UNIVERSIDADE FEDERAL DO PARÁ
CAMPUS UNIVERSITÁRIO DE TUCURUÍ
FACULDADE DE ENGENHARIA ELÉTRICA

Gabriel Lima Ferreira

VIABILIDADE TÉCNICA DE UM SISTEMA DE OCR PARA MONITORAMENTO DE SINAIS VITAIS EM MONITORES MULTIPARÂMETRO

TUCURUÍ

2025

Gabriel Lima Ferreira

VIABILIDADE TÉCNICA DE UM SISTEMA DE OCR PARA MONITORAMENTO DE SINAIS VITAIS EM MONITORES MULTIPARÂMETRO

Trabalho de Conclusão de Curso, apresentado como requisito parcial para a obtenção de grau de Bacharel em Engenharia Elétrica, pela Universidade Federal do Pará.

Orientador:
Prof. Dr. Rafael Suzuki Bayma.

TUCURUÍ

2025

Gabriel Lima Ferreira

VIABILIDADE TÉCNICA DE UM SISTEMA DE OCR PARA MONITORAMENTO DE SINAIS VITAIS EM MONITORES MULTIPARÂMETRO

Trabalho de Conclusão de Curso, apresentado como requisito parcial para a obtenção de grau de Bacharel em Engenharia Elétrica, pela Universidade Federal do Pará.

Data de aprovação: 23 de Julho de 2025.

Banca Examinadora:

Prof. Dr. Rafael Suzuki Bayma
Orientador - FEE/CAMTUC/UFPA

Prof. Dr. Cleison Daniel Silva
Avaliador Interno - FEE/CAMTUC/UFPA

Prof. Dr. Raphael Barros Teixeira
Avaliador Interno - FEE/CAMTUC/UFPA

TUCURUÍ

2025

UNIVERSIDADE FEDERAL DO PARÁ
CAMPUS UNIVERSITÁRIO DE TUCURUÍ
FACULDADE DE ENGENHARIA ELÉTRICA

Título	VIABILIDADE TÉCNICA DE UM SISTEMA DE OCR PARA MONITORAMENTO DE SINAIS VITAIS EM MONITORES MULTIPARÂMETRO
Discente	Gabriel Lima Ferreira
Matrícula	202033940013

#	BANCA EXAMINADORA	CONDIÇÃO
1	Cleison Daniel Silva (NDAE/UFPA)	Examinador interno
2	Raphael Barros Teixeira (CAMTUC/UFPA)	Examinador interno
3	Rafael Suzuki Bayma (CAMTUC/UFPA)	Orientador

Data da Defesa: 08/07/2025	Hora Início: 08:34	Hora Término: 09:10

]

Trabalho Escrito (0 a 10 pontos por critério)			
Critério	**Examinador 1**	**Examinador 2**	**Examinador 3**
Formatação	9,0	9,0	9
Linguagem (gramática e semântica)	9,0	9,0	9,5
Conteúdo técnico	9,5	9	9,0
Média	**9,2**	**9,0**	**9,2**

Defesa Oral (0 a 10 pontos por critério)			
Critério	**Examinador 1**	**Examinador 2**	**Examinador 3**
Sequência lógica de apresentação	10,0	9,5	9,5
Administração do tempo	10,0	10,0	10,0
Expressão oral	9,5	10,0	10,0
Domínio do tema	9,5	9,5	9,5
Média	**9,8**	**9,8**	**9,8**

Média por examinador	9,5	9,4	9,5
Média Final	**9,4**		
Conceito Final	**EXC**		

gov.br Documento assinado digitalmente
CLEISON DANIEL SILVA
Data: 08/07/2025 10:57:59-0300
Verifique em https://validar.iti.gov.br

Cleison Daniel Silva
Membro

gov.br Documento assinado digitalmente
RAPHAEL BARROS TEIXEIRA
Data: 08/07/2025 10:31:28-0300
Verifique em https://validar.iti.gov.br

Raphael Barros Teixeira
Membro

gov.br Documento assinado digitalmente
RAFAEL SUZUKI BAYMA
Data: 08/07/2025 11:00:56-0300
Verifique em https://validar.iti.gov.br

Rafael Suzuki Bayma
Orientador

Dedico este trabalho à minha mãe, Leidiane. Seus ensinamentos, amor, cuidado e exemplos de perseverança foram fundamentais para que eu alcançasse esta conquista. Dedico também à minha esposa, Marcilene. Agradeço pelo amor, apoio constante e por estar sempre ao meu lado, me incentivando a seguir em frente.

AGRADECIMENTOS

Agradeço primeiramente a Deus, por Seu amor infinito, misericórdia constante e sabedoria abundante. Em todos os momentos desta jornada, especialmente nos mais difíceis — quando o medo, a ansiedade e até a sombra da depressão tentaram me paralisar — foi Ele quem me sustentou. Sua presença me fortaleceu, me deu clareza, inteligência e serenidade para seguir em frente. Sem Sua luz, nada disso teria sido possível.

À minha mãe, Leidiane, minha eterna inspiração, deixo minha mais profunda gratidão. Mulher guerreira, que mesmo diante das dificuldades nunca deixou de me amar, apoiar e ensinar. Foi com ela que aprendi a ser forte, a não desistir dos meus sonhos e a seguir em frente com coragem. Seu exemplo de perseverança e amor incondicional me guiou em cada passo desta caminhada. Aos meus irmãos, Gabriella, Emanuel, Izabelle e Vanessa, agradeço pelo carinho, pela presença e por fazerem parte da minha base familiar, sempre torcendo por mim. À minha esposa, Marcilene, meu amor e companheira de vida, agradeço por cada gesto de incentivo, por todo apoio e por estar ao meu lado em todos os momentos. Sua dedicação, paciência e amor foram fundamentais para que eu pudesse chegar até aqui. Muito do que conquistei também é fruto do que construímos juntos.

Ao professor Rafael S. Bayma, minha sincera gratidão. Mais que um orientador, foi um mentor, fonte de inspiração e sabedoria. Seu exemplo de dedicação e excelência continuará a me guiar na carreira profissional. Ao professor Bruno Wallacy, meu eterno mestre e amigo, deixo meu profundo reconhecimento. Sua paixão pelo conhecimento, generosidade ao ensinar e amizade foram fundamentais para minha formação. Aprendi muito com ele — sobre ciência, humildade, ética e entusiasmo. Agradeço a todo o corpo docente, cujos ensinamentos serão fundamentais para minha carreira.

Ao senhor Franciclaudio Lisboa, meu gestor na Vale, expresso minha sincera gratidão por todo o apoio e orientação durante minha trajetória profissional. Sua liderança foi essencial para meu desenvolvimento na supervisão de componentes elétricos, e os conhecimentos e treinamentos que me proporcionou foram fundamentais para que eu pudesse crescer e me aprimorar como profissional. Agradeço por acreditar no meu potencial e por compartilhar sua experiência com generosidade.

Aos meus grandes amigos Bruno Santos, Lucas Marcley, Leonardo França e João L. Azevedo, minha gratidão sincera por cada momento compartilhado durante a graduação. Vocês não foram apenas colegas de curso, mas verdadeiros irmãos de jornada.

Agradeço também aos amigos Fabrício Viana, Victor Nascimento, Hartur Sousa e Breno dos Santos, que se tornaram verdadeiros irmãos ao longo da minha trajetória acadêmica. Cada um, à sua maneira, contribuiu para tornar essa caminhada mais leve.

"Em algum lugar, algo incrível está esperando para ser descoberto."
(Sharon Begley, 1977).

RESUMO

Este trabalho apresenta o desenvolvimento e a avaliação de um sistema em Python, baseado em técnicas de visão computacional (OpenCV) e OCR (Pytesseract), para extração automatizada de sinais vitais a partir de vídeos de monitores médicos. Foram processados quatro vídeos obtidos na internet, com diferentes resoluções, taxas de quadros e condições de iluminação, para verificar a sensibilidade do método a fatores externos. A região de saturação de oxigênio obteve acurácias entre 94 % e 100 %, enquanto batimento cardíaco variou de 69,23 % a 100 %. A extração da frequência de pulso apresentou maior dispersão, indo de apenas 2,86 % no primeiro vídeo até 100 % nos ajustes posteriores. A média geral de acertos foi de 84,29 % (erro médio de 15,71 %), evidenciando viabilidade técnica do sistema. As principais limitações identificadas estão associadas à qualidade dos vídeos — resolução, estabilidade da câmera, iluminação e ruídos visuais —, o que reforça a necessidade de ambientes controlados e calibração refinada das regiões de interesse. Este estudo de viabilidade técnica demonstra que a combinação de OpenCV e Pytesseract pode suportar soluções de monitoramento remoto em unidades de terapia intensiva, desde que respeitadas as condições mínimas de captura de imagem.

Palavras-chave: Visão Computacional; OCR; Monitoramento de Sinais Vitais; Viabilidade Técnica.

ABSTRACT

This work presents the development and evaluation of a Python-based system leveraging computer vision (OpenCV) and OCR (Pytesseract) techniques for automated extraction of vital signs from medical monitor videos. Four videos obtained online—with varying resolutions, frame rates, and lighting conditions—were processed to assess the method's sensitivity to external factors. The oxygen saturation region achieved accuracies between 94 % and 100 %, while heart rate ranged from 69.23 % to 100 %. Pulse rate extraction showed greater dispersion, from only 2.86 % in the first video to 100 % after subsequent adjustments. The overall mean accuracy was 84.29 % (mean error of 15.71 %), demonstrating the technical feasibility of the system. The main limitations identified relate to video quality—resolution, camera stability, illumination, and visual noise—underscoring the need for controlled environments and fine-tuned ROI calibration. This technical feasibility study shows that combining OpenCV and Pytesseract can support remote monitoring solutions in intensive care units, provided minimum image capture conditions are met.

Keywords: Computer Vision; OCR; Vital Signs Monitoring; Technical Feasibility.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

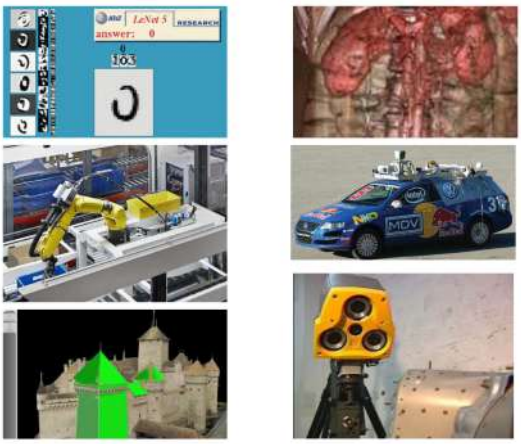
OCR	Reconhecimento Óptico de Caracteres (do inglês Optical Character Recognition)
ROI	Região de Interesse
TCC	Trabalho de Conclusão de Curso
UTI	Unidade de Terapia Intensiva
FPS	Quadro por segundo (do inglês Frames Per Second)

SUMÁRIO

1 INTRODUÇÃO

A visão computacional e o processamento de imagens têm se consolidado como tecnologias fundamentais em diversas áreas do conhecimento, incluindo reconhecimento óptico de caracteres, inspeção de máquinas, varejo, logística de armazém, imagem médica e construção de modelos 3D (fotogrametria) (SZELISKI, 2021).

Figura 1 – Aplicações de Visão computacional



Fonte: Adaptado de (SZELISKI, 2021)

Na medicina, essas tecnologias têm se destacado na busca por métodos que auxiliem os profissionais de saúde. Conforme destacado pelo professor Renato Tinós, do Departamento de Computação e Matemática da Faculdade de Filosofia, Ciências e Letras de Ribeirão Preto da USP, "a inteligência artificial vai ser usada cada vez mais e está vindo para aprimorar e auxiliar os diagnósticos na medicina, e não para substituir os médicos"(USP, 2018).

No artigo (BARBIERIA et al., 2022) diz, "A tecnologia digital e orientada por Inteligência Artificial (IA) mostrou potencial para aumentar as habilidades dos profissionais de saúde em fazer diagnósticos mais rápidos e precisos, especialmente ao usar imagens, como nas áreas de dermatologia ou oftalmologia .".

A crescente digitalização dos processos médicos e a necessidade de monitoramento de pacientes impulsionam o desenvolvimento de tecnologias para extração e análise automatizada de dados médicos. Nesse contexto, a visão computacional e o reconhecimento óptico de caracteres (OCR) surgem como soluções promissoras para transformar informações visuais capturadas de monitores hospitalares em dados digitais analisáveis.

Este trabalho propõe um sistema que integra técnicas de visão computacional e OCR para a extração automática de sinais vitais a partir de vídeos de monitores médicos. Para isso, o método emprega abordagens de processamento de imagens, incluindo a segmentação

de regiões de interesse (ROIs) e a extração de dados numéricos. A implementação, desenvolvida em Python, utiliza como principais bibliotecas OpenCV e pytesseract.

A relevância deste estudo está na possibilidade de reduzir erros manuais, facilitar o armazenamento e análise de dados clínicos e contribuir para a automação de processos médicos. Dessa forma, espera-se que a pesquisa auxilie no aprimoramento das tecnologias de monitoramento e na implementação de ferramentas acessíveis e eficientes para o setor da saúde.

1.1 Justificativa

O monitoramento contínuo de sinais vitais é essencial para o acompanhamento de pacientes em ambientes hospitalares, especialmente em unidades de terapia intensiva (UTI). No entanto, a coleta manual desses dados pode ser suscetível a erros, além de demandar tempo da equipe médica. A utilização de um sistema automatizado para extração e processamento dessas informações permite maior precisão e eficiência, auxiliando profissionais de saúde na tomada de decisões e na melhoria da qualidade do atendimento.

1.2 Objetivos

Este estudo tem como objetivo geral o desenvolvimento de um sistema para extração automática de dados de sinais vitais a partir de monitores médicos, utilizando visão computacional e OCR. Para alcançar esse objetivo, foram estabelecidas as seguintes metas:

a) Realizar uma revisão bibliográfica sobre os fundamentos de sistemas digitais e lógica de programação;

b) Implementar técnicas de segmentação de imagens para identificação de regiões de interesse;

c) Integrar a biblioteca Pytesseract OCR para reconhecimento óptico de caracteres em monitores médicos;

d) Desenvolver uma interface gráfica interativa para visualização e análise dos dados extraídos;

e) Avaliar a precisão do sistema e propor melhorias para aplicações futuras.

1.3 Estrutura da Monografia

Este trabalho está estruturado em seis capítulos.

O capítulo 1 é introdutório, apresentando a justificativa, os objetivos e a estrutura do trabalho.

No capítulo 2, é apresentada a fundamentação teórica, abordando conceitos sobre visão computacional, reconhecimento óptico de caracteres e suas aplicações na área médica.

O capítulo 3 descreve a metodologia empregada no desenvolvimento do sistema, detalhando as técnicas e ferramentas utilizadas.

No capítulo 4, são discutidos os resultados obtidos, incluindo a análise da precisão do sistema e a eficácia da extração dos dados.

O capítulo 5 apresenta as considerações finais, destacando as principais contribuições do estudo, suas limitações e sugestões para trabalhos futuros.

Por fim, são disponibilizadas as referências bibliográficas, os apêndices e anexos necessários para complementar o estudo.

2 Fundamentação Teórica

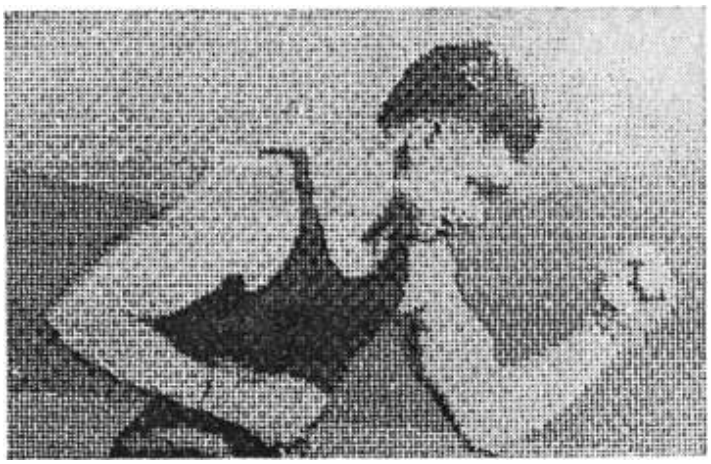
2.1 Processamento Digital de imagem

2.1.1 Origem do processo Digital de Imagens

O processamento digital de imagens é uma área do conhecimento que tem evoluído significativamente ao longo das décadas. Uma das primeiras aplicações é destacada no livro "Digital Image Processing", sendo inicialmente utilizadas na indústria jornalística. Fotos foram enviadas pela primeira vez por cabo submarino entre Londres e Nova York. Esse sistema de transmissão de imagens por cabo Bartlane reduziu o tempo necessário para transportar uma imagem através do Atlântico de mais de uma semana para menos de três horas (GONZALEZ; WOODS, 2018).

A figura 2 foi transmitida dessa forma e reproduzida em uma impressora telegráfica equipada com fontes simulando um padrão de meio-tom. Este método permitiu a reprodução de imagens com uma qualidade visual surpreendente para a época, apesar das limitações tecnológicas.

Figura 2 – Uma imagem digital produzida em 1921.



Fonte: (GONZALEZ; WOODS, 2018)

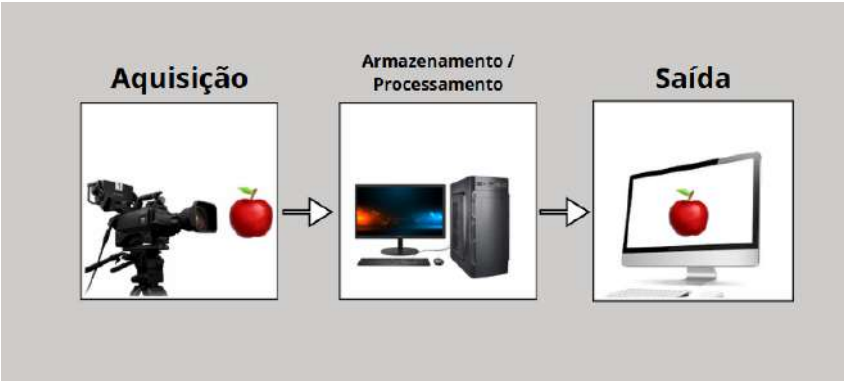
De acordo com (FILHO; NETO, 2001), mesmo diante desse avanço, essa área só ganhou mais notoridade em 1964. quando imagens da lua transmitidas por uma sonda Ranger eram processadas por computador para corrigir vários tipos de distorção inerentes à câmera de TV acoplada à sonda.

2.1.2 Sistema de processamento de imagem

Como apresentado por (FILHO; NETO, 2001), os componentes de um sistema de processamento de imagem são compostos por quatro etapas fundamentais: aquisição,

armazenamento, processamento e saída. A figura 3 ilustra de forma simplificada como essas etapas se integram para formar um sistema completo de processamento de imagem.

Figura 3 – Diagrama de um Sistema de processamento de imagem



Fonte: Autor.

2.1.2.1 Aquisição

Na etapa de aquisição de imagens, as informações visuais da cena são convertidas em sinais elétricos por meio de dispositivos ou sensores ópticos. As câmeras possuem resolução espacial e profundidade de pixel fixas. A quantização realiza o mapeamento do sinal contínuo da cena em um conjunto discreto de valores, organizados espacialmente como pixels, cada um com uma capacidade finita de representação (profundidade de pixel) (BRECKON; SOLOMON, 2013).

2.1.2.2 Armazenamento

O armazenamento de imagens digitais representa um dos principais desafios no desenvolvimento de sistemas de processamento de imagens, devido à enorme quantidade de bytes necessária para armazenar essas informações visuais. De acordo com (FILHO; NETO, 2001), armazenamento pode ser dividido em três categorias: (1) armazenamento de curta duração, enquanto ela é utilizada nas várias etapas do processamento, (2) armazenamento de massa para operações de recuperação de imagens relativamente rápidas, e (3) arquivamento de imagens.

