UNIVERSIDADE DO ESTADO DO AMAZONAS
ESCOLA SUPERIOR DE TECNOLOGIA

CÁSSIO ANTONIO TAVARES ALVES

DESENVOLVIMENTO DE SISTEMA PARA DETECÇÃO E REMOÇÃO AUTOMÁTICA DE SOMBRAS EM IMAGENS DE AMBIENTES INTERNOS

Manaus
2020

CÁSSIO ANTONIO TAVARES ALVES

DESENVOLVIMENTO DE SISTEMA PARA DETECÇÃO E REMOÇÃO AUTOMÁTICA DE SOMBRAS EM IMAGENS DE AMBIENTES INTERNOS

Projeto de pesquisa desenvolvido durante a disciplina de Trabalho de Conclusão de Curso II e apresentada à banca avaliadora do Curso de Engenharia Elétrica da Escola Superior de Tecnologia da Universidade do Estado do Amazonas, como pré-requisito para obtenção do título de Engenheiro Eletricista.

Orientador: Jozias Parente de Oliveira, Dr.

Manaus

2020

Universidade do Estado do Amazonas – UEA
Escola Superior de Tecnologia - EST

Reitor:
Cleinaldo de Almeida Costa
Vice-Reitor:
Cleto Cavalcante De Souza Leal
Diretora da Escola Superior de Tecnologia:
Ingrid Sammyne Gadelha Figueiredo
Coordenador do Curso de Engenharia Elétrica:
Israel Gondres Torné

Banca Avaliadora composta por: *Data da defesa: 01/12/2020.*
Prof. Jozias Parente de Oliveira, Dr. *(Orientador)*
Prof. Daniel Guzmán Del Río, Dr.
Prof. Karlo Homero Ferreira dos Santos, Msc.

CIP – Catalogação na Publicação

Alves, Cássio Antonio Tavares

Desenvolvimento de sistema para detecção e remoção automática de sombras em imagens de ambientes internos / Cássio Antonio Tavares Alves; [orientado por] Jozias Parente de Oliveira. – Manaus: 2020.
63 f. p.: il.

Trabalho de Conclusão de Curso (Graduação em Engenharia Elétrica). Universidade do Estado do Amazonas, 2020.

1. Processamento Digital de Imagens. 2. K-Nearest Neighbors KNN. 3. Detecção de Sombras. I. Oliveira, Jozias Parente de.

CÁSSIO ANTONIO TAVARES ALVES

DESENVOLVIMENTO DE SISTEMA PARA DETECÇÃO E REMOÇÃO AUTOMÁTICA DE SOMBRAS EM IMAGENS DE AMBIENTES INTERNOS

Projeto de pesquisa desenvolvido durante a disciplina de Trabalho de Conclusão de Curso II e apresentada à banca avaliadora do Curso de Engenharia Elétrica da Escola Superior de Tecnologia da Universidade do Estado do Amazonas, como pré-requisito para obtenção do título de Engenheiro Eletricista.

Nota obtida: 90 (nove)

Aprovada em 01/12/2020.

Área de concentração: Processamento Digital de Imagens

BANCA EXAMINADORA

Orientador: Jozias Parente de Oliveira, Dr.

Avaliador: Daniel Guzmán Del Río, Dr.

Avaliador: Karlo Homero Ferreira dos Santos, Msc.

Manaus 2020

Dedicatória

Dedico este trabalho aos meus Pais e Avós que sempre lutaram para que eu pudesse trilhar o caminho do aprendizado e do conhecimento.

Agradecimentos

Agradeço à Deus por sua misericórdia. A meus Pais que em meio as dificuldades impostas pela vida sempre deram tudo de si para que eu tivesse uma vida melhor e oportunidades de crescimento. Aos meus Avós que participaram ativamente de minha criação de minha formação como pessoa. A meus outros tantos familiares por me apoiaram desde sempre nessa caminhada. Aos meus amigos pela amizade e companheirismo.

