

Pontifícia Universidade Católica do Rio de Janeiro – PUC-Rio
Departamento de Informática
Programa de Pós-Graduação
INF2102 - Projeto Final de Programação

Bruno Rocha Gomes
Matrícula: 2112959
Orientador: Sérgio Colcher

Ferramenta para Extração de Landmarks Faciais em Vídeos .mp4

Rio de Janeiro, Brasil
2022

Pontifícia Universidade Católica do Rio de Janeiro – PUC-Rio
Departamento de Informática
Programa de Pós-Graduação
INF2102 - Projeto Final de Programação

Bruno Rocha Gomes
Matrícula: 2112959
Orientador: Sérgio Colcher

Ferramenta para Extração de Landmarks Faciais em Vídeos .mp4

Documento apresentado à
Pontifícia Universidade Católica do
Rio de Janeiro como requisito para
obtenção de nota na disciplina
INF2102 - Projeto Final de
Programação, ministrada pela
Prof.^a Dra. Clarisse de Souza.

Rio de Janeiro, Brasil
2022

Sumário

1. Especificação do Programa	4
1.1. Objetivo e escopo	4
1.2. Requisitos da aplicação	4
1.2.1. Requisitos Funcionais	4
1.2.2. Requisitos Não-Funcionais	5
2. Projeto da ferramenta	5
2.1. Arquitetura	5
2.2. Estrutura do Projeto	6
2.3. Diagramas	8
2.3.1. Caso de uso	8
2.3.2. Atividade	9
3. Testes	11
4. Documentação para o usuário	11
4.1. Instalação e configuração	12
4.2. Instruções de uso	12
5. Colaboração científica	16

1. Especificação do Programa

Esta seção contém uma descrição da ferramenta, apresentando os objetivos por trás do projeto, bem como os escopo e os requisitos funcionais e não-funcionais da aplicação.

1.1. Objetivo e escopo

Entre as diversas áreas baseadas em Machine Learning (ML) é comum nos depararmos com tarefas que envolvem detecção facial, tais como reconhecimento de expressões faciais, clusterização de rostos, localização de pessoas, detecção de vídeos *deepfakes*, entre outras. Contudo, em alguma destas, há uma importância de não apenas detectar um rosto presente em uma imagem ou vídeo, mas também as coordenadas precisas dos elementos faciais, tais como olhos, nariz e boca, para inferir, por exemplo, a posição da cabeça de uma pessoa, sua localização na mídia em questão e as expressões faciais. Contudo, apesar da necessidade, estes pontos de referências, também chamados de *landmarks*, dificilmente são contemplados nas anotações dos *datasets* onde a face humana é o elemento principal de estudo.

Nesse contexto, este trabalho propõe o desenvolvimento de uma ferramenta prática e visual para auxiliar o processo de anotação de *datasets* de vídeos contendo rostos humanos a partir da extração de *landmarks* faciais das pessoas presentes no vídeo. A aplicação também permite salvar as informações extraídas em um arquivo para o usuário futuramente utilizá-las em suas aplicações de ML.

1.2. Requisitos da aplicação

1.2.1. Requisitos Funcionais

- **RF01 – Extrair *landmarks* faciais:** a ferramenta deve ser possuir a função de extrair as *landmarks* faciais oriundas dos rostos presentes nos vídeos carregados pelo usuário;
- **RF02 – Salvar resultados obtidos:** a ferramenta deve ser capaz de salvar, no diretório escolhido pelo usuário, um arquivo *.json* contendo todas as *landmarks* obtidas, mantendo ainda, as informações referentes ao nome do vídeo, frame atual e id da pessoa;
- **RF03 – Fornecer interface gráfica:** a ferramenta deve conter uma interface gráfica de fácil uso para a aplicação;
- **RF04 – Parametrização pelo usuário:** a ferramenta deve permitir que o usuário escolha os parâmetros de extração que desejar, desde que estes sejam válidos para a ferramenta;
- **RF05 – Extração em larga escala:** a ferramenta deve ser capaz de realizar a extração de múltiplos vídeos contidos no mesmo diretório, como por exemplo *datasets*;

1.2.2. Requisitos Não-Funcionais

- **RNF01 – Usabilidade:** a ferramenta deve de fácil uso para o usuário;
- **RNF02 – Desempenho:** a ferramenta deve ser otimizada para executar com baixo custo computacional;
- **RNF03 – Confiabilidade:** a ferramenta deve gerar dados precisos acerca dos vídeos processados dentro dos intervalos de frames definidos pelo usuário;
- **RNF04 – Estabilidade:** a ferramenta deve manter o desempenho mesmo quando operando em bases de dados extensas;

2. Projeto da ferramenta

Para o desenvolvimento da ferramenta, utilizou-se os seguintes *frameworks*: Electron¹ (Versão 13.1.4), responsável por permitir a criação de uma aplicação desktop multiplataforma utilizando JavaScript, HTML e CSS; NodeJS² (Versão 14.15.0), permitindo a manipulação e controle das telas; e Bootstrap³ (Versão 5.1.3), para a adição de elementos visuais e componentes de interface da aplicação. Além disso, para o desenvolvimento do script responsável por extrair as *landmarks* faciais em vídeos, utilizou-se a linguagem de programação Python⁴ (Versão 3.8.10), juntamente com as bibliotecas OpenCV (Versão 4.5.5.64), Python Pillow (Versão 9.1.1), NumPy (Versão 1.22.4) e o módulo FaceNet-Pytorch (Versão 2.5.2).