2.1.2.3 Processamento

O processamento normalmente utiliza algoritmos que podem ser executados por software, com exceção das fases de captura e exibição das imagens. Em geral, o software é suficiente para realizar a maioria das funções de processamento de imagens. No entanto, em situações onde o computador principal enfrenta limitações, como baixa velocidade de transferência de dados, pode ser necessário utilizar hardware especializado para garantir um desempenho adequado (FILHO; NETO, 2001).

2.1.2.4 Saída

A etapa final no processamento de imagens é a representação digital de um momento ou objeto após passar por todas as etapas anteriores. Observando a figura 3, podemos entender que o resultado é a exibição do objeto de maneira digital. Esta exibição pode ocorrer em diversos formatos de monitores de vídeo ou qualquer outra forma de exibição.

2.1.3 Imagem Digital

De acordo com (PERSECHINO; ALBUQUERQUE, 2015), "Uma imagem pode ser compreendida como um sinal representativo de alguma estrutura, seja ela de natureza física, artística, conceitual etc.". Portanto, uma imagem monocromática, como ilustrado na figura 4, pode ser representada matematicamente por uma função $f(x, y)$, que expressa a intensidade da luz em cada ponto da imagem. O valor da função em um ponto com coordenadas espaciais (x, y) é proporcional ao nível de brilho (ou escala de cinza) naquele local.

Figura 4 – Uma imagem monocromática e a convenção utilizada para o par de eixos (x,y).

Fonte: (FILHO; NETO, 2001).

Com isso, podemos entender imagem como uma representação de um momento contínuo. Portanto, um modelo matemático para uma imagem monocromática, que foi tirado do (QUEIROZ; GOMES, 2001), é dado por:

$$f(x, y) = I(x, y)R(x, y),$$

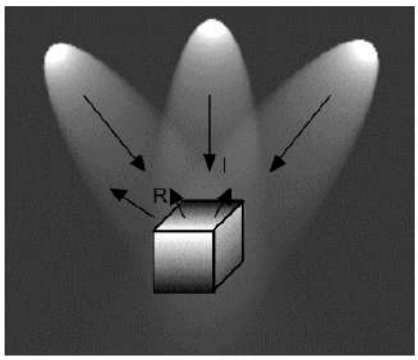
onde,

$$0 \leq I(x, y) < \infty$$

$$0 \leq R(x, y) < 1.$$

A função $f(x, y)$ representa o resultado da interação entre a iluminância $i(x, y)$, que indica a quantidade de luz que incide sobre o objeto, e as propriedades de refletância ou transmitância do próprio objeto, representadas pela função $r(x, y)$. O valor de $r(x, y)$ expressa a fração de luz incidente que o objeto reflete ou transmite no ponto (x, y) do (QUEIROZ; GOMES, 2001). Esses conceitos estão ilustrados na figura 5.

Figura 5 – Os componentes iluminância (I) e refletância (R) de uma imagem.



Fonte: (FILHO; NETO, 2001).

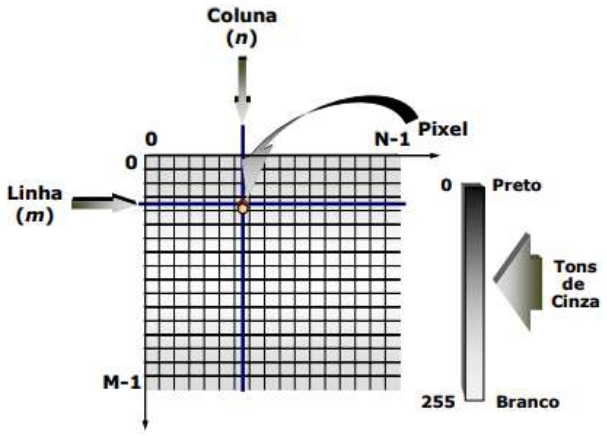
A Tabela 1 apresenta uma seleção de valores típicos de iluminância $i(x, y)$, expressa em lux, e de refletância $r(x, y)$, que representa a fração da luz refletida por diferentes superfícies. Esses parâmetros são fundamentais na modelagem de imagens digitais, pois influenciam diretamente a percepção de brilho e contraste em uma cena. Os exemplos listados abrangem desde condições de iluminação natural, como um dia ensolarado ou uma noite de lua cheia, até materiais com diferentes capacidades de reflexão, como neve, aço inoxidável e veludo preto.

Tabela 1 – Exemplos de valores para $i(x, y)$ e $r(x, y)$ (adaptado de FILHO; NETO, 2001)

Intensidade Luminosa $i(x, y)$ **(lux)**		**Refletância** $r(x, y)$	
Situação	Valor	Superfície	Valor
Dia ensolarado	900	Neve	0,93
Dia nublado	100	Parede branco-fosca	0,80
Iluminação média de escritório	10	Aço inoxidável	0,65
Noite clara de lua cheia	0,001	Veludo preto	0,01

Como os computadores não são capazes de processar imagens contínuas, mas apenas conjuntos discretos de dados numéricos, é necessário representar imagens como arranjos bidimensionais de pontos. Uma imagem digital é, portanto, a representação de uma cena visual bidimensional por meio de números binários organizados em uma matriz de pixels. Cada pixel corresponde a uma pequena unidade da imagem e armazena um valor numérico que representa a intensidade luminosa (em imagens em tons de cinza) ou a combinação de cores (em imagens coloridas) naquele ponto específico(RANKINGS, 2004). A figura 6 apresenta esse conceito.

Figura 6 – Representação de uma imagem digital bidimensional.



Fonte: (QUEIROZ; GOMES, 2001).

Para entender melhor como essas imagens digitais são formadas, é importante considerar o processo de captura da intensidade luminosa. O brilho refletido em pontos $(x, y) \in \mathbb{R}^2$ de uma determinada região do espaço, ou cena, pode ser amostrado por meio de um arranjo de sensores posicionados com espaçamentos (dx, dy). Esses sensores capturam a intensidade luminosa refletida em cada ponto, que é então quantizada em valores inteiros $I(x, y)$, resultando na formação de uma imagem (SCURI, 1999).

Os parâmetros que são apresentados na figura 6 são importantes, pois m representa a posição da linha, na qual o pixel se encontra, enquanto o n denota a posição da coluna. Se a imagem digital contiver M linhas e N colunas, o índice m variará de 0 a $M - 1$, enquanto n variará de 0 a $N - 1$. Que matematicamente é apresentado abaixo.

$$\begin{pmatrix} f(1,1) & f(1,2) & \cdots & f(1,N) \\ f(2,1) & f(2,2) & \cdots & f(2,N) \\ \vdots & \vdots & \ddots & \vdots \\ f(M,1) & f(M,2) & \cdots & f(M,N) \end{pmatrix}$$

2.2 Visão Computacional

A visão computacional é uma área interdisciplinar e dinâmica da ciência da computação que se concentra no desenvolvimento de sistemas computacionais capazes de "ver"e interpretar o mundo visual, em um processo que simula a percepção humana (MARENGONI; STRINGHINI, 2010). Esse campo inovador combina uma variedade de técnicas avançadas, incluindo processamento digital de imagens, reconhecimento de padrões, aprendizado de máquina e inteligência artificial, para extrair, analisar e interpretar informações contidas em imagens e vídeos.

De acordo com Richard Szeliski (SZELISKI, 2021),“Pesquisadores em visão compu-

tacional têm desenvolvido técnicas matemáticas e algoritmos inspirados em aspectos do sistema visual humano para replicar as capacidades de percepção visual em máquinas.". Com isso, a visão computacional é essencial para diversas aplicações tecnológicas, como veículos aéreos não triupulados com visão computacional na agricultura (SILVA et al., 2015), reconhecimento facial (BRAGA, 2013), sistemas de vigilância (MATIAS et al., 2024), entre outros. Ao integrar algoritmos sofisticados e modelos matemáticos, a visão computacional busca replicar a capacidade humana de perceber e compreender o ambiente visual, possibilitando a automação de tarefas que tradicionalmente exigiriam intervenção humana.

Surgida nas décadas de 1960 e 1970, a visão computacional se desenvolveu paralelamente ao processamento de imagens, aplicando métodos matemáticos para realçar, segmentar e entender imagens de forma automatizada (GONZALEZ; WOODS, 2018). Atualmente, a visão computacional é uma das áreas mais promissoras e amplamente exploradas. Segundo Barelli (BARELLI, 2018): "Hoje, existem diversas bibliotecas e linguagens de programação para desenvolvimento de softwares nessa área, todas elas abstraindo conceitos matemáticos e tornando seu uso mais fácil. Antes de detalhar essas novas tecnologias, é importante compreender o que é Visão computacional e como é possível fazer um computador enxergar o que acontece próximo a ele.".

2.2.1 Definição e Estrutura dos Sistemas de Visão Computacional

Um sistema que utiliza visão computacional envolve uma gama de procedimentos, que vão desde a captura da imagem até sua análise e interpretação. O processo geralmente possui seis etapas. Começa com a aquisição da imagem, seja por uma câmera, scanner ou outro dispositivo óptico, seguido pelo pré-processamento, onde são aplicadas técnicas para reduzir ruídos e ajustar o contraste. A segmentação é uma etapa crítica, na qual a imagem é dividida em regiões de interesse específicas, permitindo que partes relevantes da imagem sejam isoladas para análise detalhada (FORSYTH; PONCE, 2011).

Em seguida, a etapa de extração de características visa identificar elementos específicos, como bordas, texturas e formas, que são fundamentais para o reconhecimento de objetos ou padrões. Finalmente, a classificação e o reconhecimento são realizados, utilizando modelos de aprendizado de máquina que associam os padrões identificados a categorias ou classes conhecidas, permitindo que o sistema compreenda o conteúdo visual com precisão (SZELISKI, 2021).

A figura 7 ilustra as principais etapas de um sistema um sistema de Visão computacional.

Figura 7 – Fluxo de um sistema baseado em visão computacional

Fonte: Adaptado de (BARELLI, 2018).

2.2.2 Aquisição de imagem

A aquisição de imagem é a primeira etapa em um sistema de visão computacional e envolve a captura de dados visuais que servirão de entrada para as análises subsequentes. Nesse processo, são utilizadas câmeras e sensores de imagem que coletam informações visuais, sendo que a qualidade dessa aquisição influencia diretamente na precisão dos resultados (BRECKON; SOLOMON, 2013). Por essa razão, a escolha do dispositivo de captura de qualidade e das condições de aquisição é fundamental.

Tecnologias modernas permitem otimizar essa etapa. Câmeras de profundidade (RGBD) fornecem informações tridimensionais essenciais para certos contextos, enquanto técnicas de imagem multiespectral capturam faixas do espectro invisíveis ao olho humano, como infravermelho e ultravioleta. Além disso, a integração de sensores lidar em alguns sistemas proporciona mapeamentos precisos do ambiente tridimensional, combinando dados ópticos e de distância. Esses avanços ajudam a garantir que as imagens capturadas estejam em formato adequado para processamento subsequente (GONZALEZ; WOODS, 2018).

2.2.3 Pré-processamento

De acordo com (COSTA et al., 2022), a etapa de pré-processamento é muito importante para garantir a qualidade das imagens que serão utilizadas nas etapas subsequentes de processamento. Com isso, essa fase é essencial para reduzir ruídos, aumentar a clareza das informações e ajustar a imagem para atender às especificidades dos algoritmos utilizados nas etapas posteriores.

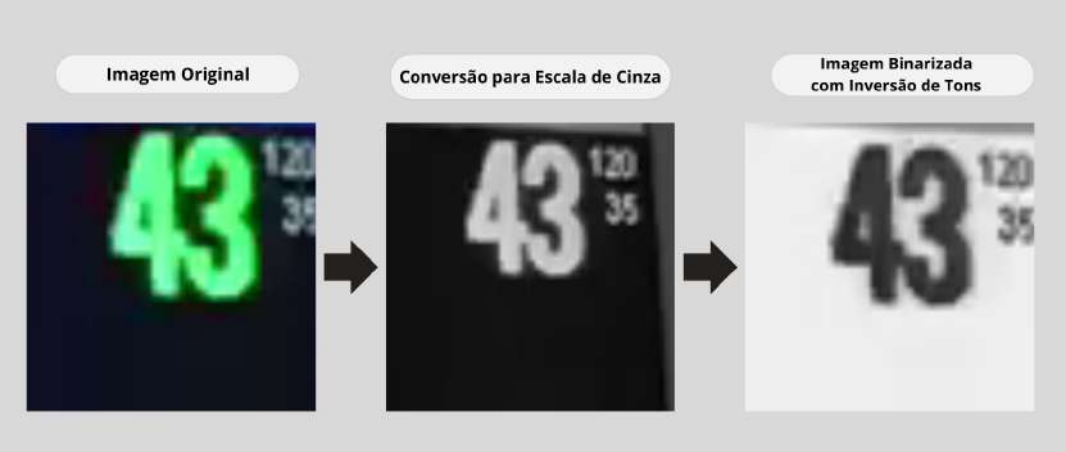
Diversas técnicas de pré-processamento têm sido amplamente empregadas. A filtragem de ruído, por exemplo, é realizada por métodos como os filtros de média e de mediana, que removem distorções indesejadas sem comprometer os detalhes importantes

da imagem e a equalização de histograma, por sua vez, melhora o contraste, realçando características relevantes que poderiam passar despercebidas (FILHO; NETO, 2001).

Outra técnica relevante é a correção geométrica, utilizada para alinhar imagens e corrigir distorções de perspectiva. Em sistemas mais avançados, algoritmos como o de transformação de Fourier são empregados para corrigir deformações ou adaptar a imagem a um formato ideal para processamento (SONKA; HLAVAC; BOYLE, 2014).

Além das técnicas citadas, este trabalho adota uma abordagem prática de pré-processamento que inclui a conversão da imagem para escala de cinza, seguida de duas etapas de binarização: a primeira destaca os caracteres e a segunda inverte os tons, otimizando a visualização dos elementos. Essa estratégia está ilustrada na Figura 8.

Figura 8 – Fluxo de Pré-Processamento: Conversão para Escala de Cinza e Binarização



Fonte: Autor.

2.2.4 Segmentação

A segmentação é o processo de dividir uma imagem em regiões significativas, separando elementos de interesse do fundo. Este passo é essencial para focar em áreas relevantes da imagem, reduzindo a complexidade computacional e melhorando a precisão das análises subsequentes (GONZALEZ; WOODS, 2018).

Os métodos de segmentação podem variar em complexidade, dependendo das características da aplicação. Técnicas clássicas como a limiarização definem valores de corte para separar regiões com base na intensidade dos pixels, sendo amplamente utilizadas em contextos simples e controlados (SONKA; HLAVAC; BOYLE, 2014). Já métodos mais sofisticados, como a detecção de contornos e bordas, utilizam algoritmos como o operador de Sobel ou o detector de bordas de Canny para identificar e delinear objetos na imagem (GONZALEZ; WOODS, 2018).

Uma abordagem complementar é o agrupamento de regiões, que combina pixels com características similares, como cor e textura, para formar regiões coesas. Este método

é particularmente útil em aplicações que requerem segmentações mais detalhadas, como análise de texturas ou objetos em imagens naturais (SONKA; HLAVAC; BOYLE, 2014).

Além disso, a segmentação desempenha um papel crítico na automação de processos e na integração de sistemas inteligentes, permitindo que algoritmos foquem apenas nas informações relevantes. Isso reduz significativamente a carga de processamento, além de abrir espaço para análises mais detalhadas e específicas.

2.2.5 Extração de características

Após a segmentação, a extração de características é realizada para identificar aspectos específicos da imagem que representam informações visuais de forma compacta. Essas características são usadas para descrever e categorizar o conteúdo visual.

Entre as técnicas de extração de características, destaca-se o Histograma de Gradientes Orientados (HOG), amplamente utilizado para descrever bordas e contornos com base nos gradientes dos pixels. O Esse método visa extrair informações sobre a orientação das arestas presentes em uma imagem, calculadas por meio de métodos de detecção de bordas.(ALMEIDA; BORGES; PAULA, 2018).

O avanço de algoritmos baseados em aprendizado profundo, como redes neurais convolucionais, tem permitido a extração de características mais complexas, muitas vezes imperceptíveis para métodos tradicionais (VARGAS; CARVALHO; VASCONCELOS, 2016).

2.2.6 Reconhecimento

Nesta fase, o sistema utiliza algoritmos de aprendizado de máquina para identificar e categorizar objetos ou padrões presentes nas imagens. Trata-se do momento em que o sistema “compreende” o conteúdo visual de forma mais detalhada, transformando dados brutos em informações interpretáveis.

Redes neurais, especialmente as redes neurais convolucionais (CNNs), revolucionaram o campo ao permitir a extração automática de características diretamente das imagens, eliminando a necessidade de engenharia manual de atributos. Essa capacidade torna as CNNs particularmente eficazes em tarefas complexas, como reconhecimento facial e classificação de imagens médicas (KRIZHEVSKY; SUTSKEVER; HINTON, 2012).

2.2.7 Resultado

Na última etapa, os dados gerados são interpretados e apresentados de forma útil e acionável. Isso pode incluir à metrologia dimensional automatizada (FELICIANO; SOUZA; LETA, 2005), sistemas para monitoramento de acidentes domésticos de idosos

(ZIZA; XIMENES, 2022) ou até mesmo para contagem automática de células em imagens obtidas por microscópio (SECRETáRIO; PIRES, 2018).

Embora o objetivo final varie conforme a aplicação específica, a precisão dessa etapa é diretamente influenciada por todas as fases anteriores. Além disso, essa etapa oferece insights valiosos que podem ser utilizados para refinar os modelos e aprimorar o desempenho do sistema em iterações futuras.

2.3 OCR (Reconhecimento Óptico de Caracteres)

O Reconhecimento Óptico de Caracteres (OCR, do inglês Optical Character Recognition) é uma tecnologia amplamente utilizada para converter texto presente em imagens ou documentos escaneados em dados textuais editáveis e pesquisáveis (MITTAL, 2020). Essa técnica combina conceitos de visão computacional e aprendizado de máquina para identificar padrões de texto e transformá-los em uma representação digital, independentemente do idioma, estilo de fonte ou contexto.

Originalmente desenvolvida para digitalizar livros e documentos impressos, a aplicação do OCR expandiu-se significativamente para áreas como processamento de formulários, digitalização de recibos, automação industrial e, mais recentemente, em aplicações médicas e científicas, onde a extração precisa de dados visuais é crucial para a tomada de decisões.

A expansão do OCR para o reconhecimento em contextos específicos, como displays de aparelhos eletrônicos, monitores e painéis de controle, exige algoritmos especializados capazes de reconhecer caracteres em posições e orientações variadas, adaptando-se a condições de luz e resolução diversificadas (BREUEL et al., 2013).

2.3.1 Histórico e Evolução do OCR

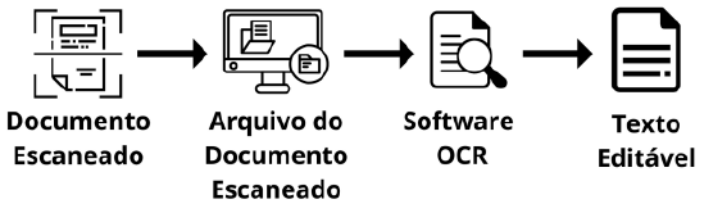
O desenvolvimento do OCR remonta ao início do século XX, com os primeiros sistemas mecânicos capazes de ler caracteres impressos. Na década de 1950, surgiram os primeiros sistemas digitais de OCR, que evoluíram significativamente até o final do século, graças ao avanço em reconhecimento de padrões e processamento de imagem (PLAMONDON; SRIHARI, 2000). A partir dos anos 2000, o OCR foi impulsionado por técnicas de aprendizado de máquina, que aprimoraram a precisão e adaptabilidade desses sistemas, permitindo o reconhecimento de textos em fontes e condições visuais variadas.

2.3.2 Funcionamento do OCR

O OCR (Reconhecimento Óptico de Caracteres) baseia-se em um conjunto de etapas para converter imagens em texto. Trata-se de um processo complexo que envolve várias fases para se obter o resultado final. Desde a aquisição da imagem até a obtenção

do texto digital (SILVA, 2025), cada etapa desempenha um papel crucial. Esse processo é melhor ilustrado na figura 9 abaixo.

