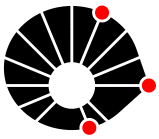
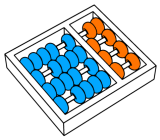




Sistema de coleta, organização e validação de assinaturas

J. L. Sardinha *P. Duduch* *E. Borin*

Relatório Técnico - IC-PFG-19-64
Projeto Final de Graduação
2019 - Novembro

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

Sistema de coleta, organização e validação de assinaturas

Joshua Sardinha* Pietro Duduch* Edson Borin*

Resumo

Este trabalho se propôs a construir um sistema computacional de criação e análise de formulários de presença de alunos em aulas. O produto final é um programa que permite a automação do processo de coleta e facilita a autenticação de assinaturas quando há uma base de comparação. As principais ferramentas utilizadas envolveram as técnicas tradicionais de Processamento de Imagens em todas as partes do processo, envolvendo secundariamente algoritmos de reconhecimento e validação de texto por redes neurais. Em seu estado final, o sistema realizou com sucesso a extração das células dos formulários e dos dados contidos nelas, obtendo uma acurácia de aproximadamente 99% na contabilização da frequência dos alunos em formulários de teste, quando utilizado com imagens de qualidade como entrada. Assim, foi possível tornar mais eficiente o processo de contagem e registro das presenças em sala de aula.

1 Introdução

A presença em aulas é tida como de fundamental importância para o aprendizado dos alunos. A correlação entre um bom desempenho universitário e a frequência em aulas é evidenciada por um estudo realizado pela Georgia College & State University[1] com estudantes de ciência da computação. No estudo foi mostrada forte correlação entre a nota no geral e a ausência, especialmente no primeiro dia de aula. Alunos que possuem presença deficiente, independente de graduados ou não, tendem a possuir notas piores.

Tão evidente é sua importância, que é prevista por lei. Durante o período escolar, é prevista uma frequência obrigatória de 75% para os alunos. Já no ensino superior, é previsto apenas que os estudantes devem ter frequência nas aulas:

> LDBE - Lei nº 9.394 de 20 de Dezembro de 1996
> Estabelece as diretrizes e bases da educação nacional.
> Art. 47. *Na educação superior, o ano letivo regular, independente do ano civil, tem, no mínimo, duzentos dias de trabalho acadêmico efetivo, excluído o tempo reservado aos exames finais, quando houver.*
> § 3º *É obrigatória a freqüência de alunos e professores, salvo nos programas de educação a distância.* [2]

*Instituto de Computação, Universidade Estadual de Campinas, 13081-970 Campinas, SP.

Deste modo, cabe às próprias instituições de ensino superior determinarem uma frequência mínima, sendo que a maioria acaba adotando a mesma taxa de 75%, inclusive a UNICAMP.

Assim, este projeto procurou automatizar e facilitar o processo de recolhimento de assinaturas, e contabilização e validação destas, como forma de incentivo ao comparecimento nas aulas e melhora da taxa de frequência dos alunos.

2 Justificativa

Apesar da importância do comparecimento às aulas, é conhecido que diversos alunos deixam de aproveitar a oportunidade de adquirir o conhecimento durante a aula, o que acaba prejudicando seu desempenho universitário. Além disso, é conhecido que os professores precisam atestar presença para os alunos e, em turmas muito grandes, é comum o uso de listas de presença onde são coletadas as assinaturas dos alunos para posterior verificação.

O próprio processo de recolhimento das assinaturas por parte do professor está sujeito a erros durante o processo de contabilização das presenças dos alunos, além de ser uma atividade extenuante e repetitiva, já que o professor precisa inspecionar a assinatura de cada aluno para contabilizar as faltas de cada aula manualmente. Considerando que existem turmas que possuem mais de cem alunos, fica evidente o trabalho e o tempo que esta atividade consome. Assim, se fosse possível realizar a obtenção das assinaturas e contabilização das faltas e presenças de cada aluno de uma forma automatizada, de modo a agilizar e tornar o processo mais eficiente, conseguindo detectar eventuais fraudes nas assinaturas, poderíamos ter uma contabilização mais precisa das faltas, um incentivo ao comparecimento nas aulas por partes dos alunos, e, consequentemente, um maior aproveitamento das aulas e melhor desempenho universitário destes.

3 Objetivos

Este projeto teve como objetivo automatizar e facilitar o processo de recolhimento das assinaturas, principalmente pelos professores. O produto final visado é um *software* com múltiplas funcionalidades, entre elas, ser capaz de:

- Receber uma lista com os identificadores (RA, no caso da Unicamp) e nomes dos alunos , armazená-la e gerar um formulário de assinaturas com os identificadores, ou RAs, e nomes dos alunos;
- Processar um formulário de autenticação com as assinaturas originais de todos os alunos, e extrair e armazenar as imagens dessas assinaturas para posterior autenticação;
- Processar um formulário de assinaturas correspondente ao formulário de uma das aulas para identificar e armazenar informações sobre os alunos presentes e ausentes e para possivelmente identificar fraudes em assinaturas;
- Gerar, com os dados armazenados, estatísticas das aulas até o momento com as tabelas de frequência, histogramas dos alunos presentes e o gráfico de evolução das presenças.

O *software* final pode gerar produtos intermediários, como as imagens das células do formulário isoladas; e saídas também primariamente já desejadas, como as tabelas de frequência. O próprio código desenvolvido pode servir como objetivo secundário de início de um sistema de automação do processo de colhimento de assinaturas, podendo ser futuramente aperfeiçoado para uso geral.

4 Desenvolvimento do Trabalho

4.1 Divisão do Problema

O problema foi separado em segmentos menores para uma melhor abordagem e implementação da solução. A primeira parte do problema consistia em gerar um formulário de presença para os alunos assinarem. Era necessário que a tabela do formulário contivesse um campo mostrando os RAs dos alunos e campos em branco para serem preenchidos com as assinaturas. Tais campos precisavam ser razoavelmente grandes para que fosse possível assinar no centro destes, de modo que as bordas da célula obstruissem o mínimo possível seu conteúdo. Também era necessário que a tabela tivesse um cabeçalho onde seriam preenchidas a data e a turma da matéria que está sendo dada.

A segunda etapa consistia na extração e recorte de cada célula do formulário para que fosse possível a utilização de seus conteúdos. Para tal, implementamos um *kernel* horizontal e aplicamos um certo número de erosões na imagem scaneada do formulário. Em seguida, aplicamos o mesmo número de dilatações na imagem para obter as linhas horizontais da tabela. Utilizamos o mesmo processo com um *kernel* vertical para obter as linhas verticais da tabela e a somamos com as horizontais para obter o *grid* da tabela. Obtido o *grid*, utilizamos métodos da biblioteca OpenCV para achar os contornos deste e obter as imagens correspondentes a cada célula da tabela. Finalmente, separamos cada imagem de cada célula e as agrupamos linha a linha.

