FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

API para desenvolvimento de agentes de póquer online

Carlos Eduardo Mesquita Frias

MESTRADO INTEGRADO EM ENGENHARIA INFORMÁTICA E COMPUTAÇÃO

Orientador: Prof. Doutor Henrique Lopes Cardoso

Coorientador: Mestre Luís Filipe Teófilo

Junho de 2014

API para desenvolvimento de agentes de póquer online

Carlos Eduardo Mesquita Frias

Mestrado Integrado em Engenharia Informática e Computação

Aprovado em provas públicas pelo júri:

Presidente: Prof. Doutor A. Augusto de Sousa

Arguente: Prof. Doutor Luís Paulo Reis

Vogal: Prof. Doutor Henrique Lopes Cardoso

_____

Julho de 2014

Resumo

A construção de agentes que simulem o comportamento de jogadores humanos em jogos que se baseiem em informação escondida e de natureza não determinista é uma área muito ativa na investigação em Inteligência Artificial. A variante Texas Hold'em do jogo de póquer fornece um contexto para o estudo da eficácia de várias técnicas de implementação de agentes e teorias por causa das propriedades que esta variante possui, bem como a necessidade de lidar com informação escondida e aleatoriedade.

A forma mais efetiva de testar, avaliar e comparar as metodologias utilizadas na construção de agentes de póquer seria poder colocar esse agente a jogar contra humanos em situação de igualdade jogando em ambiente *online*, em tempo real e com as mesmas restrições de tempo e de dinheiro. Desta forma, esta dissertação consistiu no desenvolvimento de uma API que fornece aos agentes as estruturas necessárias para jogar póquer *online*, providenciando como *input* ao agente o estado do jogo e efetuando as jogadas de acordo com o *output* recebido do agente.

O trabalho desenvolvido foi dividido em quatro fases: reconhecimento das cartas e quantias presentes no jogo, deteção do estado de jogo, integração com o simulador de póquer do LIACC e execução dos movimentos no *software* de póquer simulando a movimentação humana do rato e o uso do teclado. Foram utilizadas técnicas de visão por computador, como por exemplo, *edge detection* e *Hough lines*, para identificação das cartas e *template matching* para identificar o valor e naipe das mesmas. Para a deteção das quantias de jogo utilizou-se *optical character recognition* e para simular a movimentação do rato foram utilizadas curvas de Bézier.

Com o estudo do estado da arte realizado, obteve-se uma base de conhecimento que permitiu efetivar a construção do sistema desejado, com bons resultados de desempenho, pois os resultados mostram que a deteção dos elementos na sala de jogo, se situam, na casa dos 98% de identificação correta, quer nas cartas de jogo, quer nas quantias de jogo envolvidas. Quanto à fluidez de execução da aplicação e a correta identificação do estado de jogo, apesar de não terem sido alvo de testes de validação exaustivos, aparenta ter resultados razoáveis. Ainda de referir que o sistema desenvolvido não foi detetado, em nenhum momento, pelo sistema de segurança do *software* de póquer como sendo um jogador automático.

Abstract

The creation of agents that mimic the human behaviour in games of hidden information and with a non-deterministic nature is a very active area in Artificial Intelligence research. The Texas Hold'em Poker game is a good domain for the study of the effectiveness of various agent implementation techniques and theories because of this variant's properties, as well as the need to deal with hidden information and randomness.

The most effective way to test, evaluate and compare methodologies in the development of poker agents would be to be able to put up the agents playing against human players on an equal footing in an online environment, in real time and with the same constraints of time and money. Thus, this thesis purpose was to develop an API that provides the necessary structures to make poker agents play online, providing as an input to the agent the state of the game and making the moves according to the output received from the agent.

The work was divided into four phases: recognition of playing cards and amounts present in the poker room, detection of the state of play and integration with LIACC's poker simulator library and execution of movements in an online poker software simulating human mouse movement and the use of the keyboard. Computer vision techniques were used for the detection of cards, such as edge detection and Hough lines to identify the rectangles containing the playing cards, and template matching to identify the value and suit of the cards. For the detection of money amounts in the game we used optical character recognition and Bezier curves were used to simulate mouse movement.

With the study of the state of the art accomplished, we obtained a knowledge base that allowed us to carry out the construction of the desired system, with a good performance and results, showing that the detection of the elements in the game room are located at around 98% of correct identification, both in playing cards and money amounts involved. Regarding the fluidity of application execution and the correct identification of the game status, despite not having been subjected to rigorous validation tests, it appears to have reasonable results. Also be noted that the system was not detected by the security of the poker software system as an automatic player.

Agradecimentos

O meu primeiro agradecimento dirige-se à minha namorada pelo apoio incondicional em todos os momentos da minha dissertação, e pelas suas palavras de incentivo nos períodos de maior tensão, pois sem ela não teria concluído o curso. Deixo também uma palava de apreço aos meus familiares que sempre me encorajaram.

Agradeço igualmente ao meu orientador e coorientador, Prof. Doutor Henrique Lopes Cardoso e Mestre Luís Filipe Teófilo respetivamente, por todo o apoio e confiança depositada em mim.

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

IA	Inteligência Artificial
ASCII	American Standard Code for Information Interchange
API	Application Programming Interface
CAPTCHA	Testes de Turing complemente automatizados para diferenciação entre computadores e humanos
LIACC	Laboratório de Inteligência Artificial e Ciência da Computação da Universidade do Porto
MCTS	Monte-Carlo Tree Search
OCR	Optical Character Recognition
TCP/IP	Transmission Control Protocol-Internet Protocol

Capítulo 1

Introdução

O póquer é um jogo interessante pelo desafio que apresenta com seus elementos de sorte e informação oculta. O póquer é um jogo muito atraente, pela mistura entre sorte e habilidade, necessárias para vencer, e pelo desafio das apostas a dinheiro. É um jogo em que as recompensas monetárias são um fator aliciante, o que é evidenciado pelo aumento do número de jogadores nos últimos anos.

Atualmente, com o crescimento das tecnologias, nomeadamente a televisão e a internet, o jogo de póquer tornou-se cada vez mais popular e acessível a um público cada vez maior, não necessitando este de se deslocar a casinos. O gosto pelo jogo aleado à ambição dos jogadores promoveu o crescimento de salas de póquer *online*, tornando-se um negócio extremamente lucrativo.

A área de inteligência artificial, nomeadamente a teoria dos jogos, deu particular ênfase ao jogo de póquer, pois trata-se de um jogo com informação imperfeita. Ao contrário do jogo de xadrez, por exemplo, em que se tem acesso a toda a informação do jogo a cada momento, no póquer os jogadores possuem cartas desconhecidas para os outros oponentes. Estas características únicas tornam-no diferente de jogos com informação perfeita, e um problema complexo de solucionar computacionalmente.

A validação e teste de agentes jogadores de póquer em ambiente real é algo difícil de concretizar eficientemente. Nesse sentido, surge a ideia de desenvolver uma *API* que possibilite a execução das jogadas do agente, num *software* de póquer em ambiente online, contra utilizadores reais, de uma forma automatizada. Assim, o intuito principal desta dissertação focou-se no desenvolvimento dessa *API*[1].

[1] Durante o desenvolvimento da *API* nunca em momento algum se envolveu dinheiro real na sua aplicação ao *software* de póquer *online*, apenas dinheiro fictício, nem a *API* criada tem intuito de poder ser utilizada em jogos a dinheiro real, nem com propósito de obter vantagens em relação a adversários humanos no mesmo jogo.

1.1 Contexto

O desenvolvimento de agentes de inteligência artificial dedicados ao jogo de póquer tornou-se um tópico desafiante, muito estudado atualmente e muitas abordagens distintas têm sido desenvolvidas. No entanto, os agentes necessitam de ser testados exaustivamente, para efetivamente poder classificar de uma forma consistente a abordagem utilizada.

Idealmente, uma forma de conseguir essa consistência seria ter os agentes desenvolvidos jogando em tempo real contra utilizadores humanos, em ambiente *online*. Neste seguimento, surge a necessidade de criar uma estrutura que forneça ao agente a capacidade de jogar *online*, providenciando-lhe o estado do jogo a cada momento e efetuando as suas jogadas movimentando o rato no *software* de póquer *online*, para minimizar o risco de deteção pelo *software* de póquer da utilização de *bots*.

1.2 Motivação e Objetivos

O desafio de poder testar agentes de póquer em ambientes reais jogando contra utilizadores reais, com as mesmas restrições de tempo, em jogos de póquer online, motivou o desenvolvimento de uma API para auxiliar o agente de póquer. A API funcionará como sensores de entrada e atuadores do agente, pois irá a cada momento fornecer ao agente informação acerca do estado de jogo, e proceder às suas jogadas. A Figura 1.2.1 apresenta um diagrama ilustrativo do que se pretende obter.

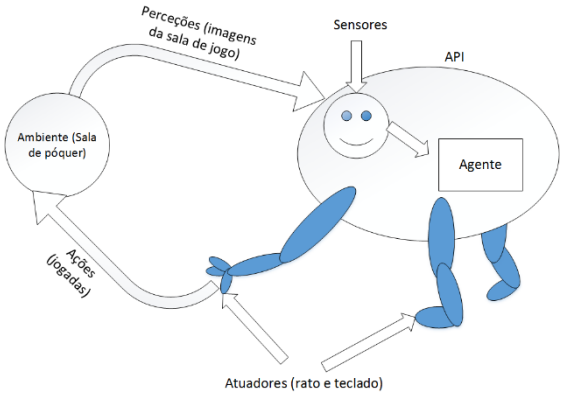


Figura 1.2.1: Ilustração simplificada do sistema a desenvolver

Pretende-se então, nesta dissertação construir uma ferramenta que auxilie no desenvolvimento e teste, em condições reais e contra jogadores humanos, de agentes de póquer. Neste sentido, os objetivos principais para esta dissertação prendem-se com o desenvolvimento de uma API que possibilite o agente a jogar póquer *online* e o desenvolvimento de um agente simples, capaz de jogar póquer online através da API desenvolvida, como prova do conceito e da operabilidade da mesma. Existe ainda um objetivo indireto da dissertação que consiste na

deteção de falhas nos *softwares* de póquer *online*. A construção de sistemas que permitem utilizar *bots* nos casinos, permite também detetar falhas e assim futuramente tomar medidas para banir os jogadores automáticos.

1.3 Estrutura da Dissertação

O restante conteúdo desta dissertação está estruturada do seguinte modo: no Capítulo 2 são explicadas as bases dos jogos de póquer, com ênfase na variante de Texas Hold'em. O Capítulo 3 é dedicado à revisão bibliográfica na qual se abordam alguns temas úteis na implementação da *API*, nomeadamente

- A definição de *bots* de póquer e a similaridade com o que se pertente realizar;
- Algumas técnicas de visão por computador para a deteção das cartas e das quantias presentes no jogo;
- O simulador de póquer desenvolvido no LIACC;
- Algumas abordagens existentes na construção de agentes de inteligência artificial jogadores de póquer.

No Capítulo 4 será efetuada uma descrição pormenorizada do problema a abordar, objetivos a atingir e a sua implementação e metodologias aplicadas na construção da API e detalhe de todas as funcionalidades, ficando o Capítulo 5 reservado para apresentação e estudo dos resultados obtidos. Finalmente, o Capítulo 6 contém as principais conclusões e possível trabalho futuro.

Capítulo 2

Póquer

O póquer é um jogo de cartas jogado por duas ou mais pessoas, muito comum em casinos. É considerado o jogo de cartas mais popular do mundo, em que os jogadores apostam que as cartas que possuem na sua mão são mais valiosas que as cartas nas mãos dos seus oponentes. Todas as apostas são colocadas num pote que, no fim do jogo, é atribuído ao jogador que não desistiu e possui a melhor mão.

O póquer tem muitas variantes, todas elas seguindo um padrão de jogo comum. Dependendo da variante, a mão pode ser formada por cartas que estão escondidas dos outros jogadores ou por uma combinação de cartas ocultas e cartas conhecidas por todos os jogadores.

2.1 Ações do jogo

Saber quando apostar e quando desistir é o aspeto chave num jogo de póquer, para minimizar perdas e maximizar os ganhos. De cada vez que uma aposta é realizada, o dinheiro dessa aposta fica no pote para ser atribuído ao jogador vencedor. As jogadas são realizadas de forma sequencial e, na sua vez, um jogador pode efetuar uma das seguintes ações:

- ***Call*:** um *Call* não é mais do que igualar a maior aposta realizada por um adversário na ronda atual.
    - ***Check*** é um tipo de *Call*, que pode ser visto como uma aposta de valor nulo, e por essa razão pode apenas ser realizada quando nenhum dos adversários realizou uma aposta na ronda atual.
- ***Raise*:** é realizado quando um jogador numa determinada ronda do jogo efetua uma aposta superior à aposta atual de maior valor. O valor da aposta não deve ultrapassar o limite máximo definido na variante do jogo de póquer.

- ***Bet*:** é um tipo de *Raise*. É uma quantidade de dinheiro, em fichas de jogo que um determinado jogador coloca na mesa de jogo numa ronda. Uma vez realizada uma aposta, os restantes jogadores, terão de pelo menos igualar essa aposta para permanecer em jogo.
- ***All-In*:** É um tipo particular de raise, em que o jogador, não tendo dinheiro suficiente para cobrir a maior aposta atual, coloca todo o seu dinheiro em fichas de jogo na mesa. Assim, o jogador aposta todo dinheiro que possui e poderá ganhar todo o que se apostou até ao momento, caso tenha a melhor mão no final do jogo. O resto do valor do pote será atribuído ao jogador com segunda mão mais valiosa. Depois de um jogador realizar um *All-in*, não poderá realizar mais apostas até final do jogo.

- ***Fold*:** Desistir significa deixar a ronda e não colocar mais apostas até à próxima ronda. Um jogador que desiste deixará de poder ganhar o valor existente no pote e perderá, evidentemente, a quantia que apostou até ao momento nessa ronda.

A variante de póquer mais popular nos dias de hoje é o Texas Hold'em, e será a variante de póquer considerada neste trabalho. O seu formato sem limite de apostas é utilizado em vários grandes eventos da World Series of Poker, que é o mais famoso campeonato de póquer do mundo, realizado anualmente em Las Vegas. Apesar de teoricamente esta variante puder ser jogada com até vinte e dois jogadores, é geralmente jogada por entre duas a dez pessoas. É uma das variantes do póquer que mais preza a posição do jogador na mesa.

2.1.1 Regras

No Texas Hold'em os jogadores recebem duas cartas cada, designadas de *hole cards* e um máximo de cinco cartas comunitárias que são colocadas na mesa. Qualquer combinação de cinco cartas de entre as duas *hole cards* e as cinco cartas comunitárias é designada de mão. Para além disso, a posição do jogador na mesa é importante para o decorrer do jogo. Existe sempre um *dealer* e, dependendo do número de jogadores, poderá existir também um *big blind* e um *small blind*. Estas posições são geralmente marcadas com fichas que vão alterando de posição de forma rotativa, no sentido contrário aos ponteiros do relógio, após cada ronda de jogo. Na Figura 2.1.1 é ilustrada uma mesa de jogo, com a posição do dealer, do *small* e *big blind*.

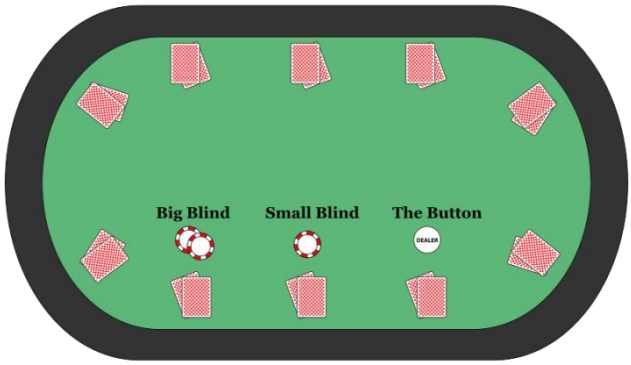


Figura 2.1.1: Posições de jogo no Texas Hold'em

Antes das cartas serem distribuídas, tanto o *small* como o *big blind* apostam uma quantia predeterminada de dinheiro no pote. A aposta do *big blind* é a aposta mínima para a mesa de jogo, enquanto que, a aposta do *small blind* é metade desse valor. Isto garante que existe no pote uma quantia inicial não nula, por isso, existirá sempre algum incentivo para ir a jogo.

O jogo é então composto por quatro rondas, e cada uma delas termina quando todos colocarem a mesma quantidade de dinheiro no pote, ou quando todos os jogadores desistirem, exceto um deles, ou quando todos os jogadores estão em *All-in* ou ainda quando todos os jogadores estão em *All-in* exceto um.

As quatro rondas são designadas por:

- ***Pre-Flop*:** em que ainda não existem cartas comunitárias;
- ***Flop*:** quando são extraídas três cartas comunitárias;
- ***Turn*:** quando a quarta comunitária é extraída;
- ***River*:** quando a quinta e última carta comunitária é extraída.