RESUMO

O estudo na área de Processamento Digital de Sinais é uma das que mais atraem pesquisadores devido ao fato de que muitos produtos ou serviços de hoje em dia necessitam de processamento de imagens, visto que os serem humanos são essencialmente visuais. O presente trabalho tem como tema o Desenvolvimento de Sistema para Detecção e Remoção Automática de Sombras em Imagens de Ambientes Internos. Onde o objetivo é tratar do assunto de processamento digital de imagens afim de desenvolver um sistema que seja capaz de detectar e remover sombras automaticamente, o que se configura como uma etapa de pré-processamento em tarefas de visão computacional. O presente trabalho apresenta um sistema de detecção de sombras baseado no método K-Nearest Neighbors (KNN), um método estatístico não paramétrico, bastante utilizado em tarefas de segmentação de imagens. A abordagem utilizada se mostra bastante interessante visto que os resultados obtidos mostram uma taxa de até 85% de acerto na detecção de zonas de sombra nas imagens.

Palavras-chave: Processamento Digital de Imagens, Visão Computacional, Sombras, Detecção, Remoção.

ABSTRACT

The study in the area of Digital Signal Processing is one of the ones that most attract researchers due to the fact that many products or services today require image processing, since human beings are essentially visual. The present work has as its theme the Development of a System for the Automatic Detection and Removal of Shadows in Images of Indoor Environments. Where the objective is to deal with the subject of digital image processing in order to develop a system that is able to detect and remove shadows automatically, which is configured as a pre-processing step in computer vision tasks. The present work presents a shadow detection system based on the K-Nearest Neighbors (KNN) method, a non-parametric statistical method, widely used in image segmentation tasks. The approach used is very interesting since the results obtained show a rate of up to 85% of correctness in detecting shadow zones in the images.

Keywords: Digital Image Processing, Computer Vision, Shadows, Detection, Removal.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

INTRODUÇÃO

A detecção de sombras em imagens e remoção de sombras em imagens é uma tarefa crucial e desafiadora em diversas situações da vida real como sistemas de vigilância, cenas internas e externas, detecção e rastreamento etc (CHONDAGAR et al., 2015). Para algoritmos de visão computacional a detecção e remoção de imagens é de suma importância pois aumenta o desempenho das aplicações de reconhecimento e classificação, como por exemplo os sistemas de vigilância de tráfego que tem desempenho prejudicado pois pode haver erro na classificação dos objetos devido a presença de áreas sob efeito de sombreamento.

Entretanto a presença de sombras em imagens pode ser benéfica para aplicações que desejam retirar características relacionadas aos objetos e iluminação de uma cena, devido ao fato de que as sombras revelam informações como forma e orientação dos objetos bem como informações sobre a fonte de luz. Adicionando o fato de imagens livres de sombras são benéficas nas áreas de edição de imagens em aplicações de realidade aumentada. Logo, a detecção e remoção de sombras em imagens é uma etapa de pré-processamento de grande importância para a visão computacional.

A implementação de um sistema capaz de detectar e remover sombras em imagens de ambientes internos envolve a utilização de conceitos utilizados nas seguintes disciplinas: Linguagem de Programação I e II, Probabilidade e Estatística, Sinais e Sistemas, Processamento Digital de Sinais, e Processamento Digital de Imagens. Além disso, o desenvolvimento do trabalho visa contribuir com a revisão bibliográfica de temas relacionados a segmentação de imagens.

Com o objetivo de detectar sombras em imagens, várias técnicas algoritmos foram desenvolvidos a fim de melhorar os inúmeros sistemas de reconhecimento de objetos e ou pessoas, sistemas de vigilância, sistemas de classificação entre outros. Com cada método sendo apropriado para uma situação específica. Vale frisar também que, os algoritmos utilizados para essa funcionalidade utilizam abordagens de alta complexidade que dificultam a implementação em sistemas de tempo real e baixo custo para essa finalidade.

No presente trabalho é apresentado um sistema cujo objetivo é detectar e remover sombras em imagens de ambientes internos, com os objetivos específicos desenvolvidos para atingir a meta do trabalho foram os seguintes:

a) conhecer as características das imagens de ambientes internos;

b) obter informações que caracterizam as sombras;

c) desenvolver algoritmo para a detecção de sombras;

d) desenvolver algoritmo para a remoção de sombras.