Para a terceira etapa era necessário extrair as informações das imagens obtidas na etapa anterior. Para tal, começamos tratando as imagens extraídas das células. Transformamos cada imagem de modo que passassem a possuir apenas níveis de cinzas e aplicamos então uma limiarização nela para que tivesse apenas pixels pretos ou brancos. Então aplicamos métodos de processamento de imagens para remover os contornos das bordas que ainda apareciam na imagem. Após isso, determinamos se a célula possuía ou não uma assinatura. Caso possuísse, era aplicado o método para validar sua assinatura com a de base presente no banco de dados. Dependendo do resultado do método, a presença seria contabilizada ou não. Nos casos de células que possuiam o número do RA do aluno, utilizamos de métodos de *OCR* para reconhecer o número presente na imagem e o armazenar no banco de dados, relacionando-o com a imagem da assinatura do respectivo aluno presente no formulário.

Finalmente, a última etapa consitia no cálculo de algumas estatísticas com os dados obtidos ao longo do processo. Após calculadas tais estatísticas, deveríamos mostrá-las através de uma interface gráfica contendo tais informações, e alguns gráficos ilustrando os dados e padrões da turma.

4.2 Ferramentas Utilizadas

No processo de desenvolvimento do trabalho e de criação do programa responsável pela análise dos formulários, múltiplas ferramentas computacionais foram utilizadas, em sua maior parte de análise e processamento de imagens. A linguagem escolhida para o desenvolvimento foi Python e a versão utilizada foi a Python 3.7.4.

4.2.1 Processamento de Imagens

Dentro da abrangente área de Processamento de Imagens, algumas técnicas específicas foram utilizadas para os propósitos de separação das células, aumento de nitidez, limpeza de ruídos, entre outros. A maior parte dos algoritmos utilizados pertencem à classe do Processamento de Imagens tradicional, o que inclui operadores morfológicos, como erosões e dilatações; limiarizações, borramentos e realces por filtragem; e operações de adição ou subtração entre duas imagens. Na maior parte das operações, foi utilizada a biblioteca OpenCV, em Python, importada como *cv2*. Essa seção é voltada para a descrição das ferramentas utilizadas.

Operadores Morfológicos Morfologia matemática é uma teoria e técnica para a análise e processamento de estruturas geométricas, baseada na teoria de conjuntos e funções aleatórias e é frequentemente aplicada ao processamento de imagens [3]. Os operadores mais básicos da aplicação da teoria são erosão, dilatação, abertura e fechamento. Os operadores mais utilizados no projeto são os dois primeiros, sendo também aplicada uma espécie de abertura.

A ideia básica da morfologia binária é sondar uma determinada imagem com um outro elemento simples, utilizando critérios para determinar como o formato desse elemento se encaixa ou escapa à imagem. Esse mesmo formato é chamado de **elemento estruturante** e é, também, uma imagem binária.

Os operadores básicos são fortemente relacionados com a adição ou subtração de Minkowski e utilizam do elemento estruturante na realização das operações. A erosão de uma imagem binária A pelo elemento estruturante B, voltada para a subtração de Minkowski, é definida por:

$$A \ominus B = \{z \in E | B_z \subseteq A\}$$

Em que B_z é a translação de B pelo vetor z, ou seja, $B_z = \{b + z | b \in B\}$. Quando o elemento estruturante B possui um centro e o centro está localizado na origem de E, então a erosão de A por B pode ser compreendida como o *locus* dos pontos pelo centro de B quando B se move em A.

Em uma noção similar, agora com a adição de Minkowski, podemos descrever o operador de dilatação:

$$A \oplus B = \bigcup_{a \in A} B_a$$

Se B possui um centro na origem, como anteriormente, então a dilatação de A por B pode ser compreendida como o *locus* dos pontos cobertos por B quando o centro de B se move dentro de A.

De uma forma geral, no projeto, os operadores morfológicos foram utilizados para erodir iterativamente objetos em imagens de forma que alguns objetos menores fossem eliminados e outros, maiores, fossem mantidos. Dessa forma, após um número idêntico de dilatações, os objetos mantidos retornam a um formato semelhante ao seu original e, por fim, são segmentados.

Limiarização e Conversão de Cores O processo de limiarização consiste em transformar o espectro de tons de uma imagem, que normalmente possui 256 tonalidades, a um espectro reduzido. Na maior parte das vezes o novo espectro possui 2 tons, gerando uma imagem binária. A binarização da imagem pela técnica de limiarização procura segmentar algum objeto, ou informação, em uma imagem e o resultado é alcançado pela comparação de cada pixel da imagem com um valor de limiar. O valor de limiar pode ser global, definido estaticamente para todos os pixels da imagem, ou local, calculado usando como parâmetros uma vizinhança de cada um dos pixels. Um exemplo de limiarização local é o cálculo da mediana dos valores dos tons dos pixels em uma vizinhança-8 de um pixel. Quando o valor do pixel é maior que essa mediana, seu novo valor é máximo, ou então, quando sua intensidade é menor que a mediana, seu novo valor é o mínimo.

No projeto a limiarização foi utilizada para a facilitação do processo de identificação de bordas e contornos, principalmente na identificação da grade de assinaturas. Foi também utilizado nas imagens de assinaturas a fim de acentuar a escrita e possibilitar a contagem da taxa de pixels pretos de cada imagem. Com os mesmos fins, também foram transformadas imagens coloridas originalmente em RGB para imagens em escala de cinza.

Filtragem em Frequência Filtros de frequência processam uma imagem no domínio da frequência. A imagem é passada por uma transformação de Fourier, multiplicada pela função filtro e então transformada de volta para o domínio espacial[4]. A atenuação de frequências altas resulta em imagens suavizadas, ou borradas, no domínio espacial, e a atenuação de baixas frequências realça as bordas. Todos os filtros de frequência podem ser implementados no domínio espacial e, se existe um *kernel* para o filtro desejado, é computacionalmente mais barato o processamento da filtragem no domínio espacial. Caso nenhum *kernel* correspondente seja encontrado para a filtragem, a realização desta no domínio da frequência é mais apropriada e pode ser mais eficiente.

A filtragem em frequência é baseada na Transformada de Fourier. O operador normalmente recebe a imagem e a função de filtragem no domínio de Fourier. Essa imagem, então, é multiplicada pelo filtro pixel a pixel, seguindo a lógica:

$$G(k,l) = F(k,l)H(k,l)$$

Em que $F(k,l)$ é a imagem de entrada no domínio de Fourier, $H(k,l)$ a função de filtragem no domínio de Fourier e $G(k,l)$, a imagem resultante filtrada. Para obter a imagem resultante no domínio espacial, $G(k,l)$ precisa ser transformada novamente usando a Transformada inversa de Fourier. Como a multiplicação no espaço de Fourier é idêntica ao processo de Convolução no domínio espacial, os filtros de frequência podem ser implementados como um filtro espacial. No projeto a filtragem em frequência ocorreu no domínio

espacial, por meio da convolução do *kernel* nas imagens. A filtragem de passa baixas foi utilizada para o realce de bordas e a facilitação do reconhecimento de contornos de objetos nas imagens dos formulários, mais especificamente as linhas da grade. Também foi utilizado um filtro que atenuou as frequências altas nas imagens resultantes das assinaturas, especificamente o filtro da mediana, a fim de remover o ruído sal-pimenta.