Ainda nesta seção, serão apresentados a arquitetura utilizada no projeto, a estrutura de pastas e arquivos da ferramenta e uma modelagem em diagramas das funcionalidades e requisitos do sistema.

2.1. Arquitetura

A arquitetura adotada para a aplicação segue o padrão de projeto *Model-View-Controller*. Nela, cada *framework* utilizado na ferramenta exerce sua determinada função. Dessa forma, o papel da componente *Model* é exercido pelo Electron juntamente com o NodeJS, o *View* é responsável pelo Bootstrap, e por fim, o *Controller* é o componente assumido pelo script Python. A Figura 1 abaixo ilustra o padrão de arquitetura do projeto.

¹ <https://www.electronjs.org>

² <https://nodejs.org/pt-br/>

³ <https://getbootstrap.com>

⁴ <https://www.python.org/downloads/release/python-3810/>

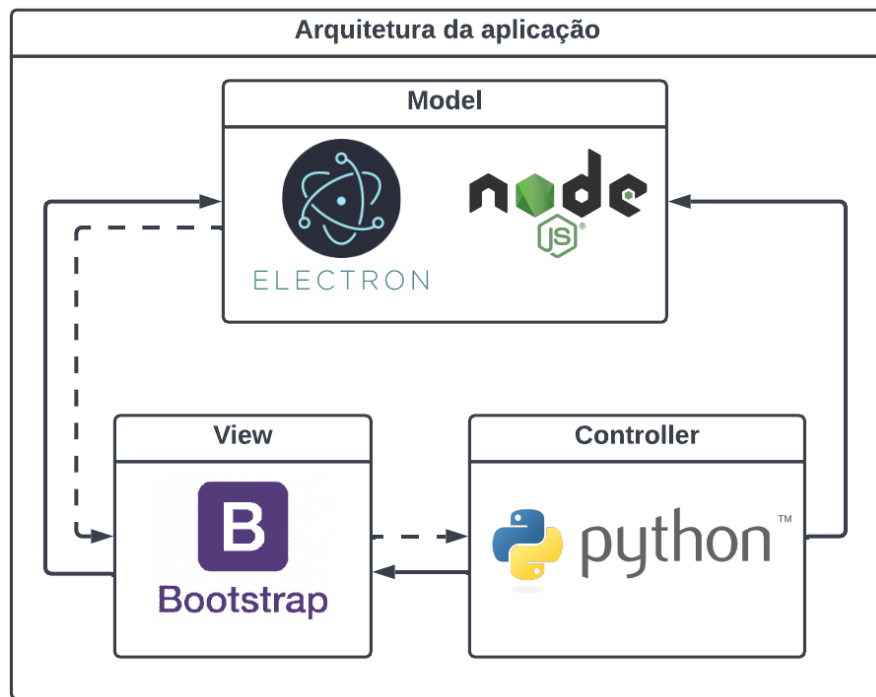


Figura 1 - Arquitetura da aplicação.

Conforme mostrado na Figura 1, o *Model*, por meio dos eventos definidos através do Electron e NodeJS, é responsável por gerenciar os estados oriundos tanto da *View*, modelando as entradas do usuário e notificando se estas correspondem ou não ao esperado, quanto do *Controller*, recebendo o retorno gerado pelo script Python. A *View*, por sua vez, é responsável por renderizar o retorno gerado pelo *Controller*, bem como enviar e receber do *Model* as entradas do usuário devidamente modeladas que serão passadas como argumento para o script Python. Por fim, o *Controller* é responsável por executar as operações e requisitos propostos pela aplicação e retornar o resultado para a *View* e *Model*.

2.2. Estrutura do Projeto

A estrutura de diretórios e arquivos do projeto corresponde ao mostrado na Figura 2. A raiz do projeto é composta pelos seguintes arquivos:

- **.gitignore**: arquivo da ferramenta Git que define arquivos ou diretórios que devem ser ignorados no repositório do projeto. Nesse caso, foram especificados o diretório *node_modules* e *test/videos*, este último contendo os vídeos utilizados no teste da aplicação e o arquivo *package-lock.json*;
- **README.md**: arquivo do tipo *markdown* responsável por descrever as principais funções do projeto, bem como as instruções de instalação e execução;
- **PFP_Documentacao_Landmark_Extractor.pdf**: este atual arquivo de documentação do projeto;

- **package.json:** arquivo de configuração do *framework* Electron responsável por especificar informações do autor, versão, nome e descrição da aplicação, bem como scripts e dependências de pacotes utilizados na implementação.

Além disso, existem ainda dois diretórios: *src* e *test*. O primeiro contém quatro subdiretórios, um para cada tipo de arquivo utilizado na implementação. Por outro lado, o segundo contém arquivos de teste usados na aplicação:

- **src:**
 - **css:** contém os arquivos *css* correspondente a cada tela da aplicação, responsáveis por adicionar a estilização nos arquivos *html*;
 - **html:** contém os arquivos *html* correspondente a cada tela da aplicação, responsáveis por definir os elementos de hipertexto, tais como imagens, palavras e arquivos de mídia;
 - **js:** contém os arquivos *JavaScript* correspondente a cada tela da aplicação, responsáveis por definir os eventos dinâmicos das páginas *html*, gerenciar o fluxo das telas existentes e controlar a chamada para o script Python. Além disso, o arquivo *main.js* destaca-se por ser o arquivo de inicialização do Electron, que configura e instancia uma janela para a aplicação, bem como define eventos de comunicação entre os demais scripts *.js*, como por exemplo, *getters* e *setters* de variáveis.
 - **py:** contém o arquivo *extract_landmarks.py*, script Python principal da ferramenta, responsável por realizar a extração das *landmarks* presentes nos vídeos de formato *.mp4* contidos no diretório de origem que é passado como argumento para o script. O mesmo também espera como argumento um diretório de destino onde será salvo o arquivo *landmarks.json* resultante e, por fim, um número inteiro positivo correspondente ao intervalo de frames a ser considerado para executar cada extração de *landmarks* no vídeo corrente. Além disso, o diretório *py* contém o arquivo *requirements.txt*, que especifica as bibliotecas e suas respectivas versões requeridas pelo script Python.
- **test:** contém os scripts responsáveis por realizar os testes automatizados da aplicação: *test.js*, que realiza os testes da ferramenta desktop gerada pelo Electron; e *test.py*, que executa o teste na função principal do script *extract_landmarks.py*. O diretório conta ainda com o arquivo *log.txt* que demonstra o resultado do teste gerado pelo Python.

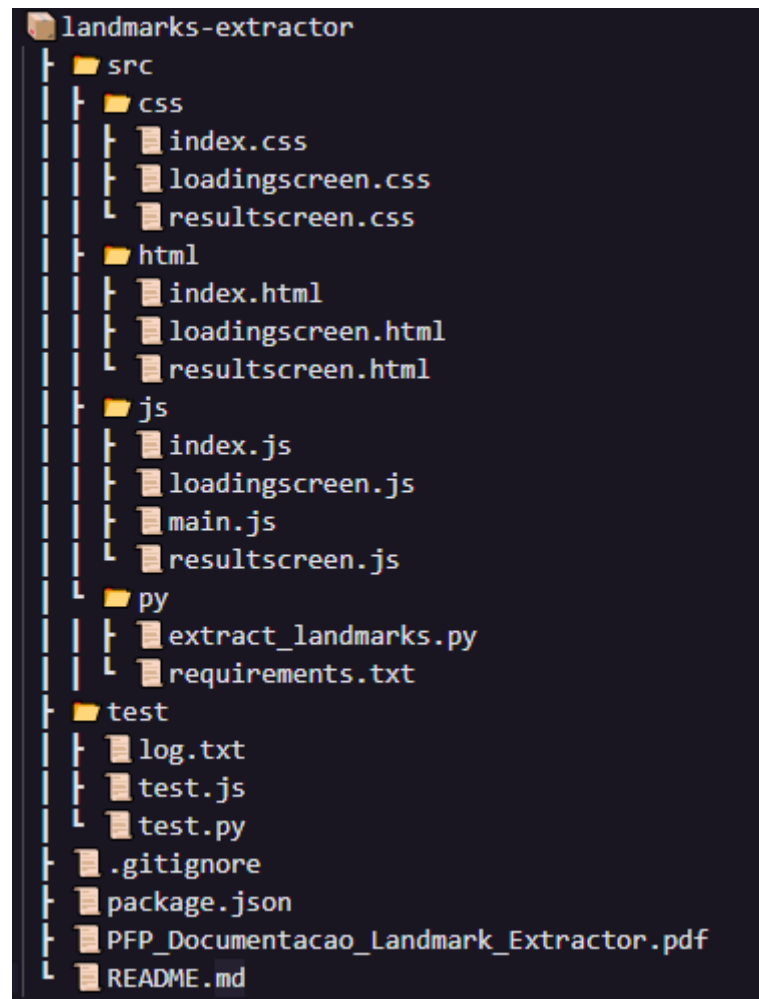


Figura 2 - Árvore de arquivos e diretórios do projeto.

2.3. Diagramas

Nesta seção serão apresentados os diagramas de caso de uso e de atividade a fim de detalhar o fluxo da aplicação.

2.3.1. Caso de uso

Com o objetivo de apresentar as funcionalidades da ferramenta, foi elaborado um diagrama de caso de uso, mostrado na Figura 3. Nele, o ator, que consiste no usuário da aplicação, é solicitado a inserir três parâmetros: caminho do diretório de origem, caminho do diretório de destino e intervalo de frames. Após executar essa tarefa, todo o resto é feito internamente pela aplicação.

O diretório de origem deverá conter os vídeos a serem processados, os quais serão carregados um por um no script Python, onde a extração das *landmarks* irá ocorrer. O intervalo de frames, por sua vez, consiste em um parâmetro que será responsável por informar de quantos em quantos frames as *landmarks* serão extraídas e, portanto, implicará diretamente nesta tarefa. Por fim, o diretório de destino será carregado pelo script Python como o diretório onde será salvo o arquivo *json* contendo todas as *landmarks* extraídas pela ferramenta.