Figura 9 – Fluxo de um sistema de OCR



Fonte: (BAZON, 2022).

Conforme a imagem acima ilustra, esse processo começa com a digitalização de um documento físico, capturada por um scanner. Em seguida, a imagem digitalizada é processada pelo software de OCR, que segmenta a imagem em partes, analisa os padrões visuais e converte os elementos reconhecidos em texto.(SILVA, 2025).

Além disso, o OCR utiliza algoritmos avançados para identificar e interpretar os caracteres, levando em consideração fatores como fontes, estilos e tamanhos variados. A precisão do OCR é continuamente aprimorada através de técnicas de aprendizado de máquina, que permitem ao sistema aprender e adaptar-se a diferentes tipos de documentos e condições de imagem em texto (SILVA, 2025).

2.3.3 Limitações do OCR

Embora o OCR tenha avançado significativamente, ele ainda enfrenta desafios em certos contextos, como no reconhecimento de caracteres em textos manuscritos ou em condições de baixa qualidade de imagem. Textos inclinados, distorcidos ou com interferências visuais, como reflexos e ruídos, podem comprometer a precisão dos sistemas OCR. Além disso, em ambientes com fontes não convencionais ou caracteres especiais, o desempenho do OCR tende a ser inferior, exigindo técnicas de ajuste fino (fine-tuning) para melhor adaptação (NAGY, 2000).

A expansão do OCR para o reconhecimento em contextos específicos, como displays de aparelhos eletrônicos, monitores e painéis de controle, exige algoritmos especializados capazes de reconhecer caracteres em posições e orientações variadas, adaptando-se a condições de luz e resolução diversificadas (BREUEL et al., 2013).

2.4 Python

Na década 1980 foi criada a linguagem Python. Sua implementação foi iniciada em dezembro de 1989 por Guido van Rossum na CWI na Holanda como uma sucessora de

outra linguagem de progrmação (TULCHAK; RCHUK, 2016).

A linguagem de programação Python é amplamente utilizada no mundo todo. É uma linguagem de uso geral, o que significa que pode ser usada para criar uma grande variedade de aplicações (YUILL; HALPIN, 2006). É considerada uma das linguagens mais comuns atualmente, devido à sua: Simplicidade, Facilidade de aprendizado, Versatilidade, Compatibilidade com a maioria dos sistemas operacionais, Capacidade de auxiliar outras linguagens.

Tulchak e Marchuk(TULCHAK; RCHUK, 2016) afirmam que : "Python é uma linguagem de programação de alto nível de uso geral amplamente utilizada. Sua filosofia de design enfatiza a legibilidade do código e sua sintaxe permite que os programadores expressem conceitos em menos linhas de código do que seria possível em linguagens como C++ ou Java".

Através do Python, hoje podemos fazer aprimoração de processos e até aplicações na medicina, que é o foco desse trabalho. Onde, que por meio dessa linguagem e o bom uso de algumas bibliotecas, certas atividades na medicina podem ganhar uma ferramenta poderosa.

2.5 Tesseract OCR

O Tesseract OCR é uma das ferramentas mais utilizadas no campo do reconhecimento óptico de caracteres, reconhecida por sua eficiência e versatilidade. Originalmente desenvolvido pela Hewlett-Packard nos anos 1980, o Tesseract passou por diversas atualizações e é atualmente mantido pela Google como um software de código aberto. Ele suporta múltiplos idiomas e pode ser integrado a diversas aplicações para conversão de imagens em texto (SMITH, 2013).

2.5.1 Histórico e Desenvolvimento

O Tesseract OCR é uma ferramenta de reconhecimento óptico de caracteres (OCR) amplamente utilizada, que passou por um longo período de desenvolvimento antes de se consolidar como um dos motores de OCR mais populares. Seu desenvolvimento inicial ocorreu entre 1984 e 1994, em um projeto conjunto entre o HP Labs, localizado em Bristol, e a divisão de scanners da HP no Colorado (PATEL; PATEL; PATEL, 2012).

Em 1995, a tecnologia foi aprimorada com foco em maior precisão e no uso de OCR para compressão de documentos. Apesar de suas inovações, o Tesseract não foi utilizado diretamente pela HP, e pouca documentação foi publicada sobre seu desenvolvimento durante esse período, caracterizado por uma abordagem em "stealth mode"(modo furtivo). Apenas em 2005 o Tesseract foi liberado como software de código aberto pela HP, permitindo sua adoção por uma comunidade mais ampla. Desde então, ele foi modernizado

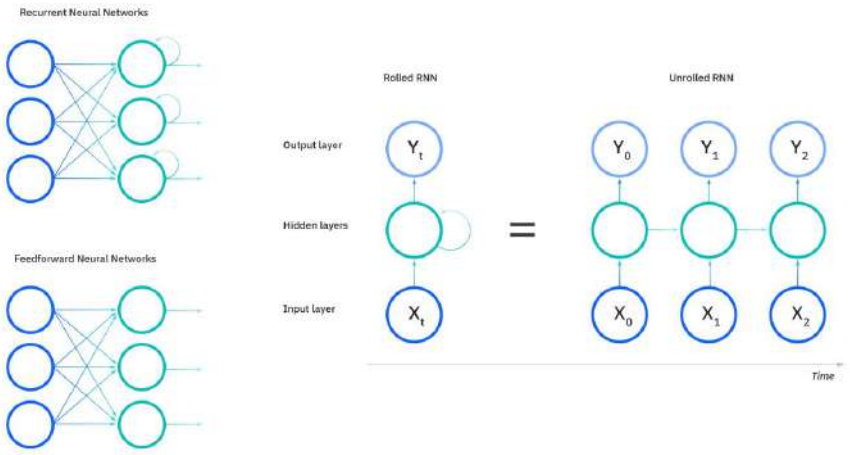
e é atualmente mantido pela Google, com suporte para múltiplos idiomas e foco em portabilidade e acessibilidade. A partir de 2010, as versões lançadas começaram a integrar redes neurais, aumentando sua eficiência e capacidade de reconhecimento (SMITH, 2013).

2.5.2 Arquitetura Neural do Tesseract: RNNs e LSTM

A partir da versão 4.0, lançada em 2021, o Tesseract passou a incorporar redes neurais profundas em sua arquitetura, substituindo o antigo sistema baseado em segmentação por um modelo mais robusto, baseado em Redes Neurais Recorrentes (RNNs) com unidades LSTM (Long Short-Term Memory) (SMITH, 2013).

As RNNs são redes neurais projetadas para processar dados sequenciais, como texto, mantendo uma memória das entradas anteriores. Isso as diferencia das redes neurais tradicionais, como as redes feedforward e convolucionais (CNNs), que tratam cada entrada de forma independente. A figura 10 ilustra essa diferença conceitual entre os dois tipos de redes.

Figura 10 – Redes neurais feedforward versus redes neurais recorrentes.



Fonte: (IBM, 2024).

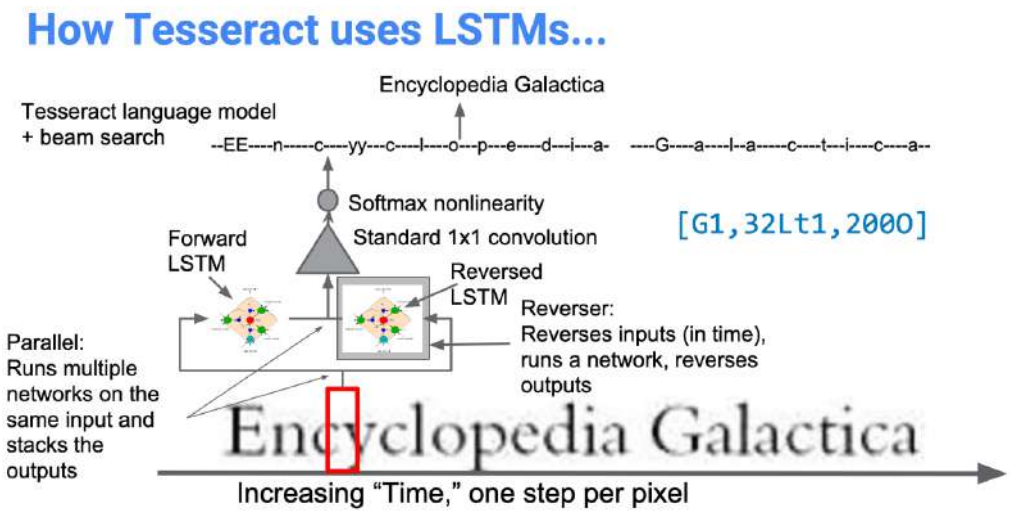
No entanto, as RNNs tradicionais enfrentam dificuldades com dependências de longo prazo devido ao problema do gradiente desaparecendo. Para superar essa limitação, foram desenvolvidas as redes LSTM, que utilizam mecanismos de controle chamados portas (gates). Segundo o artigo da IBM (2024): "Para resolver esse problema, redes LSTM possuem 'células' nas camadas ocultas da rede neural artificial, com três portas: entrada, saída e esquecimento. Essas portas regulam o fluxo de informações necessário para prever a saída da rede". Essas portas permitem que a rede aprenda a manter ou esquecer informações ao longo do tempo, tornando-a mais eficaz em tarefas sequenciais complexas, como o reconhecimento óptico de caracteres.

A adoção dessa arquitetura no Tesseract permitiu avanços significativos na precisão do OCR, especialmente em imagens com fontes variadas, ruído ou desalinhamento. Essa mudança representa um marco na evolução do Tesseract, tornando-o mais adaptável e eficiente em uma ampla gama de aplicações.

2.5.3 Leitura Sequencial com LSTM no Tesseract

Uma das maiores inovações do Tesseract 4.0 é sua capacidade de realizar leitura sequencial utilizando redes neurais LSTM. O motor processa as imagens como uma sequência temporal de pixels em cada linha de texto, semelhante à forma como seres humanos leem da esquerda para a direita (ou da direita para a esquerda, dependendo do idioma)(TIMALSINA, 2025). A figura 11 ilustra esse processo.

Figura 11 – Modo de leitura sequencial utilizando LSTM no Tesseract.



Fonte: (TIMALSINA, 2025).

Nesse modelo, a imagem de uma palavra — como "Encyclopedia Galactica"— é escaneada uma linha de pixels por vez. Cada linha da imagem (ou fatia vertical) é convertida em uma representação numérica e enviada como entrada para a rede LSTM. Ao longo desse processo, a rede vai "lembrando"dos pixels que já analisou, mantendo o contexto ao longo da sequência. Isso permite que ela reconheça caracteres inteiros mesmo quando há sobreposição, ruído ou variações na fonte.

Ao final da análise, uma camada chamada Softmax é responsável por gerar a probabilidade de cada caractere possível em cada posição da sequência. Essa camada age como uma "decisora", indicando qual letra mais provavelmente está sendo representada naquele ponto da imagem.

2.5.4 PyTesseract: Integração do Tesseract com Python

O PyTesseract é uma biblioteca desenvolvida para facilitar a utilização do Tesseract OCR dentro do ambiente Python. Ele permite extrair texto de imagens diretamente em scripts Python, simplificando a automação de processos que envolvem reconhecimento de caracteres. Suas aplicações vão além da simples leitura de texto, abrangendo desde a digitalização de documentos até sistemas avançados, como reconhecimento automático de placas de veículos com OpenCV (CHADHA et al., 2020).

A utilização do PyTesseract segue um fluxo de trabalho básico:

- Carregamento da imagem;
- Pré-processamento da imagem para otimizar a extração de texto (binarização, remoção de ruído, etc.);
- Aplicação do OCR através do PyTesseract;
- Extração e exibição do texto reconhecido.

O processamento de uma imagem pelo Tesseract OCR ocorre por meio de comandos executados via linha de comando, permitindo a extração do texto presente na imagem (PATEL; PATEL; PATEL, 2012). No caso do PyTesseract, essa funcionalidade é integrada ao ambiente Python, facilitando sua utilização em scripts automatizados. A Figura 12 apresenta um exemplo de imagem que será processada para a extração de texto utilizando o PyTesseract.

Figura 12 – Imagem contendo texto

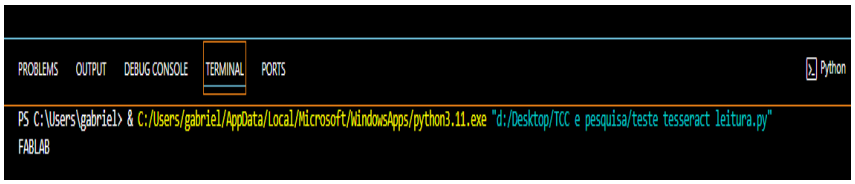
Fonte: Autor.

A extração de texto pode ser otimizada por meio de um fluxo de processamento que inclui etapas como conversão para escala de cinza e binarização. Essas técnicas são amplamente utilizadas para reduzir ruídos e melhorar o contraste entre o texto e o fundo,

facilitando o reconhecimento óptico de caracteres. Neste trabalho, essas transformações serão aplicadas em alguns casos para aprimorar a qualidade da extração de texto.

No entanto, no exemplo atual, esse pré-processamento não é necessário, pois a imagem utilizada já está em formato binário, apresentando um fundo branco e caracteres pretos. Dessa forma, a biblioteca PyTesseract pode processar diretamente a imagem e converter seu conteúdo textual em um formato editável. A Figura 13 ilustra a saída gerada, exibindo o texto extraído a partir da imagem fornecida.

Figura 13 – Saída do processamento feito pelo Tesseract na Figura 12



Fonte: Autor.

Esse fluxo de trabalho destaca a versatilidade do PyTesseract, permitindo sua aplicação em diversas áreas, como digitalização de documentos (PATEL; PATEL; PATEL, 2012), reconhecimento de sinais de trânsito de limite de velocidade via PyTesseract (MARTíN; PODRžAJ; POžRL, 2024), detecção avançada de cyberbullying (LK et al., 2025) e dentre outras.

2.6 Aplicações Médicas

A integração de tecnologias de visão computacional na área médica tem transformado profundamente a maneira como os dados de pacientes são coletados, analisados e visualizados. Segundo (CONSTâNCIO; CARVALHO, 2022), essas tecnologias desempenham um papel importante em sistemas de apoio ao diagnóstico, monitoramento de saúde e automação de processos médicos.

Essa área tem proporcionado uma variedade de aprimoramentos e melhorias em processos médicos. Estudos como de Visão Compultacional em imagens de Raio-X da região toráxica (VAZ, 2023), e o de (JUNIOR, 2024), que exploram a identificação de tubos em radiografias de tórax, exemplificam o potencial dessas tecnologias em diversas aplicações na medicina.

2.6.1 OCR na área da saúde

De acordo com (UEZIMA et al. ,2024), “reconhecimento óptico de caracteres é uma tecnologia que possibilita a extração de texto de documentos como imagens, fotografias, PDFs e diversos outros tipos de arquivos que contenham algum tipo de texto.” Embora o uso de OCR na área da medicina não seja tão abrangente quanto outras tecnologias, ele ainda desempenha um papel importante na digitalização de documentos. Na maioria

dos casos, OCR permite transformar um documento escaneado em um texto editável, facilitando a integração de informações em sistemas eletrônicos (BAZON, 2022).

A tecnologia OCR (Reconhecimento Óptico de Caracteres) pode ser uma ferramenta valiosa para os estabelecimentos de saúde ao permitir a digitalização de registros de pacientes em papel, como históricos médicos, resultados de laboratório e relatórios de imagem. Esse processo não só melhora a precisão dos dados dos pacientes, mas também facilita o acesso e o compartilhamento dessas informações entre os profissionais de saúde, promovendo uma melhor coordenação e eficiência no atendimento (KALRA, 2023).

A figura 14 ilustra os principais benefícios do uso de OCR na área da saúde, destacando como essa tecnologia pode transformar a gestão de informações médicas.

Figura 14 – Melhoria de processo com OCR

Fonte: Autor.

De acordo com OCR... (2023), todas essas vantagens são evidentes:

- **Fluxos de Trabalho Rápidos**:
  - **Melhoria no Tempo de Processamento**: Essa automação melhora significativamente o tempo de processamento e ajuda a economizar tempo para pacientes e médicos.
- **Maior Disponibilidade de Dados**:
  - **Acesso 24 Horas**: A tecnologia OCR disponibiliza dados 24 horas por dia, 7 dias por semana para os usuários.
  - **Extração de Dados Direta**: Como os dados são armazenados digitalmente, o processo de extração de dados torna-se direto, eliminando atrasos no tratamento dos pacientes.

- **Menor Investimento em Mão de Obra**:
  - **Automação de Tarefas Repetitivas**: O setor de saúde compreende várias tarefas repetitivas e tediosas que exigem uma força de trabalho substancial.
- **Minimização de Erros**:
  - **Redução da Intervenção Humana**: Os seres humanos são propensos a erros, especialmente durante processos de saúde complicados e exigentes.
  - **Precisão Aumentada**: Com o OCR, a intervenção humana fica limitada e os erros podem ser bastante minimizados.

2.6.2 Monitoramento de Pacientes

A presença de uma gama de tecnologias nos Estabelecimentos Assistenciais de Saúde é um fator que torna absolutamente essencial que o processo de gestão dessas instituições seja conduzido de maneira meticulosa (FARIA et al., 2018). Isso é fundamental para garantir que os profissionais da saúde possam observar os pacientes com tecnologias que vão proporcionar mais segurança e qualidade.

De acordo com Begg (BEGG; LAI; PALANISWAMI, 2010): "os sistemas biomédicos são projetados para coletar, processar e interpretar dados médicos.". Essa função é fundamental para a eficácia das intervenções médicas, pois permite que os profissionais de saúde acessem dados em tempo real, facilitando diagnósticos rápidos e decisões clínicas precisas.

Além disso, a integração de tecnologias nos processos de monitoramento e tratamento dos pacientes é crucial para elevar o padrão de cuidado oferecido. Tecnologias como sistemas de monitoramento contínuo, dispositivos de imagem médica e softwares de análise de dados desempenham um papel vital na coleta e interpretação de informações de saúde.

Os monitores multiparâmetro são exemplos de equipamentos essenciais que carregam informações de diversos sinais vitais de pacientes adultos, pediátricos e neonatais (BRASIL, 2025). Esses dispositivos são projetados para realizar a leitura contínua dos sinais vitais do paciente, como frequência cardíaca, pressão arterial, saturação de oxigênio, entre outros. A capacidade de monitorar múltiplos parâmetros simultaneamente permite que a equipe médica tenha uma visão abrangente e detalhada da condição de saúde do paciente, facilitando a tomada de decisões clínicas rápidas e precisas (PROLIFE, 2024).

Além de fornecer informações na tela, os monitores multiparâmetro também utilizam alarmes visuais e sonoros para alertar a equipe médica sobre qualquer alteração crítica nos sinais vitais do paciente. Esses alarmes são fundamentais para garantir que intervenções imediatas possam ser realizadas quando necessário, aumentando a segurança e a eficácia

do atendimento (FARIA et al., 2018). A figura 15 ilustra o funcionamento de um monitor multiparâmetro, destacando sua importância no monitoramento contínuo de pacientes e na manutenção de um cuidado de saúde de alta qualidade.

Figura 15 – Monitor multiparâmetro

Fonte: (PROLIFE, 2024).