Operações entre imagens Outro processo recorrentemente utilizado foi a adição e subtração entre duas imagens. As operações de adição ou subtração são tipos de sobreposição de imagens que resultam em outra imagem, de forma que, pixel a pixel, os valores dos tons são aritmeticamente somados ou subtraídos. A operação pode ser ponderada, atenuando mais o sinal de uma imagem ou outra na imagem resultante.

No projeto, a adição de imagens foi utilizada para recompor a grade de assinaturas, após a segmentação das linhas verticais e horizontais em duas diferentes imagens. Também foi utilizada a subtração para a remoção de margens em células isoladas, sendo a subtração entre a imagem original e uma imagem com a margem segmentada, tendo como resultante a imagem com o conteúdo original sem margens.

4.2.2 Redes Neurais

Para realizar a autenticação das assinaturas, ou seja, verificar se um par de assinaturas são iguais, foi utilizada uma Rede Neural Siamesa, também chamada de *SNN* (do inglês, *Siamese Neural Network*).

Redes Neurais são conjuntos de algoritmos, baseados no funcionamento do cérebro humano, que são projetados para reconhecer padrões. Elas interpretam dados através de um tipo de percepção de máquina, classificando ou agrupando entrada bruta. Os padrões que elas reconhecem são numéricos, contidos em vetores, nos quais todo tipo de dados do mundo real, seja imagens, sons ou texto, devem ser traduzidos. [6] Já uma *SNN* é um tipo específico de Rede Neural, que tem sua estrutura mostrada na Figura 1.

Neste modelo de Rede Neural, as duas redes mostradas acima não são redes diferentes, mas duas cópias da mesma rede, por isso o nome "Rede Siamesa". Elas basicamente compartilham os mesmos parâmetros. As duas imagens de entrada, $x1$ e $x2$, são passadas para a rede convolucional para gerar um vetor de características de tamanho fixo ($h(x1)$ e $h(x2)$). Assim, caso a rede seja treinada apropriadamente, se as duas imagens forem da mesma assinatura então os vetores de suas características devem ser parecidos também. Caso as duas imagens sejam de assinaturas diferentes (ou, no caso, assinaturas de pessoas diferentes) então seus vetores de características também serão diferentes. Logo, a diferença absoluta entre tais vetores será claramente diferente nos dois casos acima, e, consequentemente, o *score* de similaridade gerado pela saída da camada com a função Sigmoide também será diferente nos dois casos.[7] Caso o *score* gerado seja menor que um certo limiar L, consideraremos as assinaturas como iguais (escritas pela mesma pessoa). Caso seja maior que tal limiar, consideraremos que as assinaturas podem ter sido geradas por alunos diferentes.

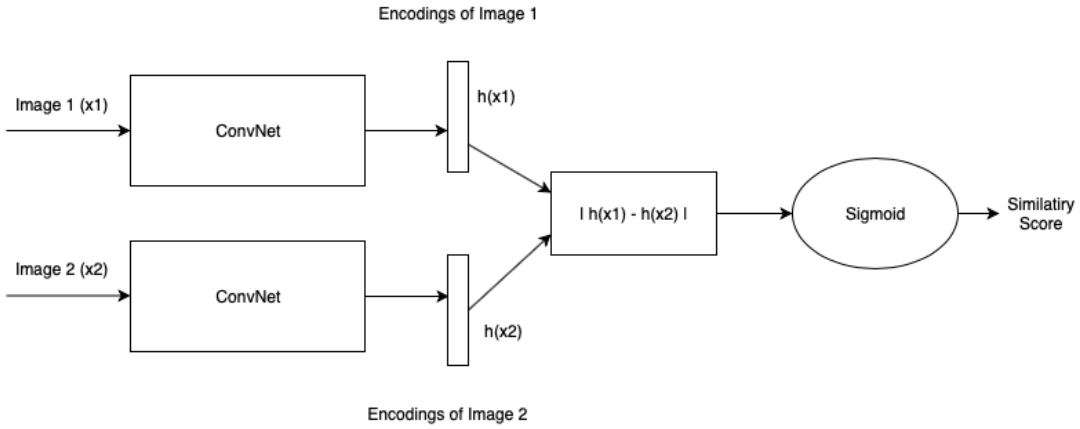


Figura 1: Ilustração da estrutura de uma *SNN*

4.2.3 Reconhecimento de Caracteres

OCR, do inglês *Optical Character Recognition*, é a tecnologia para reconhecimento de texto e caracteres, sejam impressos ou escritos à mão, dentro de imagens digitais, como fotos e documentos scaneados. O processo usual de *OCR* envolve a examinação do texto de um documento e na tradução dos caracteres presentes na imagem em código que pode ser usado para processamento de dados.[8]

Assim, com tal tecnologia, é possível traduzir de uma imagem números e texto que não possuem um significado inteligível para o computador em informação que este possa compreender e manipular.

4.2.4 Banco de Dados

Um banco de dados é uma coleção organizada de informação estruturada, ou dados, tipicamente armazenados eletronicamente em um sistema computacional. Ele é usualmente controlado por um *DBMS* (do inglês, *database management system*). Juntos, os dados, o *DBMS*e as aplicações que estão associadas com eles, são referidos como um Sistema de Banco de Dados, ou abreviados para Banco de Dados. Dados, nos tipos de Bancos de Dados mais comuns hoje em dia, são normalmente modelados em colunas e linhas em uma série de tabelas que visam tornar o processamento e busca dos dados mais eficaz. A maioria dos Bancos de Dados usa linguagem de busca estruturada (SQL) para escrever e buscar dados.[9]

No projeto foi usado **SQLite**, uma biblioteca em *C* que implementa uma pequena, rápida, auto-contida e confiável *engine* para banco de dados SQL[10]. Apesar de sua implementação em *C*, foi utilizado um *framework* para Python chamado **SQLite3**.

4.2.5 Interface

Para permitir uma revisão da análise feita pelo programa por parte do professor, foi criada uma interface que permite alterar os resultados obtidos e gerar uma tabela final. Para mostrar a tabela com imagens, botões e uma tabela interativa foi utilizada a biblioteca de Python chamada Kivy.

A biblioteca constrói a interface utilizando módulos de *layout* já existentes, como *BoxLayouts*, *Labels*, *ScrollViews* e *Buttons*, e permite a criação de módulos personalizados, como o *TableRow*, ou o *StudentHeader* do projeto. A janela gerada tem suporte para diferentes sistemas operacionais, sejam Windows, Linux ou outros.