Após a ronda *River*, se todos os jogadores estiverem de acordo para efetuar *Call* ou *Check* ao pote, segue-se o *Showdown*. Nessa fase, os jogadores começam por mostrar as suas mãos sequencialmente, e todos os jogadores, excetuando o primeiro, têm a possibilidade de não mostrar a sua mão e, portanto, perder a quantia no pote. O jogador com a melhor mão vence o pote. Se dois ou mais jogadores possuem mãos do mesmo nível, existe um empate e a quantia no pote é dividida por esses jogadores.

Em cada ronda, se todos os jogadores desistirem expeto um, esse jogador será o vencedor e receberá a quantia existente no pote, ou parte dela, caso exista uma comissão para o casino.

2.1.2 Hierarquia das mãos

Existem algumas regras que se aplicam a cada mão, independentemente da sua hierarquia:

As cartas são ordenadas individualmente por ordem decrescente de valor: Ás, Rei, Dama, Valete, Dez, Nove, Oito, Sete, Seis, Cinco, Quatro, Três, Dois e novamente, o Ás;

Os naipes não têm valor. O naipe das cartas é principalmente utilizado quando se pretende determinar se uma dada mão encaixa numa certa combinação, nomeadamente, mãos do tipo *flush* e *straight flush*.

Uma mão consiste sempre num conjunto de cinco cartas. Em jogos em que existem mais do que cinco cartas disponíveis para cada jogador, como é o caso do Texas Hold'em, a mão de cada jogador será a melhor combinação de cinco cartas de entre essas cartas.

As mãos são hierarquizadas inicialmente, por combinação e posteriormente por ordenação individual do valor das cartas: mesmo a mão mais baixa de um determinado tipo, vence todas as mãos de combinações inferiores. Por exemplo, uma mão com um par de duques e um par de ternos vence todas as mãos com apenas um par ou nenhum par. Apenas entre duas mãos da mesma categoria é que o valor das cartas é utilizado para desempatar.

Em seguida serão apresentadas as combinações de cartas que representam a hierarquia das mãos nos jogos de póquer *standard* de cinco cartas, por ordem descendente de valor:

O ***Royal Flush*** é a melhor mão possível no póquer e é constituída pela sequência Ás, Rei, Dama, Valete e Dez de um mesmo naipe.

Figura 2.1.2: Exemplo de um Royal Flush

Um ***Straight Flush*** é uma mão contendo cinco cartas em sequência de valor, todas do mesmo naipe. Quando dois jogadores chegam ao *showdown* com esta mão, as mãos são comparadas e aquela que tiver a carta com maior valor individual vence. Os Ases podem funcionar como a carta de menor valor ou como a carta de maior valor num *straight flush*, por exemplo na sequência A-2-3-4-5, o ás funciona como a carta de menor valor, no entanto, na sequência 10-V-D-R-A, o ás funciona como a carta de maior valor.

Figura 2.1.3: Exemplo de um Straight Flush

Four of a Kind ou Póquer é uma mão onde o jogador possuir quatro cartas do mesmo valor e uma carta de outro valor. Se dois jogadores possuírem um four of a kind, aquele que tiver a quinta carta, designada de kicker, com o valor maior, vence.

Figura 2.1.4: Exemplo de um Poker

Um ***Full House*** é uma mão que contém três cartas de um mesmo valor e as restantes duas têm também um valor igual, embora diferente do valor das primeiras três. Entre duas mãos de *Full House*, vence aquela em que o valor das três cartas iguais é maior.

Figura 2.1.5: Exemplo de um Full House

Um ***Straight*** ou sequência é uma mão contendo cinco cartas de valor sequencial, mas de naipes distintos. Entre duas mãos deste tipo, vence aquela em que contiver a carta de maior valor. Se a carta de maior valor das duas mãos for igual, então ocorre um empate.

Póquer

Figura 2.1.6: Exemplo de um Straight

Um ***Three of a Kind*** ou um trio é uma mão contendo três cartas com o mesmo valor e duas cartas de valores distintos. Um trio vence outro trio se o valor das três cartas iguais for superior. Em caso desse valor ser o mesmo, vence aquela que contém a carta de maior valor de entre as cartas diferentes.

Figura 2.1.7: Um exemplo de um trio

Dois pares é uma mão que contém dois pares de cartas de igual valor e uma carta com valor diferente. De entre duas mãos deste tipo, vence aquela que contenha o par de cartas iguais de maior valor. No caso do par de maior valor ser igual nas duas mãos, então é comparado o segundo par. Se ambos os pares de cartas forem iguais entre as duas mãos, então a quinta carta de cada mão decide o vencedor.

Figura 2.1.8: Exemplo de uma mão com dois pares

Um par é uma mão que contém um par de cartas com o mesmo valor, sendo as restantes três cartas de valores distintos. Entre duas mãos deste tipo, aquela cujo valor das cartas do par é maior, vence. Caso esse valor seja o mesmo, então as restantes três cartas são comparadas em

ordem descendente de valor até que seja possível determinar um vencedor. Caso os pares e as três restantes cartas tenham o mesmo valor entre as duas mãos, então ocorre um empate.

Figura 2.1.9: Um par de quatros

High Card é uma combinação de cinco cartas em que não existem cartas com o mesmo valor, as cinco cartas não estão em sequência nem são todas do mesmo naipe. Duas mãos deste tipo são ordenadas comparando a sua carta de maior valor. Se essas cartas são iguais, então são comparadas as cartas com o segundo maior valor, e assim por diante, até se determinar um vencedor. Se entre as duas mãos as cartas têm os mesmos valores, o jogo empata.

2.2 Considerações finais

Neste capítulo foram abordadas as regras do jogo de póquer na sua variante de Texas Hold'em no qual se referiram as ações possíveis e as várias fases do jogo, bem como a hierarquia das mãos, onde foram apresentados exemplos de cada uma. No Capítulo 3 vai ser dada a conhecer alguma da revisão da literatura efetuada ao longo da dissertação que serviu de base ao desenvolvimento deste projeto.

Capítulo 3

Revisão Bibliográfica

Neste capítulo é discutido o estado da arte e trabalhos realizados previamente que serviram de base ao desenvolvimento desta dissertação. Nomeadamente, será discutida a noção de póquer *bot*, as suas similaridades com o projeto a desenvolver, as diversas aplicações deste tipo existentes no mercado e a sua eventual deteção pelos *sites* de póquer *online*. Serão referidos alguns conceitos de visão por computador, nomeadamente, segmentação de imagens, e deteção de características numa imagem por *template matching*, trabalhos prévios realizados no âmbito do reconhecimento de cartas de jogo e a noção e aplicações de reconhecimento ótico de caracteres. Finalmente, será dada uma referência ao simulador de póquer do LIACC (no qual foi integrada a solução desenvolvida) e analisados os diferentes tipos de abordagens para a construção de agentes jogadores de póquer.

3.1 Póquer bots

Programas de computador criados especificamente para jogarem póquer contra humanos ou contra outros programas de computador, são designados de póquer *bots* ou simplesmente *bots*.

Os *bots* são 99% das vezes completamente baseados em *software*. Isto significa que não existe um *robot* físico a controlar o computador. Essa ideia, embora criativa, não é uma descrição adequada do que é um póquer *bot* [1].

Estes *bots* são frequentemente usados em aplicações de póquer *online*, simulando oponentes humanos legítimos ou como uma forma de fazer batota. Os *sites* de póquer não aceitam o uso de póquer *bots*. Os mais populares *sites* possuem regras nos seus termos e condições proibindo o uso de *bots*. A maioria dos *sites* tenta ativamente assegurar a não utilização deste tipo de *software*. Algumas técnicas utilizadas pelas aplicações de póquer para a deteção de *bots* são: a pesquisa do nome ou propriedades dos processos a correr na máquina do cliente para detetar possíveis *bots* comummente conhecidos, registo da posição dos cliques e

velocidade de movimentação do rato e tempo de execução da jogada. Será muito difícil um utilizador humano clicar nos botões sempre na mesma precisa localização, ou efetuar a movimentação do rato em velocidade muito grande, não coerente com a movimentação humana.

Existe uma variedade de testes para distinguir humanos de computadores num ambiente *online*, conhecidos comummente por testes *CAPTCHA* (*Completely Automated Public Turing Test to Tell Computers and Humans Apart*). Um teste *CAPTCHA* é um programa que pode gerar testes que a maioria dos humanos consegue passar mas que programas de computador não conseguem. Os testes *CAPTCHA* são essencialmente heurísticos; não existe nenhuma forma de provar que um programa de computador consiga passar um teste que um humano também consiga. Os testes *CAPTCHA* exploram as disparidades entre as capacidades do ser humano e as dos computadores descritas ao longo dos últimos anos, mas não há forma de garantir que essas disparidades não venham eventualmente a desaparecer. O teste *CAPTCHA* mais utilizado atualmente recorre à habilidade dos humanos em reconhecer textos em imagens distorcidos aleatoriamente [2].

Apesar da utilização de *bots* ser proibida, nas salas de póquer *online*, eles representam a melhor forma teórica de testar realmente um agente. Jogar contra utilizadores humanos em jogos a dinheiro é a melhor forma de testar as habilidades do agente, uma vez que os utilizadores jogam de uma forma mais consciente e empenhada nesse caso.

Existem várias aplicações de póquer *bots* disponíveis, contudo, a maior parte delas são aplicações comerciais. Nas subsecções seguintes são apresentados alguns exemplos.

3.1.1 Shanky Holdem Poker Bot

O Shanky Holdem Poker Bot [3] é um *bot* comercial que permite jogar a versão "*no limit*" do Texas Hold'em em alguns casinos *online*.

Este tipo de bot foi um dos primeiros a permitir jogar a versão "no limit" do Texas Hold'em. Possui algumas características para evitar a sua deteção como, por exemplo: um modo hide que não permite ao casino detetar o processo em execução, clique em posições aleatórias de botões, algumas mensagens pré-gravadas para introduzir no *chat*, pausas para simular a ida à casa de banho, etc. Estas características são completamente customizáveis. É também possível definir a estratégia de jogo do bot.

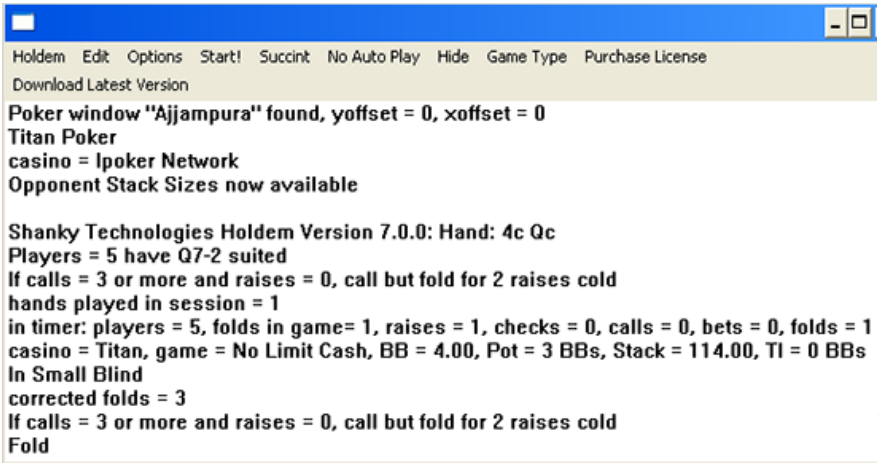


Figura 3.1.1: Interface do Shanky Holdem Póquer bot

3.1.2 Win Holdem

O WinHoldem [4] é um Póquer *bot* comercial criado por Ray Bornet. Foi uma das primeiras frameworks de Póquer *bots* programáveis. Os utilizadores podem desenvolver e compilar em linguagem C/C++ a sua lógica de inteligência artificial para incorporar no *bot*.

O WinHoldem é único Póquer *bot* disponível publicamente que admite suportar *colusion*, que é um tipo de batota em que os jogadores podem formar alianças e obter daí vantagens injustas para os restantes adversários.

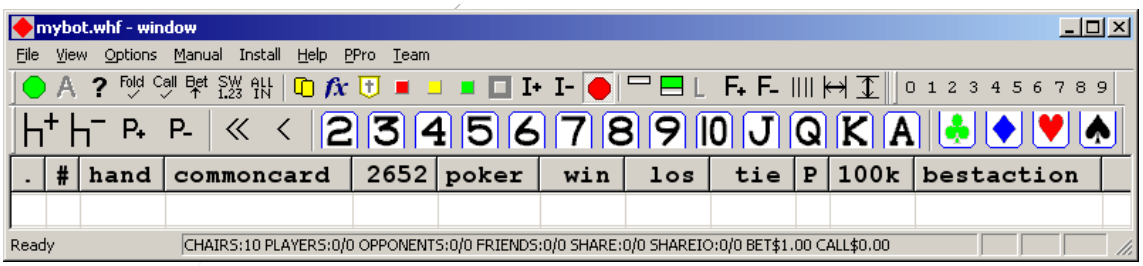


Figura 3.1.2: Interface do Win Holdem

3.1.3 Open Holdem

OpenHoldem [5] é uma *framework* de *software* aberto de análise de ecrã e motor de lógica programável para jogar *online* a variante Texas Hold'em de póquer. Esta *framework* é similar ao WinHoldem mas não inclui as capacidades de *collusion* que o WinHoldem possui. No entanto, esta *framework* inclui: um motor para interpretar ecrãs e extrair o estado de jogo; um motor lógico para efetuar as decisões baseadas no estado de jogo; uma linguagem de *scripting* simplista para descrever como essas decisões devem ser realizadas; Vários mecanismos que permitem a criação de lógicas de decisão, por meios alternativos (C++, Perl, etc).

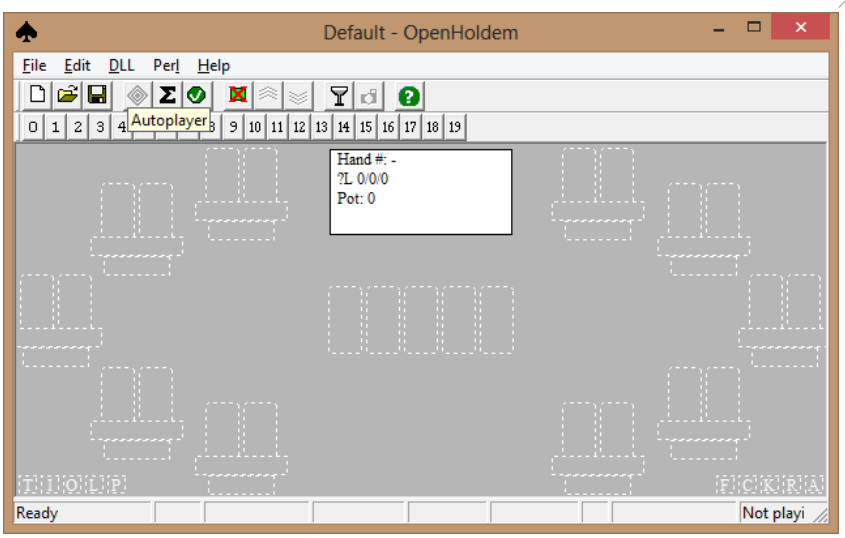


Figura 3.1.3: Interface do Open Holdem

3.2 Bots simulando comportamento humano

Nos últimos anos, os jogos *online* tornaram-se uma das atividades mais populares na *Internet*, mas a batota, como por exemplo o uso de *bots*, cresceu como uma consequência direta. Em geral, a comunidade discorda com a utilização destes *bots*, uma vez que estes permitem obter vantagens sem os esforços correspondentes. Contudo, os *bots* são difíceis de detetar, pois são desenhados para simular o comportamento de jogo humano e seguem integralmente as regras de jogo.

Embora, os *bots* possam sem grande esforço simular decisões humanas, certos comportamentos padrão dos seres humanos são difíceis de imitar devido a serem complexos problemas de Inteligência Artificial. Por exemplo, a trajetória de um avatar controlado por um jogador humano. Os jogadores controlam a movimentação do avatar baseado no seu conhecimento, experiência, intuição e na grande quantidade de informação providenciada pelo jogo. Uma vez que as decisões humanas podem não ser sempre lógicas e eficientes, como

modelar e simular movimentos realistas é ainda uma questão em aberto no campo da Inteligência Artificial. Para distinguir jogadores humanos de *bots* eficazmente, pode-se analisar as trajetórias de ambos os tipos de jogador e distingui-los de acordo com as características espaciais e temporais das suas trajetórias [6].

3.2.1 Curvas de Bézier

A simulação da movimentação humana do rato no *software* de póquer poderá ser realizada recorrendo às curvas de Bézier que são mais popularmente utilizadas em computação gráfica. As curvas de Bézier foram nomeadas em homenagem ao seu inventor, Dr. Pierre Bézier, que no início dos anos sessenta desenvolveu um modelo de construção de curvas para ser aplicado no desenho de formas, que fosse intuitivo o suficiente para ser usado por *designers* e artistas, sem que fosse necessário possuir uma sólida formação em matemática.