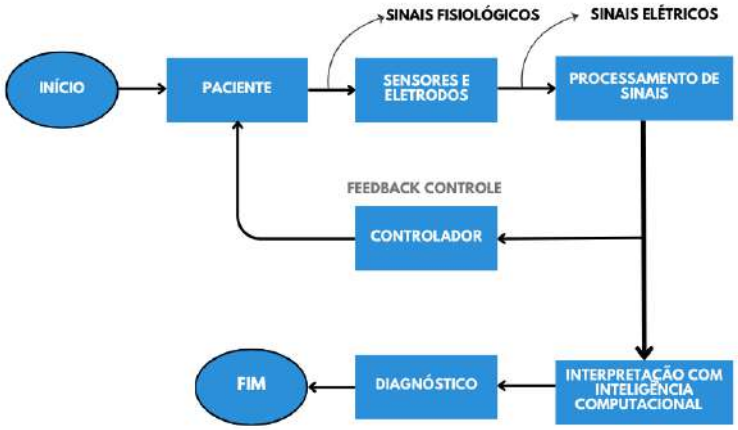
Os parâmetros monitorados possuem funções específicas e críticas no acompanhamento da condição do paciente. Os batimentos cardíacos indicam a frequência com que o coração está bombeando sangue, sendo expressos em batimentos por minuto (bpm). Valores anormais podem indicar arritmias ou insuficiência cardíaca (PROLIFE, 2024). A pressão sanguínea mede a força exercida pelo sangue nas paredes arteriais, sendo um indicador direto da circulação e função cardíaca. Pressões muito altas ou muito baixas podem levar a complicações como infartos ou choque circulatório (BEGG; LAI; PALANISWAMI, 2010). A temperatura corporal é um parâmetro essencial para identificar infecções, hipotermia ou febre, que são condições comuns em pacientes críticos (PROLIFE, 2024).

A frequência respiratória mede a quantidade de respirações por minuto, permitindo a identificação de dificuldades respiratórias ou insuficiência pulmonar. Este parâmetro é crucial para pacientes que utilizam ventilação mecânica, pois ajuda a monitorar a eficácia do suporte respiratório e a detectar possíveis complicações precocemente (PROLIFE, 2024). Além disso, a frequência respiratória é um indicador importante da condição geral do paciente, podendo sinalizar alterações no estado de saúde que requerem intervenção imediata.

Já a saturação de oxigênio, expressa em porcentagem, indica a eficiência com que o oxigênio é transportado pelo sangue. Este parâmetro é essencial no manejo de doenças respiratórias e cardiovasculares, pois permite avaliar a capacidade do sistema circulatório de fornecer oxigênio aos tecidos (BEGG; LAI; PALANISWAMI, 2010). Por fim, o nível de gás carbônico reflete a função pulmonar na eliminação do CO, sendo crucial para monitorar pacientes com doenças obstrutivas ou em ventilação assistida. A medição precisa desses parâmetros é fundamental para garantir um tratamento adequado e eficaz (PROLIFE, 2024).

Um fluxograma que representa o sistema de monitoramento biomédico é apresentado abaixo na figura 16. Este diagrama ilustra de forma clara como os componentes do sistema interagem para garantir a coleta, processamento e interpretação dos dados médicos.

Figura 16 – Fluxograma dos componentes fundamentais de um sistema de monitoramento biomédico



Fonte: Adaptado de (BEGG; LAI; PALANISWAMI, 2010).

Através deste diagrama, é possível visualizar o fluxo de informações desde o paciente até o diagnóstico, destacando a importância de cada etapa no processo de monitoramento contínuo.

2.6.3 Suporte à Decisão Clínica

A integração de visão computacional com o monitoramento de pacientes em tempo real pode representar um marco significativo para a área da saúde, especialmente em ambientes críticos como unidades de terapia intensiva. Nessas unidades, equipamentos médicos geram continuamente dados vitais, incluindo pressão arterial, frequência cardíaca e níveis de oxigênio, que são essenciais para a tomada de decisões rápidas e eficazes. O desafio, contudo, está na necessidade de monitoramento constante e na análise eficiente dessas informações para garantir que intervenções sejam realizadas no momento certo.

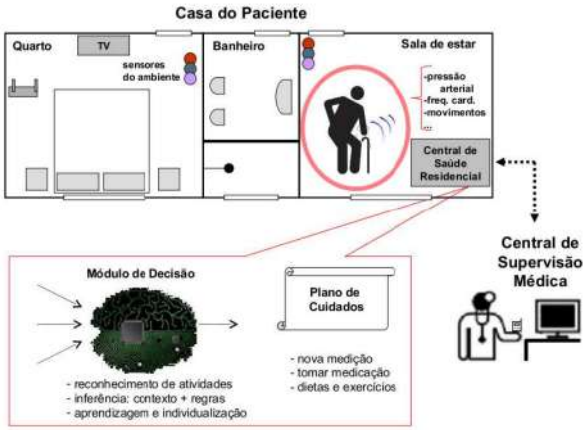
As UTIs são ambientes altamente complexos e exigem uma infraestrutura robusta e profissionais altamente qualificados para garantir a segurança e a eficácia do cuidado aos pacientes. A carga de trabalho nas UTIs é intensa, com pacientes críticos necessitando de intervenções frequentes e monitoramento contínuo. Estudos mostram que a relação paciente/enfermeiro é crucial para a qualidade do atendimento, com um aumento na razão paciente/enfermeiro correlacionado a um aumento significativo de eventos adversos (VIANA, 2012). Além disso, a pandemia de COVID-19 revelou a fragilidade das UTIs, com taxas de ocupação frequentemente ultrapassando 80%, que é considerado crítico por especialistas. A alta demanda por leitos e a escassez de profissionais qualificados agravam

ainda mais a situação, tornando o monitoramento eficiente e a rápida intervenção ainda mais essenciais (TOKARSKI, 2021).

Esses ambientes são marcados por um foco intenso na manutenção das funções vitais, muitas vezes priorizando o modelo biomédico em detrimento de abordagens mais humanísticas e relacionais. Esse cenário é exacerbado pela presença constante de tecnologia avançada e por uma rotina de atendimentos rápidos e procedimentos técnicos, que podem reduzir a interação sensível e o acolhimento dos pacientes e seus familiares (NASCIMENTO; TRENTINI, 2000).

Nesse contexto, o monitoramento remoto de pacientes surge como uma solução promissora para melhorar a eficiência e a qualidade do cuidado. O monitoramento remoto permite que médicos acompanhem a condição de seus pacientes em tempo real, independentemente da localização geográfica (CARVALHO et al., 2010). Essa abordagem é especialmente útil para pacientes com condições crônicas, que podem se beneficiar de um acompanhamento constante para minimizar riscos e prevenir complicações. Além disso, o monitoramento remoto oferece mais conforto e segurança aos pacientes, que podem receber cuidados de casa, reduzindo a necessidade de deslocamentos frequentes como apresentado na figura 17.

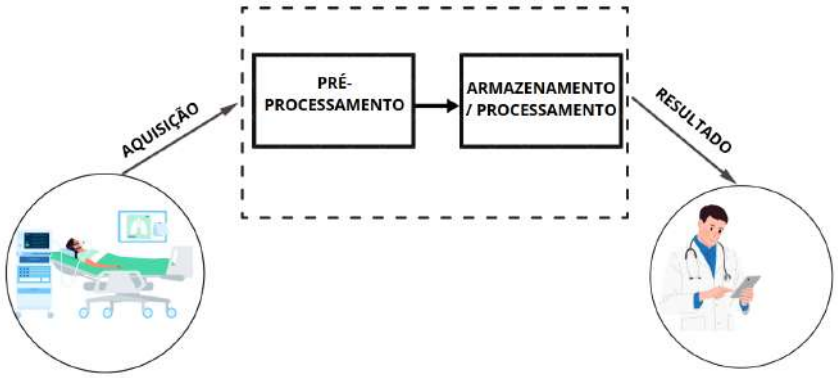
Figura 17 – Estrutura geral do SADS



Fonte: (CARVALHO et al., 2010).

Complementando essa abordagem, implementação de sistemas baseados em OCR para capturar e digitalizar dados diretamente dos monitores médicos traz uma possível solução para essa circunstância. Com o desenvolvimento dessas tecnologias permitem centralizar as informações em plataformas de análise e armazenamento, facilitando a visualização e o processamento simultâneo de grandes volumes de dados para os profissonais da saude, independentemente do lugar em que esteja, como ilustra a figura 18. Além disso, a automação desse processo reduz o risco de erros humanos, melhora a eficiência das equipes médicas e contribui para um cuidado mais ágil e preciso com os pacientes (KALRA, 2023).

Figura 18 – Processo de Monitoramento



Fonte: Autor.

Assim, qualquer inovação tecnológica, como a digitalização automatizada de dados médicos, deve buscar não apenas aprimorar a eficiência do monitoramento, mas também contribuir para um cuidado mais humanizado e integrador.

2.6.4 Desafios e Limitações das Aplicações Médicas

A integração de tecnologias aos fluxos de trabalho hospitalares são essenciais para a aprimoração do processo, porém existem algumas limitações. O ambiente de uma UTI, com sua alta demanda por monitoramento constante e intervenções urgentes, pode exacerbar essas limitações. Equipamentos médicos muitas vezes sofrem com descalibração e ruídos, o que pode gerar informações imprecisas ou interpretações errôneas dos sistemas automatizados. Em UTIs, onde as decisões precisam ser rápidas e precisas, essa limitação pode comprometer a eficácia dos diagnósticos e tratamentos (LEITE, 2011).

Muitas instituições de saúde operam com infraestruturas legadas, o que dificulta a implementação de sistemas avançados de monitoramento e diagnóstico. Além disso, a capacitação dos profissionais para manusear tais sistemas e interpretar os resultados ainda é um gargalo significativo. Esses fatores são agravados por preocupações relacionadas à privacidade dos pacientes e à segurança de dados, especialmente em cenários onde informações sensíveis são compartilhadas e armazenadas eletronicamente (LEITE, 2011). Essas preocupações exigem a criação de protocolos rigorosos para o manuseio de dados sensíveis, garantindo que a tecnologia seja não apenas eficiente, mas também ética e segura.

Além dos desafios técnicos, as condições das UTIs no Brasil apresentam limitações significativas. A distribuição desigual de leitos de UTI é um problema histórico, agravado pela pandemia de COVID-19. Estudos mostram que 73% das microrregiões brasileiras não cumprem a meta da OMS de 1 a 3 leitos de UTI para cada 100 mil habitantes, com regiões como o Norte e Nordeste sendo as mais afetadas (PITTA, 2020), como apresentado na figura 19. A escassez de leitos e a alta demanda sobrecarregam o sistema de saúde, comprometendo a qualidade do atendimento e aumentando a mortalidade (AGUIAR et

al., 2021).

Figura 19 – Aderencia à recomendação de leitos de UTI por Microrregião do IBGE

Fonte: (PITTA, 2020).

A infraestrutura precária das UTIs brasileiras também contribui para a ineficiência dos sistemas de monitoramento. Equipamentos desatualizados e falta de manutenção adequada são comuns, resultando em dados imprecisos e dificultando a implementação de tecnologias avançadas (CONTAIFER, 2024).

A não conformidade da infraestrutura das UTIs brasileiras compromete significativamente a eficácia do monitoramento contínuo dos pacientes. Um dos principais fatores é a baixa proporção de enfermeiros por paciente, em contraste com o padrão ideal adotado em países desenvolvidos. Essa desproporção sobrecarrega os profissionais de enfermagem, dificultando a vigilância constante e a resposta rápida a alterações no estado clínico dos pacientes. A presença limitada de enfermeiros especialistas e a dependência de técnicos de enfermagem para funções críticas reduzem a capacidade de interpretação precisa dos dados gerados pelos sistemas de monitoramento como é ilustrado na Tabela 2.

Tabela 2 – UTI: Realidade Brasileira x Padrões Internacionais segundo (PITTA, 2020).

Aspecto	**UTI - Países Desenvolvidos**	**UTI - SUS (Brasil)**
Proporção Enfermeiro/-Paciente	1:1	1:5
Equipe de Apoio	Todos são enfermeiros especialistas	Técnicos de enfermagem (1:2)
Monitoramento e Resposta	Reconhecimento precoce e intervenção rápida	Reconhecimento mais tardio
Segurança do Paciente	Redução de eventos adversos, menor mortalidade	Maior risco de complicações

A resistência à adoção de novas tecnologias é outro desafio. Muitos profissionais de saúde relutam em utilizar sistemas automatizados, preferindo métodos tradicionais de

monitoramento e diagnóstico (MURACA, 2025). A educação contínua e a demonstração de que essas tecnologias complementam o cuidado humano são essenciais para superar essa barreira.

Por fim, a desigualdade no acesso às inovações tecnológicas é um problema crítico. Regiões com infraestrutura limitada, como áreas rurais, enfrentam dificuldades para implementar sistemas avançados de monitoramento, aprofundando as disparidades existentes. Garantir que essas ferramentas sejam acessíveis a todos, independentemente de classe social ou localização, é fundamental para o sucesso da revolução tecnológica na saúde(MURACA, 2025).

2.6.5 Por que e Como: Fundamentação e Caminho para o Desenvolvimento

Apesar da crescente evolulação de tecnoçogias na área da saúde, a realidade das Unidades de Terapia Intensiva no Brasil ainda é marcada por limitações estruturais e tecnológicas (CONTAIFER, 2024). Embora existam equipamentos modernos capazes de transmitir dados vitais dos pacientes, a maior parte das UTIs públicas brasileiras não possui acesso a esses recursos (PITTADA, 2021). Essa carência tecnológica compromete o monitoramento contínuo e automatizado dos pacientes, exigindo atenção constante por parte das equipes médicas, muitas vezes em contextos de sobrecarga de trabalho.

Diante desse cenário, o presente trabalho propõe uma abordagem alternativa e viável, baseada no uso de algoritmos de visão computacional e técnicas de PDI. A proposta consiste na captura das informações exibidas nos monitores multiparâmetros por meio de vídeos, processando essas imagens com a biblioteca pytesseract, uma das mais consolidadas para OCR em Python. Por meio desse processamento, é possível extrair os dados vitais em tempo real e, com isso, gerar alertas automáticos quando parâmetros indicarem condições críticas.

O sistema desenvolvido é uma interface gráfica que permite ao usuário inserir vídeos gravados de monitores reais. A partir desses vídeos, os dados numéricos são lidos, interpretados e utilizados como base para a emissão de alertas — um primeiro passo para um sistema de monitoramento remoto e automatizado que poderá futuramente ser integrado a protocolos hospitalares.

No próximo capítulo, será apresentado o desenvolvimento completo do sistema: desde a aquisição e pré-processamento dos vídeos, passando pela extração de dados com OCR, até a implementação dos alertas. Esse processo representa um avanço importante rumo à automação da vigilância em tempo real em ambientes hospitalares, estabelecendo as bases para um sistema de suporte.

3 Desenvolvimento

O desenvolvimento deste trabalho está centrado na aplicação de visão computacional para o monitoramento de dados médicos, com Python como a linguagem de programação escolhida. Python foi selecionado devido à sua versatilidade e ampla disponibilidade de bibliotecas, como OpenCV para processamento de vídeo, Pytesseract para OCR e Tkinter para construção de interfaces gráficas.

O sistema projetado opera processando vídeos capturados de monitores médicos, extraindo informações vitais como frequência cardíaca e saturação de oxigênio. Esses dados são apresentados em uma interface, que não apenas permitem a visualização clara das tendências, mas também emitem alertas em caso de condições críticas. O fluxo de trabalho inclui etapas de pré-processamento das imagens, segmentação de regiões de interesse (ROIs) e aplicação de técnicas de OCR para conversão dos dados visuais em valores numéricos. Todo o processo foi projetado para ser modular, possibilitando ajustes nas configurações e expansão do sistema para novas aplicações.

Neste capítulo, são detalhadas as metodologias adotadas, incluindo a implementação técnica, os algoritmos empregados e a construção da interface gráfica. Adicionalmente, são apresentados os desafios enfrentados durante o desenvolvimento, como o tratamento de ruídos em imagens de baixa qualidade e a integração do Tesseract OCR ao sistema. As soluções implementadas, como o uso de técnicas de binarização e ajustes de parâmetros no OCR, serão exploradas, destacando a eficácia do sistema em fornecer informações precisas.

3.1 Principais Bibliotecas Utilizadas

As bibliotecas utilizadas neste projeto foram fundamentais para viabilizar cada uma das etapas, desde o processamento de vídeo até a exibição dos resultados em uma interface gráfica. A escolha dessas bibliotecas foi baseada em sua robustez, compatibilidade com Python e na capacidade de atender às necessidades específicas de manipulação de imagens, extração de texto e interatividade com o usuário.

Entre as principais bibliotecas, o OpenCV se destacou como a principal ferramenta para manipulação de imagens e vídeos. Sua flexibilidade permitiu realizar operações como fragmentação de vídeos em frames, seleção de regiões de interesse e pré-processamento das imagens, incluindo conversão para escala de cinza e aplicação de filtros de threshold. Já a biblioteca pytesseract, utilizada para integração com o Tesseract OCR, foi indispensável para a extração de texto a partir das imagens tratadas.

Além disso, bibliotecas como Tkinter e Matplotlib desempenharam um papel crucial na criação da interface gráfica e na visualização dos resultados em tempo real. O Tkinter

possibilitou a construção de uma interface intuitiva e interativa, enquanto o Matplotlib foi utilizado para gerar gráficos que facilitam a análise dos dados monitorados. Complementando essas ferramentas, bibliotecas como NumPy e Pillow (PIL) proporcionaram suporte técnico adicional, permitindo manipulação eficiente de arrays e tratamento de imagens. A Tabela 3 resume as bibliotecas principais empregadas no projeto, destacando suas funcionalidades e contribuições específicas.

Tabela 3 – Principais bibliotecas utilizadas no projeto

Biblioteca	**Função Principal**
`OpenCV`	Processamento de vídeo e manipulação de imagens, incluindo recorte e conversão para escala de cinza.
`pytesseract`	Integração com o Tesseract OCR para extração de texto das imagens tratadas.
`Tkinter`	Desenvolvimento da interface gráfica para interação com o sistema.
`NumPy`	Manipulação de arrays para representar os frames e ROIs.
`Matplotlib`	Geração de gráficos em tempo real para visualização dos dados extraídos.
`PIL (Pillow)`	Tratamento de imagens para exibição na interface gráfica.

3.2 Fragmentação de Vídeo em Frames

A captura e fragmentação de vídeos em frames individuais é essencial para o processamento de vídeo em tempo real, permitindo a análise frame a frame para a extração de informações específicas. A fragmentação do vídeo isola momentos de interesse, aplicando algoritmos de visão computacional de forma precisa. A figura 20 apresenta um exemplo de um frame capturado do vídeo utilizado no projeto.

Figura 20 – Frame capturado do vídeo de monitoramento

Fonte: Autor.

Para implementar essa funcionalidade, foi utilizada a biblioteca OpenCV. Cada frame capturado é convertido em uma estrutura manipulável no formato de array do NumPy, facilitando operações como corte e pré-processamento das imagens.

3.2.1 Período de Amostragem

Capturar todos os frames de um vídeo pode fornecer alta granularidade, mas nem sempre é necessário e pode ser computacionalmente pesado. Em um cenário real de monitoramento médico, o profissional de saúde não requer atualização de valores dezenas de vezes por segundo; costuma bastar obter um valor médio ou representativo em intervalos mais amplos (por exemplo, a cada alguns minutos). A seguir, discutem-se aspectos importantes da amostragem de frames:

- **Janelas de coleta e média temporal**: Em vez de processar cada frame individualmente, o sistema pode coletar vários frames durante uma janela curta (por exemplo, 1 segundo), extrair os valores de interesse de cada frame dessa janela e então calcular uma média (ou mediana) para representar aquele intervalo. Isso reduz o ruído de leituras pontuais e fornece um valor mais estável.
- **Frequência de atualização**: Após obter o valor médio de uma janela de 1 segundo, o sistema poderia aguardar um intervalo maior (por exemplo, 5 a 10 minutos) antes de repetir o procedimento. Em cenários de protótipo com vídeos curtos, essa estratégia não é aplicável diretamente (já que o vídeo pode ter apenas alguns segundos), mas é importante mencionar que, em implementação real, essa abordagem atende às necessidades clínicas sem sobrecarregar o processamento ou gerar excesso de dados.
- **Proposta de estratégia em sistema real**:
  - Coletar todos os frames em uma janela de 1 segundo.
  - Extrair valores de cada frame dentro dessa janela.
  - Calcular média ou mediana para reduzir ruídos esporádicos.
  - Atualizar o valor exibido e armazenado apenas a cada X minutos (5–10 min), ou definir outro intervalo conforme necessidade clínica.
  - Caso haja alerta crítico dentro da janela, pode-se emitir imediatamente, mas a tendência geral é mostrada via valor médio.

