. 3: Segmentación para extraer jugadores activos

importante, y pensando en que el objetivo del proyecto es un único proveedor, lo que implica que el formato de las mesas no va a cambiar, esta captura de información se resuelve de forma mucho más eficiente mediante la segmentación de imágenes.

Se debe tener en cuenta que este proyecto está condicionado por la poca cantidad de proveedores de servicios de póker existentes en esta región, y a que la mayoría de la comunidad de jugadores están en un único proveedor. Esto ha influido en el diseño de la solución, especialmente en el módulo de captura de información.

## 3.2. Sistema Experto

El objetivo de este módulo era, dada la información extraída de la mesa, encontrar la mejor opción de juego para conseguir maximizar los beneficios a largo plazo, ya fuera apostando, pasando, o abandonando la mano.

La solución que propongo a este problema es un sistema experto basado en casos. Esto es debido al tipo de recursos encontrados en la literatura de referencia para mejorar las acciones de los jugadores: tablas de acciones recomendadas según fase de la partida, posición en la mesa de juego y cartas del jugador [7]. El mayor problema que presentaban estas tablas residía en la falta de estandarización entre las distintas fuentes. Teniendo en cuenta el tipo de contenido encontrado, se ha decidido tratar cada una de las fases de juego por separado.

Por un lado, para la fase de preflop, se han creado matrices de 13x13 que almacenan las acciones para todas las posibles combinaciones que se pueden tener con las dos cartas privadas repartidas al jugador al inicio de la partida. Puede parecer extraño no usar una matriz triangular, pero esto es debido a que en el póker dos cartas del mismo palo tienen un valor añadido a las mismas dos cartas de palos distintos, por esto, la matriz triangular inferior guarda las acciones correspondientes a parejas de cartas de distintos palos y la triangular superior, sin la diagonal, las acciones correspondientes a parejas de cartas del mismo palo. Se muestra un ejemplo de estas tablas en la figura 4

Para los casos de las fases de flop, turn y river se ha recopilado un listado de figuras con las que seguir jugando relacionado con el valor de estas o el valor de su carta más alta, tal y como se puede observar en la figura 5, tratando cada una de las fases por separado pero con la misma metodología.

Con el fin de mejorar la calidad del sistema experto, se buscó tener en cuenta la psicología de los oponentes en el momento de tomar la decisión de juego [8]. Para esto se consideró realizar una clasificación no supervisada de los jugadores, a partir de logs de partidas donde se almacenan totas las acciones realizadas por cada uno de los participantes, con esto se quería extraer las características de distintos estilos de juegos y forzar o prever acciones de los oponentes en base a estos estilos [9].

Si bien la clusterización no parece suponer ninguna gran dificultad, el problema en este planteamiento reside en extraer un conjunto de características que definan a un jugador a partir de los logs de jugadas. Esta correcta definición de jugadores es algo que ha quedado fuera del proyecto debido a la gran dificultad y consumo de recursos que supone en contraposición con lo poco que altera el proceso actual. Pudiendo ser una investigación de lo más interesante, se consideró más oportuno excluirla del presente proyecto.

## 3.3. Paralelismo y rendimiento

Uno de los rasgos necesarios de este proyecto tenía que ver con el rendimiento en la decisión de la jugada. Esto es debido a que en las mesas de póker online existe un límite de tiempo para cada jugada. En caso que un jugador no realice ninguna acción dentro de ese tiempo, será expulsado de la partida. Por otro lado, aunque en partidas de dinero ficticio el límite de mesas jugables paralelamente esté fijado en seis, no existe tal límite en partidas de dinero real, y el objetivo, teniendo una función ganadora, es maximizar el beneficio jugando al mayor número de mesas simultáneamente.

Debido a esto, se requiere que el tiempo invertido en la captura de información sea mínimo, y la toma de decisión en base a estos datos también. De otra forma se acumularán funciones en la cola del sistema y el bot acabará expulsado de todas o gran parte de las mesas de juego.

Por los anteriores motivos, se debe prestar especial atención a aspectos referidos con el rendimiento.

Uno de los primeros elementos importantes propuestos para esta solución es el paralelismo. Resulta bastante evidente la necesidad de crear un proceso por cada mesa de juego que se encargue de la captura de pantalla correspondiente, la segmentación y la toma de decisión de forma independiente a otros procesos. Así, el número máximo de mesas simultaneas vendrá dado por la capacidad de hardware de nuestro sistema en relación con las necesidades computacionales de la extracción de datos y la posterior toma de decisión, por lo tanto, se ha buscado que estas necesidades computacionales sean mínimas.

Haciendo un análisis superficial de rendimiento sobre el código, se pudo observar que la mayor parte del tiempo del proceso se dedicaba a dos únicas funciones, el cambio de espacio de color de RGB a HSV, y el OCR.