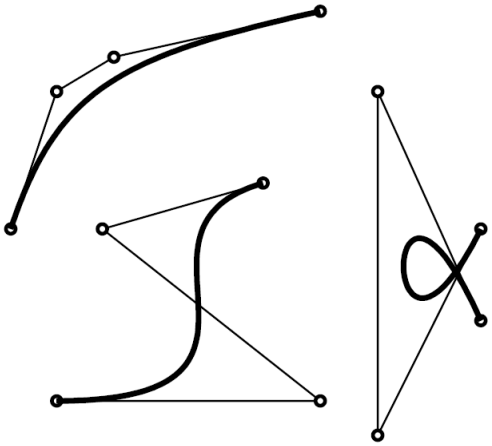


Figura 3.2.1: Exemplo de curvas de Bézier cúbicas

A Figura 3.2.1 mostra três curvas de Bézier distintas, com os seus respetivos polígonos de controlo. Cada polígono de controlo é constituído de quatro pontos de controlo que estão ligados consecutivamente por segmentos de reta. A beleza da representação de Bézier é que a curva imita a forma do seu polígono de controlo. A curva passa pelo seu primeiro e último ponto de controlo e é tangente aos segmentos de reta que os contém. Por este motivo, formas mais complexas podem ser criadas utilizando uma sequência de curvas de Bézier, uma vez que essas curvas são tangentes aos seus polígonos de controlo, é fácil unir duas curvas de Bézier de forma a elas serem contínuas e tangentes nesses pontos, mantendo assim a suavidade da curva. Na Figura 3.2.2 apresenta-se a construção de uma letra "g" utilizando a união de curvas de Bézier.

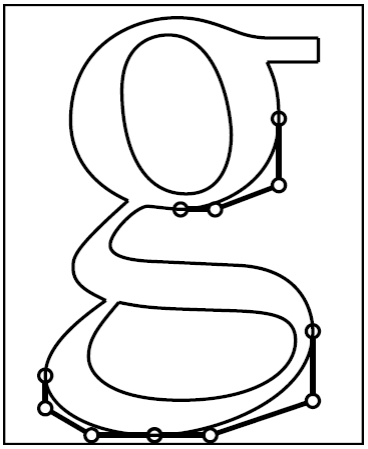


Figura 3.2.2: Os contornos da letra "g" utilizando curvas de Bézier

As equações para as curvas de Bézier são definidas parametricamente. Uma representação paramétrica de uma curva é definida por um par de coordenadas (x, y) de pontos num referencial do plano, mas em que a variável y não está definida diretamente em função da variável x, mas sim os dois valores x e y são determinados separadamente através de uma outra variável, denominada de parâmetro, que muitas vezes é a variável t que representa o tempo. [7]

Podem-se definir curvas de Bézier de qualquer grau. Uma curva de Bézier de grau n tem $n + 1$ pontos de controlo. A equação que define parametricamente uma curva de Bézier de grau n é:

$$P(t) = \sum_{i=0}^{n} {}^{n}C_i(1-t)^{n-i} \times t^i \times P_i, \text{ com } t \in [0, 1]$$

Em que, ${}^{n}C_i$ representa combinações de n, i a i, P_i representa o ponto de controlo de índice i e t representa a variável paramétrica, que varia entre 0 e 1.

A Figura 3.2.3 ilustra curvas de Bézier de diversos graus e os seus pontos de controlo.

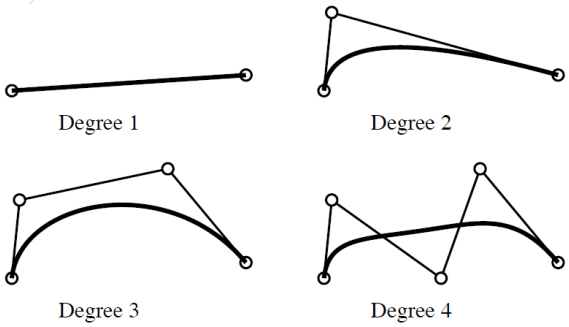


Figura 3.2.3: Curvas de Bézier de vários graus

A construção da trajetória do movimento do rato pode ser criada recorrendo às curvas de Bézier, tentando assim simular a movimentação humana [8] [9].

3.3 Sistemas de póquer *online*

Um sistema de póquer *online* é uma aplicação de software que possibilita a realização de jogos de póquer entre diversos jogadores em regime *multiplayer* conectados pela Internet. Estes sistemas normalmente fornecem várias salas de jogo, diferentes variantes do póquer, possibilidade de jogo a dinheiro real ou com dinheiro fictício.

O sistema de Póquer *online* com maior número de utilizadores é sem dúvida o *PokerStars*, considerado o site número um de póquer no mundo. O premiado *software* do *PokerStars* é gratuito e encontra-se disponível para Windows/PC e Mac/Apple, e está constantemente a ser melhorado por uma equipa de especialistas, que adicionam novas funcionalidades com base nas recomendações dos seus utilizadores, para garantir acesso à melhor experiência de jogo de póquer em ambiente online.

A *PokerStars* oferece uma ampla seleção de jogos de Póquer na *Internet* em centenas de variantes, incluindo *Texas Hold'em*, *Omaha*, *Stud*, 2-7 *Triple Draw*, *Badugi*, para além de jogos mistos como *HORSE* e *8-Game*. Todos os jogos estão disponíveis em *stakes* que se adaptam a todos os níveis de jogadores e tamanho de *bankroll*.

A Figura 3.3.1 apresenta a janela do *PokerStars* na sua variante *Texas Hold'em*.

Figura 3.3.1: Janela do *PokerStars*

Tal como se pode observar na Figura 3.3.1, estão presentes em jogo 9 jogadores, sendo a posição central e inferior a reservada para o utilizador. Normalmente as mesas no *PokerStars* são compostas por seis ou nove jogadores. No canto inferior direito pode-se observar os botões referentes às ações possíveis para o utilizador, estando esses botões visíveis apenas quanto é a sua vez de jogar. No canto inferior direito encontra-se o *chat* da sala de jogo, bem como outros menus informativos e de estatísticas. É possível configurar na sala do *PokerStars* alguns componentes, como cor da mesa do jogo, fundo de ecrã, modelo de cartas, som, entre outros.

3.4 Visão por computador

A visão por computador é uma área da computação que tem por objetivos obter informação sobre um determinado ambiente, a partir de imagens dessa cena, obter descrições explícitas e significativas da cena representada numa imagem e dotar as máquinas com capacidades visuais similares às do ser humano.

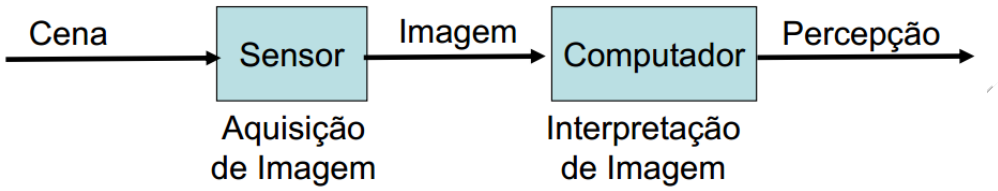


Figura 3.4.1: Etapas na aquisição de informação em Visão por Computador (imagem adaptada de [11])

Em Visão por Computador tenta-se descrever o mundo que se vê numa imagem ou conjunto de imagens e reconstruir as suas propriedades, tais como forma, iluminação e cor. É espantoso como os humanos e animais o conseguem fazer sem qualquer esforço, enquanto que, os algoritmos de visão por computador são tão propensos a erros. Este conceito errado de que a visão deve ser algo fácil de implementar data dos inícios da inteligência artificial, quando se pensava que a cognição era intrinsecamente mais difícil do que a componente de perceção [10].

A Visão por Computador está a ser atualmente utilizada numa grande variedade de aplicações do mundo real, tais como: optical character recognition (OCR) (reconhecimento de texto escrito a computador ou à mão), inspeção de máquinas, retalho, construção de modelos 3D réplicas dos objetos reais, medicina, segurança automóvel, cinema, captura de movimento, vigilância, reconhecimento de impressões digitais e biométricas, entre outras. [11]

Nas próximas secções serão indicadas algumas técnicas que foram úteis para o desenvolvimento do trabalho descrito nesta dissertação.

3.4.1 Segmentação de imagens

A segmentação de imagens é a tarefa de encontrar grupos de pixéis numa imagem que "vão juntos". Em visão por computador, a segmentação de imagens é um dos problemas mais antigos e estudados. As técnicas iniciais tendem a usar *merging* e *splitting* de regiões, que correspondem a algoritmos aglomerativos e divisivos. Os algoritmos mais recentes frequentemente otimizam algum critério global, tal como a consistência das intrarregiões e fronteiras inter-regionais ou dissimilaridade de regiões [11].

Existem essencialmente dois tipos de abordagens para efetuar a segmentação de uma imagem: abordagens baseadas em arestas que usam as fronteiras entre as regiões para segmentar a imagem, detetando alterações abruptas na intensidade e preenchendo falham nas arestas com pontes; abordagens baseadas em regiões que usam a similaridade e proximidade espacial entre os pixéis para encontrar diferentes regiões. Resumidamente, arestas são encontradas baseando-se nas diferenças entre pixéis adjacentes e regiões são encontradas baseando-se na similaridade de pixéis adjacentes. Teoricamente, ambas as abordagens deveriam obter resultados idênticos, mas isso não se verifica em termos práticos.

Figura 3.4.2: (A) Segmentação baseada em arestas (B) Segmentação baseada em regiões (imagens retiradas de **[11]**)

3.4.2 Template matching por correlação cruzada

A correlação é uma ferramenta importante no processamento de imagens, reconhecimento de padrões e em outros campos. A correlação entre dois sinais "*cross correlation*" é a abordagem *standard* para a deteção de características, bem como, base para técnicas de reconhecimento mais sofisticadas. [12]

As técnicas de *template matching* tentam responder a algumas variações da seguinte questão: A imagem contém uma parte de uma dada característica específica? Se sim, onde?

O *template matching* funciona da seguinte forma: são necessários dois componentes primários, uma imagem fonte, onde se pretende encontrar uma característica específica e a imagem *template*, que é o pedaço de imagem ou característica que se pretende encontrar e que será comparada contra a imagem fonte.

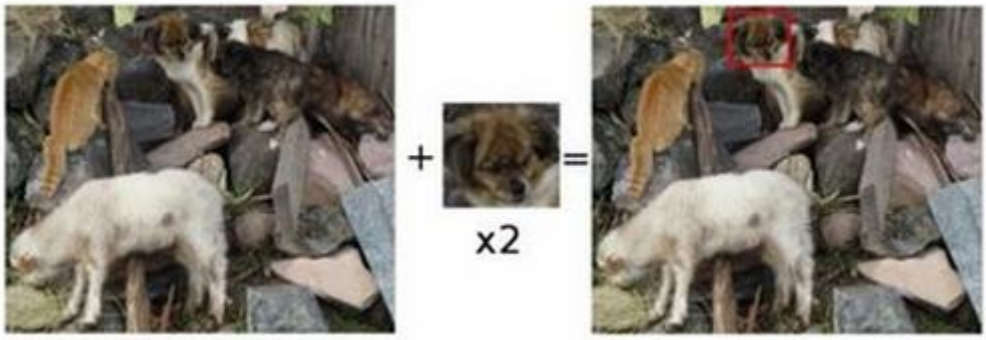


Figura 3.4.3: Template matching. Imagem fonte e template. (imagens retiradas de **[13]**)

O objetivo é detetar a zona na imagem fonte com maior similaridade com o *template*. Para determinar essa zona, temos que comparar a imagem *template* contra a imagem fonte, fazendo deslizar o *template* ao longo da imagem.

Figura 3.4.4: Template matching. Deslizando o template ao longo da imagem. (imagem retirada de [13])

Por deslizar, pretende-se dizer, movimentar o template, um pixél de cada vez, da esquerda para a direita e de cima para baixo. Em cada localização, é calculada uma métrica que representa o quão boa ou quão má a localização é, ou seja, quão similar é o template com essa área particular da região da imagem fonte.

Para cada localização do template na imagem fonte, são guardadas a métricas calculadas, numa matriz de resultados (R). Cada localização (x,y) na matriz R contém a métrica de similaridade:

A Figura 3.4.5 mostra a matriz resultado R, obtida usando a métrica de correlação cruzada normalizada. A localização mais brilhante indica o pixél, cuja área correspondente, é mais similar com o *template*. Como se pode observar, a localização marcada com um círculo vermelho é provavelmente aquela em que o valor da métrica e mais elevado, portanto, essa localização, isto é, o retângulo formado por esse ponto como vértice superior esquerdo e com comprimento e largura iguais às do *template*, é considerado uma correspondência. [13]

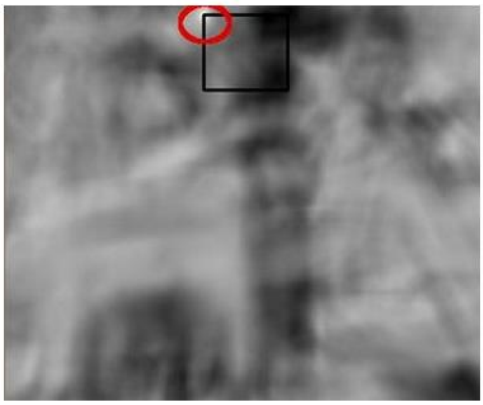


Figura 3.4.5: Template matching. Correspondência encontrada. (imagem retirada de **[13]**)

3.4.3 Reconhecimento de cartas de jogo

O reconhecimento de cartas de jogo envolve a segmentação de caracteres, transformação afim, deteção de arestas e template matching. A rotação e escalamento são considerados quando se pretende obter um reconhecimento de cartas de jogo robusto. [14]

O sistema de reconhecimento de cartas de jogo descrito em [14] engloba os seguintes passos: captura da imagem, determinação de templates, seleção da carta de jogo, escalamento da carta, rotação da carta, segmentação de caracteres e template mathing.

Uma *webcam* é utilizada no sistema de deteção descrito em [15] de cartas como um sensor para captar a imagem das cartas. A resolução da camara é importante, quanto maior resolução, melhor serão os resultados obtidos.

O valor da carta e o naipe têm de ser reconhecidos, por isso, dezoito padrões com treze valores e quatro naipes são criados como *templates*. O valor dez pode ser separado em dois, um o número zero e um com o número um. Cada imagem *template* é o retângulo mais pequeno contendo o valor ou o naipe da carta, que são depois redimensionados para um tamanho predefinido. Os *templates* são separados em duas partes: o *template* positivo e o *template* invertido. O *template* positivo contém o valor ou naipe com zero graus de rotação, enquanto que, o *template* invertido é uma rotação de cento e oitenta graus do *template* positivo.

Figura 3.4.6: Exemplos de templates positivos e invertidos

A seleção da carta envolve a construção da imagem binarizada a partir de uma imagem em escala de cinzentos utilizando *thresholding*. Em seguida, é realizada uma deteção dos contornos da carta utilizando algoritmos de deteção de arestas. Os contornos da carta são os contornos exteriores, enquanto que, os contornos do valor e naipe da carta estarão no interior, e serão extraídos em seguida.

Para distinguir as cartas de jogo de outros objetos, o sistema procura apenas os retângulos que contenham contornos no seu interior.

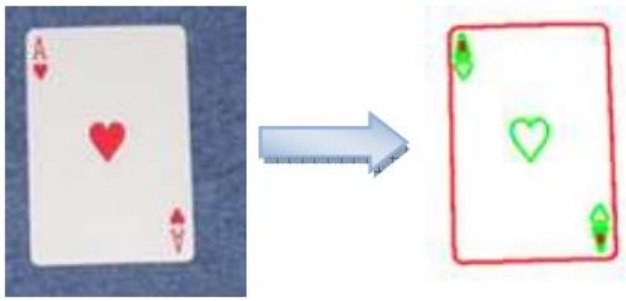


Figura 3.4.7: Deteção dos contornos da carta

Após encontrar a imagem da carta, a região de interesse é o retângulo que contém a carta, a região de interesse é então aumentada, utilizando uma interpolação bicúbica. A região que contém apenas a carta é então obtida após rotação.

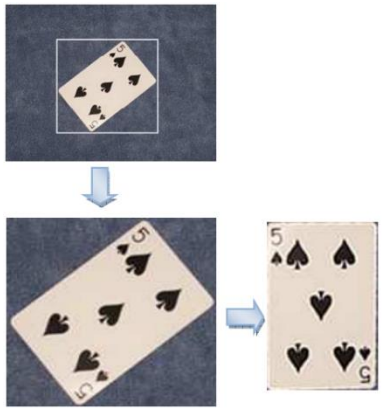


Figura 3.4.8: Rotação e escalamento da carta detetada

Para a determinação do valor da carta a solução proposta foi utilização de *template matching*, que envolve a segmentação do caracter posicionado no canto superior esquerdo da carta, que contém até três padrões. Os três padrões nesta região podem ser, o caracter representativo do valor da carta, parte do padrão do naipe da carta e parte da imagem interior da carta. Cada uma destas partes pode ser obtida por segmentação dessa região com os retângulos mais pequenos contendo contornos no seu interior. Estas imagens obtidas são comparadas com os *templates* criados inicialmente para reconhecer o valor e naipe da carta.