O trabalho é composto por quatro capítulos. O Referencial Teórico apresenta todo o fundamental teórico dos temas abordados, a Metodologia apresenta a forma de como foi realizada a pesquisa do tema, além de descrever de forma mais geral os algoritmos que compõe o sistema. O capítulo 3 expõe detalhes sobre a implementação dos módulos que formam o sistema.

O capítulo 4 apresentam as métricas de validação utilizadas para mensurar de forma quantitativa o desempenho do sistema. Também apresenta a análise dos resultados obtidos. E por fim apresenta-se a conclusão do trabalho, onde é feita uma reflexão acerca dos resultados que eram esperados e os resultados que foram obtidos pelo sistema.

1 REFERNCIAL TEÓRICO

1.1 Processamento Digital de Imagem

Processamento Digital de Imagens (PDI) está geralmente associado ao processamento de uma imagem bidimensional digital por um computador, mas este termo pode ser estendido para qualquer processamento digital de dados bidimensionais (JAIN, 1989).

Atualmente não existe uma definição formal acerca da divisão entre PDI e outras áreas como a Análise de Imagens e a Visão Computacional (GONZALEZ; WOODS, 2010). Alguns autores costumam definir PDI como uma área onde tanto a entrada quanto a saída do processo são imagens.

A área de PDI foi desenvolvida para lidar com alguns problemas relacionados as imagens, quais sejam: digitalização e codificação para fins de transmissão, impressão e armazenamento, restauração de imagens, segmentação e descrição como estágio de pré-processamento para Visão Computacional (PETROU; PETROU, 2010).

1.2 Fundamentos da Imagem

Uma imagem é definida como uma função de duas variáveis $f(x, y)$ onde x e y são as coordenadas espaciais do plano e f é a intensidade da imagem nesse ponto (GONZALEZ; WOODS, 2010). Se os valores de x e y forem finitos e discretos então essa imagem é uma imagem digital conforme apresentada na Equação 1.

$$f(x,y) = \begin{bmatrix} f(1,1) & f(1,2) & \cdots & f(1,N) \\ f(2,1) & f(2,2) & \cdots & f(2,N) \\ \vdots & \vdots & & \vdots \\ f(N,1) & f(N,2) & \dots & f(N,N) \end{bmatrix} \quad (1)$$

Onde: $0 \leq f(x,y) \leq G-1$, sendo N e G potências positivas de 2 $(N = 2^n), G = 2^m$. Cada elemento da matriz representada pela Equação 1 é chamado de elemento pictórico, elemento de imagem ou como é mais comumente chamado *pixel* (GONZALEZ; WOODS, 2010).

Cada *pixel* contém uma quantidade de bits como exemplificado na Figura 1, e essa quantidade de *pixels* determina o tipo da imagem. Imagens onde os *pixels* podem ter apenas valores 0 ou 1 são chamadas de imagens binárias, apenas as cores branco e preto podem ser representadas com esta escala. A chamada escala de cinza é uma escala composta por 256 níveis de brilho, ou seja, os bits podem conter valores entre 0 e 255, nesse tipo de imagem a intensidade

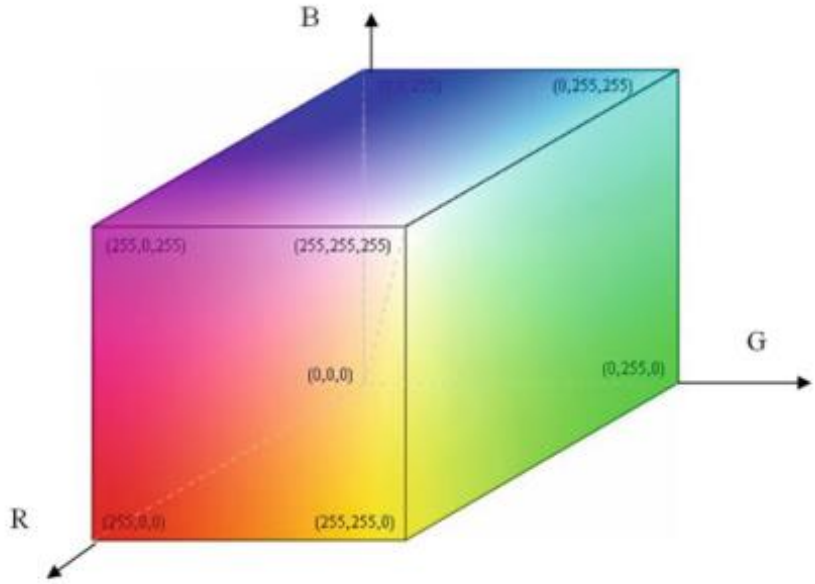
dos *pixels* pode variar no intervalo de preto, cinza e branco. Já as imagens coloridas os pixels podem ter todas as cores como vermelho, verde e azul (Figura 2) (FURHT, AKAR, ANDREWS, 2018).