También se hizo evidente que, ejecutando la extracción de información de la mesa al completo, en un loop infinito sin esperas entre iteraciones, la mayor parte de la información generada era redundante. Debido a esto, se planteó la solución de ejecutar esta extracción completa una única vez, cuando el sistema detecte que es un nuevo turno de juego del bot. Además, esta detección de nuevo turno, la cual se debe ejecutar conitnuamente, se comprobará sobre un espacio de color RGB, evitando todo coste referido a la transformación del espacio de colores.

| | A | K | Q | J | T | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | Raise | Raise | Raise | Raise | Raise | Raise | Raise | Raise | Raise | Raise | Raise | Call | Call |
| K | Raise | Raise | Raise | Raise | Raise | Call/Fold | Call/Fold | Fold | Fold | Fold | Fold | Fold | Fold |
| Q | Raise | Raise | Raise | Raise/Call | Raise/Call | Call | Call | Fold | Fold | Fold | Fold | Fold | Fold |
| J | Raise | Raise | Call | Raise | Raise | Call | Call | Fold | Fold | Fold | Fold | Fold | Fold |
| T | Raise | Call/Fold | Call | Call | Raise | Call | Call | Fold | Fold | Fold | Fold | Fold | Fold |
| 9 | Raise/Fold | Call/Fold | Call | Call/Fold | Call | Raise | Call | Fold | Fold | Fold | Fold | Fold | Fold |
| 8 | Raise/Fold | Call/Fold | Fold | Fold | Fold | Fold | Raise | Call | Fold | Fold | Fold | Fold | Fold |
| 7 | Fold | Fold | Fold | Fold | Fold | Fold | Fold | Raise | Call | Fold | Fold | Fold | Fold |
| 6 | Fold | Fold | Fold | Fold | Fold | Fold | Fold | Fold | Raise | Call/Fold | Fold | Fold | Fold |
| 5 | Call/Fold | Fold | Fold | Fold | Fold | Fold | Fold | Fold | Fold | Raise | Fold | Fold | Fold |
| 4 | Call/Fold | Fold | Fold | Fold | Fold | Fold | Fold | Fold | Fold | Fold | Fold | Fold | Fold |
| 3 | Call/Fold | Fold | Fold | Fold | Fold | Fold | Fold | Fold | Fold | Fold | Fold | Fold | Fold |
| 2 | Call/Fold | Fold | Fold | Fold | Fold | Fold | Fold | Fold | Fold | Fold | Fold | Fold | Fold |

Fig. 4: Ejemplo de Tabla Preflop

| | A | K | Q | J | T | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Poker | Raise | Raise | Raise | Raise | Raise | Raise | Raise | Raise | Raise | Raise | Raise | Raise | Raise |
| Full | Raise | Raise | Raise | Raise | Raise | Raise | Raise | Raise | Raise | Raise | Raise | Raise | Raise |
| Color | Raise | Raise | Call | Call | Call | Call | Call | Call | Call | Fold | Fold | Fold | Fold |
| Proj color fuerte | Fold | Fold | Fold | Fold | Fold | Fold | Fold | Fold | Fold | Fold | Fold | Fold | Fold |
| Proj color debil | Fold | Fold | Fold | Fold | Fold | Fold | Fold | Fold | Fold | Fold | Fold | Fold | Fold |
| Escalera | Raise | Raise | Raise | Raise | Call | Call | Call | Call | Call | Call | Fold | Fold | Fold |
| Proj escalera | Fold | Fold | Fold | Fold | Fold | Fold | Fold | Fold | Fold | Fold | Fold | Fold | Fold |
| Trio fuerte | Raise | Raise | Raise | Raise | Call | Call | Call | Call | Call | Call | Fold | Fold | Fold |
| Trio debil | Fold | Fold | Fold | Fold | Fold | Fold | Fold | Fold | Fold | Fold | Fold | Fold | Fold |
| Pareja fuerte | Call | Call | Fold | Fold | Fold | Fold | Fold | Fold | Fold | Fold | Fold | Fold | Fold |
| Pareja debil | Fold | Fold | Fold | Fold | Fold | Fold | Fold | Fold | Fold | Fold | Fold | Fold | Fold |

Fig. 5: Ejemplo de Tabla River

En cuanto a la minimización del consumo de recursos del OCR, se crea una nueva imagen incluyendo únicamente las regiones de interés binarizadas, también se limita el espacio de caracteres sobre el que se desea trabajar para conseguir mejores resultados, por ejemplo, solo dígitos o valores de las cartas. De esta manera se consigue ahorrar notablemente la cantidad de recursos hardware y el tiempo de computación empleados en estas detecciones. Se muestra un ejemplo de esto en figura 6.


Fig. 6: Fichas Jugadores OCR

Además, para la transformación a espacio HSV, se utiliza una función implementada por un miembro de la comunidad de usuarios de matlab, la cual prioriza el rendimiento a cambio de perder precisión en el cálculo de los nuevos valores. Esta pérdida de precisión no implica ningún cambio notable en el momento de aplicar los filtros de color en el nuevo espacio, en cambio, se consigue reducir el tiempo de cada transformación en aproximadamente un 40 %.