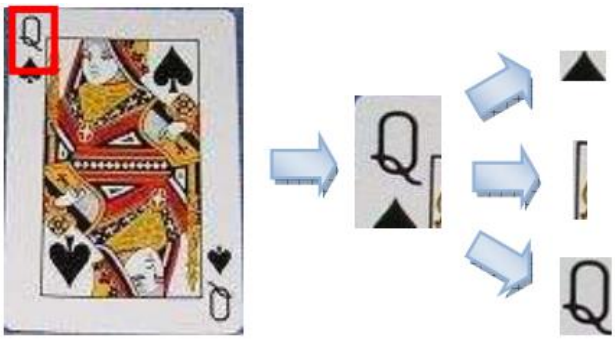


Figura 3.4.9: Extração do valor e naipe da carta

Uma alternativa para a determinação do valor das cartas do tipo numérico foi proposta em [15] e consiste em contar o número de marcas (naipes) contidos no interior da carta. Esta abordagem toma partido da simetria vertical da distribuição das marcas no interior da carta.

É possível categorizar as cartas pelo número de colunas. A primeira categoria consiste em cartas apenas com uma coluna de marcas no meio da carta, mais especificamente as cartas com o número dois e três; a segunda categoria com duas colunas de marcas mais exteriores, apenas a carta com o número quatro, e uma terceira categoria com três colunas de marcas, uma coluna interior e duas exteriores, à qual pertencem as restantes cartas numéricas.

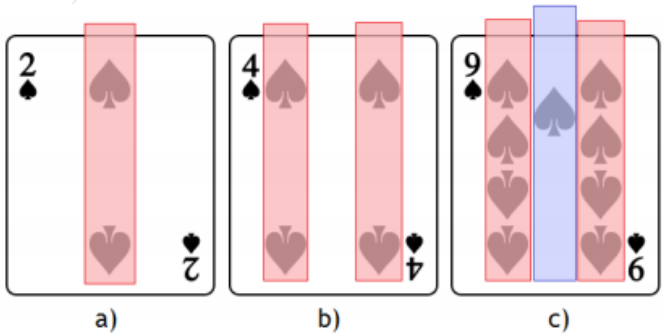


Figura 3.4.10: Categorização das cartas numéricas por número de colunas de marcas

Com esta categorização em mente, é possível contar as marcas existentes nas cartas do tipo numérico, mesmo que efeitos de iluminação impossibilitem a observação de todas as marcas.

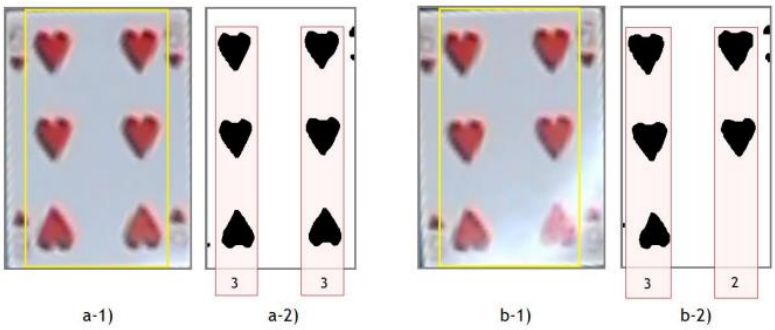


Figura 3.4.11: Contagem do número de marcas em cada coluna

3.4.4 Optical character recognition

Optical character recognition, ou em português reconhecimento ótico de caracteres, usualmente abreviado para *OCR*, refere-se ao ramo da computação que envolve a leitura de texto a partir de imagens, transformando-o num formato que seja manipulável computacionalmente, por exemplo em código *ASCII*. *OCR* é a translação mecânica ou eletrónica de imagens de texto manuscrito, ou escrito à máquina, ou impresso, usualmente capturado por um *scanner*, para um texto editável mecanicamente. No processo de *OCR*, a imagem é analisada procurando áreas claras e áreas escuras, de modo a identificar cada letra do alfabeto ou dígito numérico. Quando um caracter é reconhecido ele é convertido num código *UNICODE* [16].

Um sistema típico de *OCR* consiste em várias componentes. A Figura 3.4.12 mostra um *setup* comum de um *OCR*. O primeiro passo no processo consiste em digitalizar o documento usando um *scanner*. Quando as regiões contendo texto são localizadas, cada símbolo é extraído através de um processo de segmentação. Os símbolos extraídos podem ser pré-processados, eliminando ruído, para facilitar a extração de características no próximo passo.

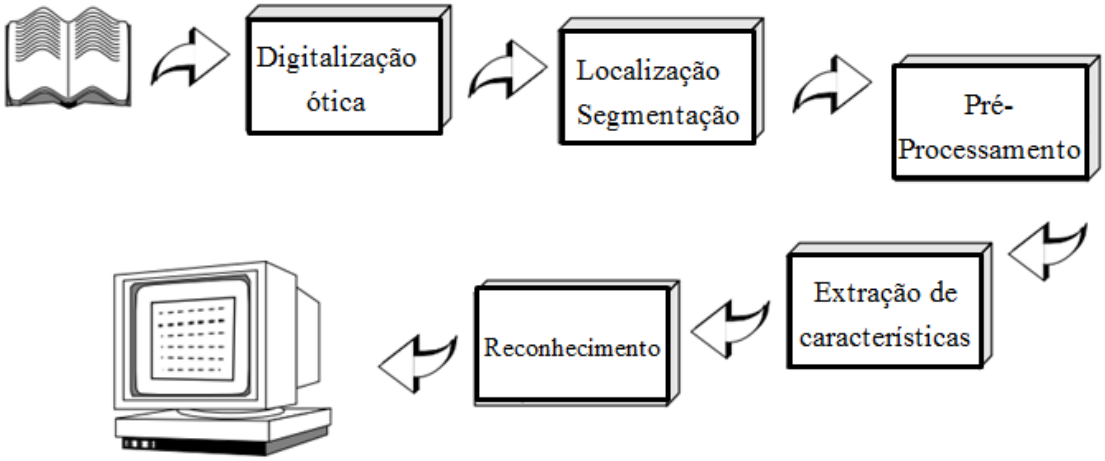


Figura 3.4.12: Componentes de um sistema de Optical character recognition. (imagem adaptada de **[17]**)

A identificação de cada símbolo é realizada por comparação entre as características extraídas e os descritores das classes de símbolos (ou fontes) obtidas através de uma fase prévia de aprendizagem. Finalmente, é usada informação contextual para reconstruir as palavras e números presentes no texto original. [17]

3.5 Simulador de Póquer do LIACC

Laboratório de Inteligência Artificial e Ciência de Computadores na Universidade do Porto (LIACC) foi criado em 1988 para promover a colaboração entre investigadores que estavam a trabalhar isoladamente nas áreas da Ciência de Computadores e Inteligência Artificial em diferentes faculdades [18].

Este laboratório desenvolveu um simulador de póquer designado de "LIACC's Texas Hold'em Simulator". Este sistema permite: a criação de ficheiros com a descrição do jogo de póquer numa linguagem criada para o efeito; gerar variantes de póquer definidas pelo utilizador através da sua interface gráfica; jogar a variante de póquer através de um simples visualizador 2D do jogo. A Figura 3.5.1 explica o fluxo de trabalho do sistema.

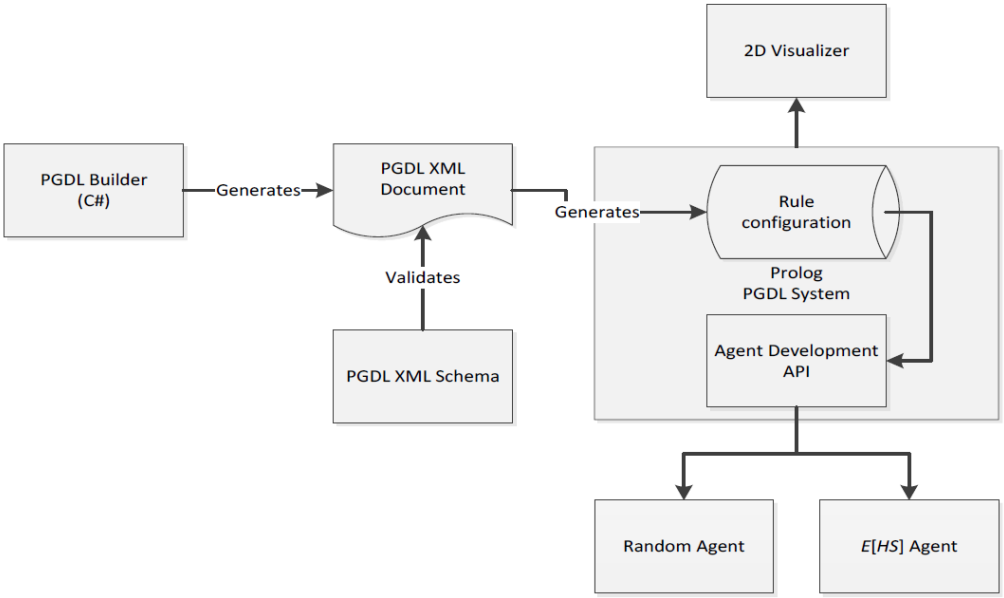


Figura 3.5.1: Simulador de póquer do LIACC (imagem retirada de **[19]**)

Com o PGDL Builder o utilizador especifica as regras da variante do jogo de póquer. Essa especificação gera um documento XML PGDL que é validado pelo XML Schema PGDL. Após a validação o documento PGDL XML é então traduzido para um arquivo Prolog que contém os termos necessários para configurar uma implementação de Poker genérica. Dois agentes são utilizados para jogar a variante de póquer baseados na força das suas mãos correntes e o jogo é apresentado num visualizador 2D onde o utilizador humano joga contra os agentes criados [19].

3.6 Construção de agentes jogadores de Póquer

Atualmente existem três abordagens principais para a construção de agentes jogadores de póquer: baseados em heurísticas, baseados em simulações e baseados em teoria de jogos. Estas abordagens focam-se na criação de uma estratégia ou política para um agente jogar, ou fornecer informação de como o agente deve reagir quando deparado com determinada situação. Uma breve discrição das abordagens para a criação de agentes de póquer é dada nas subsecções seguintes.

3.6.1 Baseados em regras

Uma abordagem baseada em regras para o póquer utiliza vários fragmentos de informação para criar uma estratégia de apostas. Por exemplo, informações como as *hole cards* do jogador, as atuais cartas comunitárias, a posição corrente do jogador na mesa e o historial de apostas

anteriores da mão fazem parte das heurísticas que determinam se um jogador deve fazer, *fold*, ou *check/call* ou *bet/raise* quando é a sua vez de agir.

Esta abordagem tenta simular a forma como os humanos descrevem como eles jogam, definindo um conjunto de regras condicionais e especificando que ação deve ser tomada de cada vez que essa situação ocorre. Apesar desta abordagem parecer razoável no início, dada a complexidade de um jogo como o póquer, esta tende a simplificar demasiado as situações, de modo a tornar praticável a computação do algoritmo, e tem sido provado ser extremamente limitada [20]. No entanto, estudos demonstram que agentes baseados em regras são ainda os mais adequados para variantes *No Limit*.

3.6.2 Baseados em fórmulas

Construindo sobre uma abordagem baseada em regras, uma abordagem baseada em fórmulas recorre a procedimentos complexos, ou fórmulas, para distinguir situações. Isto pode incrementar grandemente o número de situações identificáveis, e assim ser uma abordagem mais flexível do que sistemas baseados em regras apenas. Apesar de ser mais flexível, ainda sofre de alguns pontos fracos comuns às abordagens baseadas em regras, pois ambas são criadas sobre os mesmos pressupostos. Para além disto, um sistema construído seguindo esta abordagem é difícil de manter e de expandir, à medida que mais abstrações são incorporadas no sistema.

3.6.3 Baseados em simulações

Uma estratégia baseada em simulações é análoga à pesquisa seletiva em jogos com informação perfeita, como xadrez ou damas, mas em vez de expandir todos os nós na árvore de jogo com a mesma probabilidade, expande apenas certos nós na esperança de obter melhor informação em menor tempo analisando inicialmente os nós mais importantes. Infelizmente, em termos práticos, esta abordagem pode ser muito volátil e resultar num jogo extremamente desequilibrado, uma vez que a qualidade de simulações depende da qualidade da jogada simulada. Isto torna-o vulnerável para valores tendenciosos para certas simulações, levando a jogadas inexatas [21].

A técnica de Monte-Carlo Tree Search (MCTS) provou ser valiosa, pois é capaz de lidar com o grande espaço de estados associado com a variante "*no limit*" do póquer, contra múltiplos oponentes. Esta técnica apresentou bons resultados contra oponentes relativamente simples. Contudo, MCTS por si só não será capaz de competir contra jogadores humanos experientes, como é referido em [22].

3.6.4 Equilíbrio de Nash

A teoria de jogos é um ramo da matemática e economia que está dedicada à análise de jogos. Um conceito muito importante na teoria de jogos é o conceito de Equilíbrio de Nash para um conjunto de estratégias. Essas estratégias estão em equilíbrio de Nash se, ao alterar qualquer componente de uma dessas estratégias esta piora os seus resultados. Ou seja, um jogador não pode fazer melhor alterando a sua estratégia de jogo, considerando que os seus oponentes estão também a usar uma estratégia ótima, mais ainda, um jogador pode revelar as suas estratégias ótimas e não se tornar vulnerável ao aproveitamento dos seus oponentes [23].

Infelizmente criar uma estratégia perfeita de equilíbrio de Nash para um jogo complexo como o póquer é extremamente difícil e, neste momento, computacionalmente inviável. Ao invés, o que é atualmente usado é ε-Equilibro de Nash, que é uma aproximação da estratégia de Equilíbrio de Nash, resultando numa estratégia sub-ótima que leva a uma melhor resposta em vez de uma resposta perfeita. O valor ε é uma medida de quão distante de um equilíbrio real esta estratégia está. Se um oponente, quer humano, quer máquina, comete um erro, então a estratégia de equilíbrio pode vencer com o passar do tempo.

3.6.5 Aprendizagem por reforço

Aprendizagem por reforço é uma forma do agente ir aprendendo com as ações que executa. Ao agente não lhe é dito o que fazer, como também acontece na maioria das formas de *machine learning*, mas ao invés disso, ele terá que descobrir que ações lhe trazem maior recompensa dependendo de cada situação com que se depara. Nos casos mais interessantes e desafiantes, as ações realizadas podem não afetar somente a recompensa imediata, mas também a situação seguinte, por isso, todas as recompensas subsequentes. Estas duas características: pesquisa tentativa e erro e recompensa adiada são duas das mais importantes características da aprendizagem por reforço [24] [25].

Existiram poucas tentativas de desenvolver agentes utilizando uma metodologia de aprendizagem por reforço para o jogo de póquer. Uma abordagem para um agente que joga apenas na fase pré-flop do jogo foi descrita em [26]. Esta abordagem consiste de uma Q-Table contendo os pares estado-ação. Cada estado σ é definido por:

- **G:** Um valor representando o par de cartas que o jogador possui;
- **P:** A posição do jogador na mesa;
- **T:** O tipo de oponente;
- **A:** Um valor representando a última ação efetuada.

Para cada fase existe um tuplo correspondente (C,R) onde C é a probabilidade de efetuar um *Call* e R é a probabilidade de efetuar um *Raise*. Desta forma, quando o agente joga, ele procura na Q-Table para obter os valores de C e de R e um número aleatório N ($N \in [0,1]$) é sorteado. A escolha é realizada de acordo com a função:

$$Action = \begin{cases} Call & : N \in [0,C] \\ Raise & : N \in]C,R] \\ Fold & : N > R \end{cases}$$

3.7 Considerações finais

Neste capítulo dedicado à revisão da literatura relevante à elaboração desta dissertação foi apresentada a definição de póquer *bot* e a sua semelhança com o trabalho desenvolvido, bem como discutidas algumas técnicas de visão por computador que serviram de base no processo de reconhecimento das cartas e das quantias em jogo, nomeadamente, o template matching e o optical character recognition. Foram também abordadas as curvas de Bézier e como podem ser utilizadas para simular de um modo simplista o movimento do rato por um humano num jogo de póquer *online*. Finalmente, foram discutidas as principais abordagens existentes na construção de agentes de póquer e comparadas as suas características.

O trabalho desenvolvido teve por base os conceitos referidos ao longo deste capítulo.

Capítulo 4

Desenvolvimento do Sistema de reconhecimento e execução das jogadas

Este capítulo é dedicado à descrição e definição dos principais objetivos a atingir, bem como à especificação dos detalhes de implementação dos principais módulos do sistema implementado, nomeadamente: o reconhecimento das cartas de jogo, identificação do estado de jogo e posição do *dealer*, reconhecimento das quantias dos jogadores e valor das suas apostas, o envio automático de mensagens e a movimentação do rato e execução das jogadas. Será ainda descrita de uma forma sumária a interface gráfica e a estratégia do agente jogador de póquer criado.