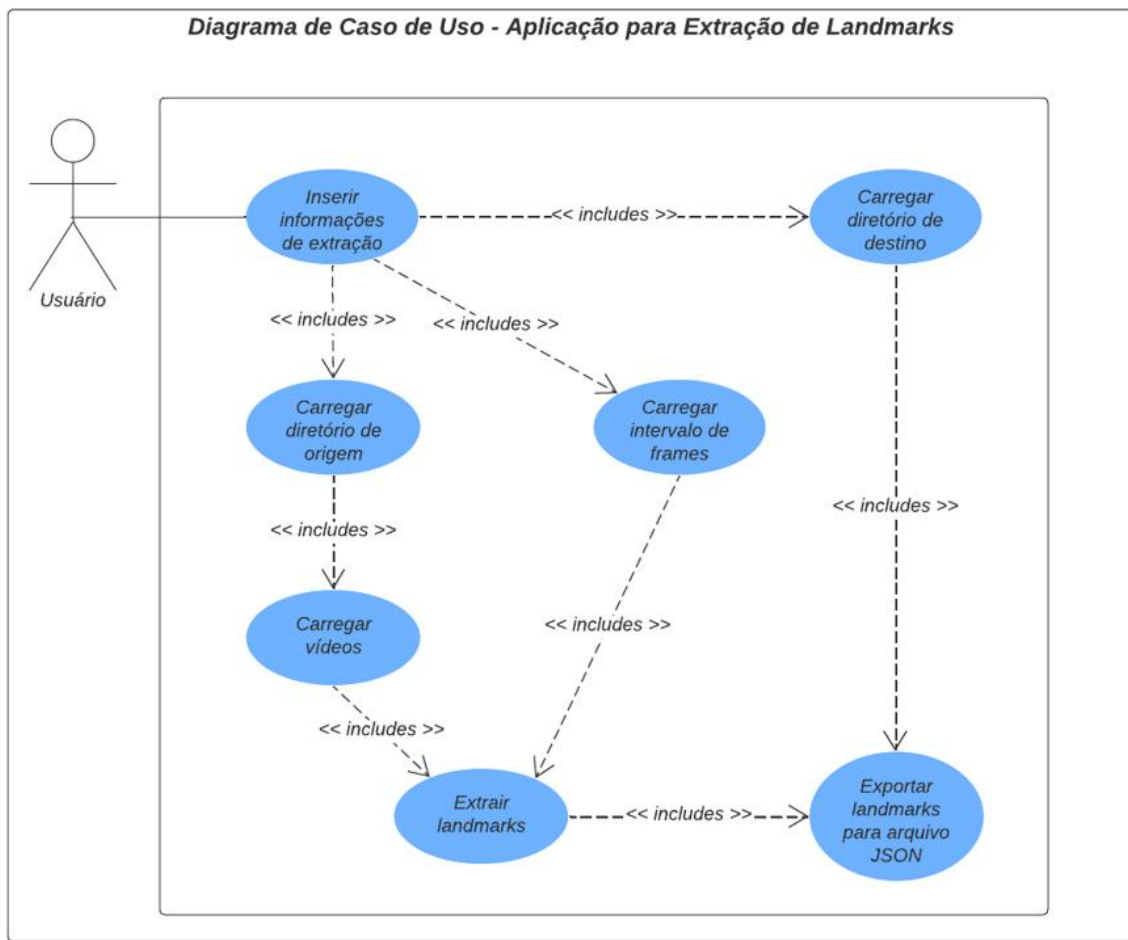


Figura 3 - Diagrama de caso de uso da ferramenta.

2.3.2. Atividade

Para demonstrar o fluxo da ferramenta detalhadamente, projetou-se também os diagramas de atividade abaixo. Estes foram divididos em dois, sendo um contendo as ações executadas pelo usuário e o outro focado nas ações internas da aplicação.

O primeiro diagrama, mostrado na Figura 4, consiste na inserção dos três parâmetros obrigatórios pelo usuário. Nele, é mostrado ainda, que a ferramenta espera que o diretório de origem informado contenha pelo menos um vídeo no formato *.mp4*. Caso contrário, o usuário será informado que o diretório é inválido e será solicitado novamente a informar este parâmetro.

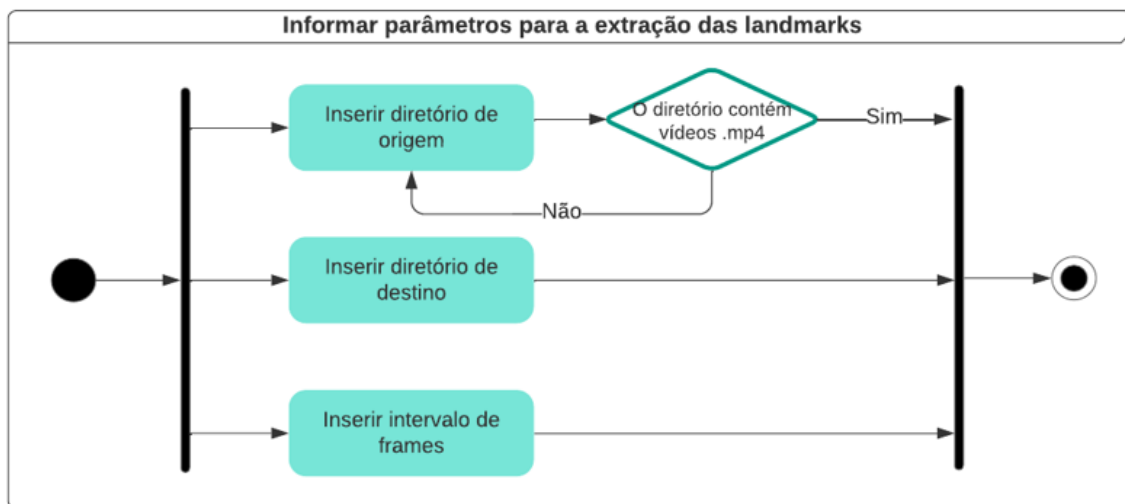


Figura 4 - Diagrama de Atividade – Inserção de parâmetros pelo usuário.