Otro de los elementos que más problemas de rendimiento causaba era la lectura de las tablas de decisión, originalmente contenidas en archivos xls y leidas con funciones del entorno de desarrollo de matlab. Debido a este bajo rendimiento se ha decidido incluir las tablas directamente en el código de la aplicación, es decir, definiendo matrices estáticas. De otra forma no habría sido posible obtener un tiempo de reacción suficiente como para jugar en ninguna mesa.

## 3.4. Integración de Elementos del Sistema

En este apartado se especifican las soluciones generadas para obtener la captura de pantalla del sistema correspondiente a la mesa de juego, enviarla a un engine de matlab, obtener la respuesta y gestionarla para su ejecución. Es decir, gestionar la interacción entre los distintos elementos del sistema y el software generado.

Uno de los primeros problemas que se plantean es el de obtener imágenes de las distintas mesas de juego y trabajar con ellas por separado. Para solventar esto se hizo uso de las funcionalidades de windows. Al inicio de la ejecución del programa se llama a una función que lista todas las ventanas del sistema y almacena en un vector global todos los handlers referidos a ventanas de juego, esto es posible hacerlo gracias a matching de strings en los títulos de las ventanas. De esta forma, a través de la lista de handlers, se guarda la información respecto a la posición global de la ventana de juego y sus tamaños. Esta información es usada posteriormente en cada thread para hacer la captura de pantalla de la zona específica de la ventana, también mediante funciones de windows, y enviarla al engine de

matlab.

Por facilidades en el envío y recepción de datos de las ventanas, se considera más simple enviar cada canal de color por separado. Una vez los datos se recogen en el engine se juntan los 3 canales y se realizan las transformaciones necesarias para recuperar la imagen original. Estas transformaciones son necesarias debido a que matlab considera el punto 0,0 de la imagen el punto superior izquierdo, en cambio los sistemas windows consideran que el punto 0,0 es el inferior izquierdo.

Cada uno de los engines de matlab genera una respuesta de acción y el thread correspondiente la recoge. En ese momento, el thread guarda la acción en una lista FIFO global con los mecanismos de sincronización necesarios. Acto seguido bloquea el envío de nuevas acciones de este thread hasta que la acción anterior se haya realizado. El proceso de simulación de input va consumiendo estas acciones mientras tenga algún elemento en la lista.

Finalmente, el proceso de simulación de input consiste en una serie de funciones que obtienen y modifican la posición actual del cursor del ratón, de forma que puede simular un desplazamiento hasta el destino deseado y generar señales de click e introducción de caracteres por teclado en caso de ser necesario.

El funcionamiento concreto del bot que se ha descrito hasta el momento se muestra en la figura 7.

## 4 RESULTADOS

Para la verificación y cuantificación de la calidad del programa se han realizado distintos tipos de mediciones según la finalidad del objeto a tratar. Desde este punto de vista, se puede clasificar la calidad del programa según tres apartados diferenciados: la captura de información, las decisiones generadas por el sistema experto, y el consumo de recursos del sistema.

### 4.1. Calidad de las detecciones

Desde el punto de vista de la detección de información, nos interesa saber cuán precisos son los datos extraídos a partir de las segmentaciones y los OCR. Para esto se ha tomado un conjunto de 50 mesas aleatorias y se ha comprobado uno a uno si los datos extraídos eran correctos.

Cabe destacar que dentro de este conjunto de datos extraídos los hay totalmente críticos, que no admiten ningún tipo de error, y generarán una respuesta incorrecta ante la mesa, y otros datos que, dentro de ser erróneos y generar peores resultados, no afectaran en exceso a la toma de decisión.

Los errores críticos en este caso son los que pertenecen a fallos en las cartas privadas, comunitarias, y el turno. En caso de existir algún error en el resto de campos, solo variará ligeramente la decisión.

TABLA 1: ACIERTOS CAPTURA INFORMACION

| | % Aciertos | Total Elementos |
|---|---|---|
| Cartas | 98.42 % | 190 |
| Conjunto cartas privadas | 98 % | 50 |
| Conjunto cartas comunitarias | 92 % | 25 |
| Turno | 100 % | 50 |
| Bote mesa | 98 % | 50 |
| Número de jugadores | 100 % | 50 |
| Posición Bot | 98 % | 50 |
| Nicks de jugadores | 90.90 % | 88 |
| Fichas de jugadores | 98.86 % | 88 |
| Jugador activo | 95.45 % | 88 |

A continuación se detalla el significado de cada uno de los campos evaluados en este apartado:

- Cartas: registra aciertos en el valor de cada una de las cartas por separado. Los fallos en este campo son críticos para bot.
- Conjunto de cartas privadas: registra porcentaje de acierto en la lectura de alguna de las cartas privadas de cada mano. Cuenta un error si hay uno o más fallos en la detección individual de estas cartas. Los fallos en este campo son críticos para bot.
- Conjunto de cartas comunitarias: de forma similar al anterior, registra aciertos en la lectura de alguna de las cartas comunitarias de cada mano. Cuenta un error si hay uno o más fallos en la detección individual de estas cartas. Los fallos en este campo son críticos para bot.
- Turno: registra cantidad de aciertos en detectar el turno que se está jugando en la mano, es decir, preflop, flop, turn o river. Los fallos en este campo son críticos para bot.
- Bote mesa: registra la precisión en la lectura de la cantidad de fichas acumuladas en el bote total de la mesa, es decir, las fichas apostadas por los jugadores en esa mano.
- Posición Bot: registra porcentaje de acierto al ubicar al bot en sentido antihorario respecto al dealer, es decir, su distancia en número de asientos, este valor es bastante relevante en la toma de decisiones.
- Nicks de jugadores: almacena la precisión en la captura de nick de los jugadores, este valor solo es importante en caso de que quiera aplicarse algun tipo de clasificación con memoria.
- Fichas de jugadores: se muestra el porcentaje de acierto en la lectura de las fichas privadas, es decir, no apostadas en la mano, de cada jugador.
- Jugador activo: registra la precisión al detectar si un jugador está participando en la mano o ya ha abandonado su mano y solo participa como espectador hasta una nueva partida.

La tabla 1 contiene los resultados de esta validación.

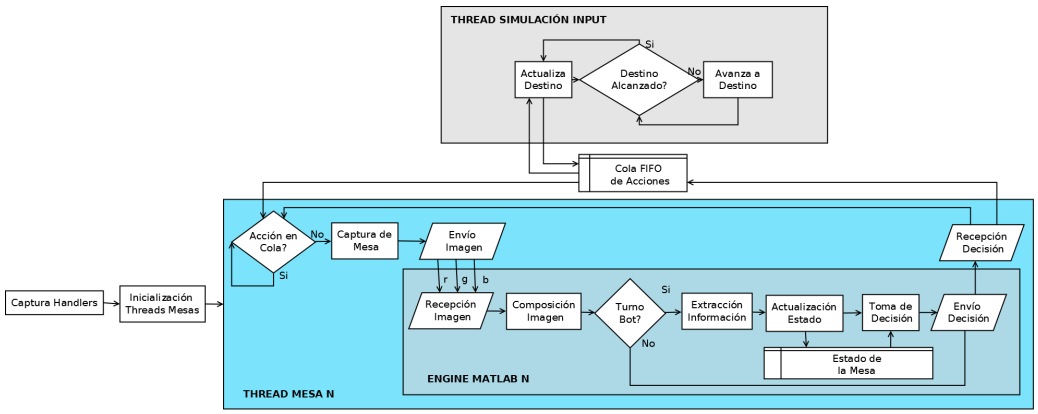



Fig. 7: Diagrama Aplicación

Se puede observar que uno de los peores resultados pertenece a la validación del conjunto de cartas comunitarias. Esto es debido, en parte, a que es el conjunto más pequeño de elementos evaluados, se obtiene a causa de 2 fallos dentro del conjunto de 25 elementos de validación. Teniendo en cuenta los otros dos valores sobre cartas validadas, se considera que este resultado no es realmente descriptivo.

Uno de los problemas que se han encontrado durante la detección y mejora del OCR de las cartas, ha sido la aparente necesidad de una falta de contexto para generar resultados válidos, es decir, contra más grande es el conjunto de caracteres sobre el que se ejecuta el OCR, más precisos son los resultados obtenidos. Es por esto que se tiene una imagen con valores de contexto, fácilmente identificables, y a esta imagen se le añaden los elementos a identificar. De esta forma es muy simple separar el conjunto de contexto del real, y se obtienen resultados mucho más precisos, aunque el coste computacional aumente ligeramente. Añadir este contexto ha supuesto que el acierto en las detecciones en las cartas aumente de un 50 % a un 98.42 %.

Se puede apreciar un ejemplo de esto en la figura 8.


Fig. 8: Ejemplo Contexto OCR

Es por esto que, debido a observarse un peor resultado en la validación final de las cartas comunitarias respecto a las cartas privadas, se cree que este valor no es todo lo descriptivo que se esperaba, y con mayores conjuntos de test se espera que el porcentaje de acierto en el conjunto de cartas comunitarias sea mayor que el de cartas privadas. Esto es debido a que el número de elementos sobre los que se aplica el OCR en las comunitarias es mayor, por lo que, en general, se consiguen unos mejores resultados en el reconocimiento.

Otro error con el que me he encontrado es el fallo ocasional en la lectura de cartas, confundiendo el valor 3 por el 5, único tipo de error que produce esta lectura de cartas. Este error podría llegar a arreglarse añadiendo más valores predefinidos a la imagen de contexto que se utiliza con el OCR. Otra posibilidad es la de realizar una correlación con los elementos concretos, así se podría detectar con cuál de los dos posibles valores se asemeja más.

Teniendo en cuenta los resultados obtenidos en la captura de datos, se puede considerar que el método utilizado genera resultados satisfactorios. Además, la metodología empleada en la solución no supone ningún tipo de límite de hardware para conseguir mejores resultados. Con un profiling más ajustado se podría extraer toda la información sin errores, sin que esto implicara ningún coste computacional extra.