> A representação mais comum das imagens coloridas, a representação RGB (red, green, blue), baseia-se na teoria tricromática de que a sensação de cor é produzida pela excitação seletiva de três classes de receptores oculares.
>
> No formato de cores de 24 bits, cada cor é representada com 8 bits. Portanto, os elementos no cubo tridimensional têm valores de (0, 0, 0) a (255, 255, 255). A cor preta é definida como (0, 0, 0) e o componente branco como (255, 255, 255). A imagem em escala de cinza é definida como a linha reta no cubo tridimensional, que é quando R = G = B (FURHT; AKAR; ANDREWS, 2018).

Figura 1: Conceito de Imagem 5x5 pixels

Fonte: (FURHT; AKAR; ANDREWS, 2018)

Figura 2: Representação tridimensional de imagens usando componentes R, G e B



Fonte: (FURHT; AKAR; ANDREWS, 2018)

Como foi discutido anteriormente as imagens digitais são constituídas por *pixels*, então pode-se definir a resolução R de uma imagem em termos de *pixels* da seguinte forma:

$$R = nm \tag{2}$$

Onde n representa a quantidade de *pixels* na horizontal e m representa os *pixels* na vertical.

1.3 Segmentação de Imagens

"A segmentação de imagem é o processo que divide uma imagem em diferentes regiões, de modo que cada região seja homogênea de acordo com algumas características ou características bem definidas" (TYAGI, 2018).

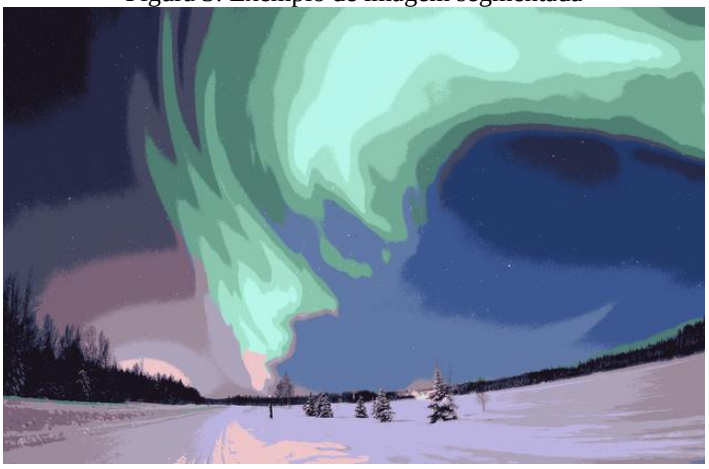
Grande parte dos algoritmos utilizados na segmentação de imagens baseia-se em duas propriedades relacionadas a intensidade da imagem: descontinuidade e similaridade.

Nos algoritmos com base em descontinuidade a abordagem utilizada é dividir a imagem, levando em consideração as alterações bruscas de intensidade, como por exemplo as bordas. Na segunda categoria de algoritmos a estratégia empregada é dividir a imagem em regiões que sejam semelhantes obedecendo a critérios predefinidos. Métodos como limiarização, crescimento de região e divisão e fusão de regiões se encaixam nessa categoria (GONZALEZ; WOODS, 2010). (Ver Figuras 3 e 4).