A Figura 5, por sua vez, diz respeito ao segundo diagrama de atividade. Este consiste no fluxo interno realizado pela ferramenta para extrair as *landmarks* faciais dos rostos presentes nos vídeos informados. Primeiramente, uma vez que um diretório de origem válido foi informado, o script irá iterar sobre cada vídeo *.mp4* presente no mesmo. Caso em nenhum dos vídeos seja detectado um rosto dentro do intervalo de frames informado, a ferramenta irá retornar um aviso ao usuário. Entretanto, caso em pelo menos um vídeo, o script tenha detectado uma face humana dentro do intervalo de frames, a mesma terá sua *landmark* extraída no frame corrente e será salva num arquivo *json* que é atualizado a cada nova *landmark* extraída. Após iterar sobre todos os vídeos, o arquivo *json* é salvo no diretório de destino informado pelo usuário e a ferramenta irá retornar uma mensagem de sucesso.

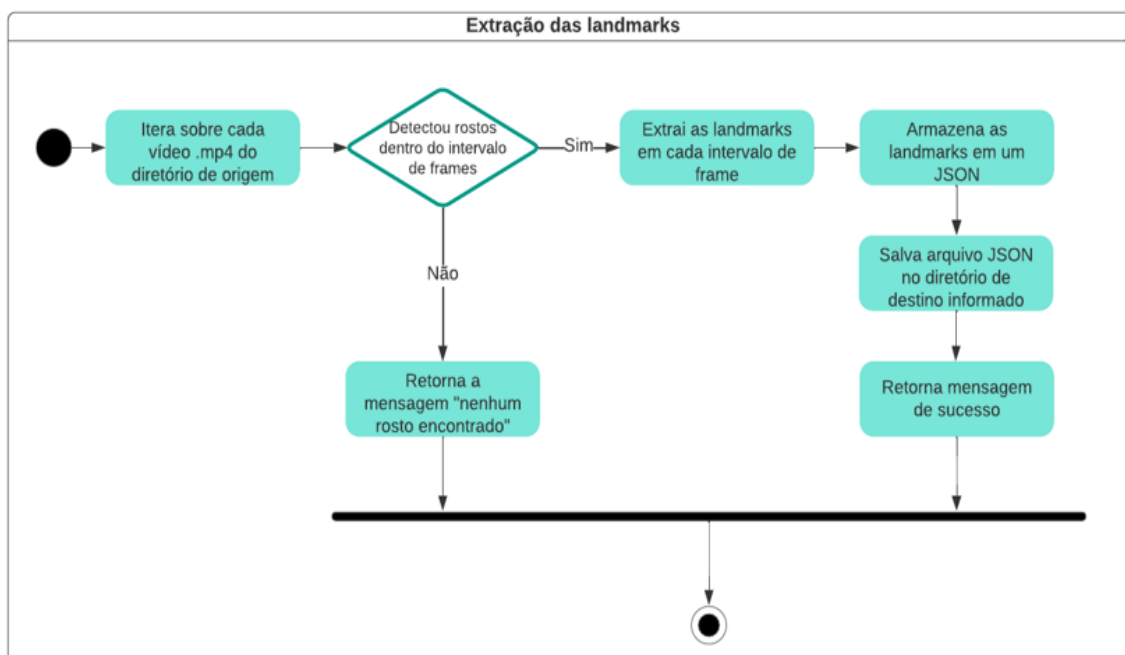


Figura 5 - Diagrama de atividade - Extração de landmarks.

3. Testes

Para validar o funcionamento da ferramenta conforme o esperado, foram realizados alguns testes automatizados. Para testar o comportamento da aplicação desktop gerada pelo Electron, utilizou-se o *framework* de testes Spectron⁵. A partir deste, foi possível checar se a janela da aplicação está visível para o usuário, se é instanciada e carregada corretamente e, por fim, se o nome da aplicação corresponde ao esperado. A Figura 6 abaixo demonstra o log dos testes realizados pelo Spectron.

```
PS D:\Meus Documentos\Mestrado\Projeto Final de Programação\landmarks-extractor> yarn test
yarn run v1.22.4
$ mocha

Landmarks Extractor launch
  ✓ Check if the window is visible
  ✓ Shows an initial window
  ✓ Has the correct title

3 passing (27s)

Done in 28.25s.
```

Figura 6 - Log de teste gerado pelo Spectron.

Além da interface gráfica da ferramenta, também foi realizado um teste unitário acerca da função principal da aplicação, a geração de *landmarks* faciais, executada pelo script *extract_landmarks.py*. Para isto, foi utilizado a biblioteca nativa do Python, *unittest*, e através dela, carregou-se o caminho de um diretório contendo vídeos *.mp4* para fins de testes e checkou-se o tipo de retorno gerado pela função principal do script, comparando-o com o tipo esperado, um *python dictionary*. A Figura 7 ilustra o log obtido durante o teste unitário do script Python.

```
PS D:\Meus Documentos\Mestrado\Projeto Final de Programação\landmarks-extractor\test> python test.py
.
-----
Ran 1 test in 12.619s

OK
```

Figura 7 - Log de teste gerado pelo unittest.

4. Documentação para o usuário

Nesta seção será apresentado um passo a passo acerca do processo de instalação e uso da aplicação. Vale ressaltar que a ferramenta proposta visa atender usuários pesquisadores que tem como objeto principal de seus estudos a face humana, tendo a necessidade de analisar e manipular as *landmarks* faciais através de modelos computacionais.