4.2.6 Outras Bibliotecas

A fim de gerar o arquivo em PDF para o formulário de assinaturas de uma classe, foi importada a biblioteca *reportlab*, principalmente os módulos *platypus* e *lib*. A documentação [5] da biblioteca demonstra os passos de como gerar o estilo do formulário, de forma a criar um cabeçalho dinâmico e um número de linhas variável.

Outras bibliotecas de Python importadas foram a *matplotlib*, para a construção dos gráficos; a *numpy*, para facilitação da manipulação dos valores numéricos e das estruturas de dados; a biblioteca *os* para análise de arquivos; e as bibliotecas *sys* e *argparse* para leitura dos parâmetros de entrada do programa.

4.3 Aplicação dos Algoritmos e Ferramentas

4.3.1 Criação do Fluxo Principal

O passo inicial no desenvolvimento do programa constituiu na criação de um fluxo principal para a definição e unificação dos comandos, além da definição de quais seriam os dados necessários à boa execução do programa e da definição da estrutura em que esses seriam armazenados de forma persistente.

O fluxo principal do programa foi mantido no arquivo nomeado *main.py*, o qual recebe os argumentos passados e os valida de acordo com uma lista de possíveis comandos e, posteriormente, de acordo com a existência ou não de outros arquivos e dados salvos. Os possíveis comandos foram definidos de forma a serem esperados logo depois no nome do programa. A execução esperada do programa, portanto, deve seguir o modelo:

```
python3 main.py [comando] [parâmetros do comando]
```

Os possíveis comandos e suas descrições seguem:

- **create-class**: Recebe o parâmetro **nome da classe** e o **arquivo CSV** com os alunos listados por RA e nome. Cria a classe no banco de dados, assim como cada um dos alunos, e salva um arquivo PDF com o formulário correspondente à classe.
- **insert-auth-form**: Recebe o parâmetro **nome da classe**, de uma classe já inserida, e a **imagem de um formulário** com a assinatura real de todos os alunos, sem

exceção. Salva as células das assinaturas autenticadas e adiciona seus caminhos ao aluno correspondente no banco de dados. Esta opção deve ser usada caso o professor tenha o formulário com todas as assinaturas de antemão.

- **add-form**: Recebe o parâmetro **nome da classe**, de uma classe já inserida, a **imagem de um formulário** com a assinatura dos alunos que estavam presente na classe e a **data** da aula. Salva as células das assinaturas, identifica quais alunos estavam presentes ou ausentes e, caso haja um formulário de autenticação inserido para a classe, procura autenticar as assinaturas. Por fim, salva todas as informações de presença e similaridade de assinaturas dos alunos desse dia de classe no banco de dados. Este comando pode ser invocado múltiplas vezes para uma mesma data, no caso de a lista de assinaturas possuir mais de uma página. Neste caso, deve ser chamado uma vez para cada página.

- **statistics**: Recebe o parâmetro **nome da classe**. Imprime as taxas de frequência de cada um dos alunos, junto com as taxas de assinaturas similares às assinaturas base (caso o formulário de autenticação tenha sido processado), e gera um histograma da frequência de alunos pelo número de presença nas aulas, bem como um gráfico com a evolução do número de presentes em cada classe para todos os formulários inseridos até o momento. Por fim, gera uma janela com uma tabela de todos os alunos da classe, mostrando, para cada data, a imagem do RA no formulário, a imagem da assinatura, a presença, similaridade e o resultado final. Pode-se alterar os valores por essa janela e gerar um arquivo CSV com o resultado final da tabela.

- **classes**: Não recebe nenhum parâmetro e imprime todas as classes já criadas.

- **clear**: Não recebe nenhum parâmetro e limpa completamente o banco de dados.

Os parâmetros de nome de classe, quando necessários, devem sempre suceder o sinalizador $-n$. Da mesma forma, os arquivos CSV e as imagens de formulários devem suceder um sinalizador $-f$. Por fim, as datas recebidas devem suceder um sinalizador $-d$.

4.3.2 Estruturação do Banco de Dados

O banco foi planejado para armazenar quatro tabelas:

- **classes**: Armazena as classes criadas, seus nomes e a menor taxa de pixels pretos das assinaturas autenticadas da classe.

- **forms**: Armazena os formulários inseridos, a classe a qual esse formulário pertence e a data da aula correspondente.

- **students**: Armazena os alunos inseridos por alguma classe, bem como seus RAs, como identificadores, seus nomes, e possivelmente o caminho para alguma assinatura de autenticação.

- **signatures**: Armazena as assinaturas identificadas. Relaciona a assinatura a algum formulário já inserido e a outro aluno já inserido. Salva também se essa assinatura é uma presença ou ausência, se está validada de acordo com uma assinatura padrão de autenticação também já inserida e o resultado final de presença do cruzamento dessas duas informações.

Com isso, é obtido o diagrama do *schema* na Figura 2.

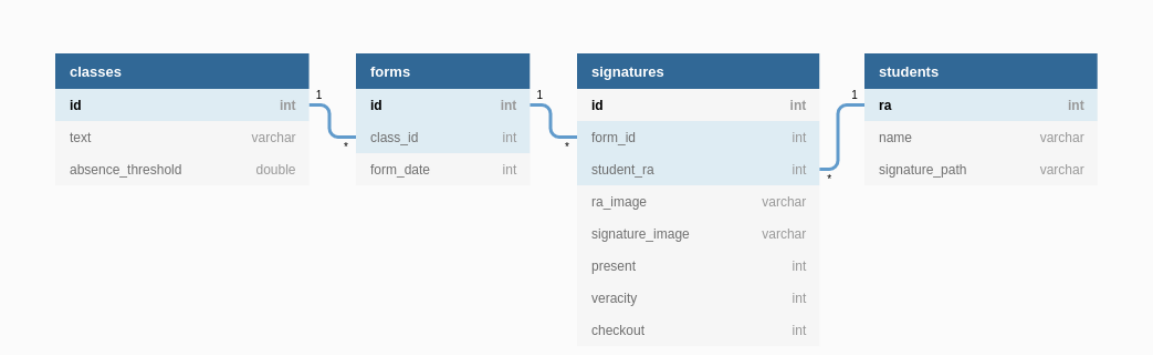


Figura 2: Diagrama do Banco de Dados da aplicação

4.3.3 Geração do Formulário

A função responsável pela geração do formulário foi colocada em um arquivo à parte, nomeado *generate_signing_sheet.py*. A função definida por *generate* é chamada no método *createClass*, após a inserção da nova classe no banco de dados.

O método recebe o arquivo em CSV com a tabela de relação dos alunos e lê o arquivo, criando uma lista de tuplas de RA e nome de cada aluno. Posteriormente, o código ordena a lista de alunos por RA e produz os dados de texto do cabeçalho do formulário, que conterá o *Curso*, com o nome do curso, um campo *Turma*, a ser preenchido e um campo *Data* também a ser preenchido. O primeiro cabeçalho só será inserido na primeira página do formulário.