4.1 Descrição do problema e principais objetivos

O principal foco desta dissertação foi a construção de uma *API* que serve de interface para testar agentes de póquer a jogar *online*, e em tempo real, contra utilizadores reais através de sistema de póquer *online*. Para cumprir este objetivo principal, foram solucionados uma série de problemas ou objetivos secundários, nomeadamente:

- Reconhecimento das cartas presentes na mesa de jogo e a posição do dealer, o que foi conseguido utilizando técnicas de visão por computador, nomeadamente, *edge detection*, linhas de Hough e *template matching*, referidas no Capítulo 3;

- Identificação os valores das apostas de cada jogador, a quantia que cada jogador ainda possui e a quantia do pote, o que foi conseguido utilizando processamento de imagem e *optical character recognition*, também referidas no Capítulo 3;

- Execução dos movimentos, com o rato e o teclado, no sistema de póquer *online*, simulando o comportamento humano, o que foi alcançado utilizando curvas de Bézier para construir as trajetórias do movimento do rato;

- Evitar a deteção pelo sistema de póquer da utilização da API. Isto poderá ser conseguido com recurso a uma máquina virtual para que o *software* de póquer não detete o processo onde corre a API, pressionar os botões em posições aleatórias, uma vez que são registadas as localizações dos cliques do rato em ficheiros *log* da aplicação de póquer, movimentos suaves do rato simulando o movimento da mão humana e escrita ocasional de mensagens pré-definidas no chat do jogo;

- Lidar com possíveis atualizações do *software* de póquer, o que foi conseguido fornecendo a possibilidade da customização da posição dos botões, posicionamento do pote e dos jogadores, bem como atualização dos *templates* utilizados para o reconhecimento das cartas de jogo;

- Integrar na API desenvolvida a biblioteca do simulador de póquer criado no LIACC e também descrito no Capítulo 3.

Um outro objetivo foi a criação de um agente de póquer simples capaz de jogar *online* através da API desenvolvida, como prova do conceito e da operabilidade da mesma. Esse agente foi criado utilizando uma abordagem baseada em regras referida no Capítulo 3.

4.2 Reconhecimento das cartas

O primeiro problema a resolver na determinação das cartas de jogo, foi o de identificar as regiões de interesse, a que contém as cartas comunitárias e a que contém as cartas da mão de cada um dos jogadores. Neste sentido foram utilizadas duas abordagens. A primeira abordagem foi a de tentar identificar os contornos das cartas, utilizando *edge detection*, e com base nesses contornos, determinar os retângulos contendo as cartas de jogo. Na Figura 4.2.1 pode-se observar uma imagem da janela de jogo, antes e depois da deteção dos retângulos contendo as cartas.

Figura 4.2.1: Deteção da região de interesse contendo cartas

No entanto, este método apresenta desvantagens, nomeadamente, a ocorrência de oclusões parciais das cartas com as fichas de jogo por exemplo, que faz com que ocasionalmente o retângulo contendo a carta de jogo não seja detetado, a Figura 4.2.2 ilustra precisamente essa situação:

Figura 4.2.2: Oclusão dificulta a deteção da região contendo cartas

No diagrama da Figura 4.2.3 apresenta-se as etapas envolvidas no processamento da imagem a fim de obter a região pretendida.

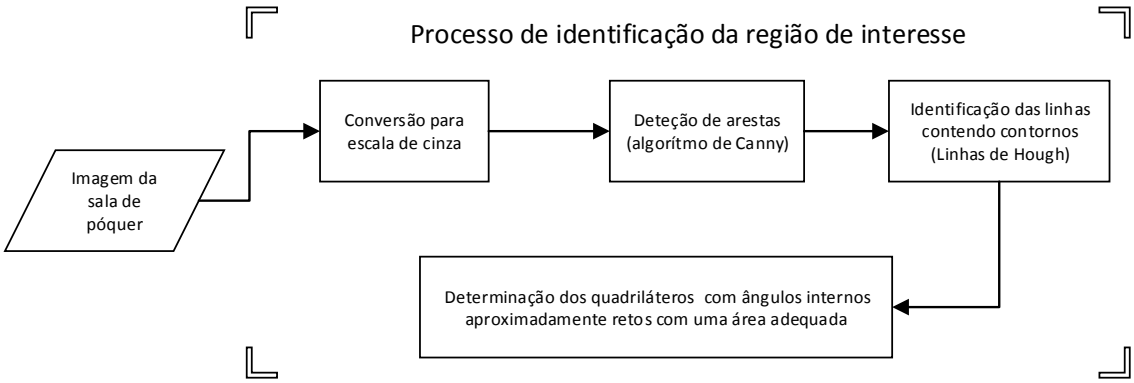


Figura 4.2.3: Etapas na deteção da região contendo cartas

O segundo método de determinação da região de interesse relativamente à posição das cartas, quer comunitárias, quer da mão, é bastante mais simples. Trata-se, precisamente, de dar a possibilidade do utilizador do sistema definir qual o retângulo que contém o conjunto de cartas, utilizando o rato na interface da aplicação, utilizando o menu de configuração descrito na secção 5.4.1. Este procedimento de configurar manualmente as regiões de interesse é especialmente útil em caso de atualizações do *software* de póquer que apresentem modificações nas posições das cartas de jogo, o que acontece com frequência.

No que diz respeito à identificação do valor e naipe das cartas de jogo na janela de póquer o método utilizado foi o *template mathing* em duas variantes: *templates* de toda a carta e *templates* apenas do canto superior esquerdo da carta, como se pode observar na Figura 4.2.4:

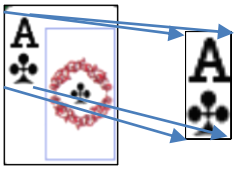


Figura 4.2.4: *Template* de toda a carta e *template* do canto

Depois de definida a região de interesse pelo utilizador, ou pela deteção descrita anteriormente, os *templates* são utilizados para "varrer" essa região da imagem da sala de póquer e assim de determinar o ponto com maior percentagem de correspondência. Se essa percentagem for superior a um determinado valor estabelecido, então está encontrada a carta.

A Figura 4.2.5 ilustra a deteção das cartas comunitárias utilizando os dois tipos de *templates*:

Figura 4.2.5: Identificação das cartas de jogo utilizando template matching

O algoritmo realizado para efetuar esta deteção, é o seguinte: É recortada a região de interesse contendo todas as cartas comunitárias (ou cartas da mão) e é percorrido um *array* contendo os *templates* de todas as cartas, e para cada um deles efetua-se o *template matching* para avaliar se existe correspondência. Se tal se verificar, então essa carta é adicionada à lista de cartas comunitárias (ou cartas da mão) e a imagem da região de interesse é atualizada, recortando-a. Se não existe correspondência, então será testado o *template* seguinte. Este procedimento é realizado, até percorrer todos os *templates*, ou até detetar um máximo de cinco

cartas comunitárias (ou duas cartas da mão). O diagrama da Figura 4.2.6 mostra esquematicamente o algoritmo implementado.

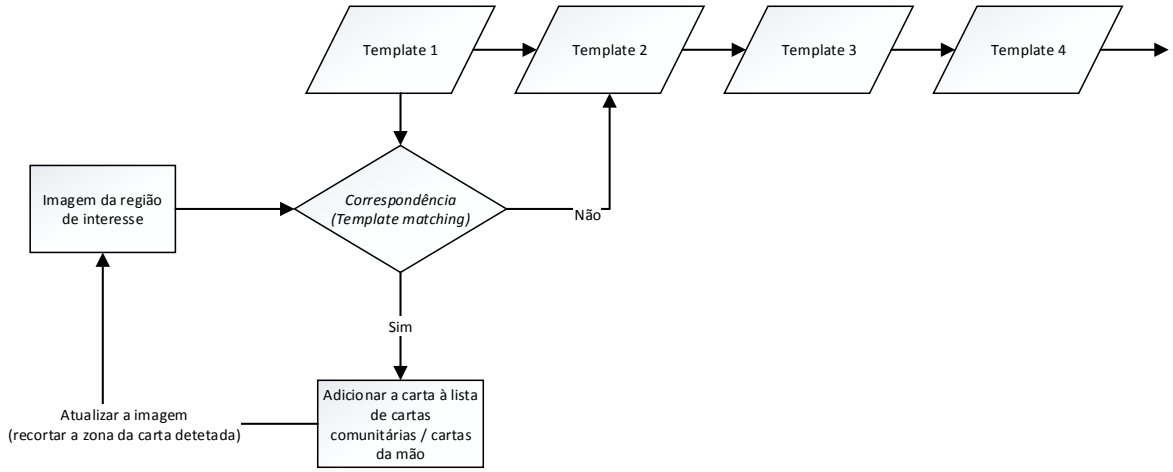


Figura 4.2.6: Etapas na identificação das cartas de jogo utilizando template matching

4.3 Identificação do estado de jogo e posição do dealer

Outra situação que foi necessária determinar a cada instante, refere-se à identificação do estado do jogo. Como já foi mencionado no Capítulo 2, um jogo de póquer é constituído por quatro rondas: *pré-flop*, *flop*, *turn* e *river*. A principal distinção entre estas rondas ou estados de jogo prende-se com a quantidade de cartas comunitárias presentes na mesa de póquer. No estado *pré-flop* não existem cartas comunitárias, enquanto que no estado *flop* existem três cartas comunitárias, no estado *turn* existem quatro e finalmente no estado *river* existem cinco.

A identificação do estado do jogo baseou-se então, na identificação do número de cartas comunitárias existentes na mesa de jogo, o qual foi conseguido utilizando os procedimentos referidos na Secção 5.1. Se o número de cartas comunitárias não tiver sido apurado corretamente, por algum motivo, então algumas medidas tiveram que ser tomadas, por exemplo: Se o número de cartas comunitárias detetadas for de apenas duas, o que não acontece em nenhuma das rondas, então seguramente que os estados de jogo possíveis naquele instante são *flop*, ou *turn* ou *river*. Outra situação a referir prende-se com a ordem na qual os estados de jogo surgem, por exemplo se o estado anterior num dado momento foi assinalado como *flop*, então seguramente que o estado seguinte não será *pré-flop*.

Desenvolvimento do Sistema de reconhecimento e execução das jogadas

Evidentemente, a deteção do final de um jogo de póquer, ocorre quando existe uma mudança do estado de jogo de um *river* para um *pre-flop*. O diagrama da Figura 4.3.1 ilustra, de uma forma simplificada, o mecanismo de deteção do estado de jogo.

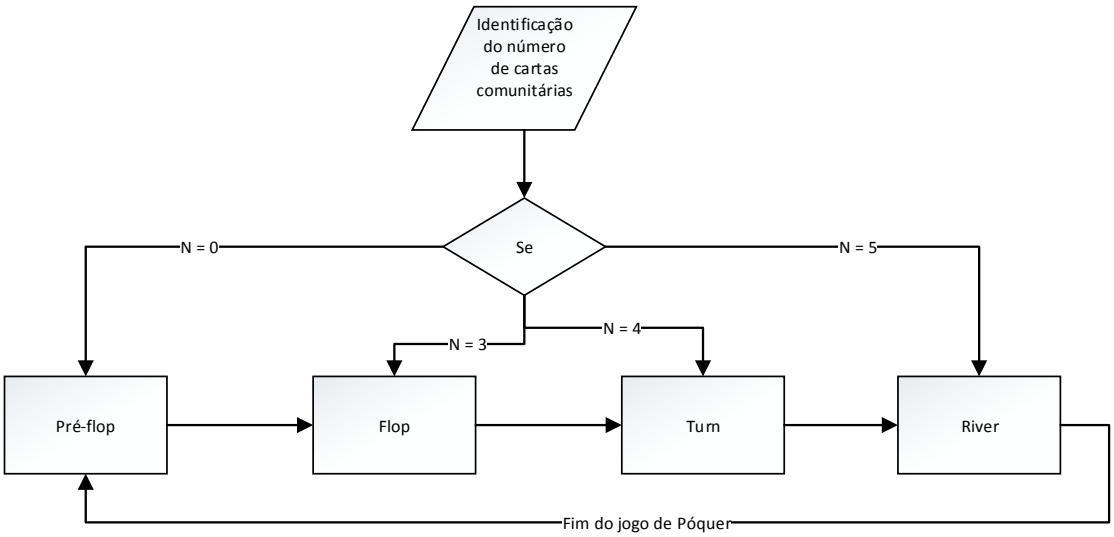


Figura 4.3.1: Diagrama de identificação do estado de jogo

A posição do jogador relativamente à do *dealer* pode ser um fator importante no jogo de póquer, pois o *dealer* é o último a jogar em cada ronda, pelo que o *dealer* e os jogadores imediatamente antes dele terão sempre mais informação relativamente às quantias em jogo e às desistências, do que os restantes adversários. Assim, é importante identificar, para cada jogo de Póquer quem é o *dealer*.

A identificação da posição do *dealer* é efetuada utilizando, também, a técnica de *template matching*. O *dealer* é marcado com uma ficha especial, pelo que a deteção da sua posição passa pela identificação da posição dessa ficha na mesa de jogo. O procedimento é o seguinte: o utilizador da aplicação tem a possibilidade de definir ou redefinir as regiões onde é possível encontrar a ficha de *dealer* para cada posição dos jogadores. Então, ter-se-á apenas que percorrer essas regiões e averiguar, utilizando *template matching*, se alguma delas contém o *template* relativo à ficha de *dealer*.

A Figura 4.3.2 mostra a identificação do dealer num determinado jogo de póquer.

Figura 4.3.2: Deteção da posição do dealer

4.4 Reconhecimento das quantias em jogo

Um problema bastante mais complexo do que a deteção das cartas de jogo é o da identificação das quantias de cada jogador e os valores das suas apostas. Essas apostas são realizadas através de fichas circulares com valores fixos, cuja dimensão e cor varia consoante o seu valor. A identificação do valor das apostas poderia ser realizada através da contagem do número de fichas de cada tipo, no entanto, oclusões parciais ou mesmo totais inviabilizam essa abordagem. Como se pode observar na Figura 4.4.1, seria extremamente complexo identificar o valor do pote e das apostas de cada jogador.

Figura 4.4.1: Identificação das quantias por contagem das fichas

Em contrapartida, existe uma outra possibilidade mais exequível que é a de extrair a quantia que cada jogador possui, utilizando *OCR*, do valor que se encontra visível em texto na

posição de cada jogador, e com base na quantia obtida no *frame* atual e no *frame* anterior extrapolar qual foi o valor da aposta efetuada.

A extração da quantia de cada jogador utilizando *OCR* é realizada com o motor de reconhecimento ótico de caracteres *Tesseract* que se encontra incorporado no *EmguCV*. De referir que o texto obtido, nem sempre é um número, também pode ser uma frase, nomeadamente, as frases "All In", "De fora" e "Lugar vazio". O diagrama da Figura 4.4.2 ilustra as etapas que são percorridas na identificação das quantias de cada jogador.

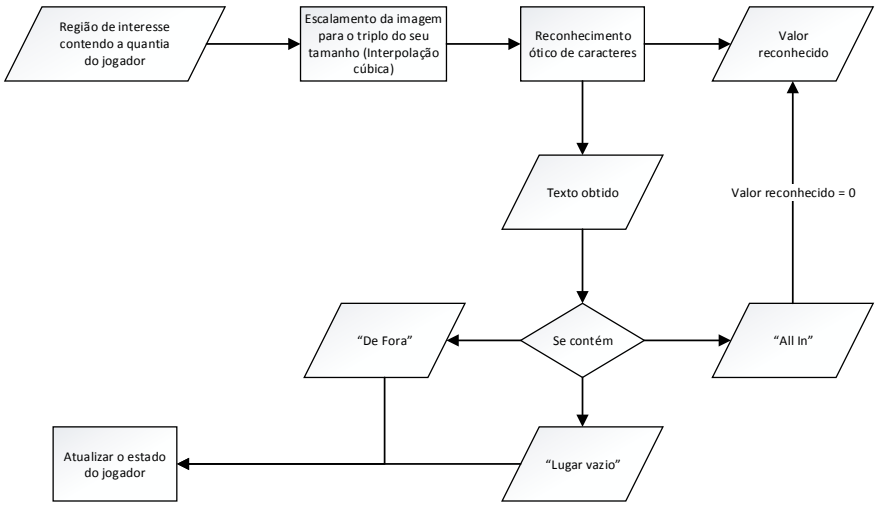


Figura 4.4.2: Etapas no reconhecimento das quantias em jogo

As regiões que contêm os valores de cada jogador podem ser definidas ou redefinidas pelo utilizador do sistema, através do menu de configuração. A cada uma dessas regiões é extraído o texto nelas contido, após uma ampliação para o triplo do seu tamanho. Esse procedimento evita falsas deteções como por exemplo a troca na identificação entre um número "5" e um número "6". Dessa imagem, são então extraídos dois textos, um com todos os caracteres possíveis e outro apenas com dígitos numéricos. Ao primeiro texto obtido, é ainda realizado um processo de limpeza de caracteres indevidos que possam ser encontrados, como sinais de pontuação, espaços, entre outros. Após este procedimento, é analisado texto obtido para averiguar se ele contém a totalidade ou parte dos textos "De Fora", "Lugar vazio" ou "All In". Caso suceda uma das duas primeiras situações, então o estado do jogador é atualizado, caso ocorra a última situação, então a quantia do jogador é atualizada para o valor zero. Se nenhuma das situações anteriores se verificar, então o valor obtido com o reconhecimento apenas de dígitos numéricos será o valor definido como a quantia do jogador.