O processo de segmentação é uma tarefa de suma importância em um sistema de análise de imagens, além de ser uma das mais difíceis de ser performada. Nível de precisão que o

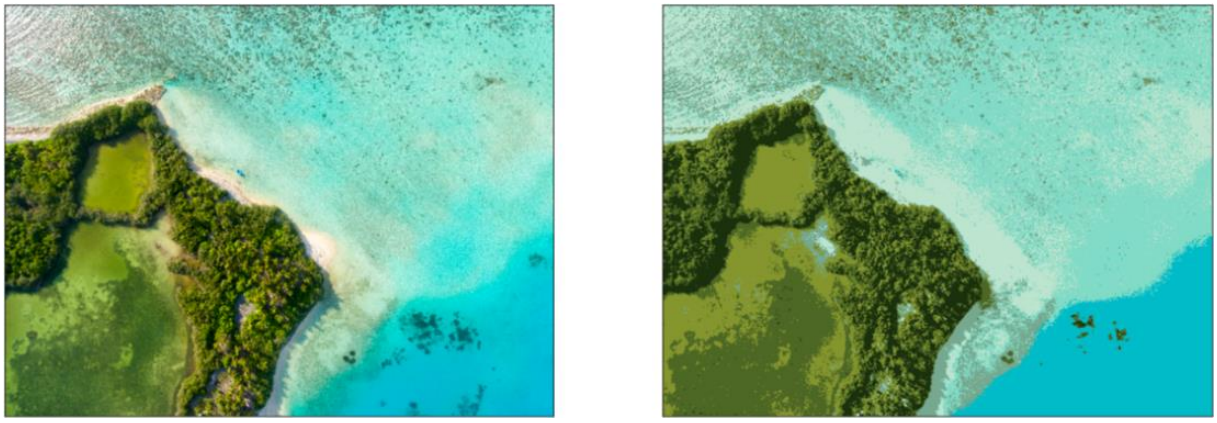
sistema de análise de imagem depende em suma do nível de precisão do processo de segmentação.

Figura 3: Exemplo de imagem segmentada



Fonte: (STRANG, 2005)

Figura 4: Imagem original e Imagem segmentada



Fonte: (CHAUHAN, 2019)

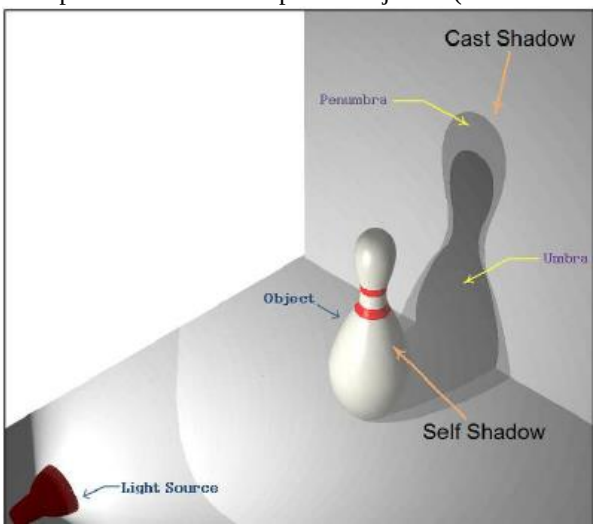
1.4 Definição de Sombra: Sombra Própria e Sombra Projetada

Sombra é um fenômeno físico que é formado quando uma fonte de luz gerada por alguma fonte é parcial ou totalmente obstruída por um objeto (SINGH; MAXTON, 2014).

As sombras têm dois tipos de classificação: sombra Própria e sombra Projetada. A sombra própria é uma parte do próprio objeto que não recebe iluminação, essa sombra se localiza na superfície oposta à superfície que é iluminada. A sombra projetada é a sombra que é produzida pela projeção da luz na direção do objeto essa sombra mantém características algumas do objeto iluminado. A sombra projetada ainda é subdividida em dois tipos: umbra e penumbra. A parte da sombra projetada que onde a luz foi

completamente bloqueada é a chamada umbra e a parte onde a luz foi parcialmente obstruída é chamada de penumbra. A Figura 5 exemplifica os tipos de sombras.

Figura 5: Tipos de Sombras: Própria e Projetada (Umbra e Penumbra)



Fonte: (AMATO et al., 2014)

1.5 Visão Computacional

O objetivo principal da área de Visão Computacional é obter informações a partir de imagens, (PRINCE, 2012) isso implica que o objeto de estudo da Visão Computacional é reconstruir e interpretar cenas naturais com base apenas no conteúdo das imagens (PETERS, 2017).

