

TCG ID

Hemanoel Brito de França
Universidade de Brasília-Faculdade do Gama
Brasília, Brasil
Email: hemanoel13@gmail.com

Lucas Xavier de Moura
Universidade de Brasília-Faculdade do Gama
Brasília, Brasil
Email: lucasxmoura@gmail.com

I. RESUMO

O presente trabalho teve como objetivo apresentar uma proposta de um sistema de reconhecimento de TCG's, utilizando a plataforma de prototipagem eletrônica Raspberry Pi. O protótipo desenvolvido utiliza uma placa "Raspberry Pi" e uma Pi câmera para realizar à captação da imagem. Por intermédio da aplicação, desenvolvida através da linguagem C/C++, o usuário irá apontar a câmera para a carta que deseja executar a identificação, capturando a imagem, que é enviada a Raspberry Pi, então a imagem passa por um pré-processamento, para assim executar a identificação.

II. INTRODUÇÃO

Poder computacional é um recurso bem difundido no mundo atual, onde milhares de celulares, computadores e outros sistemas embarcados fazem parte das nossas vidas, exemplos desses sistemas embarcados seriam o monitoramento de vídeo ou assistência ao dirigir de carros modernos. Além desses sistemas, aplicativos mais sofisticados sendo desenvolvidos estão agora usando tecnologias de visão computacional e análise de imagem, e ainda estamos na vanguarda do desenvolvimento de novos sistemas informatizados capazes de usar visão. Mas com o advento de poderosos e acessíveis dispositivos de computação e sensores visuais, como a Raspberry PI, criar sofisticados aplicativos de imagem em lugares inesperados tem se tornado uma tendência. Uma infinidade de ferramentas de software e bibliotecas que manipulam imagens e vídeos estão disponíveis para ela. Em resumo projeto busca em sua essência utilizar a visão computacional de modo a promover praticidade, para isso fez-se uso principalmente de duas ferramentas, o OpenCV e o Tesseract.

O OpenCV (Open Source Computer Vision Library) é uma biblioteca de software de visão computacional e aprendizado de máquina de código aberto, e o Tesseract que é uma ferramenta de reconhecimento óptico de caracteres(OCR). Em outras palavras, os sistemas que realizam OCR transformam uma imagem que pode conter texto impresso ou manuscrito, em um texto digital. O OCR como um processo geralmente consiste em vários subprocessos para executar com a maior precisão possível. Foi criada na HP, mas hoje é um código aberto mantido pelo Google. No projeto, para identificar as cartas, foi optado pelo o Tesseract pois ele vai ser o responsável pela conversão dos "nomes" das cartas em texto, portanto vai permitir a transformação do nome delas em dados que serão utilizados com o database.

Os TCGs "Trading card games", atualmente são compostos de milhares de cartas, que na maioria das vezes tem inventários feitos manualmente demandando uma quantidade

de tempo ou pessoas consideravelmente alto, necessitando assim de um sistema de aprimorado gerenciamento de estoque. A ideia do projeto surge nesse quesito, onde ele permitirá a identificação das cartas com precisão, reduzindo o tempo de ciclo de inventário. Portanto esse projeto é de um sistema de reconhecimento de cartas para facilitar o gerenciamento de estoque com grande coleção de cartas, de modo a automatizar esse serviço.

III. DESENVOLVIMENTO

A. Descrição de Hardware

Estrutura física do projeto consiste de duas principais Peças, a Raspberry 3B, onde é feito todo o processamento, e a Pi câmera utilizada para adquirir os samples das cartas.



Fig. 1. RaspberryPI 3 com uma Pi câmera acoplada

1) Estrutura: Foi montada uma estrutura para auxiliar na obtenção das imagens, e assim, facilitar o OCR.



Fig. 2. Suporte

Componente de hardware	Especificações	Custos
Raspberry Pi 3B	Quad Core 1.2GHz 64bit CPU, 1GB RAM, LAN and Bluetooth, 40-pin extended GPIO, 4 USB 2 ports, HDMI, Micro SD port.	R\$332,00
Pi camera	5 Megapixels	R\$99,00
Monitor	1280X720	R\$0,00
Cartão SD 1	16(GB)	R\$31,00
Material para estrutura	Estrutura de Papelão	R\$0,00
Montagem	Foi feito um esquema onde montou-se uma estrutura que serveria de apoio fixo a Pi câmera e que fosse possível colocar a carta em uma base para poder ter sua imagem captada pela Pi câmera	R\$0,00
Total		R\$462,00

TABLE I. BILL OF MATERIALS(BOM)

A estrutura foi montada em papelão e foi passado um fio metálico por dentro para ajudar na sustentação e auxiliar no foco da câmera, que é fixo por padrão.

B. Descrição de Software

1) *OpenCV/Tesseract-ocr*: O projeto tem uma questão importante que é a acurácia das identificações, assim, para diminuir a porcentagem de erros e evitar que a precisão do Tesseract caia, é realizado um pré-processamento nas imagens, de modo a aumentar as chances que o OCR interprete adequadamente as palavras. Em um primeiro momento foi utilizado um modificador de região de interesse (ROI) móvel, para facilitar a análise da OCR com múltiplas câmeras.

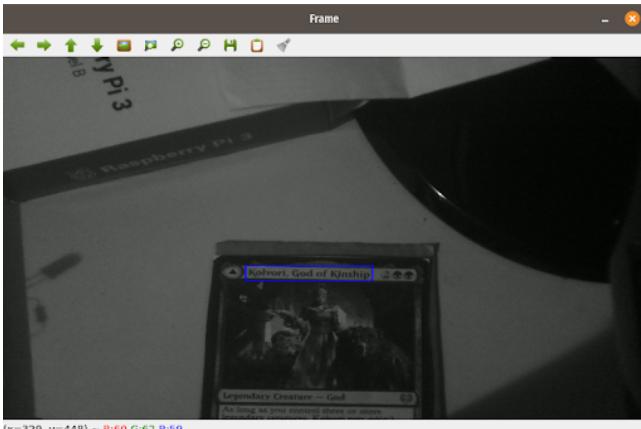


Fig. 3. Retângulo na região de interesse

Mas na etapa final do projeto foi construído um suporte para a câmera, relativamente maleável, que facilitou a configuração do foco. Na base, abaixo da câmera, coloca-se a carta e assim o algoritmo seleciona uma região de interesse automaticamente.

O processo de reconhecimento das imagens começa com o OpenCV, onde é realizado um processamento primeiro convertendo as imagens em escala de cinza.



Fig. 4. Sample

Depois de obtido a amostra desejada que contém o nome da carta, esse arquivo é passado como parâmetro para a função que realizará o reconhecimento ótico de caracteres. Essa função utiliza a API do Tesseract que faz várias operações de processamento de imagem internamente usando a biblioteca Leptonica antes de fazer o OCR. Geralmente faz um processamento muito bom, mas inevitavelmente haverá casos em que não é bom o suficiente, o que pode resultar em uma redução significativa na precisão. Portanto, seguindo a recomendação da documentação oficial, é realizado o reescalonamento da imagem, já que o Tesseract funciona melhor em imagens com DPI de pelo menos 300 dpi[4], por fim também é realizado threshold, pois a API reconhece com mais facilidade fundos brancos e texto preto.

Elvish Piper

Fig. 5. Sample

Após essas etapas de preprocessamento, a API do Tesseract, realiza sua função, que irá entregar o nome da carta como parâmetro para outra função, sendo esta última responsável por realizar uma busca no banco de dados e comparar o nome recebido pelo o OCR com os nomes contidos no database, se encontrado é adicionado no número total de cartas. Os resultados processados pelo OCR podem possuir pequenos erros como espaços a mais e caracteres não desejados, sendo necessário uma forma de corrigir ou rejeitar o resultado. Assim foi desenvolvido uma ferramenta adicional que, no momento da procura da carta no database, faz uma comparação com o nome da carta obtido pela API e os nomes das cartas no database, só assim então adicionando-a ao inventário.

2) *UI*: Com relação a UI, utilizou-se a plataforma Glade, de modo a simplificar a criação da estrutura da UI usando a biblioteca GTK. A plataforma Glade possibilita ver a UI enquanto ela é construída, oferece acesso dinâmico à documentação e acesso as características de cada elemento da UI. Sendo, também, extremamente fácil de carregar no programa, pois o GTK já possui um *loader* pronto para ela.