⁵ <https://github.com/electron-userland/spectron>

4.1. Instalação e configuração

Para a correta execução da ferramenta, é pré-requisito que o usuário tenha instalado no seu ambiente as tecnologias Python (recomenda-se a versão 3.8.10, a mesma utilizada na aplicação) e o NodeJS na sua versão LTS (*Long Term Support*). A instalação do Git⁶ é opcional, pois o usuário pode fazer o download do arquivo *.zip* contendo o código-fonte da ferramenta no link do repositório do projeto no GitHub⁷. Entretanto, caso opte pela instalação, é necessário executar o comando abaixo para obter a aplicação:

```
$ git clone https://github.com/BrunoGomes99/PFP_Landmarks_Extractor.git
```

Uma vez feito o download da ferramenta, o usuário deverá abrir o diretório raiz do projeto no terminal utilizado seguindo o comando:

```
$ cd PFP_Landmarks_Extractor
```

Caso tenha feito apenas o download do *.zip* o comando para o diretório raiz será o seguinte:

```
$ cd PFP_Landmarks_Extractor-main
```

Após estar devidamente localizado no diretório raiz da ferramenta, o usuário deverá instalar os pacotes do NodeJS contendo as dependências do projeto através do comando abaixo:

```
$ npm install
```

Se tudo ocorrer da forma esperada, a pasta *node_modules* será criada no diretório da aplicação. Em seguida, será necessário instalar as dependências do Python a fim de que o script principal do programa execute corretamente. Para isso, bastará o usuário executar o comando:

```
$ pip install -qr ./src/py/requirements.txt
```

Uma vez executados todos os comandos acima, a configuração da ferramenta estará completa. Sendo assim, o usuário será capaz de executar a aplicação através do comando:

```
$ npm start
```

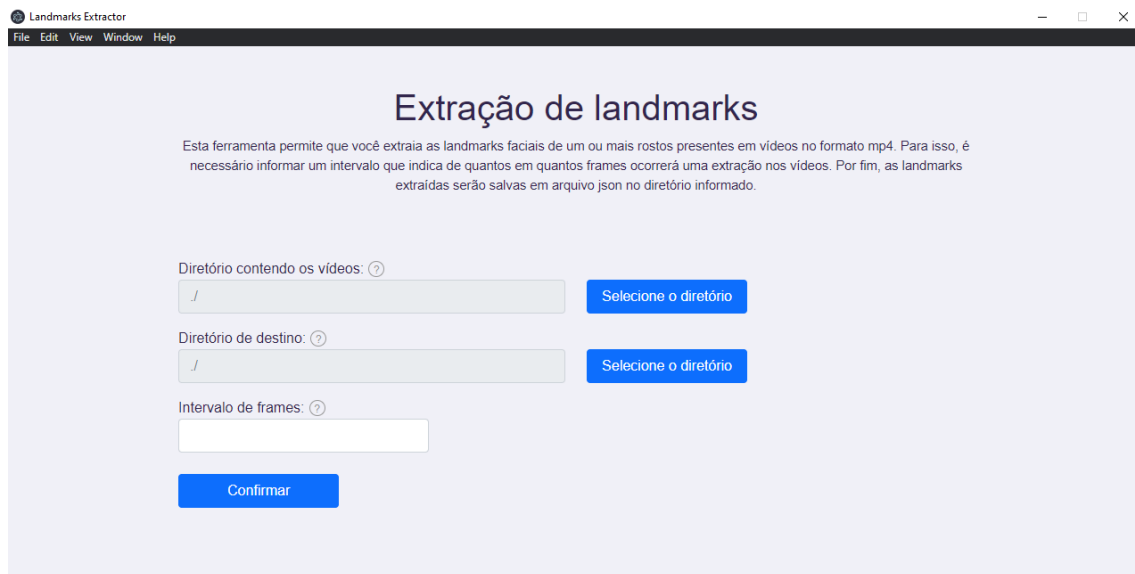
4.2. Instruções de uso

Após inicializar a ferramenta, a tela inicial, apresentada na Figura 8, será mostrada. Além do título e de uma breve descrição, esta tela também contém três campos obrigatórios: **diretório de origem**, que deverá contemplar os vídeos no formato *.mp4* a serem processados; **diretório de destino**, que consiste no local onde o arquivo *.json* com as *landmarks* resultantes será armazenado;

⁶ <https://git-scm.com>

⁷ https://github.com/BrunoGomes99/PFP_Landmarks_Extractor

intervalo de frames, que consiste em um número inteiro positivo que irá indicar de quantos em quantos frames a ferramenta realizará uma extração de *landmarks* faciais.



The screenshot shows the 'Landmarks Extractor' application window. The title bar includes 'Landmarks Extractor' and standard window controls. The menu bar contains 'File', 'Edit', 'View', 'Window', and 'Help'. The main content area has a heading 'Extração de landmarks' and a descriptive paragraph: 'Esta ferramenta permite que você extraia as landmarks faciais de um ou mais rostos presentes em vídeos no formato mp4. Para isso, é necessário informar um intervalo que indica de quantos em quantos frames ocorrerá uma extração nos vídeos. Por fim, as landmarks extraídas serão salvas em arquivo json no diretório informado.' Below this, there are three input fields: 'Diretório contendo os vídeos:' with a file icon, 'Diretório de destino:' with a folder icon, and 'Intervalo de frames:' with a numeric input field. Each of the first two fields has a blue button labeled 'Selecione o diretório' to its right. At the bottom, there is a blue button labeled 'Confirmar'.

Figura 8 - Tela inicial da ferramenta.