São criados, então, os dados do segundo cabeçalho, que indicará o conteúdo de cada uma das colunas da tabela de assinaturas. A tabela terá dois conjuntos de três colunas, indicando, cada uma, o RA, nome e o campo de assinatura de um aluno, respectivamente. Com isso, cada linha contém informação de dois alunos. Seguindo a documentação do *ReportLab* [5], foram gerados os dados correspondentes às alturas das células e, então, à maior parte do estilo da grade.

A espessura das linhas foi escolhida de forma a ser mais grossa do que os traços de uma caneta, para que a extração das células fosse facilitada posteriormente. A fonte escolhida, Helvetica, também foi escolhida porque obteve maior acurácia na leitura com a utilização do OCR. Por fim, também a altura padrão, já pré determinada, foi selecionada de forma a dar espaço para que as assinaturas dos alunos não cruzem os limites verticais. Finalmente, os objetos tabela foram construídos e o PDF é gerado com o nome da classe recém inserida.

4.3.4 Extração das Células

A extração das células é implementada no método *extract*, do arquivo *extract_table_cells.py*. O arquivo *main.py* chama tal método dentro das funções *addForm* e *insertAuthForm* após a obtenção da classe e turma no banco de dados. O método recebe como parâmetro o caminho da imagem da qual será extraída o formulário e cada célula e carrega a imagem em memória para a manipulação desta. Chamamos então a função *extract_table_cells*, passando para ela a imagem que carregamos em memória. A função então, primeiramente, aplica uma limiarização na imagem de modo que esta fique apenas com pixels brancos e pretos. Após isto, definimos dois *kernels*, um vertical e um horizontal, erodimos a imagem 3 vezes e depois a dilatamos 3 vezes verticalmente para obter as linhas verticais da tabela, e fazemos o mesmo com o *kernel* horizontal para obter as linhas horizontais da tabela.

Após isto, aplicamos uma função nas imagens resultantes para transformá-las em imagens que possuem apenas níveis de cinza. Utilizamos, então, a função *findContours* do OpenCV para achar os contornos das linhas verticais e horizontais e determinar a intersecção destas, achando então o *grid* da tabela. Chamamos então a função *filter_out_cells* que remove as células que são muito grandes ou muito pequenas do resultado. Em seguida, chamamos a função *group_by_row_and_sort_by_column* que itera nas células encontradas e as separa por linhas da tabela. Então, para cada célula, chamamos a função *removeMargins* para remover eventuais bordas remanescentes da tabela após a extração das células, a fim de conseguir um resultado melhor diminuindo a quantidade de ruído na imagem e informações que não serão utilizadas. Assim, ao término da execução deste método, temos cada célula extraída, filtrada e separada por linha.

4.3.5 Extração das Informações

A extração das informações é realizada no próprio arquivo *main.py*, especialmente nas funções *insertAuthForm* e *addForm*. Após a recepção de um nome de classe válido, de um arquivo de imagem com as assinaturas também existente e da extração das células da imagem, inicia-se uma iteração sobre as células. O nome das imagens das células segue um padrão numérico de acordo com as colunas, de forma que sempre a 1^a e a 4^a colunas contenham o RA de um aluno e sempre a 3^a e a 6^a colunas contenham a assinatura do aluno da mesma linha. Com isso, durante a iteração, são registradas as células com os RAs dos alunos e as imagens são abertas e lidas com o OCR. Caso um número seja detectado, o número é considerado o RA de um aluno do formulário e, então, passa-se a ler a imagem correspondente à sua assinatura.

Para a detecção de presença de um aluno, é utilizada a célula com o conteúdo de sua assinatura, caso exista. A imagem é lida, suas cores são convertidas para escala de cinza, ela é limiarizada localmente por uma função da biblioteca OpenCV, utilizando a limiarização local gaussiana e, por fim, é filtrada com atenuação de altas frequências por um filtro de mediana, a fim de remover ruído sal-pimenta [11]. Finalmente, com a imagem resultante, é calculada a taxa de pixels pretos em relação ao total de pixels. Essa taxa é comparada a uma taxa limiar de 1% e, quando maior, é detectada uma presença de assinatura, quando menor, é detectada uma ausência. A Figura 3 exemplifica uma célula de assinatura original

e a Figura 4 exemplifica a mesma célula após o tratamento.

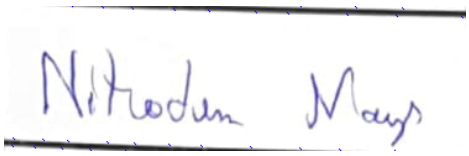


Figura 3: Célula de assinatura retirada de um formulário

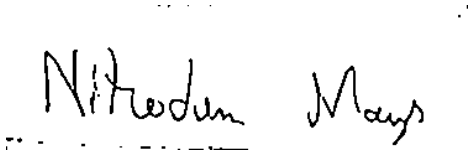


Figura 4: Imagem de assinatura resultante da qual é retirada a taxa de pixels pretos.

No caso em que existam alunos que possuam uma imagem com assinatura de autenticação salva, também é calculada a veracidade da assinatura utilizando a rede neural pré treinada. Finalmente, com as informações de presença e veracidade de assinatura, são inseridos no banco os dados de assinatura do novo formulário.

4.3.6 Geração das Estatísticas

A geração das estatísticas de presença se resume na colheita de todos os dados de assinaturas dos alunos já armazenados nos bancos e na construção de três saídas: a tabela com as frequências, o histograma dos alunos presentes e o gráfico de evolução das presenças.

Com uma classe válida, o primeiro passo é a retirada do campo *id* do objeto *classe* salvo no banco de dados. Com o identificador, então, uma segunda busca é realizada, de forma a retornar todos os formulários salvos daquelas classe. Uma terceira consulta é feita, buscando todas as assinaturas armazenadas desses formulários. Com essas consultas, tem-se, por fim, todas as assinaturas de todos os alunos já armazenadas para essa classe. Iterando sobre a lista de assinaturas, é gerado, então, um dicionário cujas chaves são os RAs dos alunos e os valores são suas presenças, somadas à medida que encontradas na lista.

A quantidade de classes dadas pode também ser calculada com os mesmos dados, contando quantas datas diferentes existem dentre os formulários.

Utilizando o dicionário com a frequência dos alunos, é feita uma contagem das frequências dos diferentes números de presença e construído outro dicionário, em cima do qual é gerado o histograma.

Finalmente, com um dicionário indexado por datas de formulários e com os valores com listas dos formulários dessas datas, uma última consulta no banco recupera um dicionário de quantas presenças cada um desses dias contém. Com esse último dicionário, por fim, é gerado o gráfico da evolução das presenças no tempo.

4.3.7 Interface Gráfica

Também foi implementada uma interface gráfica utilizando a biblioteca *open-source* para *python* chamada *Kivy*. O objetivo dela é fornecer ao usuário um *feedback* das informações extraídas e processadas dos formulários ao longo do fluxo de execução, para que este possa visualizá-las e corrigir, caso necessário, eventuais erros e classificações errôneas por parte do algoritmo.