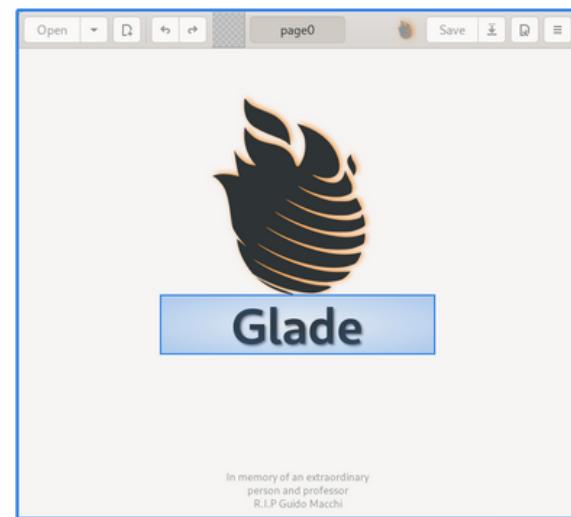


Fig. 6. Plataforma Glade

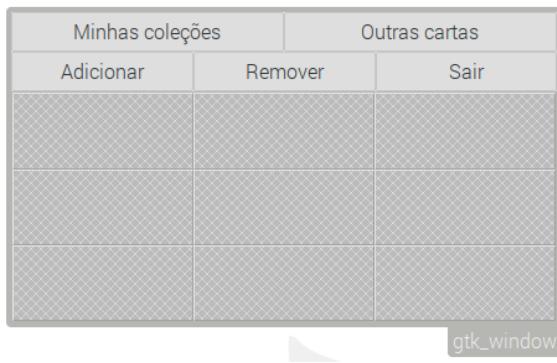


Fig. 7. Exemplo de estrutura feita na Glade

O funcionamento da UI pode ser descrito em alguns diagramas. Primeiramente, o diagrama geral Fig.8. Este mostra as funções básicas da tela de início do programa e um pouco da adição de cartas.

O diagrama de seleção de uma coleção, que mostra o funcionamento da UI ao tentar adicionar ou remover uma carta após selecionar uma coleção Fig.9.

Por fim, há o diagrama que mostra as formas de adicionar cartas às coleções Fig.10.

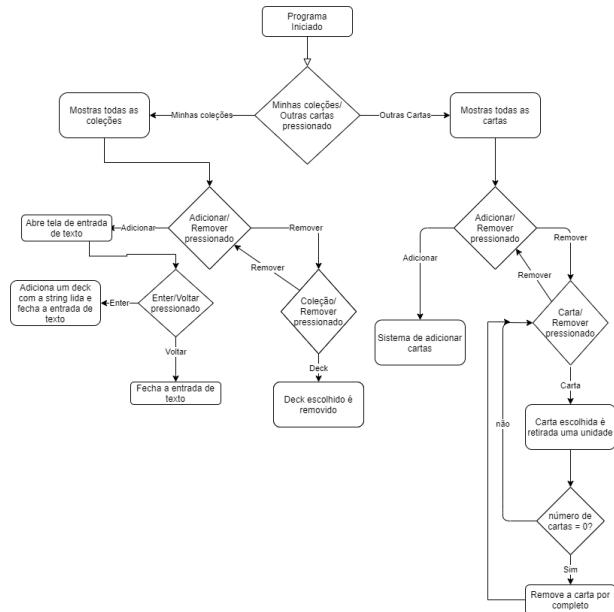


Fig. 8. Diagrama geral

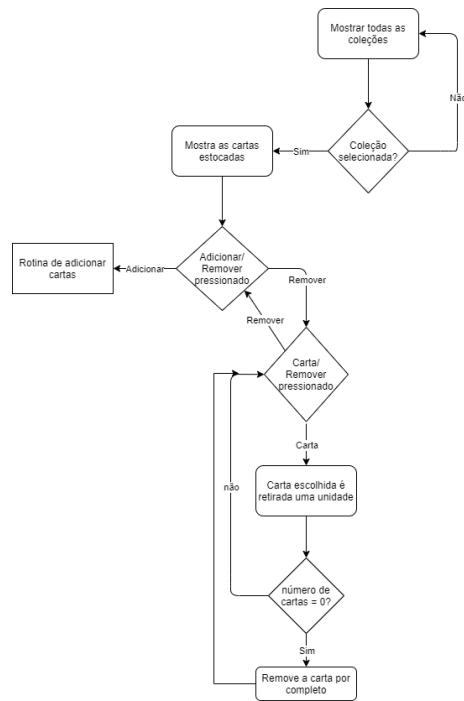


Fig. 9. Adição de cartas em uma coleção

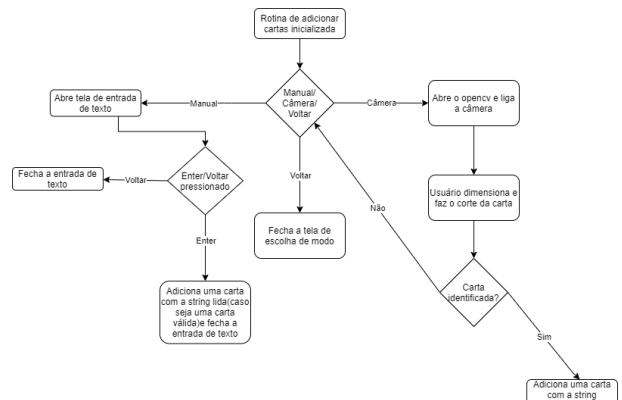


Fig. 10. Adição de cartas detalhado

C. Código

O código do projeto é bem extenso, então ele foi separado em alguns pseudocódigos que explicam grande parte de seu conteúdo.

Primeiramente temos os códigos de botões simples, como os botões "Minhas coleções" e "Outras cartas", que simplesmente modificam uma flag (Fig.11). O botão remover também modifica uma flag, mas troca a cor do botão(Fig.12). Essas flags são usadas, por exemplo, no botão "Adicionar", que serve para adicionar tanto para coleções quanto para cartas dependendo das flags (Fig.13).

Há códigos mais complexos que usam da manipulação de arquivos do linux em conjunto com o C++ para implementar suas funções, como quando o usuário clica em uma coleção para abri-la. As coleções, os nomes de cartas e os números de cartas são estocadas em simples arquivos de texto, quando o usuário clica na coleção, caso a flag de remover esteja ativa, a coleção é retirada do arquivo de coleções e seus arquivos relacionados são excluídos (cartas e números de cartas). Caso a flag de remover não esteja ativa, a coleção é aberta e uma flag é acionada(Fig.14).

Outro bom exemplo de como os arquivos .txt são utilizados é a entrada de texto. Esta funciona tanto para adicionar cartas quanto coleções. No caso de coleções, o programa verifica se essa coleção já existe e usa comandos para adicioná-la caso não exista. No caso da adição de cartas, ela procura a carta no arquivo contendo todas as cartas e, caso seja encontrada, a carta é adicionada ao final da lista conjuntamente com sua quantidade ao final de outra lista (Fig.15).

A atualização das listas de coleções e de cartas é feita pela mesma função, sendo que sua funcionalidade é definida pelas flags selecionadas. Para as coleções, é aberto um arquivo com os nomes das coleções e é criado um botão a cada nome no arquivo. No caso das cartas, um arquivo com as cartas e outro com os números de cartas é aberto("nome_da_colecao.txt" e "qnome_da_colecao.txt") e a cada linha nos arquivos é criado um botão com o número de cartas concatenado com a quantidade de cada carta (Fig.16).

Após adicionadas, as cartas podem ter sua quantidade incrementada ou decrementada. Caso a flag de remover esteja ativa, a cada click do usuário, a carta receberá o decremento de 1 na sua quantidade, caso atinja o valor zero, ela será excluída. Caso a flag de remover não esteja ativa, a cada click, o valor será incrementado em 1 (Fig.17 e Fig.18).