## 4.2. Calidad de las decisiones

Para comprobar la calidad de la toma de decisiones se han realizado observaciones sobre la ejecución del bot en partidas de dinero ficticio, se han dividido los datos recogidos en dos tablas diferenciadas, una primera para poder analizar el comportamiento del bot, y otra para valorar los beneficios conseguidos. Para la toma de datos se han ejecutado 3 sesiones de 10 minutos con 6 mesas simultáneas, el número de fichas de entrada para cada mesa era de 400. Esto influirá tanto en el número máximo de fichas perdidas como el máximo ganado, al limitar el importe ganado/perdido en los all-ins.

Primero se analizará la tabla 2, donde se encuentran los datos de verificación referentes al comportamiento del bot.

TABLA 2: COMPORTAMIENTO JUEGO

| | |
|---|---|
| Ganadas | 4 |
| Perdidas | 1 |
| Abandonadas en preflop | 144 |
| Abandonadas en flop | 7 |
| Abandonadas en turn | 0 |
| Abandonadas en river | 0 |
| Total manos jugadas | 156 |

Se debe tener en cuenta que estas pruebas de beneficios

se han hecho sobre mesas de dinero ficticio, esto implica un comportamiento bastante más aleatorio e impredecible de los oponentes. Debido a esto, se espera que el funcionamiento del sistema experto sea algo mejor en mesas de dinero real.

Por un lado, se puede observar como se ha diseñado un comportamiento tight del bot, es decir, solo juega manos de gran valor inicial. Esto se consigue apreciar debido al alto número de abandonos en el preflop.

Desde el punto de vista del diseño, se considera que este comportamiento es el más adecuado para un bot que no aplique elementos psicológicos en las apuestas. Es decir, con un comportamiento más loose se necesita la agresividad suficiente como para hacer que los oponentes abandonen sus manos. Esto último implica leer que manos tienen los otros jugadores al inicio de la partida, no solo probabilisticamente, también en base a las apuestas que realizan y los comportamientos observados hasta el momento.

Uno de los mayores problemas esperados en la toma de decisión, se encuentra en el caso de tener una figura de valor elevado. Al no realizar un análisis probabilístico condicionado, es incapaz de detectar que el oponente tiene otra figura de mayor valor, en estos casos los resultados de la mano son catastróficos, generalmente implican una pérdida total de las fichas del bot. Pese a esto, no se han encontrado casos durante la validación final, esto puede ser debido al gran filtro que supone la fase de preflop y cuestiones de azar.

Se considera que la toma de decisión en la fase de preflop es la más simple y adecuada, existen múltiples tablas que se pueden combinar para obtener distintos resultados y estilos de juego más o menos arriesgados y agresivos. En cuanto a las fases de flop, turn y river, la falta de tablas predefinidas y la falta de experiencia personal en el juego hacen suponer que podrían mejorarse.

A continuación se analizará la tabla 3, donde podemos ver los datos referentes a los beneficios obtenidos por el bot en las partidas jugadas.

TABLA 3: BENEFICIOS JUEGO

| | |
|---|---|
| Tiempo de juego | 180 min |
| Fichas por ciegas | 565 |
| Fichas apostadas | 1385 |
| Fichas ganadas | 3998 |
| Fichas perdidas | 230 |
| Fichas Iniciales | 7200 |
| Fichas Finales | 9652 |
| Total beneficios | 2452 |

Ante todo, en cuanto a los resultados globales de beneficios, se puede decir que son satisfactorios, ya que se consiguen incrementar las fichas iniciales en un 34,05 % . No obstante, poniendo esta segunda tabla en relación con la anterior, apreciamos que estos beneficios surgen de tan solo 4 partidas, dentro de los 180 minutos y las 156 manos jugadas, esto implica que el bot depende de un tipo muy concreto de manos para ganar, si este tipo de manos, por cuestiones de azar, no aparece, no habrá ningún tipo de ganancia.

Por otro lado, se puede observar que, dejando de lado las fichas obligatorias por ciegas, en caso de apostar en las rondas y abandonarl o perder, esta cantidad es bastante pequeña, tan solo de unas 230 fichas. Se puede considerar que esto es un muy buen resultado.

## 4.3. Rendimiento

Para medir el rendimiento de la aplicación y las necesidades de hardware que requiere se han utilizado las herramientas proporcionadas por el sistema operativo, el debugger de visual studio y el profiler de matlab. Los resultados obtenidos se observan en las figuras 9, 10, 11 y 12.