Figura 5: Imagem da interface gráfica com os campos de cada aluno.

A interface, ilustrada na Figura 5, mostra, para cada aluno de uma turma, seu nome e RA. Cada aluno possui, então, uma coluna **Data** com as datas de cada aula dada, uma coluna **Imagem Assinatura** com as imagens de suas assinaturas registradas em determinada data, e uma coluna **Imagem RA** com as imagens dos RAs deste aluno identificadas pelo programa. Isso, para que o usuário possa conferir se o algoritmo processou corretamente tais informações contidas nas imagens. Além disso, possui mais três colunas: **Presença**, **Similar** e **Final**. A coluna **Presença** mostra a presença do aluno em determinada data. Clicar nesse campo o faz alterar entre 'Presente' e 'Ausente' para que seja possível realizar as eventuais correções. A coluna **Similar** exibe o resultado do modelo de verificação de similaridade de assinaturas contido no projeto, mostrando 'Similar' para assinaturas que obtiveram um alto índice de similaridade com a assinatura base, 'Não similar' para assinaturas que obtiveram um baixo índice de similaridade com a assinatura base, e 'Não verificado' para assinaturas que não foram submetidas ao modelo de verificação de similariedade, pela não inserção do formulário de autenticação ou por ausência do aluno na aula, por exemplo. Clicar nesse campo alterna entre os três estados. Por fim, a coluna **Final** mostra o resultado final da presença de um aluno, relacionando os dados das duas colunas anteriores, ou seja, os dados de sua presença com os da verificação de assinatura. Existem quatro tipos de estados finais: 'A', 'PNV', 'PNS' e 'PS', que são siglas para 'Ausente', 'Presente Não Verificado',

'Presente Não Similar' e 'Presente Similar', respectivamente. Caso o aluno tenha 'Ausente' em sua coluna **Presença**, o resultado em **Final** sempre será 'A'. No caso de o aluno ter 'Presente' na coluna das presenças, se a coluna **Similar** possuir valor 'Similar', o aluno terá uma presença final 'PS', caso seja 'Não similar', terá presneça final 'PNS', e caso possua valor 'Não verificado', a presença final terá valor 'PNV'. Note que tanto o estado 'A' como 'PNS' são interpretados como faltas, 'PS' como presença, e 'PNV' também como presença, apesar de ser recomendado uma validação de seu estado por parte do usuário. Clicar neste campo alternar entre os quatro estados. Note que nas três colunas descritas acima, alterar um estado também o altera no banco de dados.

Por fim, ainda existe o botão 'Extrair CSV', que gera um CSV com as informações de presença de cada aluno presentes na interface.

4.4 Dificuldades Encontradas

Durante o desenvolvimento do projeto encontramos algumas dificuldades que tiveram de ser superadas ou contornadas. A primeira delas se apresentou ao gerarmos os formulários, pois não tínhamos familiaridade com a API que estava sendo utilizada no sistema, o que se tornou um desafio na hora de gerar as tabelas com diferentes cabeçalhos, um que seria por padrão de cada página, e um inicial na primeira página do formulário.

Também tivemos algumas dificuldades em reconstruir a grade do formulário. Como nos utilizamos das operações de erosão e dilatação para conseguir obter a grade, por vezes esta era obtida junto com muito ruído na imagem, tornando-a inutilizável ou piorando muito o resultado final por tal causa. Além disso, a separação das células extraídas por linha também se mostrou mais desafiadora do que esperávamos, sendo necessário um bom tempo de reflexão e diversas tentativas de abordagens para conseguir achar o critério que as separaria corretamente em suas respectivas linhas. E mesmo após a extração correta das células, muitas vezes, devido à posição da foto inicial, apareciam resquícios das bordas das células na imagem recortada, o que prejudicava muito a contabilização das presenças dos alunos, pois era difícil saber se o que tinha na célula era de fato uma assinatura ou apenas ruído. Assim, mais tempo e tentativas foram empregados na elaboração de uma abordagem que conseguisse remover os resquícios das bordas. A solução final foi alcançada progressivamente, primeiramente sem a remoção das margens, depois com a remoção das margens e, por fim, com a filtragem de ruído. A melhoria progressiva é visível das Figuras 6 a 12, tanto para uma célula com assinatura, quanto para uma célula vazia.

No processo de construção da interface gráfica com a tabela dos alunos e suas assinaturas para a finalização das estatísticas, dificuldades também foram encontradas para a escolha de um *framework* de interface. Inicialmente, foi criada uma página HTML pelo código Python com a disposição de uma tabela com as informações dos alunos da turma. Contudo, a página não trabalhava com eventos, como cliques, e não permitia a alteração do resultado final, ou a geração do CSV. A partir de então, buscou-se gerar a página como um *website* em um servidor, a qual seria alimentada pelo código em Python e a biblioteca Flask. Essa aproximação foi abandonada por não permitir uma interação individual dos professores com seus dados sem um sistema de autenticação. Finalmente, foi escolhida a biblioteca Kivy em Python para a construção da interface, a qual também causou problemas iniciais até a

decisão de uma implementação de layout mais simples como interface.

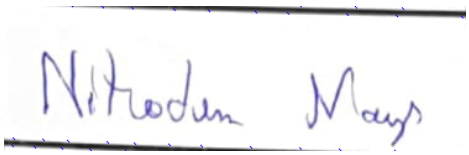


Figura 6: Célula original com assinatura e margem

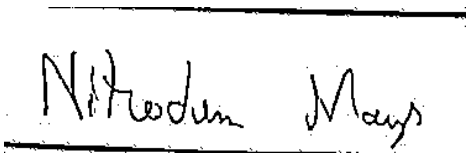


Figura 7: Imagem de assinatura com margem e limiarizada

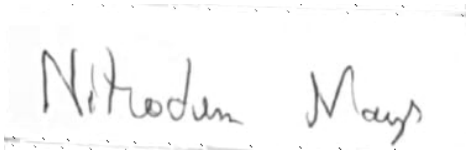


Figura 8: Célula com assinatura e margem removida

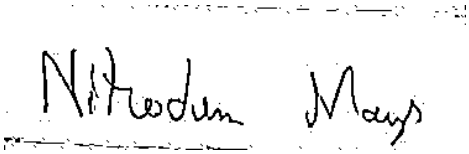


Figura 9: Imagem de assinatura com margem removida e limiarizada

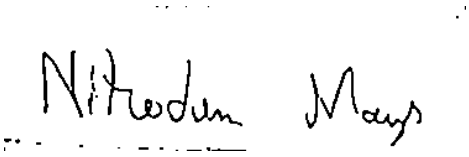


Figura 10: Imagem de assinatura resultante, com margem removida, limiarizada e filtrada com filtro passa-baixas

Figura 11: Exemplo de imagem de uma célula vazia original

Figura 12: Mesma célula vazia da Figura 11 com margens retiradas, limiarizada e filtrada com filtro passa-baixas

Outra dificuldade enfrentada foi durante a aplicação dos métodos de *OCR*. O ruído da imagem e até fatores que pareciam não tão significativos, como a fonte utilizada e o *zoom* da imagem, impactavam diretamente no resultado final obtido. E era essencial que o reconhecimento dos números dos RAs estivesse funcionando corretamente pois, do contrário, as presenças ou faltas dos alunos não poderiam ser registradas no banco de dados.