A tela inicial, possui ainda, um tratamento para impedir que parâmetros inválidos sejam passados ao script principal. O primeiro deles está na checagem se o diretório de origem informado pelo usuário contém pelo menos um vídeo no formato *.mp4*, pois caso contrário é exibido uma mensagem informando o erro, conforme mostrado na Figura 9. Além disso, também foi feita uma validação para evitar que os campos sejam passados vazios, bem como o intervalo de frames tenha valor zero, negativo ou decimal. A Figura 10 ilustra tais tratamentos.

The screenshot shows a web application window titled 'arks Extractor' with a menu bar containing 'View', 'Window', and 'Help'. The main heading is 'Extração de landmarks'. Below it, a paragraph explains the tool's purpose: extracting facial landmarks from MP4 videos. The form contains three input fields: 'Diretório contendo os vídeos:' with a value of './', 'Diretório de destino:' with a value of './', and 'Intervalo de frames:' which is empty. There are two blue buttons labeled 'Selecione o diretório' and one labeled 'Confirmar'. A red error message states: 'Nenhum arquivo .mp4 encontrado. Por favor, selecione um diretório válido.'

Figura 9 - Mensagem de vídeo .mp4 não encontrado.

This screenshot shows the same application window after some input. The 'Diretório contendo os vídeos:' and 'Diretório de destino:' fields now contain './'. The 'Intervalo de frames:' field contains '-2'. The 'Selecione o diretório' buttons are still present. Two red error messages are displayed: 'É necessário selecionar um diretório válido.' under the first directory field and 'O intervalo de frames deve ser um número inteiro positivo.' under the frame interval field. The 'Confirmar' button remains at the bottom.

Figura 10 - Validação de campos obrigatórios.

Após um preenchimento válido de todos os campos da tela inicial, o usuário deverá clicar no botão **confirmar**, o que o levará para a tela de carregamento da ferramenta, mostrado na Figura 11, onde o mesmo deverá esperar até que o processamento de todos os vídeos seja concluído.

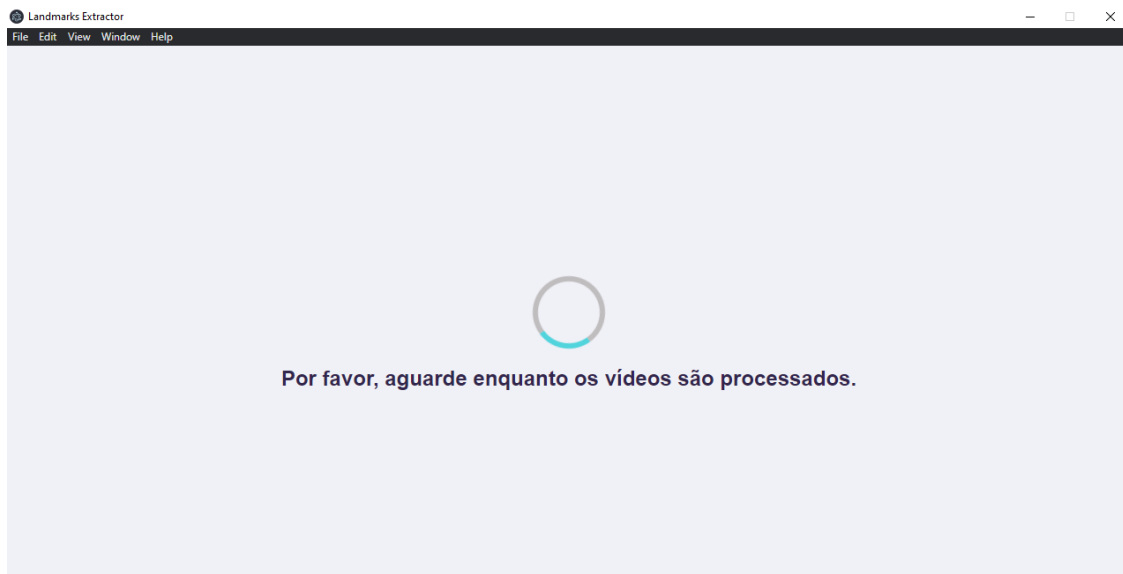


Figura 11 - Tela de carregamento da ferramenta.

Uma vez concluído todo o processo, a ferramenta irá exibir a tela final, que irá conter uma mensagem que informa ou o sucesso da extração das *landmarks* (Figura 12) ou que nenhum rosto foi encontrado dentro do intervalo informado nos vídeos analisados (Figura 13). Caso o último aviso for exibido, é possível que de fato não existam faces humanas presentes em nenhum momento dos vídeos enviados pelo usuário, assim como também há a possibilidade de que o intervalo informado por este tenha pulado alguns frames onde os rostos aparecem nos vídeos. Nesse caso, é recomendado que o usuário tente novamente com um intervalo menor a fim de considerar mais frames do vídeo. Entretanto, vale ressaltar que quanto menor o intervalo de frames, maior será o tempo de processamento dos vídeos, embora mais frames, e consequentemente, *landmarks* serão consideradas para a extração.