4.5 Simulação das ações humanas

A execução das jogadas após decisão pelo agente deverá ser algo que simule o mais exatamente possível a movimentação humana, pois pretende-se também evitar de algum modo a possível deteção pelo sistema de póquer *online* da utilização do *bot*. Com esse intuito alguns procedimentos foram planeados, nomeadamente, a movimentação do rato, o clique nos botões e o envio automático de mensagens para o *chat* do sistema de póquer.

4.5.1 Movimentação do rato e clique nos botões

A movimentação do rato por um utilizador humano não é certamente um movimento uniforme e retilíneo e a velocidade de movimentação não é constante. Também um jogador humano não irá clicar nos botões sempre na mesma posição. Neste sentido, uma adaptação para a movimentação do rato pelo sistema desenvolvido foi planeada para simular fracamente a movimentação por um jogador humano.

Assim, aplicou-se o algoritmo de construção de curvas de Bézier, descrito na Secção 3.2.1. para construir o caminho ou trajetória pela qual se movimenta o rato ao efetuar a jogada. As curvas de Bézier criadas dependem da distância entre o ponto inicial e o ponto final. De acordo com essa distância, a curva de Bézier criada terá um grau diferente, ou poderá ser a união de duas curvas e Bézier. Os pontos de controlo da curva de Bézier, à exceção do primeiro e do último, são obtidos de forma aleatória. Para tornar a trajetória um pouco mais realista é adicionado um pouco de ruído à curva de Bézier, através do modelo de Gauss, de modo a que os pontos mais próximos do início e final da curva sejam menos afetados do que os pontos a meio da trajetória. A velocidade do movimento é configurável. O diagrama da Figura 4.5.1 apresenta um esquema do algoritmo efetuado.

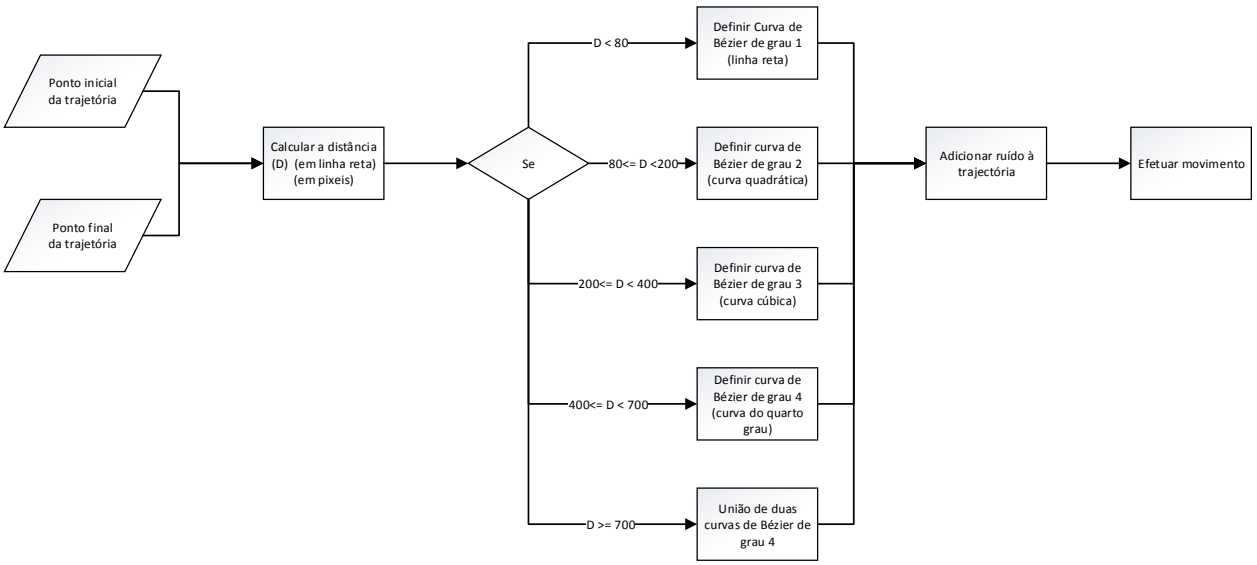


Figura 4.5.1:Movimentação do rato

Outro pormenor que foi tido em consideração foi a questão dos cliques nos botões e no tempo de reação do sistema perante a presença dos mesmos: Como um utilizador humano não clica nos botões sempre na mesma posição e o seu tempo de reação varia, devido a inúmeros fatores. Essa situação foi contemplada na implementação do sistema, adicionando aleatoriamente um tempo de reação antes da execução dos movimentos e também a escolha aleatória do ponto de clique no botão. A Figura 4.5.2 mostra numa linha amarela a trajetória efetuada pelo rato na execução de um *Fold* após decisão do agente e numa linha laranja a trajetória após essa execução.

Figura 4.5.2: Trajetória percorrida pelo rato na execução de um Fold

4.5.2 Envio de mensagens automáticas

Também com o intuito de simular o comportamento, acrescentou-se a funcionalidade de poderem ser enviadas mensagens genéricas para o *chat* do jogo. Assim, o utilizador final da plataforma poderá definir se pretende ativar esta funcionalidade e especificar quais as mensagens que pretende que sejam envidas e a periodicidade das mesmas.

4.6 Implementação

O sistema desenvolvido divide-se em dois módulos, um módulo relativo a todas as funcionalidades da API e um módulo dedicado ao agente de póquer. O módulo dedicado à API está subdividido em quatro pacotes: um deles para a implementação das funcionalidades da API, onde se destaca uma classe dedicada à deteção dos elementos na sala de póquer (cartas, quantias, dealer, entre outras), uma classe dedicada à simulação das ações com o rato, nomeadamente, a movimentação, execução das jogadas (*raise*, *fold*, *call*) e envio de mensagens para o *chat* do jogo, e uma classe dedicada à captura da imagem da janela do jogo de póquer; um pacote dedicado à interface gráfica, na qual são apresentados os dados recolhidos pela API e são apresentadas informações relativas às apostas e incidências do jogo; um pacote dedicado à base de dados SQLite para registar as quantias ganhas ou perdidas pelo agente em cada jogo de póquer disputado, a fim de puder elaborar estatísticas relativas ao agente utilizado; um pacote dedicado a testes efetuados para avaliar o trabalho desenvolvido ao longo da sua

implementação. O diagrama de pacotes da Figura 4.6.1 ilustra a estruturação da arquitetura efetuada no sistema desenvolvido.

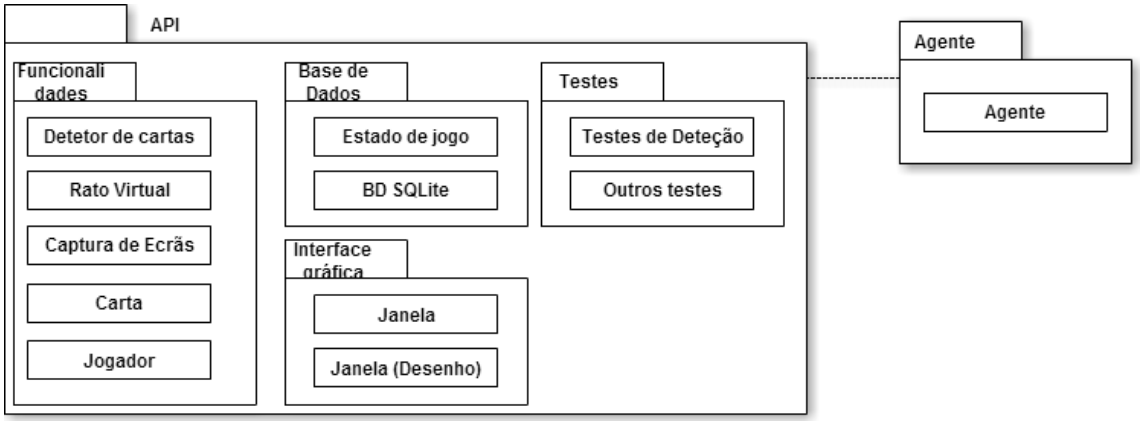


Figura 4.6.1: Arquitetura do sistema desenvolvido

Nos tópicos seguintes irão ser abordados em pormenor alguns dos aspetos mais relevantes da implementação do sistema.

4.6.1 Interface gráfica

Para apresentar de uma forma mais agradável o sistema desenvolvido, foi criada uma interface gráfica para efetuar o display dos elementos recolhidos na análise das imagens da sala de jogo. A Figura 4.6.2. apresenta a interface gráfica criada.

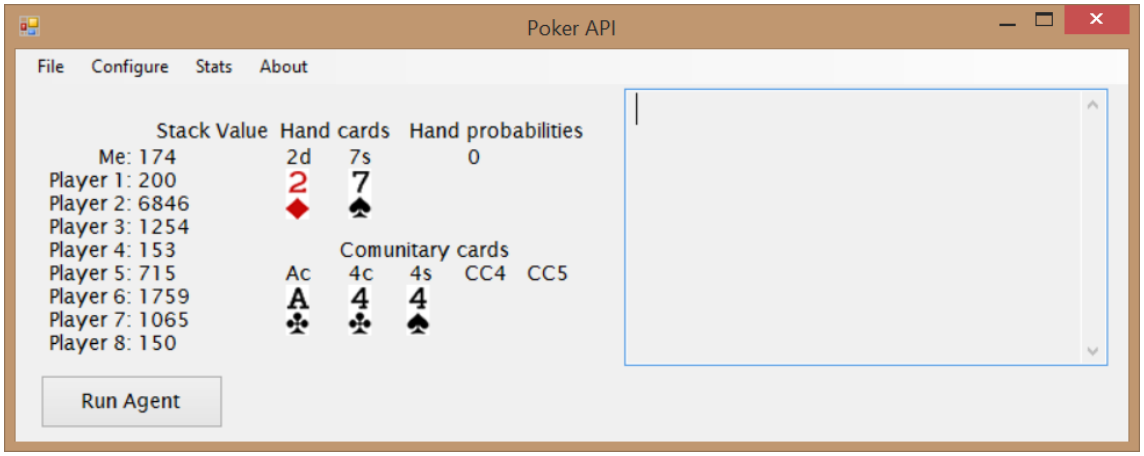


Figura 4.6.2: Interface gráfica

Desenvolvimento do Sistema de reconhecimento e execução das jogadas

Tal como é possível observar na figura 4.6.2, na interface gráfica são apresentadas as quantias que cada jogador possui, as cartas da mão, as cartas comunitárias detetadas e a probabilidade da mão, de acordo com o estado atual do jogo. Na área de texto do lado direito são apresentadas informações relativas ao decorrer do jogo, nomeadamente, os valores das apostas dos jogadores, a ronda atual e ações dos jogadores.

O botão "Run Agent" serve para lançar a execução de um *Thread* inicializando o agente e jogando na sala de póquer com base na IA definida na implementação do agente. Para tal são captados *screenshots* da janela de póquer *online* a cada segundo, e analisadas, para identificar o estado de jogo, as cartas comunitárias e da mão e as quantias atuais de cada jogador. Confrontando duas imagens consecutivas permite concluir quais os valores das apostas, se se verificarem diferenças nos valores das quantias dos jogadores.

A interface principal apresenta quatro menus: "File", "Configure", "Stats" e "About". No menu "File" existem três itens: uma opção "Open" que é utilizada para abrir imagens pré-gravadas da sala de jogo, captar os elementos detetados nessa imagem e apresenta-los na interface principal; uma opção "Save" que permite gravar a imagem da sala de jogo; e ainda uma opção "Exit" que fecha a aplicação. Ao fechar a aplicação, quer por este menu, quer por clique no botão do canto superior direito da janela, é apresentada uma mensagem de confirmação se o utilizador pretende mesmo encerrar o programa, e em caso afirmativo, são terminados todos os *Threads* da aplicação, nomeadamente o *Thread* que está a executar o agente e a aplicação encerra.

O menu "Configure" permite ao utilizador configurar as posições: das cartas comunitárias, das cartas da mão de cada jogador, dos valores de cada jogador, dos botões de jogo e das possíveis regiões contendo a ficha do dealer. Estas configurações têm especial interesse por permitem reconfigurar o sistema de deteção em caso de atualização da aplicação de póquer *online*. No tópico seguinte irá ser detalhado ao pormenor esse processo.

O menu "Stats" tem duas opções: "View Stats" para apresentar as estatísticas relativas ao desempenho do agente, nomeadamente, número de jogos disputados, número de vitórias, dinheiro acumulado ou perdido, e os ganhos/perdas médios por jogo disputado e a opção "Reset database" que permite limpar os dados da base de dados para que possa começar a contabilizar os dados para a estatística a partir desse momento, o que é útil quando queremos estudar o desempenho de um outro agente.

O menu "About" apresenta informações relativas à aplicação desenvolvida, nomeadamente, autores, propósito da aplicação e o âmbito na qual esta foi desenvolvida.

4.6.2 Reconfiguração dos elementos no sistema de Póquer

Uma das situações que se tentou solucionar foi a de lidar com as possíveis atualizações do *software* de póquer *online*, que por vezes podem apresentar ligeiras mudanças na localização dos elementos da sala de jogo em relação à versão anterior, o que podia inviabilizar o sistema de deteção implementado. Por forma a contornar o problema, fornece-se ao utilizador a possibilidade de reconfigurar as posições desses elementos através do menu "Configure". A Figura 4.6.3 ilustra a utilização deste menu para reconfigurar a posição dos botões na sala de póquer.

Figura 4.6.3: Processo de reconfiguração dos botões do jogo

A figura apresenta os ecrãs que surgem após clicar na opção de configuração dos botões, assim é apresentada a informação para selecionar o botão de desistência e apresentada a imagem da sala de póquer para o utilizador selecionar a região onde se localiza o botão de desistência, este processo é repetido até que todos os botões estejam configurados. A informação da localização de cada botão é gravada em ficheiro de texto, que é carregada sempre que se inicie a aplicação. O procedimento de configuração dos restantes elementos: região das cartas comunitárias, região das cartas da mão de cada jogador, localizações das quantias de cada jogador e possíveis localizações para a ficha do dealer é idêntico ao processo descrito anteriormente.

Apesar de este procedimento, solucionar alguns dos problemas que surgem com a atualização do *software* de póquer *online*, não responde a outras possíveis mudanças, como por exemplo a alteração dos modelos das cartas, ou do seu tamanho. Para essa situação, a solução seria alterar os *templates* existentes, o que seria um processo moroso. Deste modo, pretende-se reafirmar que a solução implementada não é aplicável a todos os *softwares* de póquer *online*, e a todas as suas versões, mas tem características que permitem a sua alteração e adequação.

4.6.3 Funcionalidades da API

A API criada fornece ao agente todas as informações extraídas da sequência de imagens captadas da sala de póquer *online*, nomeadamente, informações relativas ao estado de jogo (pré-*flop*, *flop*, *turn* ou *river*), informações relativas aos jogadores, se estão presentes ou ausentes e o valor das suas quantias e suas apostas, informações relativas às cartas de jogo, que cartas comunitárias estão presentes (se não for o estado pré-flop), que cartas da mão possui o utilizador e qual a probabilidade de com aquela mão de jogo, o utilizador ganhar o encontro.

Informa ainda, qual a posição do *dealer*, pois pode ser um fator a ser considerado na implementação da IA do agente. E por fim, informa se naquele momento é a vez de o utilizador jogar ou não, assim, o programador do agente pode aproveitar os momentos em não tem que efetuar uma jogada para puder enviar mensagens para o *chat* de jogo, ocasionalmente. As imagens da sala de jogo são obtidas a partir do *screenshot* da janela do *software* de póquer em intervalos de um segundo, da seguinte forma: é procurado o processo a correr no computador que contém no seu nome um determinado elemento, no caso a palavra "Texas Hold'em", e a partir do apontador para a janela desse processo obtém-se a imagem. No caso de se utilizar uma máquina virtual, a abordagem teria de ser diferente, ter-se ia que captar o screenshot de todo o ecrã e aí, com processos de visão por computador identificar qual a região referente à janela da sala de póquer.

A API fornece ainda funções para a execução de jogadas, segunda uma arquitetura baseada em eventos, nomeadamente a função *fold*, *call*, *all in* e *raise*, esta com várias possibilidades: um *raise* normal em que se eleva o valor apresentado na sala de jogo por defeito, um *raiseMin* em que se eleva a aposta com o valor mínimo, um *raiseOneHalf* em que se eleva a aposta para mais metade do valor do pote, um *raisePot* que eleva a aposta para mais uma vez o valor do pote e ainda um *raise(val)* em que pode ser especificado, no argumento da função, qual o valor que se pretende elevar a aposta. O programador do agente terá apenas que implementar uma função relativa à IA do agente e chamar as funções de *fold*, *call*, *allin* e *raise* referidas, quando pretender efetuar as ações.

Outras funções da API são a possibilidade de definir uma lista de mensagens automáticas que serão enviadas para o *chat* da sala de jogo ocasionalmente, numa periodicidade que pode ser definida pelo programador do agente, a possibilidade de configurar a velocidade do rato em três níveis: lento, médio e rápido e ainda a possibilidade de o programador do agente puder definir se pretende que possam ser acrescentadas fichas de jogo, na eventualidade de ficar sem fichas e, nesse caso, definir um teto máximo que dinheiro que está disposto a perder.