Como os vídeos de teste neste trabalho são curtos (alguns segundos a minutos), a estratégia real de coleta em intervalos de 5–10 minutos não pôde ser implementada. No entanto, mencionar esse planejamento demonstra que se considerou a eficiência e aplicabilidade em ambiente real de monitoramento continuo.

3.2.2 Fonte dos Dados de Teste

Para uma análise estruturada, este estudo recorreu a quatro vídeos obtidos em plataformas de livre acesso na internet, selecionados com o objetivo de compor um conjunto heterogêneo capaz de desafiar o sistema em diferentes condições de captura e devidamente referenciado ao longo do capítulo de resultados, onde se indica o caminho para cada vídeo.

Embora todos tenham em comum a exibição contínua de sinais vitais em monitores multiparamétricos, eles diferem deliberadamente do modelo e condições de iluminação. Dois deles apresentam iluminação uniforme e contraste elevado, oferecendo uma situação relativamente ideal para o OCR, enquanto os demais exibem reflexos, zonas de sombra e curtos instantes de subexposição, simulando a realidade de ambientes hospitalares onde a luz ambiente muda com frequência.

A figura 21 exibe um frame representativo de cada um desses vídeos. A fim de aproximar-se do fluxo esperado em um cenário clínico, definiu-se que, ao longo da reprodução, seria realizada exatamente uma leitura por segundo para cada parâmetro monitorado — frequência cardíaca, saturação de oxigênio e frequência de pulso.

Figura 21 – Base de vídeos utilizados no TCC

Fonte: Autor.

Os quatro arquivos foram escolhidos não apenas pelas diferenças técnicas já mencionadas, mas também pela facilidade de comprovar sua procedência pública: todos se enquadram em licenças que permitem reutilização acadêmica e, por não exibirem pacientes reais, eliminam preocupações éticas de confidencialidade. O conteúdo visual foi capturado em em ambientes hospitalares, fato que assegura diversidade de modelos de monitores e de layouts gráficos na tela. Essa variedade enriquece a avaliação: se o sistema mantiver desempenho consistente ao longo desses cenários contrastantes, aumenta-se a evidência de que o método de pré-processamento, extração e pós-tratamento é suficientemente geral para aplicações futuras em ambiente hospitalar.

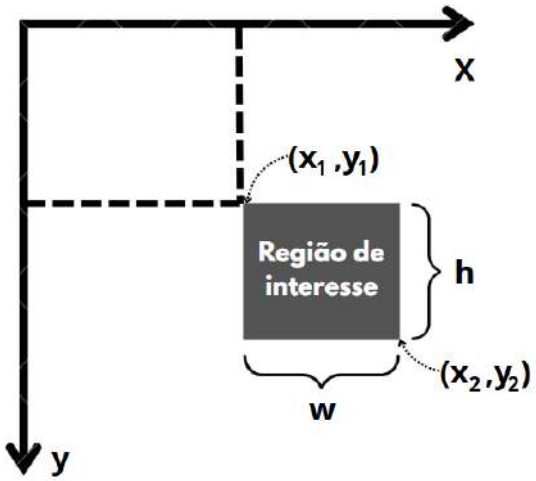
Em síntese, o conjunto de vídeos funciona como um banco de testes versátil: abrange

diversas resoluções, taxas de quadros distintas, variações de iluminação e ocorrência de alertas visuais. Ao registrar leituras de forma uniforme, gera-se uma série temporal coerente entre os arquivos, possibilitando, no capítulo de resultados, a análise comparativa estruturada.

3.3 Seleção de Regiões de Interesse (ROIs)

Região de Interesse é uma área específica de uma imagem ou vídeo que é selecionada para análise ou processamento. No contexto da biblioteca OpenCV, a ROI é definida utilizando coordenadas retangulares que delimitam a área de interesse. Essas coordenadas são representadas por quatro valores: (x, y, w, h), onde (x, y) são as coordenadas do canto superior esquerdo da região, e (w, h) são a largura e altura da região, respectivamente. A figura 22 ilustra essa noção geométrica, mostrando como as ROIs são delimitadas dentro de um frame.

Figura 22 – Noção geométrica de uma Região de interesse



Fonte: Autor.

Essa abordagem foi necessária devido à quantidade de informações exibidas no painel de monitoramento de pacientes como ilustra a figura 20. O objetivo principal era focar exclusivamente nos valores numéricos relevantes, como frequência cardíaca, saturação de oxigênio e frequência de pulso, ignorando elementos visuais desnecessários que poderiam atrapalhar na leitura do valor na imagem.

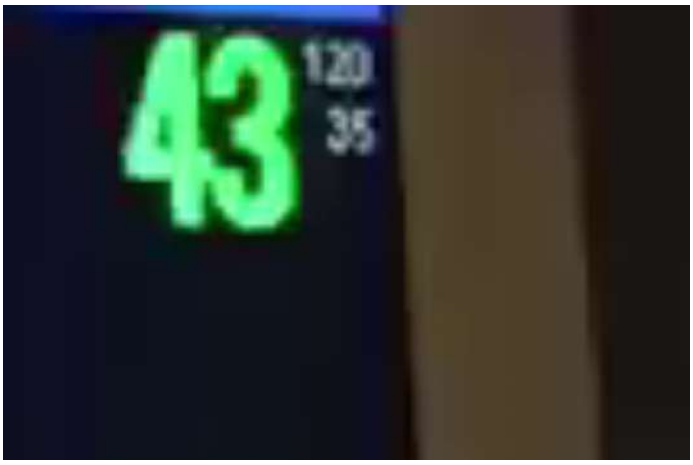
A seleção das ROIs foi implementada manualmente, utilizando coordenadas retangulares para delimitar as áreas específicas dentro de cada frame. Este processo envolveu a identificação precisa das regiões onde os valores numéricos de interesse eram exibidos, garantindo que apenas as informações relevantes fossem capturadas para posterior processamento. Com uma função, desenvolvida com a biblioteca OpenCV, recebe essas coordenadas e retorna os segmentos de imagem que contêm os valores de interesse. Esta

função é fundamental para isolar as áreas de interesse e facilitar a extração precisa dos dados.

Cada ROI foi associada a um parâmetro clínico específico, como frequência cardíaca, saturação de oxigênio ou frequência de pulso. Essas associações foram configuradas no sistema como um dicionário, onde cada chave representa um parâmetro clínico e o valor correspondente é a coordenada da ROI. Esta abordagem modular permite ajustes fáceis e rápidos, caso seja necessário alterar as áreas de interesse ou adicionar novos parâmetros ao sistema.

A figura 23 mostra um exemplo de uma das ROIs selecionadas no painel de monitoramento, destacando a área que contém os valores de saturação de oxigênio. Esta imagem ilustra como as ROIs são delimitadas e como o sistema foca nas informações relevantes, ignorando elementos visuais desnecessários. A precisão na seleção das ROIs é crucial para garantir a qualidade dos dados extraídos e a eficácia do sistema de monitoramento.

Figura 23 – Região de interesse



Fonte: Autor.

A definição manual das ROIs, embora trabalhosa, oferece um alto grau de precisão e controle sobre as áreas de interesse. Este método permite que o sistema seja adaptado a diferentes cenários e necessidades clínicas, garantindo que os dados mais relevantes sejam sempre capturados e analisados. Além disso, a modularidade da configuração das ROIs facilita a manutenção e a atualização do sistema, tornando-o flexível e escalável para futuras expansões.

3.4 Tratamento das Imagens

O tratamento das imagens extraídas das Regiões de Interesse é uma etapa essencial para garantir a precisão e a confiabilidade do sistema. Este processo envolve técnicas que simplificam a visualização e destacam os elementos de interesse, preparando as imagens

para análise pelo Pytesseract. A seguir, são apresentadas as etapas realizadas e os motivos pelos quais essas abordagens foram implementadas.

3.4.1 Preparo Visual

De acordo com (AWARI, 2023), para garantir uma extração de texto mais precisa pelo OCR, é essencial realizar um pré-processamento na imagem. Esse processo pode envolver várias etapas, como a conversão da imagem para tons de cinza, a binarização para destacar o texto, e a remoção de ruídos que possam interferir na leitura. Essas técnicas ajudam a melhorar significativamente a qualidade do texto extraído, tornando-o mais claro e legível para o reconhecimento óptico de caracteres.

Na preparação das imagens das ROIs, inicialmente foi feito a conversão para tons de cinza que é uma etapa do pré-processamento. Essa conversão reduz a complexidade visual ao eliminar as cores, que podem introduzir ruídos ou dificultar o reconhecimento dos caracteres. Com isso, o sistema passa a focar exclusivamente nos contornos e formas dos números presentes nas imagens, otimizando a precisão.

Uma outra etapa crucial para aumentar a eficiência do OCR, foi feito a inversão dos pixels das imagens tratadas, transformando pixels pretos em brancos e vice-versa. Essa abordagem é especialmente eficaz para lidar com imagens onde os números estão dispostos em fundos complexos, como áreas escuras ou texturizadas. A inversão de pixels destaca os caracteres de interesse, criando um contraste mais nítido entre o texto e o fundo.

Além dos passos anteriores, foi de suma importância o redimensionamento das imagens das Regiões de Interesse. As imagens foram ampliadas em 6,6 vezes o tamanho original. Esse redimensionamento aumenta a densidade de informações visíveis nos caracteres, melhorando a clareza para o reconhecimento pelo OCR.

A Figura 24 apresenta o resultado de uma ROI após todos os procedimentos de tratamento de imagem aplicados.

Figura 24 – ROI tratada

Fonte: Autor.

3.5 Integração do Pytesseract OCR e OpenCV

A integração entre o Pytesseract OCR e o OpenCV desempenhou um papel central no desenvolvimento deste trabalho, permitindo a combinação de ferramentas de visão computacional com algoritmos avançados de reconhecimento óptico de caracteres. Essa sinergia possibilitou a criação de um sistema robusto para análise e extração de dados numéricos diretamente de vídeos de monitores médicos, atendendo às exigências de precisão e eficiência em aplicações de monitoramento.

O OpenCV foi utilizado para capturar e processar frames de vídeo, permitindo a segmentação precisa das Regiões de Interesse. Esses recortes foram então submetidos ao Pytesseract OCR para a conversão dos elementos visuais em texto. Embora o sistema tenha demonstrado alta eficiência na maioria das leituras, algumas situações apresentaram desafios específicos, como a ausência temporária de números nas ROIs durante alertas nos monitores médicos. A Figura 25 ilustra uma dessas situações, onde o quadrante da ROI não apresenta valores numéricos devido ao alerta visual emitido.

Figura 25 – Exemplo de ROI sem valores numéricos devido a alertas do monitor

Fonte: Autor.

3.5.1 Extração de valores de imagens

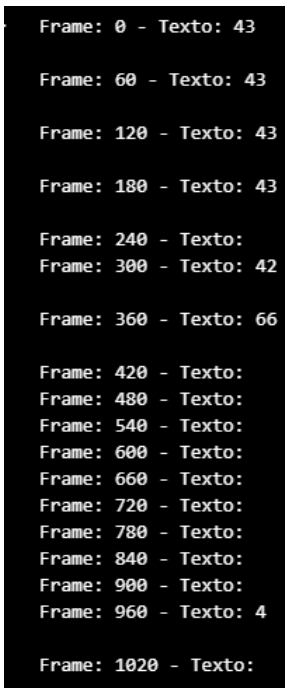
No núcleo do processo de extração de dados, o OpenCV desempenhou um papel importante ao fornecer ferramentas robustas para capturar frames, identificar as ROIs e realizar ajustes de pré-processamento. Essas operações prepararam as imagens para o Pytesseract, que realizou a conversão de dados visuais em texto.

A integração harmoniosa dessas blibiotecas foi fundamental para garantir que o sistema conseguisse processar dados em tempo real, mesmo em cenários com variações de iluminação ou ruídos visuais apresentados nos vídeos. Além disso, técnicas de pré-processamento como binarização e remoção de ruído foram aplicadas para melhorar a qualidade das imagens antes da análise pelo Tesseract. Essas etapas foram essenciais para

aumentar a precisão do reconhecimento de caracteres, especialmente em ambientes com condições adversas.

Durante a etapa de extração, valores reconhecidos nas ROIs foram validados e exibidos no terminal, permitindo ajustes contínuos nos parâmetros do sistema. A Figura 26 apresenta os valores capturados diretamente pelo OCR, evidenciando como os dados foram estruturados durante a validação inicial.

Figura 26 – Resultados obtidos de algumas leituras de imagens com o Pytesseract



Fonte: Autor.

3.5.2 Comandos e Configurações no Pytesseract (psm e oem)

A leitura de texto em imagens pode ser influenciada por diversos fatores, incluindo a maneira como os textos estão geometricamente estabelecidos ou simbologias, como ilustrado na figura 27. Textos podem aparecer em diferentes formatos, tamanhos e orientações, o que pode dificultar o reconhecimento óptico de caracteres.

Para lidar com essas variações, é essencial configurar corretamente os parâmetros do PyTesseract, garantindo uma leitura mais eficiente. A função de leitura do PyTesseract permite passar configurações específicas que ajudam a otimizar o reconhecimento de texto, adaptando-se às características da imagem (ROSEBROCK, 2021). Para otimizar o desempenho do OCR, comandos específicos foram configurados no projeto, garantindo uma leitura melhor dos valores numéricos. Entre os principais comandos utilizados, destacam-se:

- `-psm 6`: Define o modo de segmentação da página, ideal para imagens contendo caracteres individuais em vez de blocos de texto.

Figura 27 – Diversidade de padrões de textos

Fonte: (ROSEBROCK, 2021).

- `-oem 3`: Especifica o uso do motor de OCR baseado em redes neurais, proporcionando maior precisão nas leituras, especialmente para caracteres pequenos ou complexos.

No código, os comandos utilizados dentro da função de leitura foram configurado como `-oem 3 -psm 6 outputbase digits`. A configuração desses parâmetros é fundamental para adequar o Tesseract OCR ao cenário de monitoramento médico descrito neste trabalho. Segundo Uezima et al. (UEZIMA et al., 2024), o uso do motor de OCR baseado em redes neurais é especialmente eficaz em aplicações que exigem alta precisão, como no reconhecimento de caracteres em imagens complexas.

3.5.3 Apresentação e Representação Gráfica dos Dados

De acordo com (IMPORTANCE. . . , 2025) "Um gráfico é uma representação gráfica de dados, na qual os dados são representados por símbolos, como barras em um gráfico de barras, linhas em um gráfico de linhas ou fatias em um gráfico de pizza". Essa representação visual torna os dados mais significativos e acessíveis, permitindo que decisões sejam tomadas de maneira mais eficaz.

Os gráficos oferecem várias vantagens importantes. Eles simplificam dados complexos, tornando-os mais fáceis de entender e memorizar. Onde, (ISLAM, 2025) destaca essas vantagens como:

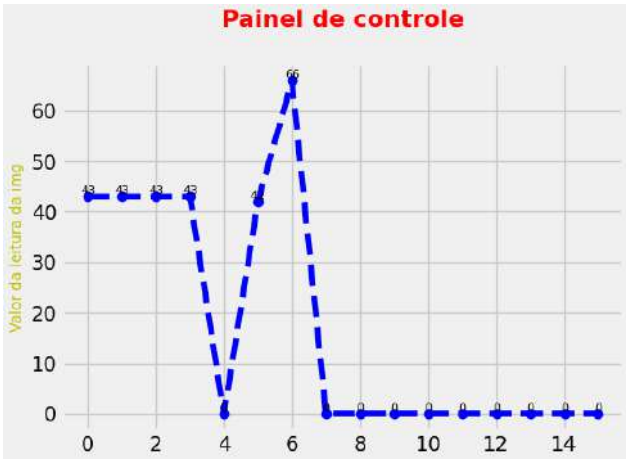
- **Clareza aprimorada:** Simplificam dados complexos, tornando-os mais fáceis de entender.
- **Memória aprimorada:** Recursos visuais são mais memoráveis do que dados brutos.
- **Comunicação eficaz:** Facilitam a comunicação clara e concisa de insights.
- **Identificação de padrões:** Revelam tendências e padrões ocultos.

- **Maior engajamento:** Visuais são mais envolventes e chamam mais atenção do que tabelas de números.

Partindo desse raciocinio, após a extração inicial, os dados numéricos reconhecidos pelo Tesseract foram processados e representados graficamente utilizando a biblioteca Matplotlib. Os gráficos dinâmicos gerados permitiram uma visualização clara dos valores monitorados, destacando padrões e anomalias.

A Figura 28 apresenta um exemplo de gráfico criado a partir dos dados extraídos, ilustrando a eficiência do sistema em visualizar alterações nos parâmetros monitorados.

Figura 28 – Representação gráfica dos valores extraídos



Fonte: Autor.

Essa integração entre OpenCV, Tesseract OCR e ferramentas gráficas resultou em um sistema adaptável e que possibilitou o desenvolvimento de uma interface, assunto esse que será abordado posteriormente.

3.6 Interface gráfica e Feedback Visual

Uma interface gráfica amigável é essencial para garantir uma experiência positiva ao usuário. Ela deve ser intuitiva, acessível e eficiente, permitindo que os usuários realizem suas tarefas com facilidade e satisfação (O. . . , 2024). Ou seja, elementos como layout claro, ícones compreensíveis e navegação intuitiva são fundamentais para criar uma interface que não apenas atenda às necessidades dos usuários, mas também os envolva de maneira agradável.

De acordo com (AMSTEL, 2010), " O Design de Interação é mais uma proposta para trazer aquilo que falta à Engenharia no desenvolvimento de novas tecnologias: a preocupação com o usuário. Seu diferencial perante propostas mais antigas como a Interação Humano-Computador e a Ergonomia é que ele não trata da solução de problemas, mas sim da intermediação entre pessoas". Portanto, a interface gráfica desempenha um papel

fundamental na interação do usuário com o sistema, oferecendo uma maneira intuitiva e eficiente de visualização e controle.

3.6.1 Desenvolvimento da Interface Gráfica

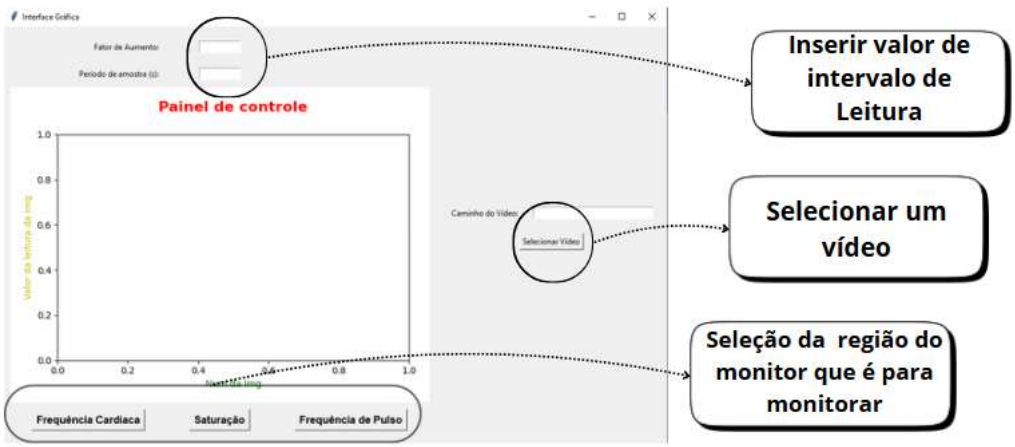
A interface gráfica desenvolvida para o sistema de monitoramento de dados médicos foi projetada com o objetivo de proporcionar uma interação intuitiva para os usuários. A ideia central da interface é permitir que os usuários insiram vídeos gravados de monitores médicos reais, a partir dos quais os dados numéricos são lidos, interpretados e utilizados como base para a emissão de alertas.