Por fim temos a lógica de adição de cartas pela câmera. A imagem da carta é obtida a partir do "raspistill", ela é cortada na região de interesse, transformada na escala de cinza e transformada em preto e branco, com a função "threshold" para evidenciar os caracteres do nome da carta. Esta imagem pré-processada é enviada à API do OCR, que devolve o possível nome da carta. Esse nome é comparado com os nomes no arquivo contendo todos os nomes. Para isso, os nomes no arquivo foram ordenados do maior para o menor, em quantidade de caracteres. Na comparação, os nomes que são maiores que o nome obtido no OCR são desconsiderados, a string obtida no OCR é comparado caractere por caractere e, caso tenha mais que 80% de compatibilidade, a carta é adicionada. Isso permitiu cobrir mais casos, inclusive aqueles

que mudam uma letra ou outra da palavra e o caso em que são adicionados caracteres no final da palavra, aumentando, assim, a taxa de acerto(Fig.19).

```
1 #Para os botoes minhas coleções/outras cartas
2 set_flag(índice_do_botao) # 1 = "minhas
    coleções" e 3 = "outras cartas"
3 atualiza_a_lista()
4
```

Fig. 11. Pseudocódigo dos botões "Minhas coleções" e "Outras cartas"

```
1 #Para o botao remove
2 if flag == 1:
3     flag = 0
4     cor = "azul"
5
6 else if flag == 0:
7     flag = 1
8     cor = "vermelho"
```

Fig. 12. Pseudocódigo do botão "Remover"

```
1 #Para o botao adicionar.
2 if flag == "minhas coleções" OR flag == "Outras
    cartas" :# Usado para adicionar coleções ou cartas
    avulsas
3     mostra_entrada_de_texto()
4
5 else if flag == "Coleção_n": # somente é ativado
    dentro de uma coleção para adicionar cartas
6     mostra_seleção_de_opções_de_entrada()# as opções
    são a câmera ou a entrada de texto para as cartas
```

Fig. 13. Pseudocódigo do botão "Adicionar"

```
1 #Ao clicar em uma coleção.
2 file = nome_da_colecao.txt #Abre o arquivo .txt com
    o nome da coleção
3
4 if botao_remover_ativado == 1: #botao remover está
    ativado
5     system("cp Decks.txt aux2.txt"); # copia o
    arquivo com os decks para um arquivo auxiliar
6
7     while(1): #Procura a linha que o widget está
8         temp = gtk_grid_get_child_at(GTK_GRID(dc_grid
    ), 1, n);
9         if temp == nome_da_colecao:
10             break
11         n++
12
13     system("sed n 'd' aux.txt > Decks.txt") # Remove
    a linha encontrada e joga no arquivo das coleções
14     system("rm file") # remove o arquivo contendo as
    cartas
15     system("rm qfile") # remove o arquivo contendo o
    número de cartas
16     atualiza_a_lista()
17
18 else:
19     colecao_atual = nome_da_colecao # O nome da
    coleção vira a coleção que sera aberta
20     atualiza_lista()
21     set_flag(colecao_aberta)#flag que indica que a
    coleção foi aberta, a mesma flag usada nos outros
    botões
```

Fig. 14. Pseudocódigo das coleções

```

1 #Botao enter. Essa rotina serve tanto para adicionar
  cartas quanto colecoes, usando a entrada de texto
2
3 if flag == 1:# Adicionar deck
4     file = open(nome_da_colecao.txt) # abre o arquivo
      com nome da colecao
5
6     if file == NULL: #se o arquivo não existe, vamos
      adicionar
7         system("echo nome_da_colecao >> Decks.txt")
      # Adiciona o nome da colecao no final da lista
8         system("touch nome_da_colecao.txt") #
      Adiciona o arquivo com os nomes de cartas da coleção
9         system("touch nome_da_colecao.txt")# Adiciona
      o arquivo com os numeros de cartas da coleção
10
11 if flag == 2: # Adiciona uma carta a partir da
      entrada de texto
12     if(procura_carta(nome_da_carta)):# Procura a
      carta no arquivo de cartas
13         system("echo nome_da_carta >> nome_da_colecao
      .txt") # Adiciona a carta no final da colecao
14         system("echo 1 >> qnome_da_colecao.txt") #
      Adiciona 1 ao numero de cartas
15
16 if flag == 2:
17     if (procura_carta(nome_da_carta)):
18         system("echo nome_da_carta >> AllCards.txt"
      ) # Adiciona a carta no final da colecao de outras
      cartas
19         system("echo 1 >> qAllCards.txt") # Adiciona
      1 ao numero de cartas

```

Fig. 15. Pseudocódigo da entrada de texto manual

```

1 #rotina de atualizar listas, tanto de cartas quanto a
  de colecoes
2
3 for i in lista_de_cartas_e_colecoes: # iteramos e
  destruimos todos os itens da lista
4     destroy(i)
5
6 if(flag == 1 ): # se as coleções estiverem escolhidas
7     file = open("Decks.txt") # abre o arquivo de
  colecoes
8
9     for colecao in file:
10        adiciona_linha_na_lista()
11        botao = cria_botao(colecao)
12        adiciona_botao_na_lista(botao)
13        conecta_botao(funcao_do_botao_de_colecao)
14
15 if(flag == 2 || flag==3): # Caso uma colecao esteja
  selecionada ou outras cartas
16     file1 = open("nome_da_colecao.txt") # cartas da
  colecao
17     file2 = open("qnome_da_colecao.txt") # numeros de
  cartas
18     for carta,numero_de_cartas in file1, file2:
19         adiciona_linha_na_lista()
20         botao = cria_botao(numero_de_cartas+"x"+carta
      )
21         adiciona_botao_na_lista(botao)
22         conecta_botao(funcao_da_carta)

```

Fig. 16. Pseudocódigo da atualização da lista

```

1 #funçao conectada as cartas e colecoes
2
3 if (rem_flag == 1):# Modo de remover cartas
4     n=0
5     while(1): # procura a linha da carta
6         temp = gtk_grid_get_child_at(GTK_GRID(dc_grid
      ), 1, n)
7         if temp == botao_da_carta:
8             break
9         n++
10
11    system("cp qcolecao_atual.txt > aux.txt") #copia
  os numeros de cartas para um arquivo auxiliar
12    file = open("aux.txt")
13
14    row == 0
15    for line in file:
16        if row == n:
17            val = int(line) # Pegamos o valor da
  carta
18            val--           # Diminuimos o valor em 1
19            if(val==0):   # se for zero retiramos a
  carta
20                system("cp actual_deck.txt > aux2.txt
      ") # copia as cartas para um arquivo auxiliar
21                system("sed 'nd' aux2.txt >
      actual_deck.txt") # remove a linha da carta
22                system("sed 'nd' aux.txt >
      qactual_deck.txt") # remove a linha do numero
23                else:
24                    system("echo val >> qcolecao_atual.txt")
      #adicionamos o valor novo a colecao
25                    row++
26
27 else: # Modo de adicionar
28
29     while (1): # procura a linha da carta
30         temp = gtk_grid_get_child_at(GTK_GRID(dc_grid
      ), 1, n)
31         if temp == botao_da_carta:
32             break

```

Fig. 17. Pseudocódigo da interação com as cartas parte 1

```

33        n++
34
35        system("cp qcolecao_atual.txt > aux.txt")
36        row == 0
37
38        for line in file:
39            if row == n:
40                val = int(line) # Pegamos o valor da
  carta
41                val++ # Diminuimos o valor em 1
42                system("echo val >> qcolecao_atual.txt"
      ) # adicionamos o valor novo a colecao
43                row++
44
45 atualiza_a_lista()

```

Fig. 18. Pseudocódigo da interação com as cartas parte 2

```

1 #Rotina para adicionar cartas pela camera(engloba
  varias funções)
2
3 #Obtenção e pre-processamento da imagem
4 system("raspistill -o image.jpg -br 50 -sh 100 -sa -
  100 -ex auto -t 700 -q100")# tira uma foto da carta
5 frame = imread("image.jpg")# abre o arquivo da carta
6 cvtColor(frame(cut), grayscale, COLOR_RGB2GRAY) #
  corta a carta na região de interesse e transforma na
  escala de cinza
7 threshold(grayscale, grayscale, 200, 255,
  THRESH_BINARY + THRESH_OTSU) # transforma a carta em
  preto e branco
8 #Essa função é usada para evidenciar as letras
9
10 #Aplicação do OCR
11 image = pixRead("image.jpg") #carregamos a imagem
12 api_set_image(image)# colocamos a imagem na API
13 ocrResult = api_GetUTF8Text() # Pegamos do OCR
14
15 #Verificamos se a carta existe
16 file = open("names2.txt") # abrimos o arquivo com
  todos os nomes de cartas
17 for carta in file:
18     if comparech(carta, ocrResult) > 80:# compara a
      similaridade entre as cartas
19         nome2 = carta # se a similaridade for >80% a
      carta foi encontrada
20         break

```

Fig. 19. Pseudocódigo da adição de cartas pela câmera

IV. RESULTADOS

A. Resultados Teóricos Esperados

Para o obtenção de resultados o projeto utiliza de etapas que são:Pré-processamento da imagem, depois a determinação da localização do Texto, seguindo da segmentação da carta, realizando o pós-processamento para enfim fazer o reconhecimento do nome da carta.