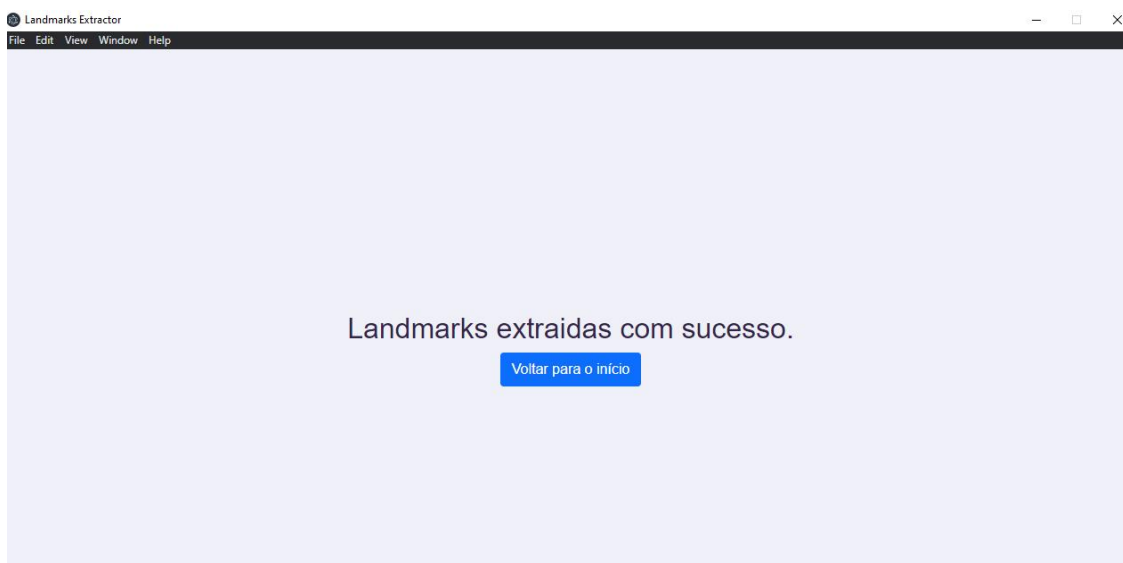


Figura 12 - Tela final - Retorno de sucesso.

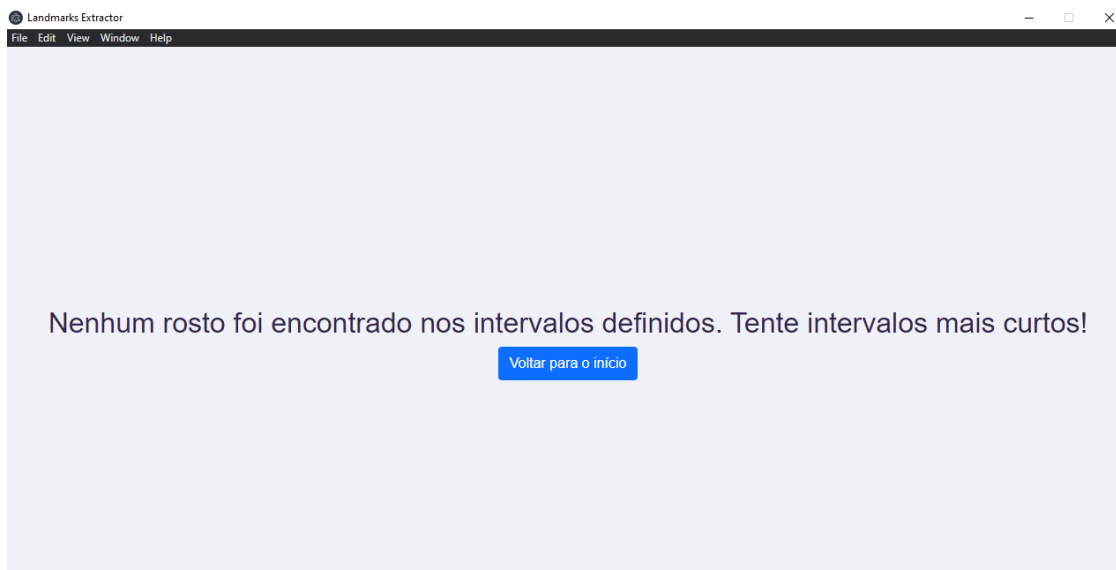


Figura 13 - Tela final - Retorno de rosto não encontrado.

5. Colaboração científica

A ferramenta apresentada foi desenvolvida com o intuito de facilitar a obtenção de uma das informações mais cruciais dentro do contexto de detecção e reconhecimento facial através de *machine learning*, as *landmarks*. A aplicação foi projetada tendo em vista contribuir com o desenvolvimento de um projeto de dissertação sobre detecção de *deepfakes*, vídeos maliciosos que substituem o rosto original de uma pessoa por outra, geralmente colocando esta última dentro de um discurso fora de contexto. Dessa forma, para essa tarefa é essencial que o modelo treinado seja capaz de capturar não só as imagens das faces presentes nos vídeos em cada frames, mas também suas *landmarks* faciais, uma vez que estas oferecem as coordenadas precisas de elementos como olhos, nariz e boca, possibilitando investigar possíveis vestígios de manipulação dentro dessas regiões.

Apesar de pensado no contexto de *deepfakes*, a ferramenta faz-se útil para qualquer outro estudo científico onde a face humana é o principal objeto de estudo. Além disso, visto que os *datasets* voltados para esse fim raramente disponibilizam as *landmarks* em suas anotações, a ferramenta permite que grandes volumes de dados tenham tal informação extraída com poucas interações do usuário e através de uma interface gráfica simples e amigável. Dessa forma, o projeto irá beneficiar o trabalho de outros pesquisadores que necessitam do uso de *landmarks* faciais.

Embora a ferramenta ofereça praticidade para a anotação de tais *datasets* e tenha se mostrado eficiente, o não reconhecimento das faces, e consequentemente das *landmarks*, em vídeos ou imagens de baixa qualidade, iluminação e com obstruções do rosto humano presente, ainda pode ser um problema para o pesquisador, uma vez que a falta de informações nessas circunstâncias pode comprometer o resultado da pesquisa de um outro usuário.