Mais desafios se apresentaram na validação das assinaturas dos alunos. A pesquisa e escolha de um modelo apropriado ao problema, junto à decisão de se utilizaríamos uma rede já projetada ou se estruturaríamos a nossa própria rede, demandaram ainda mais esforços. Após a escolha por usar Redes Neurais Siamesas já projetadas, foi preciso achar uma adequada. Inicialmente, chegamos a escolher um modelo e treiná-lo, porém sua acurácia no conjunto de dados de teste era de aproximadamente 50.6%, uma medida inaceitável para nossos objetivos no projeto. Assim, mais tempo foi empregado na procura de outra rede, finalmente achando um modelo pronto e já treinado[12], o qual apenas carregamos no código e utilizamos.

Por fim, a dificuldade que gerou mais erros nos resultados foi a de falsos positivos e falsos negativos, ou seja, a conclusão de que uma célula que tem uma assinatura é uma ausência e a conclusão de que uma imagem sem assinatura é uma presença, respectivamente. Existem dois tipos específicos de imagens que se aproximam do limiar, que são imagens sem assinatura, mas com riscos de caneta de outras assinaturas; e imagens com assinaturas muito pequenas. Duas células com esses casos são demonstradas nas Figuras 13 e 14 e são casos de possíveis falsos positivos e negativos.

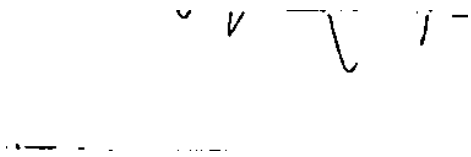


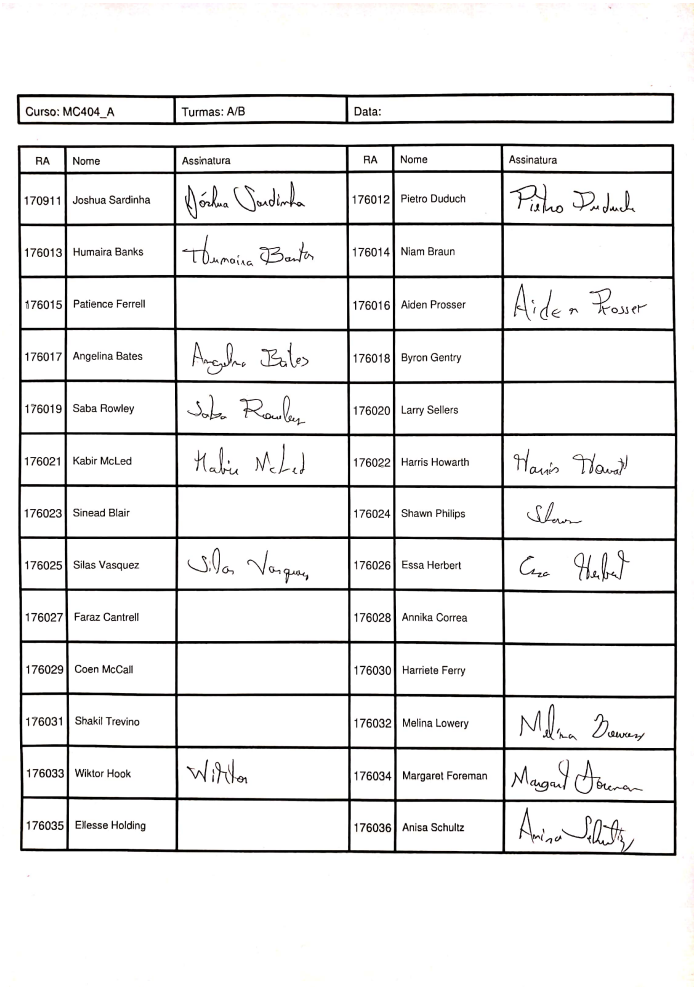
Figura 13: Exemplo de possível falso negativo com traços de uma assinatura na célula superior

Figura 14: Exemplo de possível falso positivo com uma assinatura curta

5 Resultados

Para a reprodução de testes e comprovação dos resultados, foi gerado um formulário de exemplo com 42 alunos fictícios, colocados em um arquivo CSV também passado ao programa. Foram impressas 3 cópias do formulário e as células foram assinadas de forma aleatória, como mostra uma das páginas, na Figura 15. Todas as imagens que foram passadas ao programa foram tiradas de um celular e escaneadas com um aplicativo. O formulário resultante teve 2 páginas e o PDF foi gerado corretamente.

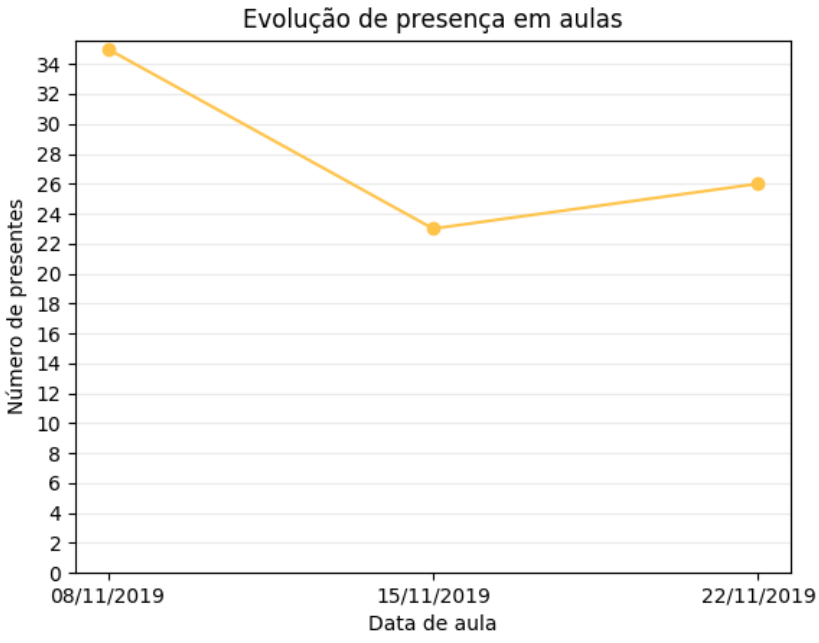
Cada uma das 3 cópias do formulário foi inserida no programa com uma diferente data de aula. Propositalmente, algumas assinaturas foram feitas muito curtas, outras invadindo outras células e uma das fotos cortou para fora parte da tabela. No fim, na soma dos 3 formulários, 126 células de assinatura foram analisadas. Com a adição de todos os formulários e todas as células, foram obtidos 124 resultados corretos e 2 resultados falsos, com um falso positivo e outro falso negativo. A saída do teste final realizado também compreendeu os gráficos das Figuras 15 e 16. Pelos testes realizados com alguns outros formulários de teste, chegamos que, com fotos de boa qualidade de nitidez e sem cortes, a acurácia da análise das presenças fica entre 99% e 100%, diminuindo de acordo com a deteorização da qualidade da imagem.