Fig. 20. Sample de uma imagem de cartal ideal



Fig. 21. Token criado a partir nome da carta

Burgeoning

Fig. 22. Output no terminal

B. Resultados Reais

Nos primeiros testes ,foram testadas em um celular várias resoluções mostradas na Tabela 1 e na Tabela 2.

Resolução	176x144	320x180	352x288	640x480	720x480
Resultado	Ruim	Ruim	Ruim	Ruim	Ruim/Médio

TABLE II.

Resolução	800x600	1024x768
Resultado	Médio	Bom

TABLE III.

É possível que as resoluções mais baixas possam ser usadas com o uso de processamentos na imagem antes de ser usada pela api, apesar do resultado desfavorável. Por fim, a piCâmera, que foi escolhida para o projeto, possui resolução alta (5MP).

Foram testadas diversas cartas mostradas nas figuras abaixo. Elas são fotografadas, transformadas para a escala de cinza e passam por um tratamento chamado *thresholding* que transforma a imagem em preto e branco, destacando as letras.

Para os três primeiros casos, as cartas foram identificadas com sucesso e foram adicionadas à coleção (Fig.23, Fig.24, Fig.25, Fig.26, Fig.27, Fig.28, Fig.29, Fig.30 e Fig.31).

No quarto caso (Fig.32) foi simulado um ambiente com pouca iluminação e percebe-se que a qualidade da *sample* caiu muito (Fig.33), as letras se tocam e fica mais difícil para a API fazer o reconhecimento. Assim a carta não foi identificada e nada mudou na lista da coleção (a última carta que aparece é da situação anterior)(Fig.34).

Por fim, no quinto caso, o texto da carta foi identificado erroneamente pela API, mas a ferramenta de procura conseguiu achar uma carta em que as letras correspondiam em mais de 80%(Fig.35, Fig.36 e Fig.37).



Fig. 23. Carta do caso 1

Shineshadow Snarl

Fig. 24. *Sample* do caso 1

```
ui Estimating resolution as 350
OCR->Mountain
Minhas coleções Outras cartas
Adicionar Remover Sair
1xMountain
1xShineshadow Snarl
Carta localizada!---->Mountain
Warning: Invalid resolution 0 dpi. Using 70 instead.
Estimating resolution as 498
Estimating resolution as 452
OCR->Shineshadow Snarl
nome2 = ocrResult
nome =>Shineshadow Snarl
Carta localizada!---->Shineshadow Snarl
```

Fig. 25. Procura e adição da carta no caso 1



Fig. 26. Carta do caso 2

Mountain

Fig. 27. *Sample* do caso 2

```
ui Empty page!!
OCR->
Minhas coleções Outras cartas
Adicionar Remover Sair
1xShineshadow Snarl
Carta nao localizada!
Warning: Invalid resolution 0 dpi. Using 70 instead.
Estimating resolution as 347
Estimating resolution as 350
OCR->Mountain
nome2 = ocrResult
nome =>Mountain
Carta localizada!---->Mountain
```

Fig. 28. Procura e adição da carta no caso 2



Fig. 29. Carta do caso 3

Chaos Maw

Fig. 30. *Sample* do caso 3

```
ui Estimating resolution as 439
Empty page!!
Estimating resolution as 439
Empty page!!
OCR->
nome2 = ocrResult
nome =>
Carta nao localizada!
Warning: Invalid resolution 0 dpi. Using 70 instead.
Estimating resolution as 422
Estimating resolution as 425
OCR->Chaos Maw
nome2 = ocrResult
nome =>Chaos Maw
Carta localizada!---->Chaos Maw
```

Fig. 31. Procura e adição da carta no caso 3

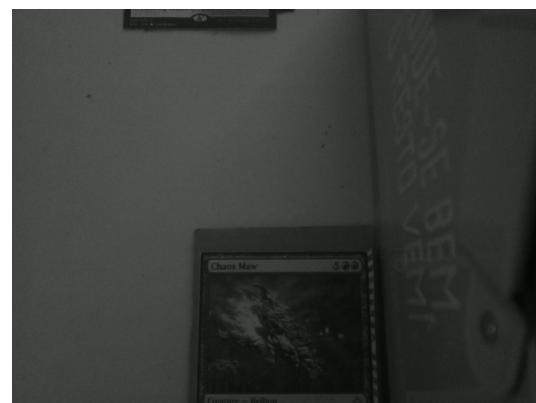


Fig. 32. Carta do caso 4

Chaos Maw

Fig. 33. *Sample* do caso 4

```
ui Estimating Resolution as 425
OCR->Chaos Maw
nome2 = ocrResult
nome =>Chaos Maw
Carta localizada!---->Chaos Maw
Warning: Invalid resolution 0 dpi. Using 70 instead.
Estimating resolution as 338
Estimating resolution as 453
Empty page!!
Estimating resolution as 453
Empty page!!
OCR->
nome2 = ocrResult
nome =>
Carta nao localizada!
```

Fig. 34. Procura e adição da carta no caso 4



Fig. 35. Carta do caso 5



Fig. 36. Sample do caso 5

```

ui
  nome2 = ocrResult
  nome =>Shineshadow Snarl
  Carta localizada:---->Shineshadow Snarl
  1
  cmd:l1sed '3d' aux2.txt > col7.txt
  Warning: Invalid resolution 0 dpi. Using 70 instead.
  Estimating resolution as 430
  Estimating resolution as 431
  OCR->Gaze of Granite QQ
  nome2 = ocrResult
  nome =>Gaze of Granite QQ
  Carta localizada:---->Gaze of Granite
  1

```

Fig. 37. Procura e adição da carta no caso 5

mesmo com os processamentos das imagens, ainda requer uma boa foto para funcionar adequadamente. Durante o trabalho foi realizado diferentes configurações da Pi Câmera para obter o melhor angulo, iluminação e outros aspectos físicos que aprimorariam os resultados. Os passos futuros seriam principalmente na UI, onde seria criado mais funcionalidades e também seria refinada, além de suporte a outras câmeras, outros tipos de TCG.

REFERENCES

- [1] https://scryfall.com/?utm_source=mci
- [2] <https://www.mtjson.com>
- [3] <https://acervodigital.ufpr.br/bitstream/handle/1884/44742/Kamke.pdf?sequence=2isAll>
- [4] <https://tesseract-ocr.github.io/tessdoc/ImproveQuality.html>
- [5] <https://nanonets.com/blog/ocr-with-tesseract/>

V. CONCLUSÃO

Atualmente o inventário realizado pelas pessoas que possuem milhares de cartas é feito em planilhas ou cadernos. Com o objetivo principal de automatizar esse trabalho, foi pensado a criação uma ferramenta que facilitasse gerenciar cartas, realizar pesquisas e criar coleções, a partir da identificação física da carta. Para isso era necessário a criação de um aplicativo com GUI, que permitisse gerenciar habilmente a coleção, além do desenvolvimento de uma aplicação de visão computacional, onde a OCR era seu principal componente, que seria responsável por fazer a identificação das cartas, transformando o nome delas em texto digital.