4.6.4 Integração no simulador de Póquer do LIACC na API desenvolvida

Na API desenvolvida foi integrada a biblioteca do simulador de póquer do Laboratório de Inteligência Artificial da Faculdade de Engenharia da Universidade do Porto, permitindo assim que todas as ações do jogo sejam validadas, nomeadamente, as jogadas, as apostas, as cartas observadas e as rondas do jogo. A incorporação desta biblioteca na API permite também obter o valor das probabilidades da mão do utilizador a cada momento, algo que pode ser útil na elaboração da inteligência artificial do agente de póquer.

4.6.5 Agente de póquer

Para testar a API desenvolvida foi criado um agente simples de póquer baseado em regras. Houve o cuidado de separar o agente da API, para que o sistema desenvolvido ficasse mais modular e para que o desenvolvedor do agente não necessite de conhecer os detalhes da implementação da API. Isto foi conseguido através da utilização de delegações e eventos em C#.

Quando a API necessita da decisão para uma jogada por parte do agente, envia um evento para a classe agente que, com base na sua implementação decide a jogada a efetuar. O mesmo acontecendo, embora em sentido oposto, para a execução dessa mesma jogada. Os parâmetros enviados ao agente nesse evento são: um objeto da classe *CardDetector*, o qual contém a informação das cartas detetadas na mesa de jogo e cartas da mão; uma lista de objetos da classe Player, que contém informação de quais os jogadores ativos, que quantias possuem e valores das suas apostas; um valor real que informa a probabilidade da mão atual ser uma mão vencedora, valor esse obtido pela biblioteca referida na Secção 4.5.4 e um boleano referindo se se trata ou não da sua vez de jogar. Assim, com base nestes parâmetros é possível definir estratégias de jogo para decidir qual a ação: *call/check*, *raise* (*bet* ou *all in*), *fold* e envio de mensagens para o *chat*. Essa ação é então enviada, pelo mesmo processo de delegação usando eventos, para a API, através da função *doAction*.

Para verificar a usabilidade da API desenvolvida foi criado um agente simples que utilizasse todas as funcionalidades da API.

O diagrama da seguinte Figura 4.6.4 apresenta a estratégia seguida pelo agente.

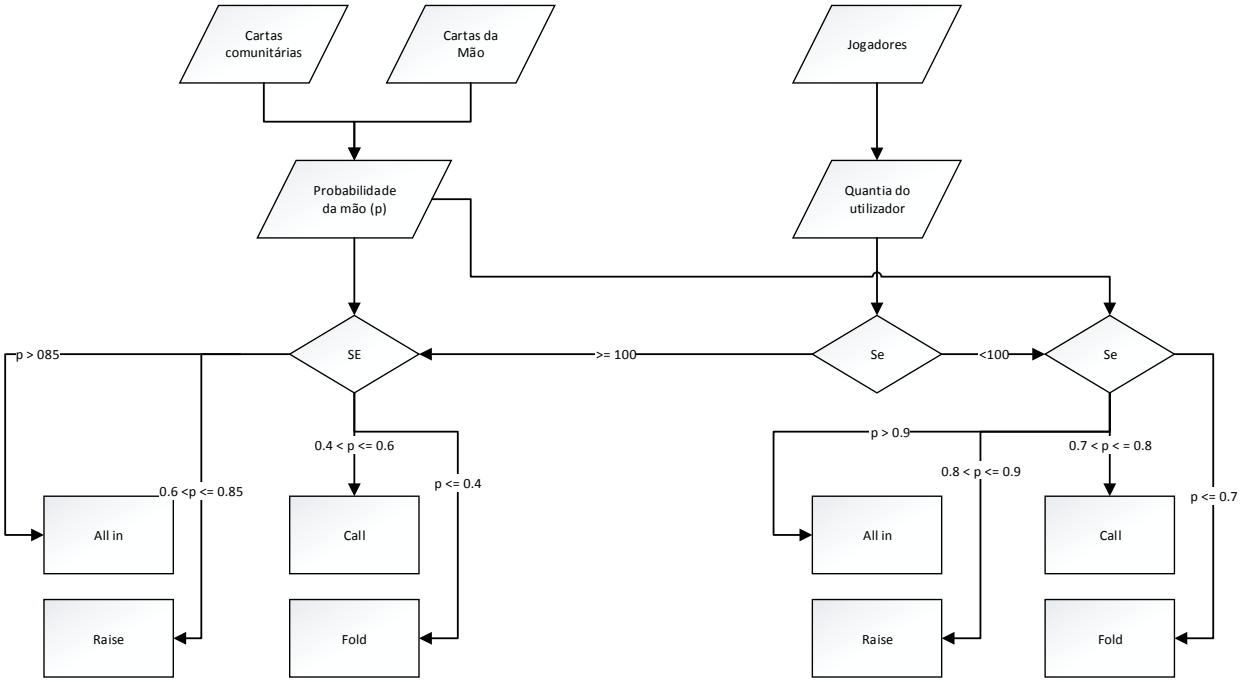


Figura 4.6.4: Estratégia baseada em regras do agente de póquer

4.7 Considerações finais

Este Capítulo 4 foi dedicado às especificações e detalhe da implementação do trabalho efetuado nesta dissertação. Foram mencionados os principais objetivos a atingir e estratégias utilizadas para os alcançar. Foram descritos ao pormenor os métodos utilizados para identificação das cartas, estado de jogo e dos valores das quantias e apostas dos jogadores e expostos os procedimentos adotados para a execução das jogadas no *software* de póquer, emulando o comportamento do movimento humano. Foram ainda apresentadas as funcionalidades da API e o aspeto gráfico da interface criada e como foi integrada a biblioteca do simulador de póquer existente no LIACC. Finalmente, foi descrito como foi implementado o agente simples para jogar póquer *online* utilizando a API desenvolvida.

Capítulo 5

Resultados e Discussão

5.1 Testes efetuados e discussão dos resultados

Para validar e analisar a eficiência dos métodos implementados, foram realizados testes de validação ao longo do desenvolvimento, com maior incidência nos processos de reconhecimento dos elementos presentes nas imagens da sala de póquer. Assim, foram conduzidos os seguintes testes: identificação das cartas de jogo, identificação das quantias dos jogadores, posição do *dealer*, movimentação do rato e execução do agente a jogar no *software* de póquer *online*. Para os testes de identificação dos elementos na sala de póquer foram obtidas duas amostras de imagens: uma com duzentas imagens da sala de póquer com resolução 1016×728 e outra com trinta e cinco imagens da sala de póquer com resolução 1158×826, ambas obtidas a partir de *screenshots* adquiridos em intervalos de cinco segundos a partir do *software* de póquer *online* PokerStars. Para o teste relativo à movimentação do rato foram criados dois vídeos mostrando a realização de uma jogada por um jogador humano e pelo sistema desenvolvido e para o teste de execução do agente a jogar póquer *online* através da API desenvolvida, colocou-se o agente a jogar ininterruptamente ao longo de uma hora. Os detalhes destes testes são explicados nas subsecções seguintes.

5.1.1 Identificação das cartas de jogo e da posição do dealer

O procedimento realizado para testar a deteção das cartas de jogo e da posição do *dealer* foi o de registar em ficheiro de texto as cartas comunitárias presentes em cada imagem, para ambas as amostras. Em seguida criou-se um *script* que lesse o ficheiro de texto e que recolhesse as cartas comunitárias das duzentas imagens, pelo procedimento de reconhecimento das cartas

da API e efetuasse a comparação entre os dois dados. Assim, os resultados obtidos estão apresentados na Tabela 5.1.1. e na Tabela 5.1.2.

Tabela 5.1.1: Resultados do teste de identificação do valor das cartas na amostra com resolução $\mathbf{1158 \times 826}$

Valor	Ás	Dois	Três	Quatro	Cinco	Seis	Sete	Oito	Nove	Dez	Valete	Dama	Rei	Total
Nº de cartas	3	6	4	6	4	4	4	10	3	4	0	28	21	97
Nº de cartas identificadas	3	6	4	6	4	4	4	10	3	4	0	28	21	97
Nº de cartas não identificadas	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Percentagem	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%

Tabela 5.1.2: Resultados do teste de identificação do valor das cartas na amostra com resolução $\mathbf{1016 \times 728}$

Valor	Ás	Dois	Três	Quatro	Cinco	Seis	Sete	Oito	Nove	Dez	Valete	Dama	Rei	Total
Nº de cartas	49	15	40	44	29	60	32	27	42	11	2	29	73	453
Nº de cartas identificadas	46	13	39	43	25	57	32	25	41	10	2	28	70	431
Nº de cartas não identificadas	3	2	1	1	4	3	0	2	1	1	0	1	3	22
Percentagem	93,9%	86,7%	97,5%	97,7%	86,2%	95,0%	100,0%	92,6%	97,6%	90,9%	100,0%	96,6%	95,9%	95,1%

Os principais problemas na identificação das cartas de jogo prendem-se com a resolução da imagem, uma vez que se otimizou a deteção para a resolução de 1158×826, e para resoluções diferentes é necessário fazer um redimensionamento da imagem, neste caso com interpolação cúbica, o que afeta ligeiramente a imagem obtida. O que se pode observar nos resultados obtidos nas duzentas imagens com resolução 1016×728, o que leva a um aumento ligeiro de deteções incorretas, com maior incidência nas cartas de valor dois e cinco. Por exemplo o cinco de ouros presente na Figura 5.1.1 não foi detetado.

Figura 5.1.1: Não deteção do cinco de ouros

Outro fator que pode contribuir para falsas deteções prende-se com a similaridade entre alguns templates, por exemplo um ás de copas e um ás de espadas são muito similares, à exceção da cor. O que pode ser resolvido pela deteção da densidade da cor, se a região possui elevada concentração de cor vermelha, então basta comparar com os vinte e seis templates de cartas vermelhas, evitando assim este problema.

Em jeito de conclusão pode-se referir que para a resolução de 1158×826, resolução essa para a qual a aplicação foi otimizada, a deteção das cartas é praticamente perfeita, e para resoluções diferentes poderá ocorrer um aumento ligeiro de deteções erradas ou não deteções que será tanto maior quanto mais afastada for a resolução da imagem da resolução ideal.

Relativamente à deteção da posição do dealer, o procedimento realizado foi idêntico, isto é, foi registado manualmente qual o dealer em cada uma das imagens e verificou-se a correspondência com a identificação do *dealer* utilizando o mesmo procedimento da API desenvolvida. Assim, os resultados obtidos estão apresentados nas Tabelas 5.1.3 e 5.1.4.

Tabela 5.1.3: Resultados do teste de identificação do dealer na amostra com resolução $\mathbf{1158 \times 826}$

Posição do dealer	jogador 0	jogador 1	jogador 2	jogador 4	jogador 7	Total
Nº de vezes	4	19	4	7	1	35
Nº de identificações corretas	4	19	4	7	1	35
Nº de identificações incorretas	0	0	0	0	0	0
Percentagem	100,0%	100,0%	100,0%	100,0%	100,0%	100,0%

Tabela 5.1.4: Resultados do teste de identificação do dealer na amostra com resolução $\mathbf{1016 \times 728}$

Posição do dealer	jogador 0	jogador 1	jogador 2	jogador 3	jogador 4	jogador 5	jogador 6	jogador 7	jogador 8	Total
Nº de vezes	14	0	47	38	0	17	32	52	0	200
Nº de identificações corretas	14	0	47	38	0	17	32	52	0	200
Nº de identificações incorretas	0	0	0	0	0	0	0	0	0	0
Percentagem	100,0%	100,0%	100,0%	100,0%	100,0%	100,0%	100,0%	100,0%	100,0%	100,0%

Como se pode observar os resultados de deteção da posição do dealer foram infalíveis para as duas amostras utilizadas mas, no entanto, como o procedimento de deteção quer das cartas, quer da posição do *dealer* assenta no mesmo conceito, *template matching*, a resolução da imagem também é um fator que potencialmente influencia a eficiência da deteção.

5.1.2 Identificação das quantias de jogo

Quanto à identificação das quantias de jogo o procedimento realizado no teste foi similar aos anteriores, isto é, foi criado um ficheiro de texto, por cada amostra de imagens, contendo as quantias observadas em cada imagem, separadas por espaços. Por exemplo, a linha "167 200 7345 732 -1 -1 1899 953 198" representa as quantias dos jogadores (cujos índices variam de 0 a 8), sendo que o valor -1 representa que o jogador está de fora ou o lugar está vazio.

Tal como os testes anteriores, foi criado um script que lesse o ficheiro e comparasse com os valores das quantias obtidas pelo procedimento utilizado na API. As Tabelas 5.1.5 e 5.1.6 apresentam os resultados obtidos.

Tabela 5.1.5: Resultados na deteção das quantias na amostra com resolução $\mathbf{1158 \times 826}$

Tipo	**All In**	**De Fora / lugar vazio**	**Valor normal**	**Total**
Nº de quantias	96	33	186	315
Nº de identificações corretas	96	29	186	311
Nº de identificações incorretas	0	4	0	4
Percentagem	100,0%	87,9%	100,0%	98,7%

Tabela 5.1.6: Resultados na deteção das quantias na amostra com resolução $\mathbf{1016 \times 728}$

Tipo	All In	De Fora / lugar vazio	Valor normal	Total
Nº de quantias	62	84	1654	1800
Nº de identificações corretas	62	67	1648	1777
Nº de identificações incorretas	0	17	6	23
Percentagem	100,0%	79,8%	99,6%	98,7%

Assim, observa-se que na amostra de trinta e cinco imagens o único problema de deteção das quantias em jogo trata-se do caso em que o jogador está de fora, ou o lugar está vazio. Nestes casos três situações podem ocorrer: ou no lugar da posição do jogador pode ler-se "De Fora" ou "Lugar vazio" ou a letra "R". Esta letra surge, menos frequentemente, quando o utilizador que está a entrar na sala de póquer está a escolher qual a quantia com que vai iniciar o seu jogo. A Figura 5.1.2 mostra estas três situações referidas.

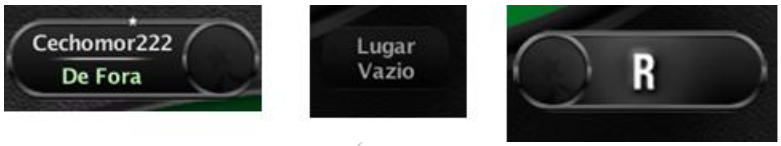


Figura 5.1.2: Situações que podem ocorrer quando o jogador não está presente

Nos casos em que na posição do jogador aparece a letra "R", como mostra na Figura 5.1.2. o valor detetado é o valor "11", pois a imagem recortada que é lida pelo OCR contém apenas a parte inferior da letra "R", causando assim uma má deteção. Nos outros dois casos, a imagem recortada para leitura por OCR contém a as frases "De Fora" ou "Vazio" que são regra geral, bem interpretadas.

Foi referido na Secção 4.4. que para a leitura das quantias se redimensiona a região de interesse para o triplo do seu tamanho, para melhorar os resultados. De facto, executando o teste para a amostra de duzentas imagens, sem proceder ao redimensionamento, os resultados pioram ligeiramente, como se pode observar na Tabela 5.1.7.

Tabela 5.1.7: Resultados na deteção das quantias sem o redimensionamento

Tipo	All In	De Fora / lugar vazio	Valor normal	Total
Nº de quantias	62	84	1654	1800
Nº de identificações corretas	58	66	1646	1770
Nº de identificações incorretas	4	18	8	30
Percentagem	93,5%	78,6%	99,5%	98,3%

Pode observar-se aqui que alguns "All In" foram falsamente identificados como quantias. A Figura 5.1.3 mostra um exemplo de uma deteção errada.

Figura 5.1.3: Um exemplo de um "All In" mal identificado

Os resultados obtidos com redimensionamento da região de interesse foram substancialmente melhores, pois a deteção das quantias é baseada no reconhecimento ótico de caracteres que é mais tolerante a variações de tamanhos. Outra situação que ocorre é a identificação incorreta de caracteres que acontece, por exemplo, entre o número 5 e o número 6, que apesar do redimensionamento e pós processamento da zona da imagem não soluciona o problema a 100%, e a má definição da zona da imagem que contém a quantia do jogador, pois se a região definida que contém o valor da quantia do jogador for muito grande, então pode-se identificar outros elementos existentes nessa zona como dígitos e consequentemente obter um valor errado.

Para averiguar qual o fator de escalamento a utilizar para obter o melhor rácio de deteção das quantias sem prejudicar o desempenho da aplicação, foi conduzido um teste para a deteção das quantias em jogo, na amostra de 35 imagens com resolução 1158×826. Assim, o teste efetuou a deteção das quantias utilizando sequencialmente fatores de escalamento cada vez maiores, iniciando-se no fator 1.0 até ao fator 5.0, com intervalos de uma casa decimal. A Figura 5.1.4. mostra o gráfico da eficiência de deteção em função do fator de escalamento.