O desenvolvimento da interface começou com a definição dos controles necessários para garantir uma operação simples. Foram incluídos três botões principais, cada um representando uma região de interesse dos monitores multiparamétricos: os valores de batimento cardíaco, frequência de pulso e oxigenação. Esses botões permitem ao usuário selecionar quais dados deseja monitorar.

Além disso, a interface possui um campo de frequência de amostra, onde o usuário pode definir o intervalo de tempo em que as informações serão capturadas, lidas e interpretadas. Este campo é essencial para ajustar a periodicidade da coleta de dados conforme as necessidades específicas do monitoramento.

Outro elemento importante da interface é o botão para selecionar um vídeo no computador. Como esta versão do sistema é um protótipo de teste e não para implementação em uma UTI, é crucial que os usuários possam facilmente carregar vídeos gravados para análise. Para facilitar a compreensão dos elementos da interface gráfica, a figura 29 ilustra os principais campos e controles disponíveis e sua estrutura.

Figura 29 – Frame capturado do vídeo de monitoramento



Fonte: Autor.

Para alcançar esse objetivo, foi adotada uma estratégia de design centrada no usuário. Cada elemento da interface foi cuidadosamente planejado para garantir que

as operações fossem diretas e sem complicações. Os botões e campos de entrada foram posicionados de maneira lógica e visível, facilitando a navegação e a interação.

3.6.2 Funcionalidades e Interação do Usuário

Como apresentado anteriormente, a interface gráfica do sistema foi projetada com foco na usabilidade, utilizando a biblioteca Tkinter devido à sua simplicidade e eficiência no desenvolvimento de aplicações em Python. Essa escolha também se justificou pela ampla documentação e suporte fornecido por sua comunidade ativa, o que facilitou o processo de implementação e solução de problemas.

As funcionalidades do sistema foi desenvolvida para ser intuitiva e funcional, atendendo às necessidades de monitoramento de dados e processamento de vídeos. A estrutura contempla múltiplas funcionalidades, como:

- **Seleção de Vídeos**: Um seletor de arquivos permite ao usuário buscar vídeos armazenados no computador de forma prática e rápida.
- **Configuração de Parâmetros**: A frequência de amostragem define o intervalo de tempo entre cada captura de dados, permitindo que o usuário ajuste a periodicidade da coleta conforme as necessidades do monitoramento. Esses ajustes garantem que o sistema seja flexível e preciso, adaptando-se a diferentes cenários e requisitos de monitoramento.
- **Seleção de Regiões de Interesse**: Botões dedicados permitem ao usuário selecionar as regiões de interesse dos monitores multiparamétricos, como valores de batimento cardíaco, frequência de pulso e oxigenação, facilitando o monitoramento específico de cada parâmetro.

A organização visual da interface facilita o acesso a essas funcionalidades. Por exemplo, a área de exibição de vídeos permite a visualização em tempo real do processamento em andamento, enquanto a seção de gráficos exibe dados monitorados de maneira dinâmica, favorecendo análises imediatas e tomadas de decisão.

3.6.3 Funcionalidades de Alerta e Testes

O alerta é uma parte essencial para um sistema de monitoramento. Essas notificações são usadas para informar aos usuários que uma ação foi concluída com sucesso ou para informar aos usuários sobre algum erro ou problema ocorrido (AWARI, 2023). Um exemplo de alerta visual é o dos monitores multiparâmetro apresentado na figura 30, onde a equipe pode ser avisada através de informações na tela e alarmes com luzes piscando ou alarmes sonoros que são disparados quando algum sinal vital do paciente atinge níveis

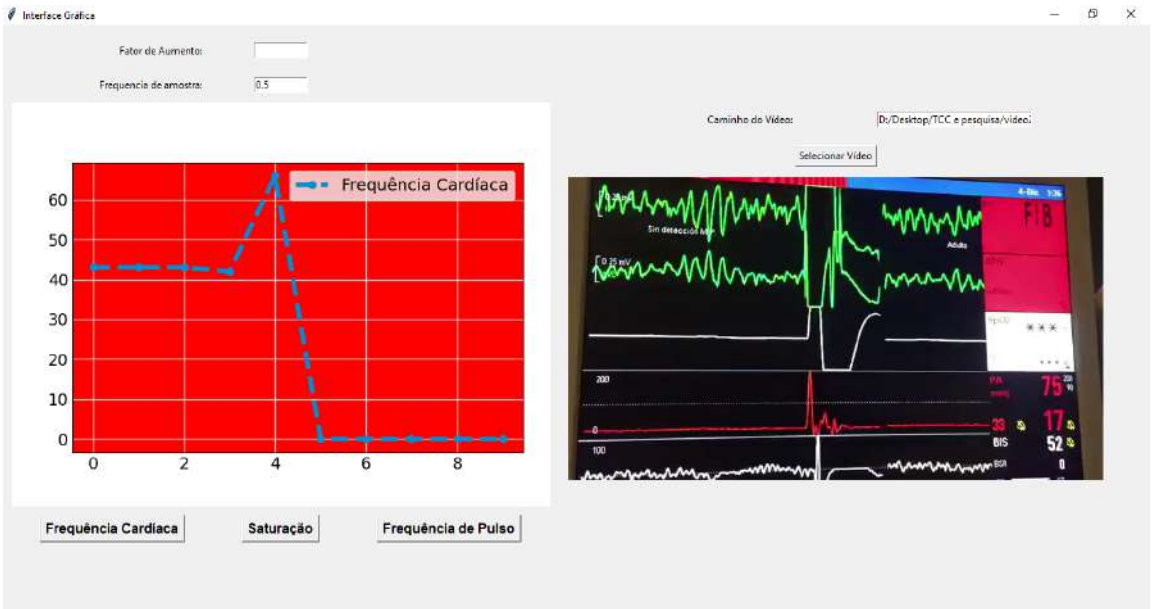
diferentes dos programados, que são considerados normais para o corpo humano (DRAKE, 2025).

Figura 30 – Alerta no monitor multiparâmetro.

Fonte: Autor.

Em relação a interface, cenários onde os valores monitorados ultrapassam limites predefinidos, a interface incorpora alertas visuais que mudam de cor automaticamente. Essa funcionalidade é essencial para chamar a atenção do usuário em situações críticas, permitindo respostas rápidas e eficientes. A Figura 31 ilustra a interface exibindo alertas em níveis críticos.

Figura 31 – Interface gráfica com alerta visual em nível crítico.



Fonte: Autor.

A compreensão dos níveis normais de saturação de oxigênio e batimento cardíaco é fundamental para o monitoramento e aplicação da lógica no alerta da interface. Onde, (DRAKE, 2025) destaca que:

- **Funcionamento do Aparelho**: O aparelho de medição de saturação funciona melhor quando não há esmalte nas unhas e a mão está em temperatura corporal normal.
- **Nível Ideal de Saturação**: A saturação de oxigênio ideal geralmente varia entre 95% e 100%. Se o nível de saturação estiver abaixo de 92%, é recomendado procurar auxílio médico imediatamente.
- **Valores Críticos**: Saturações abaixo de 90% podem indicar a presença de doenças que comprometem a eficiência das trocas gasosas entre pulmão e sangue, como asma, pneumonia, enfisema e insuficiência cardíaca.
- **Necessidade de Oxigênio Suplementar**: Pessoas com baixa saturação sanguínea podem necessitar de oxigênio suplementar para melhorar a oxigenação.

Os alertas são valores máximos e mínimos que o paciente pode apresentar. Para as grandezas específicas analisadas neste estudo, os valores de referência foram definidos de acordo com parâmetros médicos e estão detalhados na tabela 4.

Tabela 4 – Valores críticos dos principais sinais vitais segundo (SAúDE, 2023).

Parâmetro	**Mínimo Crítico**	**Máximo Crítico**
Frequência Cardíaca (FC)	< 40 bpm	> 150 bpm
Saturação de O_2 (SpO_2)	$< 90\%$	$> 100\%$ (hiperventilação)
Frequência de pulso	< 40 bpm	> 150 bpm

Esse processo iterativo de ajustes e refinamentos reflete a natureza dinâmica do desenvolvimento da interface, onde melhorias constantes são necessárias para aprimorar a experiência do usuário e buscar um produto mais intuitivo.

4 Resultados da Extração de Dados

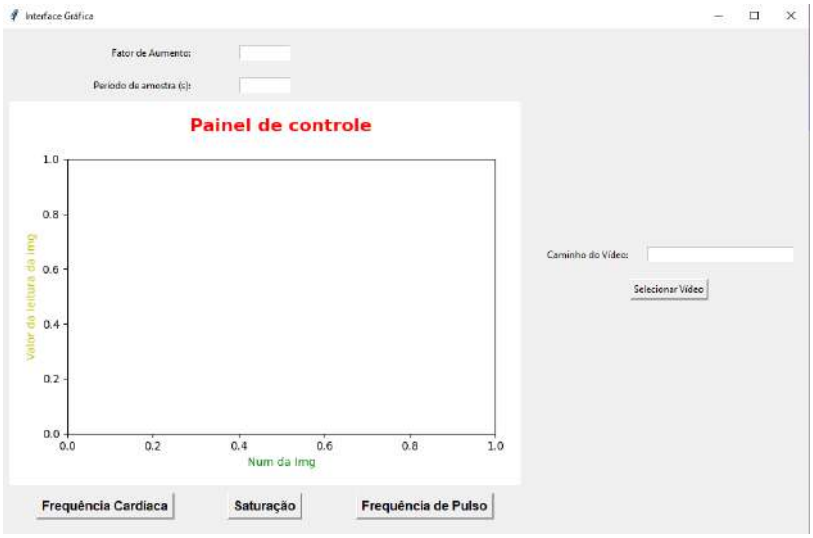
Este capítulo apresenta os resultados obtidos a partir do processo de extração de dados realizado com o sistema desenvolvido. O foco principal está na leitura e interpretação de três parâmetros fisiológicos essenciais: batimento cardíaco, saturação de oxigênio e frequência de pulso. Essas variáveis foram escolhidas por representarem indicadores críticos de estabilidade clínica e constituírem as principais regiões de interesse utilizadas tanto na etapa de desenvolvimento quanto na validação do sistema proposto.

A análise dos resultados busca avaliar a capacidade da ferramenta em realizar a extração confiável dessas informações a partir de vídeos reais de monitores multiparâmetros, com diferentes níveis de qualidade e características de gravação. O desempenho do sistema, portanto, está diretamente relacionado à sua robustez frente a diferentes condições visuais e como luminosidade.

Para garantir uma apresentação clara e organizada, o capítulo foi estruturado em quatro seções principais. Cada uma dessas seções corresponde à análise individual de um dos vídeos utilizados como base de testes. Em cada caso, será descrito o comportamento do sistema na extração dos dados ao longo do tempo, permitindo avaliar a consistência, estabilidade e precisão das leituras obtidas.

Todos os vídeos foram processados na interface interativa desenvolvida especialmente para este projeto. Essa interface foi desenhada para facilitar a seleção das ROIs e exibir as leituras de forma clara e intuitiva, proporcionando uma visualização dinâmica dos parâmetros extraídos. A Figura 32 ilustra essa interface, que foi essencial no processo de análise e validação dos dados extraídos.

Figura 32 – Interface para visualização das leituras

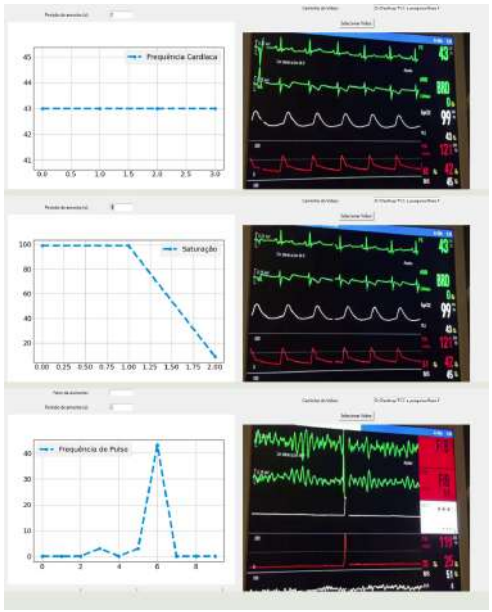


Fonte: Autor.

4.1 Análise do primeiro vídeo

O primeiro teste foi realizado no vídeo: Primeiro Vídeo, no qual são seleciondas três regiões de interesse para cada um dos parâmetros analisados. As ROIs foram definidas da seguinte forma: Frequência Cardíaca (530, 25, 47, 50), Saturação de Oxigênio (525, 155, 60, 50) e Frequência de Pulso (525, 195, 64, 42). A figura 33 ilustra a representação gráfica nos primeiros segundos de vídeo.

Figura 33 – Interface para visualização das leituras do Pimeiro Vídeo



Fonte: Autor.

A Tabela 5 apresenta os resultados das leituras corretas para o primeiro vídeo, que tem duração de 34 segundos.

Tabela 5 – Eficiência do Tesseract na leitura de valores - Vídeo 1

Região de Interesse	**Total de Leituras**	**Leituras Corretas**	**Taxa de Acerto (%)**
Batimento Cardíaco	35	31	88,57
Saturação de O_2	35	33	94,28
Frequência de Pulso	35	3	2.86

A baixa taxa de acerto na frequência de pulso pode estar diretamente associada a fatores como a presença de alertas visuais sobrepostos, o ângulo de captura da imagem e a distância entre a câmera e o monitor. Esses elementos podem ter comprometido a legibilidade dos caracteres, dificultando a extração precisa dos valores.

Por outro lado, as demais regiões de interesse apresentaram uma qualidade de imagem mais favorável, o que resultou em um desempenho significativamente melhor do Tesseract na leitura dos dados. Para que o reconhecimento óptico de caracteres ocorra de forma eficiente, a região analisada deve atender a determinados padrões de qualidade, como nitidez, contraste e clareza na exibição dos números.

Conforme ilustrado na Figura 34, as imagens relacionadas à frequência de pulso não atenderam plenamente a esses critérios, apresentando baixa qualidade e impactando negativamente o processo de extração dos dados.

Figura 34 – Frames da ROI Frequência de Pulso do vídeo 1

Fonte: Autor.

Especificamente, observou-se que as imagens para essa região de interese apresentam baixa nitidez, dificultando a distinção clara dos caracteres; baixo contraste entre os números e o fundo, o que prejudica o OCR; além de interferências visuais, como os alertas.

4.2 Análise do Segundo vídeo

O segundo vídeo analisado neste trabalho pode ser acessado por meio do seguinte link: Segundo vídeo. Seguindo a mesma metodologia aplicada ao primeiro vídeo, foram definidas três Regiões de Interesse para a extração dos dados numéricos exibidos no monitor multiparâmetro que é apresentado na figura 35.

Figura 35 – Monitor multiparâmetro - Vídeo 2

Fonte: Autor.

Apesar da semelhança metodológica do primeiro, este vídeo apresenta algumas particularidades que influenciaram diretamente na mudança dos valores das ROIs e do reconhecimento óptico de caracteres. Fatores como a posição da câmera em relação à tela e a qualidade da gravação (resolução, iluminação e foco) exigiram ajustes nos vetores das ROIs e, consequentemente, impactaram a taxa de acerto da extração dos dados. As regiões escolhidas foram: Frequência Cardíaca (850, 184, 120, 55), Saturação de Oxigênio (838, 314, 140, 90) e Frequência de Pulso (980, 184, 105, 45).

O vídeo possui uma duração total de 1 minuto e 11 segundos. A análise foi realizada em intervalos de 2 segundos, totalizando 36 quadros distintos para cada uma das três variáveis monitoradas. Os resultados dos valores extraídos estão sintetizados na Tabela 6, onde são apresentados o total de leituras realizadas, o número de leituras corretas e a respectiva taxa de acerto percentual.

Tabela 6 – Eficiência do Tesseract na leitura de valores - Vídeo 2

Região de Interesse	**Total de Leituras**	**Leituras Corretas**	**Taxa de Acerto (%)**
Batimento Cardíaco	36	30	83.33
Saturação de O_2	36	34	94.44
Frequência de Pulso	36	29	80.56

A análise dos resultados revela um desempenho satisfatório do sistema na extração dos dados do segundo vídeo. A Saturação de Oxigênio apresentou a maior taxa de acerto (94,44%), indicando que a região correspondente possuía características visuais favoráveis, como boa nitidez, contraste adequado e ausência de interferências visuais. A Frequência Cardíaca também obteve um desempenho consistente, com 83,33% de acerto, o que demonstra que, apesar de pequenas variações na qualidade da imagem, os dígitos permaneceram legíveis na maior parte dos quadros.

Por outro lado, a Frequência de Pulso, embora tenha apresentado uma taxa de acerto inferior (80,56%), ainda assim superou significativamente o desempenho observado no primeiro vídeo. Isso sugere que a ausência dos alertas e a melhoria na qualidade da gravação contribuíram para uma extração melhor. No entanto, pequenas oscilações de foco e reflexos pontuais ainda podem ter impactado negativamente algumas leituras.

4.3 Análise do terceiro vídeo

O terceiro vídeo analisado neste estudo foi: Terceiro Vídeo, que tem duração de 1 minuto e 42 segundos. As novas delimitações são: Frequência Cardíaca (735, 180, 120, 70), Saturação de Oxigênio (749, 319, 180, 78) e Frequência de Pulso (881, 180, 100, 55).

Foram analisados quadros a cada 2 segundos ao longo de todo o vídeo, totalizando 52 leituras para cada variável observada. Após a extração dos dados, cada leitura foi

comparada manualmente com os valores exibidos no vídeo, garantindo a verificação da acurácia das detecções. Os resultados consolidados estão apresentados na Tabela 7.

Tabela 7 – Eficiência do Tesseract na leitura de valores - Vídeo 3

Região de Interesse	**Total de Leituras**	**Leituras Corretas**	**Taxa de Acerto (%)**
Batimento Cardíaco	52	36	69.23
Saturação de O_2	52	49	94.23
Frequência de Pulso	52	50	96.15

Os resultados indicam uma redução na taxa de acerto da leitura da variável Batimento Cardíaco em relação aos vídeos anteriores. Essa queda está associada principalmente à instabilidade da câmera e à presença de uma luz direta sobre a região de leitura em certos momentos. A Figura 36 mostra essa diferença, comparando a visualização normal com a interferência luminosa.

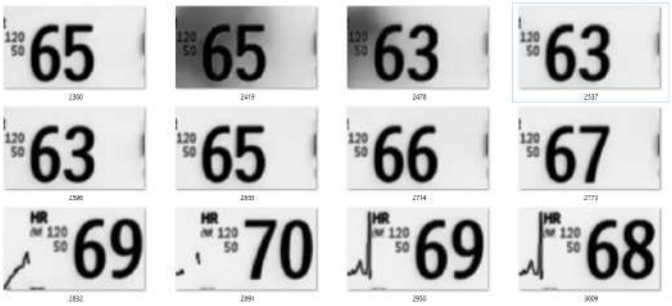
Figura 36 – Condição normal e com ruído

Fonte: Autor.

Durante a análise dos quadros extraídos, observou-se que essas interferências provocaram flutuações na nitidez da região-alvo. Além disso, a iluminação excessiva comprometeu o contraste entre os caracteres e o fundo da tela, dificultando o reconhecimento automático. Esses obstáculos explicam a maior quantidade de erros registrados na leitura da frequência cardíaca.

A Figura 37 ilustra essas limitações, destacando os impactos da iluminação irregular e das oscilações da câmera na qualidade da extração de dados. É possível observar como esses fatores introduzem ruídos visuais que prejudicam o reconhecimento óptico de caracteres, tornando a leitura imprecisa ou até inviável em determinados momentos. Esse problema ressalta a necessidade de ajustes no posicionamento da câmera e no controle da iluminação para minimizar interferências e melhorar a acurácia do sistema.

Figura 37 – Frames com baixa qualidade



Fonte: Autor.

4.4 Análise do Quarto vídeo

O quarto vídeo utilizado neste trabalho está disponível em: Quarto Vídeo, e possui uma duração total de 4 minutos e 55 segundos. No entanto, os testes foram conduzidos apenas nos primeiros 34 segundos.