Os resultados no início possuíam uma taxa de acerto impreciso, variando bastante com a carta utilizada, então foram realizadas etapas de pré-processamento através do OpenCV , onde foi selecionado a região de interesse(ROI) para ser cortada na imagem, e criar uma amostra que passa por outras etapas de pré-processamentos usando o OpenCV. Isso foi necessário devido outros problemas obtidos como ruídos, as fontes utilizadas nas cartas e contraste baixo. A API do OCR, o Tesseract, também foi utilizado nesse aspecto de processamento, utilizando o algorítimo próprio para fazer o processamento da imagem. Os resultados depois dessas etapas pareceram promissores mas ainda encontramos problemas, talvez devido a impressão das cartas ou outros aspectos desconhecidos. Então, para resolver isso, foi feito um algoritmo que compara o nome obtido pela OCR e o database das cartas, se houvesse uma porcentagem alta de confiança, essa que o próprio OCR calcula, ocorre a comparação com o database, se encontrado, a carta é adicionada na coleção. Os desafios se concentram bastante na foto obtida das cartas, a OCR,

APPENDIX

C:\Users\lxm10\Desktop\SOE\codigo_final.cpp

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <gtk/gtk.h>
5 //#include <gtk/gtkx.h>
6 #include <tesseract/baseapi.h>
7 #include <leptonica/allheaders.h>
8 #include "opencv2/opencv.hpp"
9 #include "opencv2/core.hpp"
10 #include <pthread.h>
11 #include <cstddef>
12
13
14 //window
15 GtkWidget* window;
16
17 //containers
18 GtkWidget* main_frame;
19 GtkWidget* first_btns;
20 GtkWidget* sec_btns;
21 GtkWidget* dc_win;
22 GtkWidget* dc_viewp;
23 GtkWidget* dc_grid;
24 GtkWidget* text_entry;
25 GtkWidget* method_selector;
26
27
28
29 //buttons
30 GtkWidget* m_decks;
31 GtkWidget* m_cards;
32 GtkWidget* add_btn;
33 GtkWidget* rem_btn;
34 GtkWidget* exit_btn;
35 GtkWidget* back_btn;
36 GtkWidget* enter_btn;
37 GtkWidget* camera_btn;
38 GtkWidget* manual_btn;
39 GtkWidget* back_btn2;
40
41 //text entry
42 GtkWidget* get_usr_entry;
43
44 //rows of cards and decks
45 GtkWidget* label[1000];
46 GtkWidget* button[1000];
47 char actual_deck[1024];
48
49
50 //builder
51 GtkBuilder* builder;
52
```

```
53 //aux
54 char tmp[1024];
55 char tmp2[1024];
56 int row = 0;
57 int cd_flag = 0;
58 int rem_flag = 0;
59 int flag_card = -1;
60
61 void on_m_decks_clicked(GtkButton*);
62 void on_m_cards_clicked(GtkButton*);
63 void on_add_btn_clicked(GtkButton*);
64 void on_rem_btn_clicked(GtkButton*);
65 void on_exit_btn_clicked(GtkButton*);
66 void on_back_btn_clicked(GtkButton*);
67 void on_enter_btn_clicked(GtkButton*);
68 void on_camera_btn_clicked(GtkButton*);
69 void on_manual_btn_clicked(GtkButton*);
70 void on_back_btn2_clicked(GtkButton*);
71 void on_row(GtkButton*);
72 void on_row2(GtkButton*);
73 void delete_rows();
74 void update_list();
75 int comparech(const char* str1, const char* str2);
76
77
78 //ocr
79 using namespace std;
80 using namespace cv;
81 char* nome2;
82 char tesse();
83 int quadro();
84 int cards();
85 int tessete(char* name);
86
87
88 int main(int argc, char* argv[])
89 {
90     system("export LC_ALL=C");
91     gtk_init(&argc, &argv);
92
93
94     builder = gtk_builder_new_from_file("Ui_teste.glade");
95     //load windows
96     window = GTK_WIDGET(gtk_builder_get_object(builder, "gtk_window"));
97     text_entry = GTK_WIDGET(gtk_builder_get_object(builder, "text_entry"));
98     method_selector = GTK_WIDGET(gtk_builder_get_object(builder,
99                                     "method_selector"));
100    g_signal_connect(window, "destroy", G_CALLBACK(gtk_main_quit), NULL);
101
102
103    gtk_builder_connect_signals(builder, NULL);
```

Fig. 38. Código

```
104 //load buttons
105 m_decks = GTK_WIDGET(gtk_builder_get_object(builder, "m_decks"));
106 m_cards = GTK_WIDGET(gtk_builder_get_object(builder, "m_cards"));
107 add_btn = GTK_WIDGET(gtk_builder_get_object(builder, "add_btn"));
108 rem_btn = GTK_WIDGET(gtk_builder_get_object(builder, "rem_btn"));
109 back_btn = GTK_WIDGET(gtk_builder_get_object(builder, "back_btn"));
110 enter_btn = GTK_WIDGET(gtk_builder_get_object(builder, "enter_btn"));
111 camera_btn = GTK_WIDGET(gtk_builder_get_object(builder, "camera_btn"));
112 manual_btn = GTK_WIDGET(gtk_builder_get_object(builder, "manual_btn"));
113 back_btn2 = GTK_WIDGET(gtk_builder_get_object(builder, "back_btn2"));
114
115 //usr entry
116 get_usr_entry = GTK_WIDGET(gtk_builder_get_object(builder,
117     "get_usr_entry")); ↗
118
119 //load list
120 dc_win = GTK_WIDGET(gtk_builder_get_object(builder, "dc_win"));
121 dc_viewp = GTK_WIDGET(gtk_builder_get_object(builder, "dc_viewp"));
122 dc_grid = GTK_WIDGET(gtk_builder_get_object(builder, "dc_grid"));
123
124
125 //connect signals
126 g_signal_connect(m_decks, "clicked", G_CALLBACK(on_m_decks_clicked), NULL);
127 g_signal_connect(m_cards, "clicked", G_CALLBACK(on_m_cards_clicked), NULL);
128 g_signal_connect(add_btn, "clicked", G_CALLBACK(on_add_btn_clicked), NULL);
129 g_signal_connect(rem_btn, "clicked", G_CALLBACK(on_rem_btn_clicked), NULL);
130 g_signal_connect(exit_btn, "clicked", G_CALLBACK(gtk_widget_destroy),
131     window); ↗
131 g_signal_connect(back_btn, "clicked", G_CALLBACK(on_back_btn_clicked), ↗
132     NULL);
132 g_signal_connect(enter_btn, "clicked", G_CALLBACK(on_enter_btn_clicked), ↗
133     NULL);
133 g_signal_connect(back_btn2, "clicked", G_CALLBACK(on_back_btn2_clicked), ↗
134     NULL);
134 g_signal_connect(manual_btn, "clicked", G_CALLBACK(on_manual_btn_clicked), ↗
135     NULL);
135 g_signal_connect(camera_btn, "clicked", G_CALLBACK(on_camera_btn_clicked), ↗
136     NULL);
136
137 //start window
138
139 //mudar cor do rem_btn
140 GdkColor color;
141 gdk_color_parse("blue", &color);
142 gtk_widget_modify_bg(rem_btn, GTK_STATE_NORMAL, &color);
143
144 gtk_widget_show(window);
145 gtk_widget_hide(text_entry);
146 gtk_widget_hide(method_selector);
147
148 gtk_main();
```

Fig. 39. Código

```
149
150
151     return 0;
152 }
153
154 void on_m_decks_clicked(GtkButton* b)
155 {
156     cd_flag = 1;
157
158     update_list();
159 }
160
161
162 void on_m_cards_clicked(GtkButton* b)
163 {
164     cd_flag = 3;
165     sprintf(actual_deck, "%s", "AllCards");
166     update_list();
167 }
168
169 void on_add_btn_clicked(GtkButton* b)
170 {
171     if (cd_flag == 1 || cd_flag == 3)
172     {
173         gtk_widget_show_all(text_entry);
174     }
175     if (cd_flag == 2)
176     {
177         gtk_widget_show_all(method_selector);
178     }
179 }
180
181 void on_rem_btn_clicked(GtkButton* b)
182 {
183     if (rem_flag == 1)
184     {
185         GdkColor color;
186
187         gdk_color_parse("blue", &color);
188
189         gtk_widget_modify_bg(GTK_WIDGET(b), GTK_STATE_NORMAL, &color);
190         rem_flag = 0;
191     }
192     else if (rem_flag == 0)
193     {
194         GdkColor color;
195
196         gdk_color_parse("red", &color);
197
198         gtk_widget_modify_bg(GTK_WIDGET(b), GTK_STATE_NORMAL, &color);
199         rem_flag = 1;
200     }
}
```

Fig. 40. Código

```
201
202 }
203
204
205 void on_row(GtkButton* b)
206 {
207     //make file string
208     char txt[] = ".txt";
209     char file_name[1024];
210     sprintf(file_name, "%s", gtk_button_get_label(b));
211     strcat(file_name, txt);
212
213     if (rem_flag == 1)
214     {
215         char buff[100];
216         int n = 0;
217         while (1)
218         {
219             GtkWidget* temp = gtk_grid_get_child_at(GTK_GRID(dc_grid), 1, n);
220             if (GTK_BUTTON(temp) == b)
221                 break;
222             n++;
223
224         }
225         n++;
226         sprintf(buff, "%d", n);
227
228         system("cp Decks.txt aux2.txt");
229         char cmd_append[] = "sed ";
230         strcat(cmd_append, buff);
231         strcat(cmd_append, "d' ");
232         strcat(cmd_append, "aux2.txt > Decks.txt");
233
234         system(cmd_append);
235
236         char remove_file[1024] = "rm ";
237         sprintf(file_name, "%s", gtk_button_get_label(b));
238         strcat(file_name, ".txt");
239         strcat(remove_file, file_name);
240
241         system(remove_file);
242
243         char remove_file2[1024] = "rm q";
244         sprintf(file_name, "%s", gtk_button_get_label(b));
245         strcat(file_name, ".txt");
246         strcat(remove_file2, file_name);
247
248         system(remove_file2);
249
250         update_list();
251
252
```

Fig. 41. Código