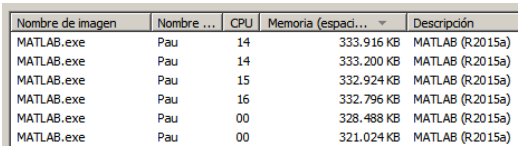



Fig. 9: CPU/Memoria consumida por engines de matlab

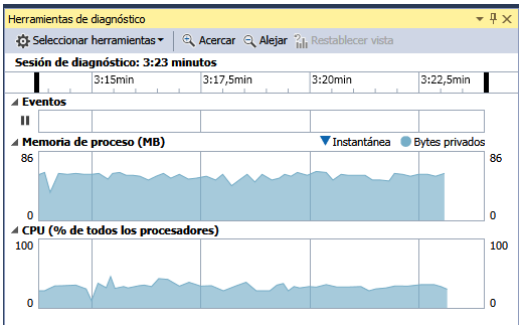



Fig. 10: Output debugger VisualStudio

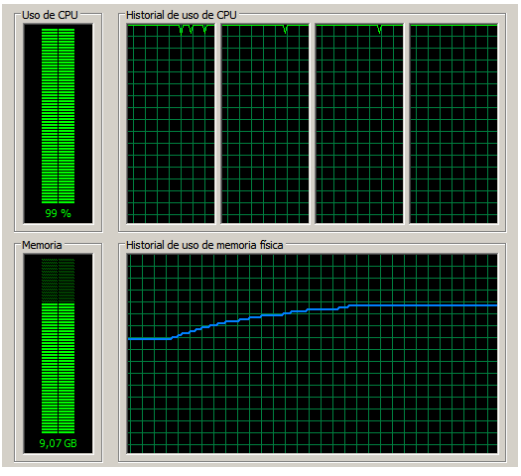
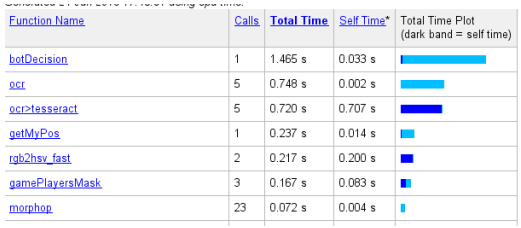



Fig. 11: Grafica de recuros del sistema



| Function Name | Calls | Total Time | Self Time* | Total Time Plot (dark band = self time) |
|---|---|---|---|---|
| botDecision | 1 | 1.465 s | 0.033 s | |
| ocr | 5 | 0.748 s | 0.002 s | |
| ocr>tesseract | 5 | 0.720 s | 0.707 s | |
| getMyPos | 1 | 0.237 s | 0.014 s | |
| rgb2hsv_fast | 2 | 0.217 s | 0.200 s | |
| gamePlayersMask | 3 | 0.167 s | 0.083 s | |
| morphop | 23 | 0.072 s | 0.004 s | |

Fig. 12: Grafica profiler Matlab

En la figura 9 se puede observar la cantidad de memoria que necesita cada uno de los engines de matlab para su entorno. Ciertamente es una cantidad de memoria muy elevada si se quieren tener muchas mesas en paralelo, para este caso se pueden observar las seis mesas con las que se han realizado las pruebas de rendimiento.

Seguidamente, en la figura 10, se muestra el consumo de los threads en C++, sin incluir los consumos de los engines.Según estos valores, esta parte del código no parece suponer un problema importante de consumo de memornia ni procesador.

En la figura 11 se ve como se dispara el uso de recursos del sistema debido a la ejecución del bot, y como crece el consumo de memoria conforme se inicializan los distintos engines de matlab.

Finalmente, en la figura 12 se puede observar que funciones, dentro del código matlab, son las que más tiempo consumen, así como el tiempo total de cada extracción de información / toma de decision del programa.

Gracias a estos gráficos se puede afirmar que, debido a la paralelización, se consigue utilizar al máximo la CPU del ordenador, y que el punto crítico de nuestro sistema está en el consumo de memoria. Partiendo de un estado previo de 6.3GB de memoria ocupada, entre los recursos necesitados por los engines de matlab y los necesitados por la propia aplicación y los cambios de contexto de los threads, la ejecución del bot ocupa 2.7GB de memoria principal.

Se ha conseguido rebajar enormemente la carga de trabajo respecto a las versiones iniciales del programa, y los objetivos buscados en este punto son dos, maximizar el uso eficiente de la CPU, y minimizar el consumo de memoria.

Así, se observa que el consumo de la CPU resulta óptimo, con lo que podemos asegurar de que no se desaprovecha este recurso del sistema por falta de trabajo o debido a esperas en la transferencia de datos. Tan solo quedaría comprobar que cantidad de recursos se están destinando a los cambios de contexto entre los distintos threads.

En cuanto al consumo de memoria de las distintas partes de la aplicación, es un aspecto al que también se le ha prestado atención durante el desarrollo del sistema, no obstante, es un punto que podría llegarse a mejorar con un estudio posterior centrado en el rendimiento.

Se ha podido observar que, si bien la función de matlab no implica un tiempo crítico en la toma de decisión, parece ser que el problema principal reside en la carga del contexto del engine. Esta observación es tan solo una suposición, debido a que por mucho que se modifique la carga de trabajo de la función matlab, sigue habiendo ocasiones en las que el bot, debido a la falta de velocidad en el proceso de toma de decisión, está cerca de ser expulsado de la partida.