Apesar do monitor utilizado ser do mesmo modelo apresentado no vídeo anterior, o cenário de gravação apresenta diferenças significativas em termos de iluminação e posicionamento da câmera. Esses fatores influenciam diretamente na qualidade da captura e, consequentemente, na precisão da extração dos dados. Neste caso específico, as condições ambientais se mostraram favoráveis, resultando em um desempenho excepcional do sistema de leitura.

Com as variações no enquadramento e nas condições de luz, foi necessário redefinir os vetores que delimitam as regiões de interesse. As novas coordenadas utilizadas foram: Frequência Cardíaca (751, 180, 115, 70), Saturação de Oxigênio (749, 319, 180, 78) e Frequência de Pulso (881, 180, 100, 55). Essas áreas foram ajustadas manualmente com base em testes preliminares, visando maximizar a acurácia da detecção.

Durante a análise, foram extraídas 34 amostras para cada uma das três variáveis monitoradas, respeitando um intervalo de 1 segundo entre os quadros selecionados. O desempenho do OCR Tesseract foi avaliado com base na taxa de acertos em relação ao valor real exibido no vídeo. Os resultados obtidos estão organizados na Tabela 8.

Tabela 8 – Eficiência do Tesseract na leitura de valores - Vídeo 4

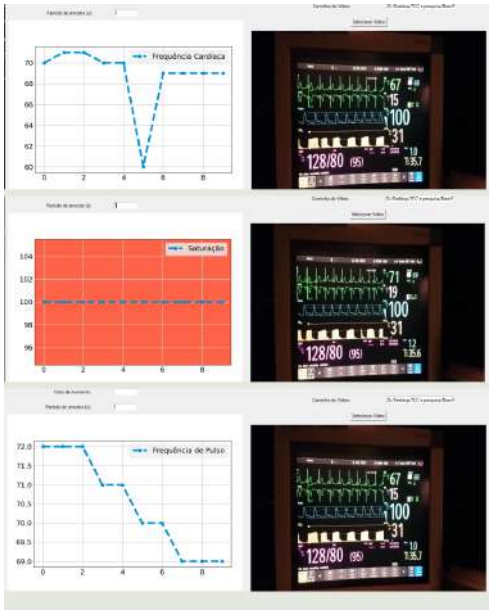
Região de Interesse	**Total de Leituras**	**Leituras Corretas**	**Taxa de Acerto (%)**
Batimento Cardíaco	34	34	100
Saturação de O_2	34	34	100
Frequência de Pulso	34	34	100

Os resultados demonstram uma leitura ideal por parte do sistema, com 100% de acerto em todas as regiões avaliadas. Esse desempenho pode ser atribuído à combinação de fatores como iluminação uniforme, ausência de interferência visual e estabilidade da

gravação. Esses elementos contribuíram para que os caracteres exibidos nos monitores fossem reconhecidos com clareza pelo mecanismo de OCR.

Além da avaliação numérica, é importante destacar como os dados são representados na interface gráfica desenvolvida durante o projeto. A Figura 38 apresenta a forma como os valores são exibidos em tempo real, evidenciando a organização visual e a precisão das leituras automáticas. A representação gráfica permite o acompanhamento simultâneo das variáveis, reforçando a aplicabilidade do sistema em contextos clínicos e laboratoriais.

Figura 38 – Resultados exibidos na interface gráfica para o Vídeo 4



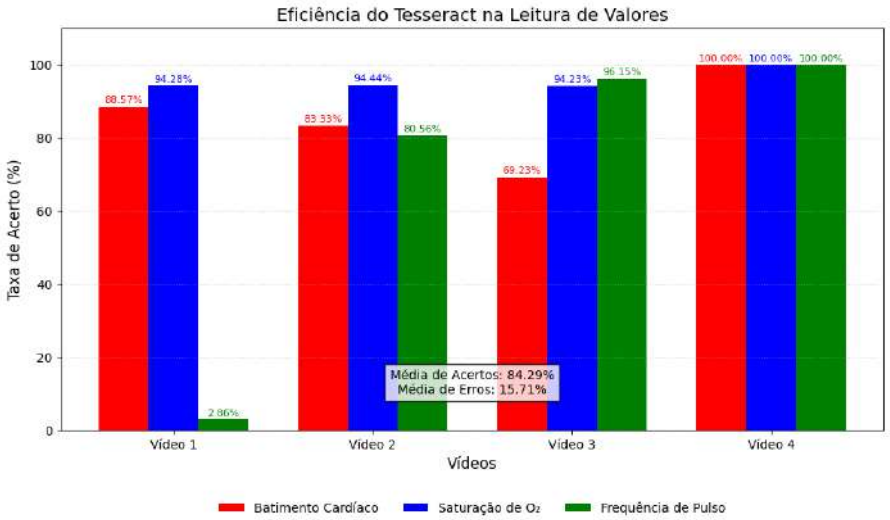
Fonte: Autor.

A consistência dos resultados neste vídeo destaca a importância de um ambiente controlado para capturas mais eficazes. Fatores como boa iluminação, estabilidade da câmera e ausência de ruídos visuais são essenciais para a confiabilidade do processo de extração de dados por meio de visão computacional.

4.5 Visão Geral dos Resultados

A análise geral dos resultados obtidos nos quatro vídeos evidencia variações significativas na eficiência do sistema de OCR, refletindo a influência direta das condições de captura sobre o desempenho da extração automática de dados. Conforme ilustrado na Figura 39, as taxas de acerto não seguem um padrão uniforme entre os diferentes vídeos e variáveis, o que indica que fatores externos, como qualidade da imagem, estabilidade da câmera, iluminação e presença de ruídos visuais, desempenham um papel crucial na precisão do reconhecimento óptico. Essas variações também revelam a sensibilidade às características específicas de cada gravação, como o posicionamento e estabilidade da câmera em relação ao monitor, a resolução do vídeo e a nitidez dos dígitos exibidos.

Figura 39 – Resultados gerais obtidos no trabalho



Fonte: Autor.

Observa-se que a região de Saturação de O_2 obteve desempenho consistente, com taxas de acerto elevadas (entre 94% e 100%), demonstrando que a qualidade da imagem nessa área foi adequada para uma extração precisa dos valores. Em contraste, a região de Batimento Cardíaco apresentou resultados mais variados (entre 69,23% e 100%), sugerindo que fatores como o ângulo de captura e a presença de interferências influenciaram a leitura dos dados.

Notavelmente, a região de Frequência de Pulso apresentou uma sensibilidade maior a ruídos e interferências. Enquanto o primeiro vídeo registrou uma taxa de acerto extremamente baixa (2,86%), os ajustes efetuados nos vídeos subsequentes proporcionaram melhorias significativas, alcançando índices entre 80,56% e 100%.

Com uma média geral de acertos de aproximadamente 84,29% (e, consequentemente, uma média de erros de 15,71%), os resultados demonstram um desempenho satisfatório do sistema, embora ressaltem a necessidade de otimizações, principalmente em condições adversas de captura de dados. Este panorama reforça a importância de se manter um ambiente controlado e de realizar ajustes finos na delimitação das regiões de interesse para aumentar a confiabilidade dos dados extraídos.

Embora as taxas de acerto variem entre os diferentes tipos de medida, é importante destacar que os erros observados estão mais fortemente associados à qualidade dos vídeos do que à natureza da variável monitorada. Fatores como resolução da gravação, iluminação, foco, ângulo da câmera e presença de elementos visuais sobrepostos influenciaram diretamente a legibilidade dos caracteres e, consequentemente, a eficácia do OCR.

5 CONSIDERAÇÕES FINAIS

Este trabalho teve como objetivo o desenvolvimento de uma interface gráfica para a extração automática de dados numéricos exibidos em monitores hospitalares, utilizando técnicas de Reconhecimento Óptico de Caracteres. A proposta visa automatizar um processo que, tradicionalmente, depende da observação manual por parte dos profissionais de saúde, oferecendo uma solução que alia precisão, agilidade e acessibilidade. Para isso, foi implementado um conjunto de processamento que integra múltiplas etapas: pré-processamento de imagem com a biblioteca OpenCV, responsável por realçar características visuais relevantes e reduzir ruídos; segmentação de Regiões de Interesse, que delimita as áreas específicas da tela onde os dados são exibidos; e reconhecimento óptico de caracteres com o Pytesseract OCR, que converte as informações visuais em dados estruturados.

A validação do sistema foi realizada sobre quatro vídeos de referência, nos quais foram monitorados a frequência cardíaca, a frequência de pulso e a saturação de oxigênio. A análise dos resultados indicou taxas de acerto que variaram conforme as condições de captura: em cenários controlados e com boa iluminação, observou-se precisão próxima de 100 %; em ambientes com ruídos visuais ou instabilidade da câmera, as taxas mantiveram-se acima de 69 %, evidenciando robustez do algoritmo mesmo em situações adversas. A média geral de acertos—aproximadamente 84,3 %—revela o potencial do trabalho.

Os procedimentos de ajuste de parâmetros do OCR, aliados ao refinamento das ROIs, mostraram-se determinantes para mitigar efeitos de baixa resolução e reflexos. Além disso, a interface gráfica desenvolvida viabiliza a exibição em tempo real das leituras, com alerta automático para valores críticos, oferecendo suporte imediato a profissionais de saúde.

5.0.1 Perspectivas Futuras

Como evolução natural, projeta-se a implementação de um módulo de captura ao vivo, no qual uma câmera de boa resolução seria fixada em posição estável, diretamente em frente ao monitor hospitalar. Nesse cenário, cada frame recebido seria submetido ao mesmo fluxo de processamento:

- captura e estabilização da imagem;
- aplicação de pré-processamento;
- segmentação dinâmica das ROIs com calibração automática conforme o ângulo de visão;
- extração e validação dos valores via OCR;

- transmissão instantânea dos dados, por rede local ou internet, para dispositivos móveis e sistemas de registro eletrônico de saúde.

Essa arquitetura em tempo real permitirá o monitoramento contínuo, a emissão de alertas preditivos com base em tendências, e a integração direta com prontuários eletrônicos, reforçando a segurança do paciente e a agilidade na tomada de decisão.

O sistema desenvolvido aponta para uma viabilidade técnica promissora em ambientes clínicos, ao propor a automatização da leitura de monitores hospitalares. A adoção de câmeras fixas e processos de calibração automática pode tornar a solução mais precisa e escalável, favorecendo respostas mais rápidas em unidades de terapia intensiva. Dessa forma, este trabalho representa um passo inicial rumo a inovações no monitoramento hospitalar, contribuindo para um atendimento mais acessível, ágil e confiável.

Além das melhorias já implementadas, destaca-se a possibilidade de incorporar técnicas adicionais de pré-processamento de imagem, como correção de perspectiva, equalização de histograma, filtragem adaptativa e transformações geométricas, com o objetivo de compensar distorções causadas por ângulos desfavoráveis de captura ou variações de iluminação. Essas abordagens podem contribuir significativamente para o aumento da taxa de acerto do OCR, especialmente em ambientes não controlados. No entanto, a escolha adequada dessas técnicas depende do contexto específico de cada aplicação, sendo necessária uma investigação mais aprofundada para determinar quais métodos são mais eficazes em diferentes cenários clínicos.

Referências

AGUIAR, L. M. M. et al. Perfil de unidades de terapia intensiva adulto no brasil: revisão sistemática de estudos observacionais. *CNN Brasil*, 2021. Disponível em: <https://www.scielo.br/j/rbti/a/sDnLGny8cZgQtVVfX5q3X7G/?format=pdf>. Citado na página 40.

ALMEIDA, A.; BORGES, C.; PAULA, I. Aplicação do descritor hog e classificador svm no reconhecimento de poses humanas em imagens de profundidade. In: *ESCOLA REGIONAL DE INFORMÁTICA DO PIAUÍ (ERI-PI), 4.* Teresina: Sociedade Brasileira de Computação, 2018. p. 119–124. Citado na página 26.

AMSTEL, F. van. Afinal, o que é design de interação? *IFD*, 2010. Acessado em: 17 de Abril de 2025. Disponível em: <https://www.ifd.com.br/design/afinal-o-que-e-design-de-interacao/>. Citado na página 52.

AWARI. Design de notificações em ui design: Comunicando informações e alertas de forma visualmente atraente. *Awari*, Awari, Julho 2023. Acessado em: 29 de Abril de 2025. Disponível em: <https://awari.com.br/design-de-notificacoes-em-ui-design-comunicando-informacoes-e-alertas-de-forma-visualmente-atraent>. Citado 2 vezes nas páginas 48 e 54.

BARBIERIA, R. R. et al. Reimagining leprosy elimination with ai analysis of a combination of skin lesion images with demographic and clinical data. *thelancet Regional Health*, May 2022. Disponível em: <https://doi.org/10.1016/j.lana.2022.100192>. Citado na página 15.

BARELLI, F. *Introdução à Visão Computacional: Uma abordagem prática com Python e OpenCV*. Casa do Código, 2018. ISBN 9788594188588. Disponível em: <https://books.google.com.br/books?id=CA5ZDwAAQBAJ>. Citado 2 vezes nas páginas 23 e 24.

BAZON, L. S. TraduÇÃo de textos em imagens de histÓrias em quadrinhos utilizando visÃo computacional e ocr. Cachoeiro de Itapemirim, 2022. Disponível em: <https://repositorio.ifes.edu.br/bitstream/handle/123456789/2874/ARTIGO_TRADUCAO_DE_TEXTOS_EM_IMAGENS_DE_HISTORIAS_EM_QUADRINHOS_UTILIZANDO_VISAO_COMPUTACIONAL_E_OCR.pdf?sequence=1&isAllowed=y>. Citado 2 vezes nas páginas 28 e 34.

BEGG, R.; LAI, D. T.; PALANISWAMI, M. *Computational Intelligence in Biomedical Engineering*. Hardcover. [S.l.]: Wiley, 2010. Citado 3 vezes nas páginas 35, 36 e 37.

BRAGA, L. F. Z. VeÍculos aÉreos nÃo tripulados com visÃo computacional na agricultura: AplicaÇÕes, desafios e perspectivas. 2013. Disponível em: <https://bdta.abcd.usp.br/directbitstream/46b694a0-715b-462d-a8b7-546ea4ef259d/Braga_Luiz_Filipe_Zenicola.pdf>. Citado na página 23.

BRASIL, M. d. S. *Manual da Tecnovigilância: abordagens de vigilância sanitária de produtos para a saúde comercializados no Brasil*. Brasília, DF: Ministério da Saúde, 2025. Citado na página 35.

BRECKON, T.; SOLOMON, C. *Fundamentos de Processamento Digital de Imagens - Uma Abordagem Prática com Exemplos em Matlab*. 1. ed. [S.l.]: LTC, 2013. Citado 2 vezes nas páginas 19 e 24.

BREUEL, T. M. et al. High-performance ocr for printed english and fraktur using lstm networks. In: *Proceedings of the 12th International Conference on Document Analysis and Recognition (ICDAR)*. [S.l.]: IEEE, 2013. Citado 2 vezes nas páginas 27 e 28.

CARVALHO, S. T. et al. Monitoramento remoto de pacientes em ambiente domiciliar. In: INSTITUTO DE COMPUTAçãO – UNIVERSIDADE FEDERAL FLUMINENSE (UFF), INSTITUTO DE INFORMáTICA – UNIVERSIDADE FEDERAL DE GOIáS (UFG), UNIVERSIDADE DO ESTADO DO RIO DE JANEIRO (UERJ). *XXVIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. 2010. Disponível em: <http://sbrc2010.inf.ufrgs.br/anais/data/pdf/salao/st01_03_salao.pdf>. Citado na página 38.

CHADHA, A. et al. License plate recognition system using opencv & pytesseract. *CSI Journal of Computing*, v. 3, n. 3, July-September 2020. Citado na página 32.

CONSTâNCIO, A. S.; CARVALHO, D. F. Aplicações de visão computacional na saúde: revisão de literatura incrementada com técnicas de processamento de linguagem natural. *Research, Society and Development*, v. 11, n. 10, 2022. Citado na página 33.

CONTAIFER, J. Distribuição de utis ainda é muito desigual no brasil, mostra estudo. *AMIB*, Novembro 2024. Acesso em: 19 nov. 2024. Disponível em: <https://www.amib.com.br/noticias/distribuicao-de-utis-ainda-e-muito-desigual-no-brasil-mostra-estudo>. Citado 2 vezes nas páginas 40 e 41.

COSTA, V. L. et al. Pré-processamento de imagens de baixa resolução utilizando deep learning baseado em um autoencoder. In: *XL Simpósio Brasileiro de Telecomunicações e Processamento de Sinais - SBrT 2022*. Sta. Rita do Sapucaí, MG: [s.n.], 2022. Citado na página 24.

DRAKE, C. *Monitor Multiparamétrico: 5 Fatores que Você Deve Analisar Antes de Comprar um*. 2025. Acessado em: 29 de Abril de 2025. Disponível em: <https://cmosdrake.com.br/blog/monitor-multiparametrico-5-fatores-que-voce-deve-analisar-antes-de-comprar-um/#:~:text=O%20monitor%20possui%20alarmes%20sonoros%20que%20s%C3%A3o%20disparados,informa%C3%A7%C3%B5es%20na%20tela%20e%20alarmes%20com%20luzes%20piscando>. Citado na página 55.

FARIA, J. T. et al. Procedimento operacional padrão de operação do monitor multiparamétrico dixtal 2010. *Anais do V Congresso Brasileiro de Eletromiografia e Cinesiologia e X Simpósio de Engenharia Biomédica*, Universidade Federal de Uberlândia, Uberlândia, Brasil, p. 727, 2018. Citado 2 vezes nas páginas 35 e 36.

FELICIANO, F. F.; SOUZA, I. L. de; LETA, F. R. VisÃo computacional aplicada À metrologia dimensional automatizada: ConsideraÇÕes sobre sua exatidÃo. *ENGEVISTA*, v. 7, n. 2, p. 38–50, dezembro 2005. Citado na página 26.

FILHO, O. M.; NETO, H. V. *Processamento Digital de Imagens*. 1. ed. [S.l.]: Editora XYZ, 2001. Citado 6 vezes nas páginas 11, 18, 19, 20, 21 e 25.

FORSYTH, D.; PONCE, J. *Computer Vision: A Modern Approach*. 2. ed. Upper Saddle River, NJ: Prentice Hall, 2011. Citado na página 23.

GONZALEZ, R. C.; WOODS, R. E. *Digital Image Processing*. 4. ed. Upper Saddle River, NJ: Pearson, 2018. Citado 4 vezes nas páginas 18, 23, 24 e 25.

IBM. *O que é uma rede neural recorrente (RNN)?* 2024. Acessado em: 11 de Abril de 2025. Disponível em: <https://www.ibm.com/br-pt/think/topics/recurrent-neural-networks>. Citado na página 30.

IMPORTANCE of Graphical Representation of Data. *QS Study*, 2025. Acessado em: 21 de Abril de 2025. Disponível em: <https://qsstudy.com/importance-of-graphical-representation-of-data/>. Citado na página 51.

ISLAM, M. T. Graphical representation of data. *Statistical Aid*, 2025. Acessado em: 18 de Abril de 2025. Disponível em: <https://www.statisticalaid.com/graphical-representation-of-data/>. Citado na página 51.

JUNIOR, A. R. B. *Aplicação de Visão Computacional na Identificação de Tubos em Radiografias de Tórax*. Projeto Final de Graduação — PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO, Rio de Janeiro, Julho 2024. Citado na página 33.

KALRA, K. Ocr na área da saúde - automatize processos usando ocr no setor médico. *Nanonets Blog*, Março 2023. Disponível em: <https://nanonets.com/blog/ocr-for-healthcare/>. Citado 2 vezes nas páginas 34 e 38.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: *Advances in Neural Information Processing Systems*. [S.l.: s.n.], 2012. v. 25. Citado na página 26.

LEITE, C. R. M. *Arquitetura Inteligente Fuzzy para Monitoramento de Sinais Vitais de Pacientes: Um Estudo de Caso em UTI*. Dissertação (Mestrado) — Universidade Federal do Rio Grande do Norte, Natal, RN, 2011. Citado na página 39.