```
253     }
254     else
255     {
256
257         sprintf(actual_deck, "%s", gtk_button_get_label(b));
258         //reset grid
259         update_list();
260         cd_flag = 2;
261     }
262 }
263
264 void on_back_btn_clicked(GtkButton* b)
265 {
266     gtk_editable_delete_text(GTK_EDITABLE(get_usr_entry), 0, -1);
267     gtk_widget_hide(text_entry);
268 }
269
270 void on_enter_btn_clicked(GtkButton* b)
271 {
272     if (cd_flag == 1)
273     {
274
275         printf("%s\n", gtk_entry_get_text(GTK_ENTRY(get_usr_entry)));
276         char txt[] = ".txt";
277         char file_name[1024], file_namecp[1024];
278
279         sprintf(file_name, "%s", gtk_entry_get_text(GTK_ENTRY(get_usr_entry)));
280         strcpy(file_namecp, file_name);
281         strcat(file_name, txt);
282
283         FILE* f1 = fopen(file_name, "r");
284         if (f1 == NULL)
285     {
286
287             char start_cmd[2048] = "echo ";
288             char start_cmd2[2048] = "echo ";
289             char end_cmd[] = " >> Decks.txt";
290             char end_cmd2[] = "touch ";
291             char end_cmd3[] = "touch ";
292
293             //adiciona o deck a colecao
294             strcat(start_cmd, file_namecp);
295             strcat(start_cmd, end_cmd);
296             system(start_cmd);
297
298             //cria o arquivo do deck
299             //strcat(start_cmd2,end_cmd2);
300             strcat(end_cmd2, file_name);
301             system(end_cmd2);
302
303             //cria o arquivo de n de cartas
304             strcat(end_cmd3, "q");
```

Fig. 42. Código

C:\Users\lxm10\Desktop\SOE\codigo_final.cpp

```
305         strcat(end_cmd3, file_name);
306         system(end_cmd3);
307
308         update_list();
310
311         gtk_editable_delete_text(GTK_EDITABLE(get_usr_entry), 0, -1);
312         gtk_widget_hide(text_entry);
313         return;
314     }
315
316     update_list();
317
318     gtk_editable_delete_text(GTK_EDITABLE(get_usr_entry), 0, -1);
319     gtk_widget_hide(text_entry);
320 }
322
323 if (cd_flag == 2)
324 {
325     char txt[] = ".txt";
326     char card_name[1024];
327
328     char file_name[1024], file_namecp[1024];
329     char append_cmd[1024] = "echo ";
330     char append_cmd2[1024] = "echo ";
331     char actual_deck_q[1024] = "q";
332     char actual_deck_aux[1024] = "q";
333     char actual_deck_aux2[1024] = "q";
334
335     strcpy(actual_deck_aux, actual_deck);
336     strcpy(actual_deck_aux2, actual_deck);
337
338     strcat(actual_deck_aux, txt);
339
340     sprintf(card_name, "%s", gtk_entry_get_text(GTK_ENTRY(get_usr_entry)));
341     int flag_card = tesste(card_name);
342     if (flag_card < 0) {
343         return;
344     }
345     //add card to deck
346     strcat(append_cmd, card_name);
347     strcat(append_cmd, ">>");
348     strcat(append_cmd, actual_deck_aux);
349     system(append_cmd);
350
351     strcat(append_cmd2, "1");
352     strcat(append_cmd2, " >> ");
353     strcat(actual_deck_q, actual_deck_aux2);
354     strcat(actual_deck_q, txt);
355
356
```

Fig. 43. Código

C:\Users\lxm10\Desktop\SOE\codigo_final.cpp

```

357     strcat(append_cmd2, actual_deck_q);
358     printf("%s\n", append_cmd2);
359     system(append_cmd2);
360
361     //make buttons and append to grid
362     gtk_editable_delete_text(GTK_EDITABLE(get_usr_entry), 0, -1);
363     gtk_widget_hide(text_entry);
364     update_list();
365
366 }
367
368 if (cd_flag == 3)
369 {
370     char txt[] = ".txt";
371     char card_name[1024];
372
373     char file_name[1024], file_namecp[1024];
374     char append_cmd[1024] = "echo ";
375     char append_cmd2[1024] = "echo ";
376     char actual_deck_q[1024] = "q";
377     char actual_deck_aux[1024] = "q";
378     char actual_deck_aux2[1024] = "q";
379
380     strcpy(actual_deck_aux, "AllCards");
381     strcpy(actual_deck_aux2, "AllCards");
382
383
384     strcat(actual_deck_aux, txt);
385
386     sprintf(card_name, "%s", gtk_entry_get_text(GTK_ENTRY(get_usr_entry)));
387     int flag_card = tesste(card_name);
388     if (flag_card < 0) {
389         return;
390     }
391     //add card to deck
392     strcat(append_cmd, card_name);
393     strcat(append_cmd, ">>");
394     strcat(append_cmd, actual_deck_aux);
395     system(append_cmd);
396
397     strcat(append_cmd2, "1");
398     strcat(append_cmd2, " >> ");
399     strcat(actual_deck_q, actual_deck_aux2);
400     strcat(actual_deck_q, txt);
401
402     strcat(append_cmd2, actual_deck_q);
403     printf("%s\n", append_cmd2);
404     system(append_cmd2);
405
406     //make buttons and append to grid
407     gtk_editable_delete_text(GTK_EDITABLE(get_usr_entry), 0, -1);
408     gtk_widget_hide(text_entry);

```

Fig. 44. Código

```
409         update_list();
410     }
411
412 }
413
414 void delete_rows()
415 {
416     GList* children, * iter;
417     children = gtk_container_get_children(GTK_CONTAINER(dc_grid));
418     for (iter = children; iter != NULL; iter = g_list_next(iter))
419     {
420         gtk_widget_destroy(GTK_WIDGET(iter->data));
421     }
422     g_list_free(children);
423 }
424
425 void update_list()
426 {
427     delete_rows();
428     if (cd_flag == 1)
429     {
430
431
432
433
434
435     FILE* f1 = fopen("Decks.txt", "r");
436     if (f1 == NULL)
437     {
438         printf("File error!\n");
439         exit(1);
440     }
441
442     //make buttons and append to grid
443     row = 0;
444     while (1)
445     {
446         if (fgets(tmp, 1024, f1) == NULL)
447         {
448             fclose(f1);
449             break;
450         }
451         tmp[strlen(tmp) - 1] = 0;
452
453         gtk_grid_insert_row(GTK_GRID(dc_grid), row);
454
455         //add buttons
456         button[row] = gtk_button_new_with_label(tmp);
457         gtk_button_set_alignment(GTK_BUTTON(button[row]), 0.0, 0.5);
458         gtk_grid_attach(GTK_GRID(dc_grid), button[row], 1, row, 1, 1);
459         g_signal_connect(button[row], "clicked", G_CALLBACK(on_row), NULL);
460         gtk_widget_show(button[row]);
```

Fig. 45. Código

```
461         row++;
462
463     }
464 }
465
466 if (cd_flag == 2)
467 {
468     char txt[] = ".txt";
469     char actual_deck_aux[1024] = "q";
470     char actual_deck_aux2[1024] = "q";
471     strcpy(actual_deck_aux, actual_deck);
472     strcat(actual_deck_aux, txt);
473
474     delete_rows();
475
476     FILE* f1 = fopen(actual_deck_aux, "r");
477     strcat(actual_deck_aux2, actual_deck_aux);
478     FILE* f2 = fopen(actual_deck_aux2, "r");
479     if (f1 == NULL || f2 == NULL)
480     {
481         printf("File error!\n");
482         exit(1);
483     }
484
485 //make buttons and append to grid
486 row = 0;
487 while (1)
488 {
489     if (fgets(tmp, 1024, f1) == NULL || fgets(tmp2, 1024, f2) == NULL)
490     {
491         fclose(f1);
492         fclose(f2);
493         break;
494     }
495     tmp[strlen(tmp) - 1] = 0;
496     tmp2[strlen(tmp2) - 1] = 0;
497
498     gtk_grid_insert_row(GTK_GRID(dc_grid), row);
499     strcat(tmp2, "x");
500     strcat(tmp2, tmp);
501
502     button[row] = gtk_button_new_with_label(tmp2);
503     gtk_button_set_alignment(GTK_BUTTON(button[row]), 0.0, 0.5);
504     gtk_grid_attach(GTK_GRID(dc_grid), button[row], 1, row, 1, 1);
505     g_signal_connect(button[row], "clicked", G_CALLBACK(on_row2), NULL);
506     gtk_widget_show(button[row]);
507     row++;
508
509
510
511 }
512 }
```

Fig. 46. Código