## 5 Conclusiones y trabajo futuro

Ha sido muy didáctico comprobar que en el mundo real, por muy interesante que parezca cualquier metodología, es importante seguir el principio KISS. Crear soluciones simples resulta mucho más efectivo la mayoría de ocasiones, sobre todo cuando los recursos de tiempo y de equipo de desarrollo son tan limitados como en este proyecto.

Uno de los puntos que me habría gustado abordar es la clasificación no supervisada de jugadores según su estilo de juego. De esta forma se podría contemplar la faceta psicológica del póker y aportaría mejores resultados en la toma de decisiones. No obstante, la extracción de características de los jugadores a partir de los logs de acciones es una tarea compleja. Esto es debido a que pese a estar acostumbrados a extraer características sobre imágenes, el tipo de datos almacenados en un log no tiene nada que ver con lo anterior, por lo tanto el vocabulario o el tipo de relaciones que forman estos datos se alejan de los típicos usados en los ejemplos de visión por computador.

Otro de los objetivos interesantes a abordar, desde un punto de vista más matemático que informático, es el uso de funciones de probabilidad condicionada [10], combinado con modelos ocultos de markov [11] y la clasificación mencionada anteriormente. Como se apuntó en el apartado del estado del arte, esto podría llegar a requerir de supercomputadores para obtener los resultados deseados.

Para mí ha sido un proyecto realmente interesante, pero que requería mucho más tiempo del que imaginaba en un principio. Debido a la falta de experiencia, hice una planificación inicial extremadamente optimista que no se ajustaba a la realidad. No obstante, aunque me haya sentido sobrepasado en muchos momentos, estoy satisfecho con los resultados obtenidos y creo que este proyecto ofrece ramas de continuación muy diversas, como la optimización del rendimiento, mejorar el sistema experto o añadir factores psicológicos en la toma de decisiones.

## Agradecimientos

Quisiera agradecer a Aura Hernàndez-Sabaté, mi tutora en este trabajo de fin de grado, todo el apoyo, confianza y paciencia demostrados durante estos meses de trabajo, los consejos sin los cuales no podría haber finalizado y las ayudas que han hecho de este un mejor proyecto.

A mis profesores, que me han formado como ingeniero, y me han proporcionado la actitud, métodos y herramientas

necesarias para desarrollar este proyecto.

Y por último, a todos mis familiares, amigos y compañeros, por el soporte, los ánimos, y los momentos aportados durante estos años de carrera que finalizan con este trabajo.

## REFERENCIAS

[1] González Fidalgo, Eduardo. Análisis competitivo de la Empresa. Universidad de Oviedo.

[2] Computer Poker Research Group. University of Alberta. poker.cs.ualberta.ca.

[3] Brains Vs. AI. School of computer Science. Carneige Mellon University. www.cs.cmu.edu/brains-vs-ai.

[4] Torbjrn Lofterud. Developing and running autonomous pokerbots at online casinos. www.youtube.com/watch?v=BxgKMwWKb3I.

[5] Using RGB or HSV. dsp.stackexchange.com/questions/2687/why-do-we-use-the- hsv-colour-space-so-often-in-vision-and-image-processing.

[6] Vanrell, Maria., Valveny, Ernest., López Pena, António. Curso Online de Detección de Objetos. Universidad Autónoma de Barcelona, Coursera.

[7] Estrategias, jugadas y tablas para Texas Hold'em poker. www.texasholdemplus.com/tablas.html.

[8] Higer, Matthew. Internet Texas Hold'em, Winning Strategies from an Internet Pro.