LK, S. et al. Detecção avançada de cyberbullying: integrando pytesseract, demoji e bert para análise abrangente de conteúdo textual e visual. In: *4ª Conferência Internacional sobre Análise de Sentimentos e Aprendizado Profundo (ICSADL)*. Bhimdatta, Nepal: IEEE, 2025. p. 18–20. Adicionado ao IEEE Xplore em 27 de março de 2025. Citado na página 33.

MARENGONI, M.; STRINGHINI, D. Introdução à visão computacional usando opencv. *UFRGS*, 2010. Citado na página 22.

MARTíN, J. M.; PODRžAJ, P.; POžRL, T. Implementação básica de um algoritmo de reconhecimento de sinais de trânsito de limite de velocidade via pytesseract. In: *Conferência Internacional sobre Tendências Emergentes em Redes e Comunicações de Computadores (ETNCC)*. Windhoek, Namíbia: IEEE, 2024. p. 23–25. Adicionado ao IEEE Xplore em 04 de dezembro de 2024. Citado na página 33.

MATIAS, G. C. et al. Uma solução iot de segurança utilizando visão computacional para potencializar a proteção escolar. *Instituto Federal de Educação Ciência e Tecnologia do Rio Grande do Norte (IFRN)*, 2024. Citado na página 23.

MITTAL, R. Text extraction using ocr: A systematic review. In: DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING, AMITY SCHOOL FOR ENGINEERING AND TECHNOLOGY, AMITY UNIVERSITY UTTAR PRADESH, NOIDA (UP), INDIA. *IEEE Xplore*. [S.l.], 2020. ISBN 978-1-7281-5374-2. Citado na página 27.

MURACA, F. das C. F. L. Os avanços e desafios da saúde em 2025. *Migalhas de Peso*, Janeiro 2025. Em 2025, a tecnologia transforma a saúde com inovações como IA, telemedicina e medicina personalizada. No entanto, desafios como resistência e desigualdade no acesso persistem. Disponível em: <https://www.migalhas.com.br/pilulas-de-informacao/os-avancos-e-desafios-da-saude-em-2025>. Citado na página 41.

NAGY, G. Twenty years of document image analysis in pami. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, IEEE, v. 22, n. 1, p. 38–62, 2000. Citado na página 28.

NASCIMENTO, E. R. P. do; TRENTINI, M. O cuidado de enfermagem na unidade de terapia intensiva (uti): teoria humanística de paterson e zderad. *Revista Brasileira de Enfermagem*, v. 53, n. 1, 2000. Acessado em: 17 de novembro de 2024. Disponível em: <https://www.scielo.br/j/reben/a/example>. Citado na página 38.

O que é Interface Amigável? *Aprendendo Sobre*, 2024. Acessado em: 21 de Abril de 2025. Disponível em: <https://aprendendosobre.com.br/glossario/o-que-e-interface-amigavel/>. Citado na página 52.

OCR na Saúde: O que é? Casos de uso, benefícios, desvantagens. *Shaip Blog*, Maio 2023. Disponível em: <https://pt.shaip.com/blog/ocr-in-healthcare/>. Citado na página 34.

PATEL, C.; PATEL, A.; PATEL, D. Optical character recognition by open source ocr tool tesseract: A case study. *International Journal of Computer Applications*, v. 55, n. 10, p. 50–56, October 2012. Citado 3 vezes nas páginas 29, 32 e 33.

PERSECHINO, A.; ALBUQUERQUE, M. P. de. Processamento digital de imagens: conceitos fundamentais. *Centro Brasileiro de Pesquisas Físicas - Coordenação de Atividades Técnicas*, Rua Dr. Xavier Sigaud 150, Rio de Janeiro, RJ. CEP: 22290-180. Brasil, Outubro 2015. Citado na página 20.

PITTA, d. C. I. *Maioria dos brasileiros vive em áreas abaixo da meta da OMS de UTIs públicas*. 2020. Acessado em: 11 de Abril de 2025. Disponível em: <https://www.cnnbrasil.com.br/saude/maioria-dos-brasileiros-vive-em-areas-abaixo-da-meta-da-oms-de-utis-publicas/>. Citado 3 vezes nas páginas 11, 39 e 40.

PITTADA, I. Maioria dos brasileiros vive em áreas abaixo da meta da oms de utis públicas. *CNN Brasil*, Junho 2021. Disponível em: <https://classic.exame.com/bussola/contagio-elevado-utis-lotadas-e-jovens-em-risco-a-nova-face-da-pandemia/?utm_source=copiaecola&utm_medium=compartilhamento>. Citado na página 41.

PLAMONDON, R.; SRIHARI, S. N. On-line and off-line handwriting recognition: A comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2000. Citado na página 27.

PROLIFE. *Quais são os parâmetros básicos de um monitor multiparamétrico?* 2024. Acessado em: 20 nov. 2024. Disponível em: <https://prolife.com.br/quais-sao-os-parametros-basicos-de-um-monitor-multiparametro/>. Citado 2 vezes nas páginas 35 e 36.

QUEIROZ, J. E. R. de; GOMES, H. M. Introdução ao processamento digital de imagens. *RITA*, VIII, n. 1, 2001. Citado 3 vezes nas páginas 20, 21 e 22.

RANKINGS, S. I. Simplificando a obtenção e a utilização de imagens digitais: scanners e câmeras digitais. Pontifícia Universidade Católica de São Paulo, 2004. Citado na página 21.

ROSEBROCK, A. Tesseract page segmentation modes (psms) explained: How to improve your ocr accuracy. *PyImageSearch Blog*, 2021. Acessado em: 21 de Abril de 2025. Disponível em: <https://pyimagesearch.com/2021/11/15/tesseract-page-segmentation-modes-psms-explained-how-to-improve-your-ocr-accuracy>. Citado 2 vezes nas páginas 50 e 51.

SAúDE, T. *Frequência Cardíaca*. 2023. Acessado em: 29 de Abril de 2025. Disponível em: <https://www.tuasaude.com/frequencia-cardiaca/>. Citado 2 vezes nas páginas 11 e 56.

SCURI, A. E. *Fundamentos da Imagem Digital*. Tecgraf/PUC-Rio, 1999. Acessado em: 05 de Janeiro de 2025. Disponível em: <https://www2.tecgraf.puc-rio.br/ftp_pub/curso-cgi/Notas_de_Aula/scuri.pdf>. Citado na página 22.

SECRETáRIO, J. H. A.; PIRES, R. Uso de visÃo computacional para contagem automÁtica de cÉlulas em imagens obtidas por microscÓpios. *REGRASP*, v. 3, n. 3, p. 4–17, junho 2018. ISSN 2526-1045. Citado na página 27.

SILVA, A. Reconhecimento ótico de caracteres: como funciona? *Pixforce*, 2025. Disponível em: <https://pixforce.ai/pt-br/reconhecimento-otico-de-caracteres/>. Citado na página 28.

SILVA, G. G. da et al. VeÍculos aÉreos nÃo tripulados com visÃo computacional na agricultura: AplicaÇÕes, desafios e perspectivas. *Universidade Católica Dom Bosco - UCDB*, 2015. Disponível em: <https://anaisonline.uems.br/index.php/ecaeco/article/view/2832>. Citado na página 23.

SMITH, R. W. History of the tesseract ocr engine: What worked and what didn't. In: *Proceedings of SPIE 8658, Document Recognition and Retrieval XX*. Mountain View, CA: SPIE, 2013. v. 8658, p. 865802. Disponível em: <https://doi.org/10.1117/12.2010051>. Citado 2 vezes nas páginas 29 e 30.

SONKA, M.; HLAVAC, V.; BOYLE, R. *Image Processing, Analysis, and Machine Vision*. 4. ed. Boston, MA: Cengage Learning, 2014. Citado 2 vezes nas páginas 25 e 26.

SZELISKI, R. *Computer Vision: Algorithms and Applications*. 2. ed. [S.l.]: Springer, 2021. Citado 3 vezes nas páginas 15, 22 e 23.

TIMALSINA, A. How does tesseract for ocr work? *Docsumo Blog*, 2025. Acessado em: 14 de junho de 2025. Disponível em: <https://www.docsumo.com/blog/tesseract-ocr>. Citado na página 31.

TOKARSKI, M. Contágio elevado, utis lotadas e jovens em risco: a nova face da pandemia. *Exame*, Junho 2021. Disponível em: <https://classic.exame.com/bussola/contagio-elevado-utis-lotadas-e-jovens-em-risco-a-nova-face-da-pandemia/?utm_source=copiaecola&utm_medium=compartilhamento>. Citado na página 38.

TULCHAK, L. V.; RCHUK, History of python. , 2016. Citado na página 29.

UEZIMA, L. K. et al. Utilizando o tesseract ocr para reconhecimento de textos à mão. In: UNIVERSIDADE PRESBITERIANA MACKENZIE (UPM). *Faculdade de Computação e Informática*. São Paulo, SP, Brazil, 2024. Citado 2 vezes nas páginas 33 e 51.

USP, J. da. *Inteligência artificial ajuda em diagnósticos de doenças*. 2018. Acesso em: 09 de Agosto de 2024. Disponível em: <https://jornal.usp.br/atualidades/inteligencia-artificial-ajuda-em-diagnosticos-de-doencas/>. Citado na página 15.

VARGAS, A. C. G.; CARVALHO, A. M. P.; VASCONCELOS, C. N. Um estudo sobre redes neurais convolucionais e sua aplicação em detecção de pedestres. *Instituto de Computação, Universidade Federal Fluminense*, Niterói, Brasil, 2016. Citado na página 26.

VAZ, G. C. *Aplicação de Modelos de Deep Learning para Qualificação da Área da Engenharia Biomédica: Um Estudo de Caso de Visão Computacional em Imagens de Raio-X da Região Torácica*. Dissertação (Dissertação de Mestrado) — Universidade Estadual de Campinas, Campinas, 2023. Citado na página 33.

VIANA, D. R. A. P. P. O real cenários das utis brasileiras: Mudanças que nos preocupam. *Novidades SOTIBA*, Junho 2012. Disponível em: <https://novidadessotiba.com/o-real-cenarios-das-utis-brasileiras-mudancas-que-nos-preocupam>. Citado na página 37.

YUILL, S.; HALPIN, H. *Python*. 2006. Citado na página 29.

ZIZA, L. N.; XIMENES, T. S. S. Desenvolvimento de um sistema para o monitoramento de acidentes domésticos de idosos utilizando visão computacional. *Faculdade de Tecnologia da Unicamp*, 2022. Disponível em: <https://periodicos.unb.br/index.php/ripe/article/view/41819>. Citado na página 27.

listings xcolor

Apêndices

APÊNDICE A – Código do trabalho

Tabela 9 – Visão Geral das Principais Partes do Código

Módulo / Classe / Função	Descrição
`Imports & Configuração`	Importa bibliotecas (threading, OpenCV, tkinter, matplotlib, pytesseract, PIL) e define o caminho do executável do Tesseract.
`cortar_imagem()`	Recebe imagem e retângulo (x, y, w, h); extrai a ROI para o OCR.
`melhorar_imagem()`	Converte ROI para tons de cinza, aplica threshold binário e amplia a imagem para melhorar acurácia do OCR.
`reconhecer_inteiro()`	Executa o Tesseract em modo dígitos, limpa o texto e converte para inteiro (0 se falhar).
`ProcessadorDeVideo`	Classe iteradora que lê frames de `VideoCapture` a cada período, recorta, pré-processa e faz OCR em cada ROI, armazenando os resultados.
`FrameDeVideo`	`tk.Frame` com widgets para selecionar o vídeo e exibir o frame processado em tempo real.
`FrameDoGrafico`	`tk.Frame` que contém: entrada de período, botões para cada ROI e um gráfico Matplotlib embutido mostrando a série temporal.
`Processador`	Extensão de `ProcessadorDeVideo` que integra GUI: atualiza gráfico, exibe frame no `FrameDeVideo` e sinaliza alertas colorindo o fundo do plot.
`Controlador`	`threading.Thread` que executa o loop de captura de forma assíncrona, mantém a interface responsiva e encerra no fim do vídeo ou ao fechar a janela.
`Janela`	Classe principal (`tk.Tk`) que monta a GUI, instancia frames, gerencia seleção de ROI e encerra tudo de forma limpa ao fechar.

```
import threading
import time
import tkinter as tk
import tkinter.filedialog as filedialog

import cv2
import matplotlib.pyplot as plt
import numpy as np
import pytesseract

from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from PIL import Image, ImageTk
```

```
# Ajuste o caminho para o execut vel do Tesseract no seu sistema
    :
pytesseract.pytesseract.tesseract_cmd = r"C:\Program␣Files\
    Tesseract-OCR\tesseract.exe"

def cortar_imagem(image, rect):
    x, y, w, h = rect
    return image[y:y+h, x:x+w]

def melhorar_imagem(imagem, fator_aumento=6):
    altura, largura = imagem.shape[:2]
    cinza = cv2.cvtColor(imagem, cv2.COLOR_BGR2GRAY)
    _, thresh = cv2.threshold(cinza, 128, 255, cv2.THRESH_BINARY)
    ampliada = cv2.resize(
        thresh,
        (int(largura * fator_aumento), int(altura * fator_aumento
            ))
    )
    return ampliada

def reconhecer_inteiro(imagem_binaria: np.ndarray,
                       opcoes: str="--oem␣3␣--psm␣6␣outputbase␣
                           digits") -> int:
    texto = pytesseract.image_to_string(imagem_binaria, config=
        opcoes).strip()
    return int(texto) if texto.isdigit() else 0

class ProcessadorDeVideo:
    def __init__(self, fonte_de_video, espec_rois: dict,
        periodo_de_amostragem: float):
        self.fonte = fonte_de_video
        self.espec_rois = espec_rois
        self.fps = self.fonte.get(cv2.CAP_PROP_FPS) or 30
        self.tempo_por_frame = 1.0 / self.fps
        self.periodo_de_amostragem = periodo_de_amostragem
        self.dados = {nome: [] for nome in espec_rois}
        self.tempo_passado = 0.0

    def __iter__(self):
        return self
```

```
    def __next__(self):
        while self.tempo_passado < self.periodo_de_amostragem:
            ok, frame = self.fonte.read()
            if not ok:
                raise StopIteration
            time.sleep(self.tempo_por_frame)
            self.tempo_passado += self.tempo_por_frame

        self.tempo_passado -= self.periodo_de_amostragem
        resultado = {}
        for nome, cfg in self.espec_rois.items():
            roi = cfg["roi"]
            corte = cortar_imagem(frame, roi).copy()
            melh = melhorar_imagem(corte)
            inteiro = reconhecer_inteiro(melh)
            resultado[nome] = inteiro
            self.dados[nome].append(inteiro)
        return resultado

class FrameDeVideo(tk.Frame):
    def __init__(self, master):
        super().__init__(master)
        tk.Label(self, text="Caminho do Vídeo:").grid(row=0,
            column=0, sticky="e")
        self.entry = tk.Entry(self, width=30)
        self.entry.grid(row=0, column=1, padx=5, pady=5)
        tk.Button(self, text="Selecionar Vídeo", command=self.
            _selecionar).grid(row=1, column=0, columnspan=2)
        self.video_label = tk.Label(self)
        self.video_label.grid(row=2, column=0, columnspan=2)

    def _selecionar(self):
        caminho = filedialog.askopenfilename(title="Selecione o
            Vídeo")
        if caminho:
            self.entry.delete(0, tk.END)
            self.entry.insert(0, caminho)

    def get_caminho(self):
        return self.entry.get().strip()

    def atualizar(self, frame, tamanho):
```

```
        img = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        img = cv2.resize(img, tamanho)
        img = ImageTk.PhotoImage(Image.fromarray(img))
        self.video_label.config(image=img)
        self.video_label.image = img

class FrameDoGrafico(tk.Frame):
    def __init__(self, master, config, tamanho_grafico):
        super().__init__(master)
        tk.Label(self, text="Per odo␣(s):").grid(row=0, column
            =0, sticky="e")
        self.entry_periodo = tk.Entry(self, width=10)
        self.entry_periodo.insert(0, str(config.get("
            periodo_padrao", 1.0)))
        self.entry_periodo.grid(row=0, column=1)
        self.tamanho_grafico = tamanho_grafico

        fig, ax = plt.subplots()
        ax.set_xlabel("Amostra")
        ax.set_ylabel("Valor␣OCR")
        ax.set_title("Leitura␣em␣tempo␣real")
        self.canvas = FigureCanvasTkAgg(fig, master=self)
        self.canvas.get_tk_widget().grid(row=1, column=0,
            columnspan=3)
        self.ax = ax

        for i, nome in enumerate(config["rois"].keys()):
            tk.Button(
                self,
                text=nome,
                command=lambda n=nome: master.seleciona_roi(n)
            ).grid(row=2, column=i)

    def plotar(self, dados, nome_atual):
        self.ax.clear()
        serie = dados.get(nome_atual, [])
        ultimos = serie[-self.tamanho_grafico:]
        self.ax.plot(ultimos, marker="o", linestyle="--")
        self.ax.set_title(nome_atual)
        self.canvas.draw()

    def get_periodo(self):
```

```
        try:
            return float(self.entry_periodo.get())
        except ValueError:
            return 1.0

class Processador(ProcessadorDeVideo):
    def __init__(self, fonte, config, periodo, frame_grafico,
       frame_video, nome):
        super().__init__(fonte, config["rois"], periodo)
        self.frame_grafico = frame_grafico
        self.frame_video = frame_video
        self.nome_atual = nome
        self.config = config

    def __next__(self):
        res = super().__next__()
        ok, frame = self.fonte.read()
        if ok:
            tamanho = (
                self.frame_grafico.tamanho_grafico * 20,
                self.frame_grafico.tamanho_grafico * 15
            )
            self.frame_video.atualizar(frame, tamanho)
        self.frame_grafico.plotar(self.dados, self.nome_atual)
        return res

class Controlador(threading.Thread):
    def __init__(self, frame, config):
        super().__init__()
        self.frame = frame
        self.config = config
        self.rodando = False
        self.processador = None

    def run(self):
        while self.rodando:
            try:
                valores = next(self.processador)
                print(valores)
            except StopIteration:
                self.rodando = False
```

```
class Janela(tk.Tk):
    def __init__(self, config):
        super().__init__()
        self.title("Leitura␣OCR␣em␣V deo")
        self.config_rois = {"rois": config["rois"], "
           periodo_padrao": config["periodo_padrao"]}
        tamanho = config["tamanho_grafico"]
        self.frame_graf = FrameDoGrafico(self, self.config_rois,
           tamanho)
        self.frame_graf.grid(row=0, column=0)
        self.frame_vid = FrameDeVideo(self)
        self.frame_vid.grid(row=0, column=1)
        self.control = Controlador(self, self.config_rois)
        self.protocol("WM_DELETE_WINDOW", self._on_close)

    def seleciona_roi(self, nome):
        if self.control.rodando:
            self.control.rodando = False
            self.control.join()
        periodo = self.frame_graf.get_periodo()
        cap = cv2.VideoCapture(self.frame_vid.get_caminho())
        self.control.processador = Processador(cap, self.
           config_rois, periodo,
                                               self.frame_graf,
                                                  self.frame_vid,
                                                  nome)
        self.control.rodando = True
        self.control.start()

    def _on_close(self):
        if self.control.rodando:
            self.control.rodando = False
            self.control.join()
        self.destroy()

if __name__ == "__main__":
    configuracao = {
        "tamanho_grafico": 10,
        "periodo_padrao": 1.0,
        "rois": {
            "Frequ ncia␣Card aca": {
                "roi":  (751, 180, 115, 70),
```

```
                "limites": {"superior": 150, "inferior": 40},
            },
            "Satura  o": {
                "roi": (749, 319, 180, 78),
                "limites": {"superior": 100, "inferior": 85},
            },
            "Frequ ncia de Pulso": {
                "roi": (881, 180, 100, 55),
                "limites": {"superior": 150, "inferior": 40},
            },
        }
    }
    app = Janela(configuracao)
    app.mainloop()
```

Listing A.1 – Código-fonte completo do processamento de vídeo com OCR e interface gráfica