```
513     if (cd_flag == 3)
514     {
515         char txt[] = ".txt";
516         char actual_deck_aux[1024] = "q";
517         char actual_deck_aux2[1024] = "q";
518         strcpy(actual_deck_aux, "AllCards");
519         strcat(actual_deck_aux, txt);
520
521         delete_rows();
522
523         FILE* f1 = fopen(actual_deck_aux, "r");
524         strcat(actual_deck_aux2, actual_deck_aux);
525         FILE* f2 = fopen(actual_deck_aux2, "r");
526         if (f1 == NULL || f2 == NULL)
527         {
528             printf("File error!\n");
529             exit(1);
530         }
531
532         //make buttons and append to grid
533         row = 0;
534         while (1)
535         {
536             if (fgets(tmp, 1024, f1) == NULL || fgets(tmp2, 1024, f2) == NULL)
537             {
538                 fclose(f1);
539                 fclose(f2);
540                 break;
541             }
542             tmp[strlen(tmp) - 1] = 0;
543             tmp2[strlen(tmp2) - 1] = 0;
544
545             gtk_grid_insert_row(GTK_GRID(dc_grid), row);
546             strcat(tmp2, "x");
547             strcat(tmp2, tmp);
548
549             button[row] = gtk_button_new_with_label(tmp2);
550             gtk_button_set_alignment(GTK_BUTTON(button[row]), 0.0, 0.5);
551             gtk_grid_attach(GTK_GRID(dc_grid), button[row], 1, row, 1, 1);
552             g_signal_connect(button[row], "clicked", G_CALLBACK(on_row2), NULL);
553             gtk_widget_show(button[row]);
554             row++;
555
556         }
557     }
558 }
559
560 }
561 void on_row2(GtkButton* b)
562 {
563     if (rem_flag == 1)
```

Fig. 47. Código

```
565     {
566         char buff[100];
567         int val_n = 0;
568         while (1)
569     {
570
571             GtkWidget* temp = gtk_grid_get_child_at(GTK_GRID(dc_grid), 1,
572                                         val_n);
573             if (GTK_BUTTON(temp) == b)
574                 break;
575
576             val_n++;
577
578         }
579         sprintf(buff, "%d", val_n);
580
581
582
583         system("rm aux.txt");
584         char cmd_append[] = "cp ";
585         strcat(cmd_append, "q");
586         strcat(cmd_append, actual_deck);
587         strcat(cmd_append, ".txt ");
588         strcat(cmd_append, "aux.txt");
589         system(cmd_append);
590
591         FILE* f1 = fopen("aux.txt", "r");
592
593         if (f1 == NULL)
594     {
595             printf("File error!\n");
596             return;
597         }
598
599
600         char cmd_append2[] = "rm ";
601         strcat(cmd_append2, "q");
602         strcat(cmd_append2, actual_deck);
603         strcat(cmd_append2, ".txt");
604         system(cmd_append2);
605
606         char cmd_append3[] = "touch ";
607         strcat(cmd_append3, "q");
608         strcat(cmd_append3, actual_deck);
609         strcat(cmd_append3, ".txt");
610         system(cmd_append3);
611
612         row = 0;
613         while (1)
614     {
615
```

Fig. 48. Código

```
616         if (fgets(tmp, 1024, f1) == NULL)
617         {
618             fclose(f1);
619             break;
620         }
621         tmp[strlen(tmp) - 1] = 0;
622         char cmd_append4[2048] = "echo ";
623
624         if (row == atoi(buff))
625         {
626
627
628             int val = atoi(tmp);
629             val--;
630
631
632
633             sprintf(tmp, "%d", val);
634             strcat(cmd_append4, tmp);
635
636
637             strcat(cmd_append4, " >> ");
638             strcat(cmd_append4, "q");
639             strcat(cmd_append4, actual_deck);
640             strcat(cmd_append4, ".txt");
641
642
643             system(cmd_append4);
644
645
646             if (val == 0)
647             {
648                 char buff2[100];
649                 sprintf(buff2, "%d", atoi(buff) + 1);
650                 char cmd_append12[] = "cp ";
651                 strcat(cmd_append12, actual_deck);
652                 strcat(cmd_append12, ".txt");
653                 strcat(cmd_append12, " aux2.txt");
654
655                 system(cmd_append12);
656
657                 char cmd_append11[] = "sed ";
658                 strcat(cmd_append11, buff2);
659                 strcat(cmd_append11, "d' ");
660                 strcat(cmd_append11, "aux2.txt > ");
661                 strcat(cmd_append11, actual_deck);
662                 strcat(cmd_append11, ".txt");
663                 printf("cmd:11%s\n", cmd_append11);
664                 system(cmd_append11);
665
666                 char cmd_append14[] = "cp q";
667                 strcat(cmd_append14, actual_deck);
```

Fig. 49. Código

```
668         strcat(cmd_append14, ".txt");
669         strcat(cmd_append14, " aux2.txt");
670
671         system(cmd_append14);
672
673         char cmd_append13[] = "sed ";
674         strcat(cmd_append13, buff2);
675         strcat(cmd_append13, "d' ");
676         strcat(cmd_append13, "aux2.txt > q");
677         strcat(cmd_append13, actual_deck);
678         strcat(cmd_append13, ".txt");
679
680         system(cmd_append13);
681     }
682
683
684     }
685     else
686     {
687
688         strcat(cmd_append4, tmp);
689
690
691         system(cmd_append4);
692         strcat(cmd_append4, " >> ");
693         strcat(cmd_append4, "q");
694         strcat(cmd_append4, actual_deck);
695         strcat(cmd_append4, ".txt");
696
697         system(cmd_append4);
698     }
699     row++;
700 }
701 update_list();
702
703 }
704 else
705 {
706     {
707         char buff[100];
708         int val_n = 0;
709         while (1)
710     {
711
712         GtkWidget* temp = gtk_grid_get_child_at(GTK_GRID(dc_grid), 1,
713                                         val_n);
714         if (GTK_BUTTON(temp) == b)
715             break;
716
717         val_n++;
718 }
```

Fig. 50. Código

```
719         }
720         sprintf(buff, "%d", val_n);
721
722
723
724         system("rm aux.txt");
725         char cmd_append[] = "cp ";
726         strcat(cmd_append, "q");
727         strcat(cmd_append, actual_deck);
728         strcat(cmd_append, ".txt ");
729         strcat(cmd_append, "aux.txt");
730         system(cmd_append);
731
732
733         FILE* f1 = fopen("aux.txt", "r");
734         if (f1 == NULL)
735         {
736             printf("File error!\n");
737             return;
738         }
739
740
741         char cmd_append2[] = "rm ";
742         strcat(cmd_append2, "q");
743         strcat(cmd_append2, actual_deck);
744         strcat(cmd_append2, ".txt");
745         system(cmd_append2);
746
747         char cmd_append3[] = "touch ";
748         strcat(cmd_append3, "q");
749         strcat(cmd_append3, actual_deck);
750         strcat(cmd_append3, ".txt");
751         system(cmd_append3);
752
753         row = 0;
754         while (1)
755         {
756
757             if (fgets(tmp, 1024, f1) == NULL)
758             {
759                 fclose(f1);
760                 break;
761             }
762             tmp[strlen(tmp) - 1] = 0;
763             char cmd_append4[2048] = "echo ";
764
765             if (row == atoi(buff))
766             {
767
768
769                 int val = atoi(tmp);
```

Fig. 51. Código