O processo da Visão Computacional é uma tarefa desafiadora, pois parte do problema é devido a complexidade apresentada por dados visuais (PRINCE, 2012). Na Figura 6 temos um exemplo de como os programas de visão computacional podem ser afetados pela presença de sombras, nessa imagem um algoritmo de segmentação foi aplicado com objetivo de detectar o objeto, pode-se observar que a imagem foi erroneamente segmentada pela aplicação, no primeiro caso ocorreu uma subsegmentação causada pela forte intensidade da sombra própria do objeto e no segundo caso ocorreu uma sobresegmentação devido a influência da sombra projetada do objeto (ECINS; FERMÜLLER; ALOIMONOS, 2014).

Assim, como uma etapa de pré-processamento a detecção de sombras e sua remoção pode tornar tarefas como a segmentação mais precisas, logo, tornado outras aplicações na parte de visão computacional mais robustas.

Figura 6: Exemplo de problemas de segmentação causados pela presença de sombras

Fonte: (ECINS; FERMÜLLER; ALOIMONOS, 2014)

1.6 Abordagem Estatística não Paramétrica para Detecção de Sombras

Muitos dos testes em estatística são realizados tomando como base amostras que seguem certas suposições denominadas parâmetros, estes tipos de testes são chamados testes paramétricos. As suposições paramétricas contêm amostras que:

a) são sorteados aleatoriamente de uma população com distribuição normal;

b) consistem em valores em uma escala de medição de intervalo ou razão;

c) são adequadamente grandes;

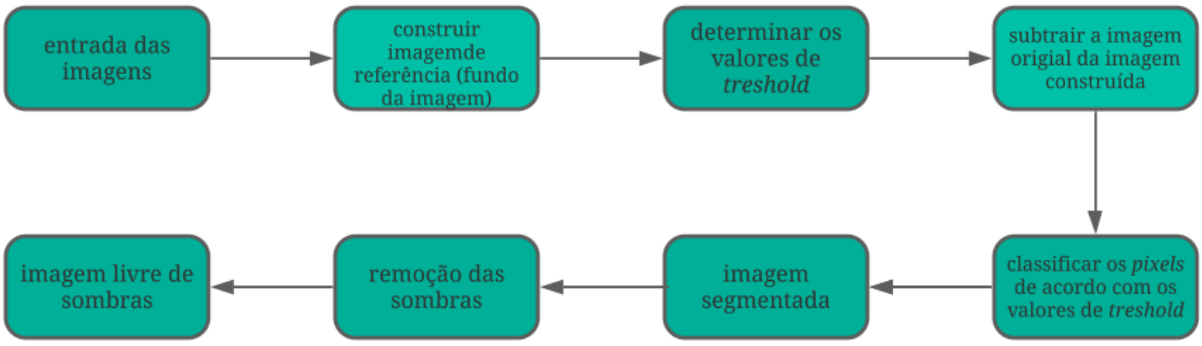
d) assemelham-se aproximadamente a uma distribuição normal.

Quando uma amostra não tiver uma alguma dessas características então ela viola as suposições de testes paramétricos (CORDER; FOREMAN, 2009). Para tratar de situações desse tipo convém o uso de outra abordagem estatística que não demande que a amostra tenha todos os parâmetros ou regras. Essa nova abordagem é chamada de Estatística não paramétrica. Os testes não paramétricos abrangem técnicas onde os dados não estão vinculados a nenhuma distribuição particular.

Abordagens estatísticas não paramétricas se mostram efetivas para a detecção de sombras em imagens de ambientes internos. Nessa abordagem um novo espaço de cores é baseado no espaço RGB é desenvolvido com a finalidade de diferenciar o fundo que está sob efeito de sombra do fundo comum. Depois que um novo espaço de cores é desenvolvido o próximo passo é criar um algoritmo que tem a função de subtrair o fundo da imagem.

Este algoritmo em primeiro lugar deve construir uma imagem de referência que represente o fundo e então determinar um valor de *threshold* apropriado que será usado para a operação de subtração, assim obtendo uma taxa de detecção e por fim classificando os pixels (DAS et al., 2017). (Ver Figura 7)

Figura 7: Fluxograma simplificado do sistema



FONTE: (DAS et al., 2017)

Uma comparação entre as abordagens utilizadas para detecção de sombras é mostrada na Tabela 1.