Curso: MC404_A	Turmas: A/B	Data:

RA	Nome	Assinatura	RA	Nome	Assinatura
170911	Joshua Sardinha		176012	Pietro Duduch	
176013	Humaira Banks		176014	Niam Braun	
176015	Patience Ferrell		176016	Aiden Prosser	
176017	Angelina Bates		176018	Byron Gentry	
176019	Saba Rowley		176020	Larry Sellers	
176021	Kabir McLeod		176022	Harris Howarth	
176023	Sinead Blair		176024	Shawn Philips	
176025	Silas Vasquez		176026	Essa Herbert	
176027	Faraz Cantrell		176028	Annika Correa	
176029	Coen McCall		176030	Harriete Ferry	
176031	Shakil Trevino		176032	Melina Lowery	
176033	Wiktor Hook		176034	Margaret Foreman	
176035	Ellesse Holding		176036	Anisa Schultz	

Figura 15: Página de um relatório de teste



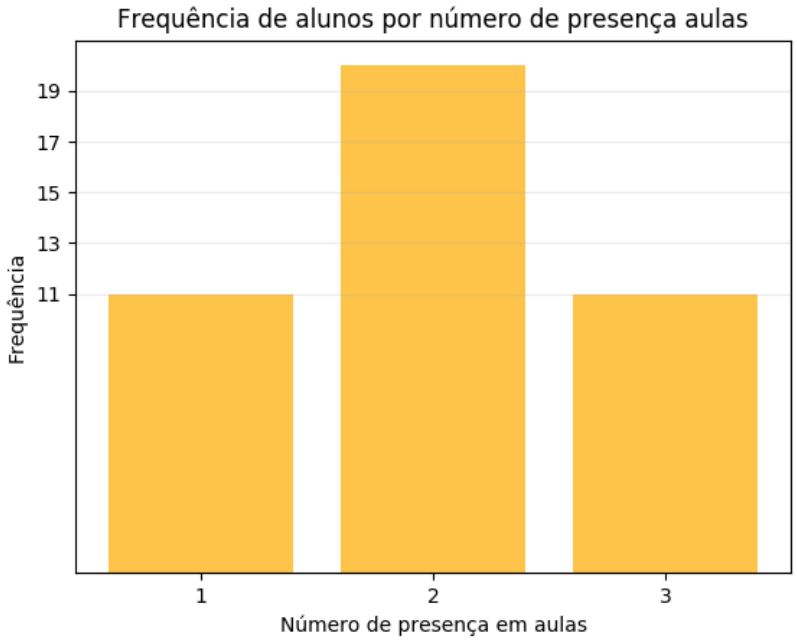


Figura 16: Histograma da frequência de alunos por números de presença como saída do caso de teste

Possíveis problemas que afetam os resultados são fotos borradas, que complicam a leitura das células de RA por meio do OCR, ou fotos que cortam parte da grade, dificultando o reconhecimento de uma célula. O maior fator para o alcance de acurácia perfeita do código desenvolvido é a entrega também de um formulário com assinaturas nos espaços adequados.

6 Conclusões

Neste projeto foi possível exercitar e investigar diversos conceitos de processamento de imagens e de aprendizado de máquina, bem como projetar um sistema para automatizar e otimizar a coleta das assinaturas dos alunos, a contabilização de suas faltas e a validação de suas assinaturas. Para o futuro, espera-se que melhorias ainda sejam implementadas ao projeto, como uma melhor extração e filtragem das células, uma rede neural com acurácia maior para autenticação das assinaturas, melhorias no método de determinação de presença ou ausência de assinaturas em uma célula, e, por fim, poderia se desenvolver mais a interface gráfica de visualização final dos dados obtidos no processo, de modo a implementar novas funcionalidades nesta, como uma opção de exibir o formulário de uma determinada turma ao clicar no campo de uma data específica, e melhorias gráficas na interface, deixando-a mais intuitiva e proporcionando uma melhor experiência ao usuário.

7 Agradecimentos

Agradecemos ao nosso orientador, professor Edson Borin, pelo suporte e disponibilização dos *scripts* base, ao professor Hélio Pedrini, por suas contribuições e ideias para lidar com os desafios relacionados ao processamento das imagens, e a Deus, por ter nos sustentado até aqui e continuar nos sutentando a cada dia.

Referências

[1] YAO, Jenq-Foung; CHANG, Tsu-Ming. *Correlação entre presença e nota*. Georgia College & State University, 146–147.

[2] Art. 47, § 3 da Lei de Diretrizes e Bases - Lei 9394 (1996). Disponível em: `https://www.jusbrasil.com.br/topicos/11688556/paragrafo-3-artigo-47-da-lei-n-9394-de-20-de-dezembro-de-1996`. Acesso em: 10 de nov. 2019.

[3] SERRA, Jean. *Image Analysis and Mathematical Morphology*, ISBN 0-12-637240-3, 1982.

[4] JAIN, Anil. *Fundamentals of Digital Image Processing*, Prentice-Hall, 1986, Cap. 8.

[5] DOCUMENTAÇÃO DA BIBLIOTECA REPORTLAB. Disponível em: `https://www.reportlab.com/docs/reportlab-userguide.pdf`. Acessado em: 10 nov. 2019

[6] BHADRA, Rajarshi. *What is a Neural Network?*. Disponível em: `https://towardsdatascience.com/what-is-a-neural-network-a02b3c2fe3fa`. Acessado em: 10 nov. 2019

[7] LAMBA, Harshall. *One Shot Learning with Siamese Networks using Keras*. Disponível em: `https://towardsdatascience.com/one-shot-learning-with-siamese-networks-using-keras-17f34e75bb3d`. Acessado em: 10 nov. 2019

[8] *What Is OCR And What Is It Used For?*. Disponível em: `https://docparser.com/blog/what-is-ocr/`. Acessado em: 10 nov. 2019

[9] *What Is a Database*. Disponível em: `https://www.oracle.com/database/what-is-database.html#WhatIsDBMS`. Acessado em: 10 nov. 2019

[10] *What Is SQLite?*. Disponível em: `https://www.sqlite.org/index.html`. Acessado em: 10 nov. 2019

[11] `https://en.wikipedia.org/wiki/Salt-and-pepper_noise`

[12] `https://github.com/Aftaab99/OfflineSignatureVerification`