```
C:\Users\lxm10\Desktop\SOE\codigo_final.cpp
771             val++;
772
773             sprintf(tmp, "%d", val);
774             strcat(cmd_append4, tmp);
775
776             strcat(cmd_append4, " >> ");
777             strcat(cmd_append4, "q");
778             strcat(cmd_append4, actual_deck);
779             strcat(cmd_append4, ".txt");
780
781
782             system(cmd_append4);
783
784
785
786
787         }
788         else
789     {
790
791             strcat(cmd_append4, tmp);
792
793
794             system(cmd_append4);
795             strcat(cmd_append4, " >> ");
796             strcat(cmd_append4, "q");
797             strcat(cmd_append4, actual_deck);
798             strcat(cmd_append4, ".txt");
799
800             system(cmd_append4);
801         }
802         row++;
803     }
804     update_list();
805 }
806 }
807
808 }
809
810 void on_camera_btn_clicked(GtkButton* b)
811 {
812     flag_card = cards();
813     if (flag_card > 0) {
814
815
816
817     char txt[] = ".txt";
818     char card_name[1024];
819
820     char file_name[1024], file_namecp[1024];
821     char append_cmd[1024] = "echo ";
822     char append_cmd2[1024] = "echo ";
```

16

Fig. 52. Código

```
823     char actual_deck_q[1024] = "q";
824     char actual_deck_aux[1024] = "q";
825     char actual_deck_aux2[1024] = "q";
826
827     strcpy(actual_deck_aux, actual_deck);
828     strcpy(actual_deck_aux2, actual_deck);
829
830     strcat(actual_deck_aux, txt);
831
832
833
834
835     //add card to deck
836     strcat(append_cmd, nome2);
837     strcat(append_cmd, ">>");
838     strcat(append_cmd, actual_deck_aux);
839     system(append_cmd);
840
841     strcat(append_cmd2, "1");
842     strcat(append_cmd2, " >> ");
843     strcat(actual_deck_q, actual_deck_aux2);
844     strcat(actual_deck_q, txt);
845
846     strcat(append_cmd2, actual_deck_q);
847
848     system(append_cmd2);
849     update_list();
850 }
851 }
852 void on_manual_btn_clicked(GtkButton* b)
853 {
854     gtk_widget_show(text_entry);
855
856 }
857 void on_back_btn2_clicked(GtkButton* b)
858 {
859     gtk_widget_hide(method_selector);
860 }
861
862
863 int cards() {
864
865
866     quadro();
867     tesse();
868
869     FILE* f1 = fopen("names2.txt", "r");
870     if (f1 == NULL)
871     {
872         printf("names2.txt nao existe\n");
873         exit(1);
874     }
```

Fig. 53. Código

```
875 //search for card
876 printf("nome =>%s\n", nome2);
877 while (1)
878 {
879     if (fgets(tmp, 1024, f1) == NULL)
880     {
881         printf("Carta nao localizada!\n ");
882         break;
883     }
884     tmp[strlen(tmp) - 1] = 0;
885     int diff = comparech(tmp, nome2);
886
887     if (diff > 80)
888     {
889         strcpy(nome2, tmp);
890         printf("Carta localizada!---->%s\n ", nome2);
891         fclose(f1);
892         return 1;
893     }
894 }
895
896 fclose(f1);
897
898
899
900
901     return -1;
902 }
903 char tesse()
904 {
905
906     char* outText;
907     tesseract::TessBaseAPI* api = new tesseract::TessBaseAPI();
908     // Initialize tesseract-ocr with English, without specifying tessdata path
909     if (api->Init(NULL, "eng")) {
910         fprintf(stderr, "Could not initialize tesseract.\n");
911         exit(1);
912     }
913     Pix* image = pixRead("sample.png");
914     api->SetPageSegMode(tesseract::PSM_AUTO);
915     api->SetImage(image);
916     api->Recognize(0);
917     Boxa* boxes = api->GetComponentImages(tesseract::RIL_TEXTLINE, true, NULL, ↵
918         NULL);
919     //printf("Found %d textline image components.\n", boxes->n);
920     char* ocrResult;
921     for (int i = 0; i < boxes->n; i++) {
922         BOX* box = boxaGetBox(boxes, i, L_CLONE);
923         api->SetRectangle(box->x, box->y, box->w, box->h);
924         ocrResult = api->GetUTF8Text();
925         int conf = api->MeanTextConf();
926         //fprintf(stdout, "Box[%d]: x=%d, y=%d, w=%d, h=%d, confidence: %d, ↵
```

Fig. 54. Código

```
926         text: %s",
927             //i, box->x, box->y, box->w, box->h, conf, ocrResult);
928     boxDestroy(&box);
929 }
930 int a;
931 //if(ocrResult==NULL)
932 //strcpy(nome2,"A");
933 printf("OCR->%s\n", ocrResult);
934 // Destroy used object and release memory
935 api->End();
936
937 delete api;
938
939
940
941 //delete [] outText;
942
943 pixDestroy(&image);
944 char lixo[] = "A";
945
946 if (ocrResult == NULL)
947     nome2 = lixo;
948 else
949     nome2 = ocrResult;
950
951 printf("nome2 = ocrResult\n");
952
953 return 0;
954 }
955 int quadro()
956 {
957     int i = 0;
958
959     Mat frame;
960     Mat resized_frame;
961     Mat grayscale;
962     Mat grayscalecp;
963
964     char* outText;
965
966
967     //int x=980;
968     //int y=1230;
969     //int u=580;
970     //int l=63;
971     int x = 970;
972     int y = 1230;
973     int u = 580;
974     int l = 63;
975     //Rect text_rect(x,y,200,20);
976 }
```

Fig. 55. Código

```

C:\Users\lxm10\Desktop\SOE\codigo_final.cpp 20
977     system("raspistill -o image.jpg -br 50 -sh 100 -sa -100 -ex auto -t 700 -q   ↵
978         100");
979     frame = imread("image.jpg");
980
981     Rect cut(x, y, u, l);
982     imwrite("aaaaa.png", frame(cut));
983
984     cvtColor(frame(cut), grayscale, COLOR_RGB2GRAY);
985     threshold(grayscale, grayscale, 200, 255, THRESH_BINARY + THRESH_OTSU);
986     //double scale_up_x = 3;
987     //double scale_up_y = 3;
988     //Mat scaled_f_up;
989     //resize(grayscale, scaled_f_up, Size(), scale_up_x, scale_up_y,           ↵
990             INTER_NEAREST);
990     imwrite("sample.png", grayscale);
991
992
993     return 0;
994 }
995
996
997 int tesste(char* name) {
998     FILE* N_1 = NULL;
999     N_1 = fopen("names.txt", "r");
1000    if (N_1 == NULL) {
1001        printf("Erro 2: Arquivo de Referência vocabulario_nota1 não pode ser      ↵
1002            aberto\n");
1003    }
1004    char* buffer1 = 0;
1005    long length1;
1006    if (N_1)
1007    {
1008        fseek(N_1, 0, SEEK_END);
1009        length1 = ftell(N_1);
1010        fseek(N_1, 0, SEEK_SET);
1011        buffer1 = (char*)malloc(length1);
1012        if (buffer1)
1013        {
1014            fread(buffer1, 1, length1, N_1);
1015        }
1016        fclose(N_1);
1017    }
1018    if (strstr(buffer1, name) != NULL) {
1019        printf("Carta localizada!\n");
1020        return 0;
1021    }
1022    else
1023        printf("Por favor tente novamente");
1024    free(buffer1);
1025    return -1;

```

Fig. 56. Código

C:\Users\lxm10\Desktop\SOE\codigo_final.cpp

```
1026
1027 }
1028
1029 int comparech(const char* str1, const char* str2)
1030 {
1031     int diff = 0;
1032     int str1_size = (int)strlen(str1); //tmp
1033     int str2_size = (int)strlen(str2); //input
1034     if (str2_size + 1 < str1_size)
1035     {
1036         return 0;
1037     }
1038     if (strlen(str1) > strlen(str2))
1039     {
1040
1041         for (int i = 0; i < strlen(str2); i++)
1042         {
1043             if (str1[i] - str2[i] != 0)
1044             {
1045                 diff++;
1046             }
1047         }
1048
1049
1050         return 100 - (diff * 100) / str1_size;
1051     }
1052     else
1053     {
1054         for (int i = 0; i < strlen(str1); i++)
1055         {
1056             if (str1[i] - str2[i] != 0)
1057             {
1058                 diff++;
1059             }
1060         }
1061
1062
1063         return 100 - (diff * 100) / str1_size;
1064     }
1065 }
1066
1067
```

21

Fig. 57. Código