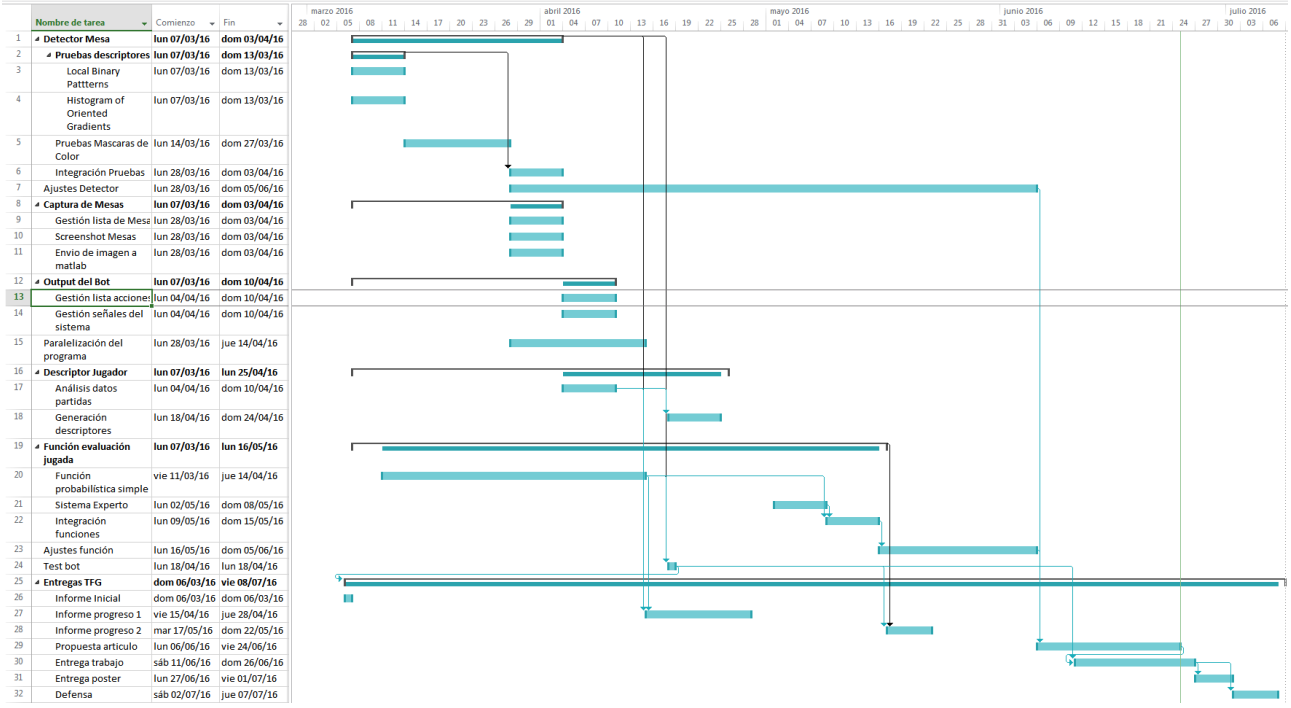
[9] Sklansky, David. The Theroy of Poker. Two Plus Two Publishing.

[10] Murru, Giovanni. Nash Equilibrium and Game Theroy on Poker Texas Holdem. Sapienza, Universita di Roma.

[11] Understanding Hidden Markov Models. valserb.wordpress.com/2011/08/02/understanding-hidden-markov-models.

## APÉNDICE

### A.1. Planificación del desarrollo del proyecto



| | Nombre de tarea | Comienzo | Fin |
|---|---|---|---|
| 1 | **Detector Mesa** | **lun 07/03/16** | **dom 03/04/16** |
| 2 | **Pruebas descriptores** | **lun 07/03/16** | **dom 13/03/16** |
| 3 | Local Binary Pattterns | lun 07/03/16 | dom 13/03/16 |
| 4 | Histogram of Oriented Gradients | lun 07/03/16 | dom 13/03/16 |
| 5 | Pruebas Mascaras de Color | lun 14/03/16 | dom 27/03/16 |
| 6 | Integración Pruebas | lun 28/03/16 | dom 03/04/16 |
| 7 | Ajustes Detector | lun 28/03/16 | dom 05/06/16 |
| 8 | **Captura de Mesas** | **lun 07/03/16** | **dom 03/04/16** |
| 9 | Gestión lista de Mesa | lun 28/03/16 | dom 03/04/16 |
| 10 | Screenshot Mesas | lun 28/03/16 | dom 03/04/16 |
| 11 | Envio de imagen a matlab | lun 28/03/16 | dom 03/04/16 |
| 12 | **Output del Bot** | **lun 07/03/16** | **dom 10/04/16** |
| 13 | Gestión lista acciones | lun 04/04/16 | dom 10/04/16 |
| 14 | Gestión señales del sistema | lun 04/04/16 | dom 10/04/16 |
| 15 | Paralelización del programa | lun 28/03/16 | jue 14/04/16 |
| 16 | **Descriptor Jugador** | **lun 07/03/16** | **lun 25/04/16** |
| 17 | Análisis datos partidas | lun 04/04/16 | dom 10/04/16 |
| 18 | Generación descriptores | lun 18/04/16 | dom 24/04/16 |
| 19 | **Función evaluación jugada** | **lun 07/03/16** | **lun 16/05/16** |
| 20 | Función probabilística simple | vie 11/03/16 | jue 14/04/16 |
| 21 | Sistema Experto | lun 02/05/16 | dom 08/05/16 |
| 22 | Integración funciones | lun 09/05/16 | dom 15/05/16 |
| 23 | Ajustes función | lun 16/05/16 | dom 05/06/16 |
| 24 | Test bot | lun 18/04/16 | lun 18/04/16 |
| 25 | **Entregas TFG** | **dom 06/03/16** | **vie 08/07/16** |
| 26 | Informe Inicial | dom 06/03/16 | dom 06/03/16 |
| 27 | Informe progreso 1 | vie 15/04/16 | jue 28/04/16 |
| 28 | Informe progreso 2 | mar 17/05/16 | dom 22/05/16 |
| 29 | Propuesta articulo | lun 06/06/16 | vie 24/06/16 |
| 30 | Entrega trabajo | sáb 11/06/16 | dom 26/06/16 |
| 31 | Entrega poster | lun 27/06/16 | vie 01/07/16 |
| 32 | Defensa | sáb 02/07/16 | jue 07/07/16 |



Fig. 13: Diagrama Gantt del desarrollo