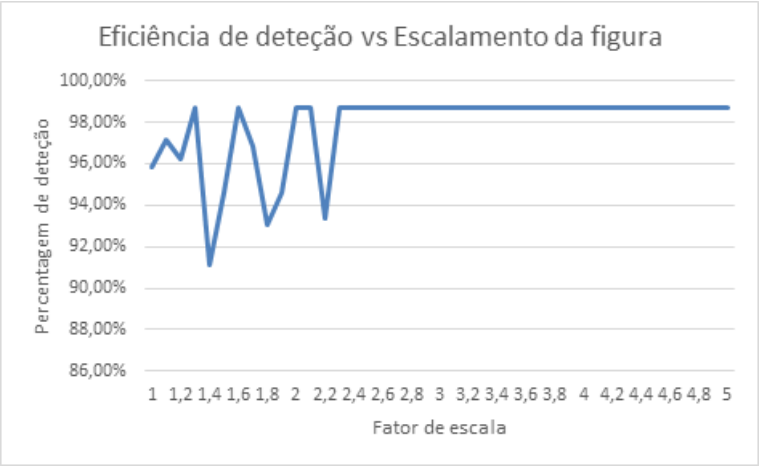


Figura 5.1.4: Eficiência de deteção em função do fator de escalamento

Pelo gráfico é possível observar que para escalamentos superiores a 2,3, a eficiência tende a estabilizar em próximo dos 99%.

As principais consequências da má identificação das quantias dos jogadores e das cartas de jogo são as de o agente tomar decisões tendo por base informações erradas, o que pode levar à perda de dinheiro, ou à desistência (efetuar uma jogada de *fold*) quando à partida poderia ser proveitosa outra decisão.

5.1.3 Movimentação do rato

Para testar a movimentação do rato e execução das jogadas foi criado um pequeno vídeo contendo a execução de duas jogadas na sala de póquer, uma executada pelo sistema desenvolvido e outra por um utilizador humano real. Esse vídeo foi apresentado a vinte e sete pessoas com o intuito de averiguar se o observador conseguia distinguir qual das duas jogadas correspondia ao jogador humano. Assim, os resultados obtidos foram que 4 em 27 dos inquiridos identificou a jogada do *bot* como sendo a jogada do utilizador humano. O que corresponde a cerca de 15% dos inquiridos persepcionou a jogada do sistema como sendo um jogador real. A Tabela 5.1.8 apresenta a estatística recolhida.

Tabela 5.1.8: Resultados no teste de identificação do *bot*

	Visualizou o vídeo uma única vez	**Necessitou de *replay***	**Total**
Nº de pessoas	12	15	27
Identificou o *bot*	8	15	23
Percentagem	66,7%	100,0%	85,2%

De salientar que alguns dos inquiridos necessitaram ver o vídeo pela segunda vez para efetuar a sua decisão e todos esses conseguiram identificar corretamente o *bot*.

Os resultados obtidos, numa primeira análise são insatisfatórios, no entanto, deve-se realçar o fato que os observadores humanos têm uma capacidade de identificação dos movimentos do ser humano muito mais apurada do que qualquer sistema de deteção automático de deteção de *bots*.

5.1.4 Experimentação do agente a jogar *online*

Ao longo do desenvolvimento do projeto por diversas vezes se testou o a agente a jogar o jogo de póquer *online* através da API, como forma de *debug* da aplicação. No entanto, um teste mais rigoroso foi realizado no final do projeto para validação do trabalho desenvolvido, que consistiu em colocar o agente a jogar póquer *online*, a dinheiro fictício, ininterruptamente durante uma hora a fim de averiguar se o *software* de póquer conseguia detetar a presença de um *bot*. Durante esse tempo o agente jogou sem falhas de maior, não tendo sido identificado pelo *software* de póquer. A única falha a apontar, foi que por quatro vezes o agente ficou sem fichas de jogo e não detetou essa situação a fim de as adicionar. Essa situação foi resolvida manualmente.

A situação de acrescentar fichas foi contemplada no desenvolvimento da API, embora a deteção da ausência de fichas e consequente adição apresentou limitações. Poderá ser algo a ser desenvolvido em trabalho futuro.

5.2 Considerações finais

Neste Capítulo foram descritos os testes de validação aplicados após e durante o desenvolvimento da API, bem como referidos e analisados os resultados obtidos. Foram identificados os problemas deparados e que referidas as principais causas dos mesmos.

Capítulo 6

Conclusões e trabalho futuro

Neste capítulo são apresentadas as principais conclusões, que têm como base o trabalho desenvolvido na construção de uma API que permite auxiliar e avaliar agentes de póquer em ambiente de jogo *online*. No final do capítulo são apresentados alguns aspetos que não foram contemplados para o projeto e que podem ter especial interesse para o desenvolvimento de um trabalho mais intenso no futuro, sobre o tema estudado.

O propósito deste trabalho de investigação foi o de desenvolver uma ferramenta que funcionasse como uma camada de perceção para o jogo de póquer, fiável, num ambiente controlado e *online*, bem como uma outra camada de execução no software de póquer *online* das jogadas decididas pelo agente. Um objetivo secundário foi o da criação de um agente de IA para utilizar a API e poder, com esta, jogar póquer *online*.

Os requisitos para a camada de perceção incluíam o reconhecimento das cartas de jogo, quantias em jogo e o seguimento completo do fluxo de jogo. Toda esta informação recolhida tem o objetivo de ser fornecida ao agente de póquer, que de acordo com a sua implementação decide qual o movimento a efetuar, o qual comunica com a camada de execução da API.

Neste documento foi introduzida, de um modo breve, o jogo de póquer, bem como alguma da literatura existente útil no desenvolvimento do trabalho, nomeadamente, conceitos de visão por computador e metodologias de construção de agentes de póquer. Esta revisão permitiu concluir que existiam muitos conceitos que serviram de apoio ao desenvolvimento e solução aos problemas que surgiram pelo caminho.

A primeira etapa a executar, e a mais importante, foi a de desenvolver a camada de reconhecimento dos elementos existentes na sala de póquer. Assim, o reconhecimento das cartas de jogo bem como a deteção da posição do *dealer* basearam-se em técnicas de visão por computador como *edge detection* e *template matching*, e o reconhecimento das quantias de jogo

baseou-se em *optical character recognition*. Os resultados obtidos nestes reconhecimentos são prometedores uma vez que rondam os 98,7% de sucessos, no entanto, alguns fatores podem perturbar estas identificações como a dimensão da janela da sala de póquer, a má definição das regiões de interesse pelo utilizador, e fatores externos à API como, possíveis atualizações do *software* de póquer. Os resultados apresentados referem-se aos resultados obtidos para resoluções de 1016×728 e de 1158×826.

Uma segunda etapa, foi a identificar com base na sequência de imagens obtidas da janela do *software* de póquer, o fluxo de jogo, nomeadamente os estados de jogo, as jogadas efetuadas pelos jogadores e valores das apostas. Para esta etapa, a principal metodologia baseou-se na contagem do número de cartas, valores das quantias dos jogadores e posição do *dealer*. Para este tópico não foram conduzidos testes de validação, também porque depende da correta identificação das cartas e das quantias em jogo. Algumas simplificações foram necessárias nesta etapa, nomeadamente, as ações dos jogadores, pois apesar de no *software* de póquer surgir por extenso a ação executada pelo jogador, esse texto é visível apenas por breves instantes, impossibilitando a sua captura na maioria das situações. Assim, por exemplo, quando um jogador faz *Fold* e não se consegue captar essa situação, então assume-se que o jogador efetuou um *Call*. O único problema dessa simplificação é o de assumir que o jogador ainda está em jogo quando na realidade já desistiu.

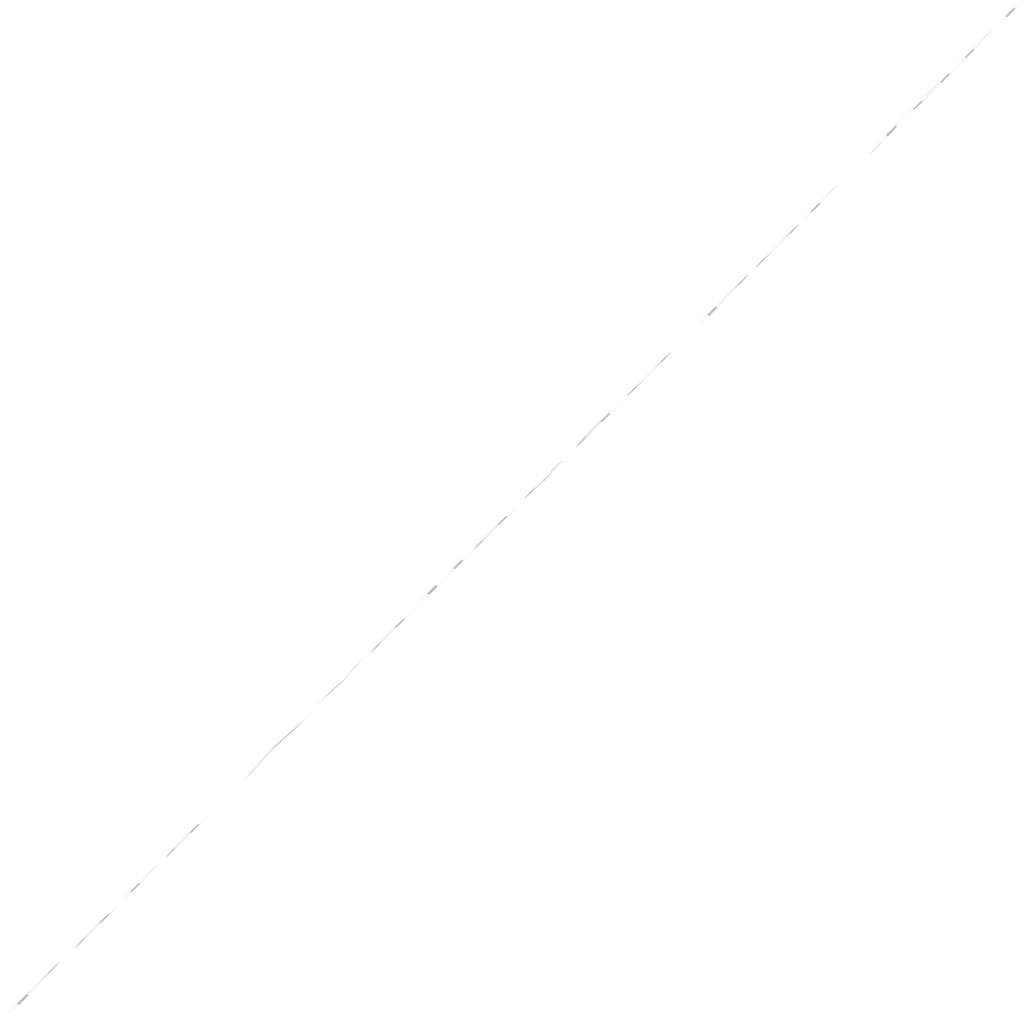
A última etapa da construção da API foi criar a camada de execução, para tal foram considerados alguns métodos para evitar a deteção automática pelo sistema de póquer da utilização de *bots*, para tal procedeu-se a um planeamento da movimentação do rato e clique dos botões que simulasse um jogador real, e a adaptação da API para que possa ser utilizada uma máquina virtual, impossibilitando assim ao *software* de póquer inspecionar detetar a execução do *bot* por pesquisa na lista de processos. Apesar de esta situação ter sido planeada, por impossibilidade de tempo e de uma máquina adequada, esta funcionalidade não foi testada.

Para verificar a operacionalidade da API construída foi criado um agente de póquer simples que utiliza todas as funcionalidades providenciadas por esta. Assim, constatou-se que o correto funcionamento de todas as funcionalidades.

De um modo global, penso que foram atingidos os objetivos principais propostos para este projeto e o resultado final, apesar de ser passível de melhorias, apresenta uma qualidade e desempenho satisfatórios.

Assim, algumas possíveis melhorias ao projeto desenvolvido passariam por tornar a API desenvolvida adaptável a mais sistemas de póquer *online*, para além do PokerStars. Providenciar um mecanismo de avaliação do agente de póquer, e não apenas registar a estatística de ganhos/perdas por jogo. Possibilitar o teste de agentes de forma intensiva a jogar com dinheiro real e analisar os resultados obtidos, uma vez que neste projeto não se testou o agente com dinheiro real. Estudar, mais aprofundadamente, as movimentações e cliques de

jogadores humanos no jogo de póquer *online* e adequar a API desenvolvida em conformidade. Melhorar a eficiência de deteção dos elementos do jogo, para diferentes resoluções da imagem de jogo, aplicando conjuntamente aos existentes outras metodologias de visão por computador e melhorar a funcionalidade de acrescentar fichas quando estas acabam. Finalmente, seria interessante também utilizar um agente mais complexo e construído com outras abordagens para testar a utilização da API nesses casos.



Referências

[1] T. Bakker, "Poker Bots – Part 3: Detection pt.1," [Online]. Available: http://bakker.cc/blog/2011/11/poker-bots-detection-stealth-p1/. [Acedido em 10 fevereiro 2014].

[2] P. GOLLE e N. DUCHENEAUT, "Preventing Bots from Playing Online Games," em *ACM Computers in Entertainment*, 2005.

[3] Shanky Technologies, "Holdem poker bot," [Online]. Available: http://www.bonusbots.com/holdempokerbot.htm. [Acedido em 10 fevereiro 2014].

[4] WINHOLDEM.NET, "WINHOLDEM.NET Simply the best way to play Texas Holdem," [Online]. Available: http://www.winholdem.net/. [Acedido em 10 fevereiro 2014].

[5] Openholdem, "openholdembot A programmable Texas Holdem style poker botting platform," [Online]. Available: https://code.google.com/p/openholdembot/. [Acedido em 10 fevereiro 2014].

[6] K.-T. Chen, A. Liao, H.-K. K. Pao e H.-H. Chu, "Game Bot Detection Based on Avatar Trajectory," *Entertainment Computing - ICEC 2008,* pp. 94-105, 2008.

[7] Q. Chen e G. Wang, "A class of Bézier-like curves," *Computer Aided Geometric Design 20,* pp. 29-39, 2003.

[8] O. Rdschel , "Rational motion design - a survey," *Computer-Aided Design,* vol. 30, pp. 169-178, 1998.

[9] B. Ravani e Q. J. Ge, "Geometric Construction of Bézier Motions," *Journal of Mechanical Design,* vol. 116, pp. 749-755, 1994.

[10] R. Szeliski, Computer Vision: Algorithms and Applications, Springer, 2010.

[11] M. A. Boden, "Mind As Machine: A History of Cognitive Science," em *Mind As Machine: A History of Cognitive Science*, Oxford, England, 2006.

[12] J. P. Lewis, "Fast Template Matching," em *Canadian Image Processing and Pattern Recognition Society*, Quebec City, 1995.

[13] OpenCV, "Template Matching," OpenCV, [Online]. Available: http://docs.opencv.org/doc/tutorials/imgproc/histograms/template_matching/template_matching.html. [Acedido em 10 fevereiro 2014].

[14] C. Zheng e R. Green, "Playing Card Recognition: Using Rotational Invariant Template Matching," em *Proceedings of Image and Vision Computing*, Hamilton, New Zealand, 2007.

[15] P. Martins, L. P. Reis e L. Teófilo, "Poker Vision: Playing Cards and Chips Identification Based on Image Processing," em *Springer-Verlag*, Berlin, 2011.

[16] P. a. S.K.Saxena, "Optical Character Recognition," em *VSRD International Journal of Computer*

Science & Information Technology, 2012.

[17] L. Eikvil, "Optical Character Recognition," em *citeseer. ist. psu. edu/142042. html*, 1993.

[18] LIACC, "About," 2014. [Online]. Available: http://www.liacc.up.pt/about. [Acedido em 9 fevereiro 2014].

[19] J. C. Correia, L. F. Teófilo, H. L. Cardoso e L. P. Reis, "A Poker Game Description Language," *IEEE/WIC/ACM International Conferences on Web Intelligence (WI) and Intelligent Agent Technology (IAT),* 2013.

[20] L. F. Teófilo, N. Passos, L. P. Reis e H. L. Cardoso, "Adapting Strategies to Opponent Models in Incomplete Information Games: A Reinforcement Learning Approach for Poker," em *Springer-Verlag*, Berlin Heidelberg, 2012.

[21] T. Schauenberg, "Opponent modelling and search in poker," em *Master thesis*, University of Alberta, 2006.

[22] G. Gerritsen, "Combining Monte-Carlo Tree Search and Opponent Modelling in Poker," em *Master Thesis*, Maastricht, 2010.

[23] D Billings, A Davidson, J Schaeffer, and D Szafron, "The challenge of poker," em *Artificial*, 2002.

[24] R. S. S. Barto e A. G., Reinforcement Learning: An Introduction, A Bradford Book, 1998.

[25] J. P. Marques, "Reinforcement Learning applied to the game of Poker," Faculdade de Engenharia da Universidade do Porto, Porto, 2013.

[26] N. Passos, "Poker Learner : Reinforcement Learning Applied to Texas Hold ' em Poker," em *Master thesis*, Faculdade de Engenharia da Universidade do Porto, 2011.

[27] PokerStars.net, "PokerStars," PokerStars.net, [Online]. Available: http://www.pokerstars.net/. [Acedido em 11 fevereiro 2014].

[28] R. J. e W. I., "Computer poker: A review," em *Artificial intelligence*, 2011.