Tabela 1: Comparativo entre abordagens de detecção de sombras

Método	**Vantagem**	**Desvantagem**
Abordagem estatística não paramétrica	Efetivo para imagens de ambientes internas	A precisão é bastante baixa para imagens externas.
Abordagem estatística paramétrica	Independente da cena e dos objetos	Falha na detecção de sombras projetadas indiretas.
Abordagem determinística não baseada em modelo (DNM1)	Alta capacidade de lidar com vários tamanhos de projeção de sombra e robusta ao ruído.	Falha na detecção de sombras projetadas indiretas.
Abordagem determinística não baseada em modelo (DNM2)	Alta flexibilidade.	Falha na detecção de sombras projetadas indiretas.

Fonte: (DAS et al., 2017)

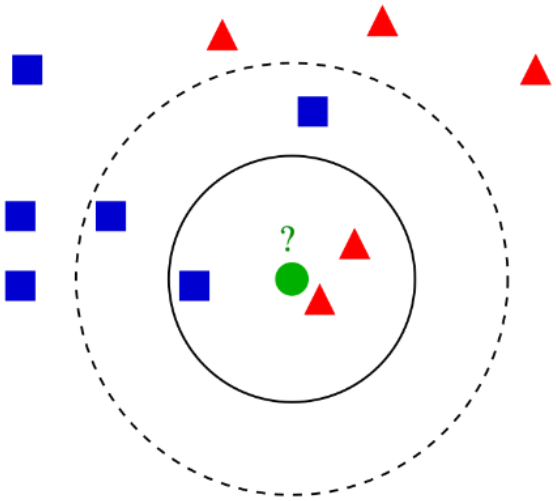
Como mencionado anteriormente a abordagem estatística não paramétrica é eficaz para aplicações para a detecção de sombras em ambientes internos. Essa será abordagem utilizada para o desenvolvimento do projeto.

1.7 Classificador KNN (K-Nearest Neighbors)

O KNN é um método de classificação não paramétrico baseado em memória e não necessita de modelo para ajustes (HASTIE; TIBSHIRANI; FRIEDMAN, 2008).

O KNN funciona da seguinte forma: Dado um ponto x_0 tomamos então os k pontos mais próximos no espaço de características. Esses pontos são os chamados pontos de treinamento $x(r), r = 1, 2, 3, \ldots, k$, a partir disso o próximo passo é então classificar o elemento x_0 de acordo com a classe do elemento de maior quantidade presente nos k elementos vizinhos (HASTIE; TIBSHIRANI; FRIEDMAN, 2008). A Figura exemplifica de modo visual como é feita a classificação por meio do método KNN.

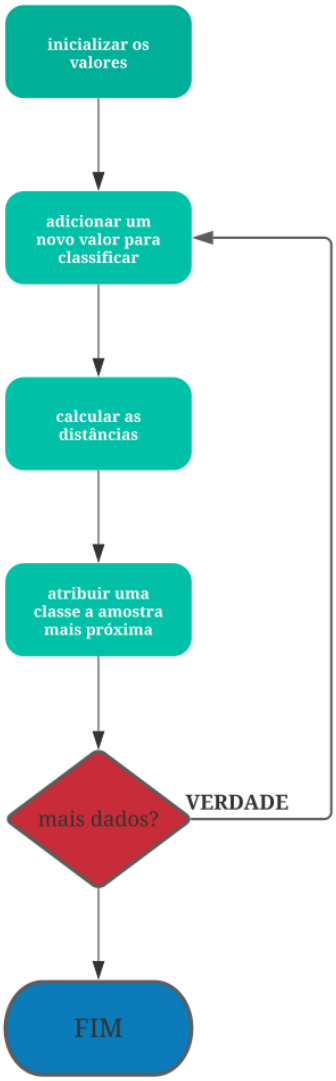
Figura 8: Exemplo de classificação por meio do método KNN



Fonte: (AJANKI, 2007)

O método KNN pode ser sintetizado de acordo com o fluxograma mostrado na Figura 9.

Figura 9: Fluxograma do processo de classificação KNN



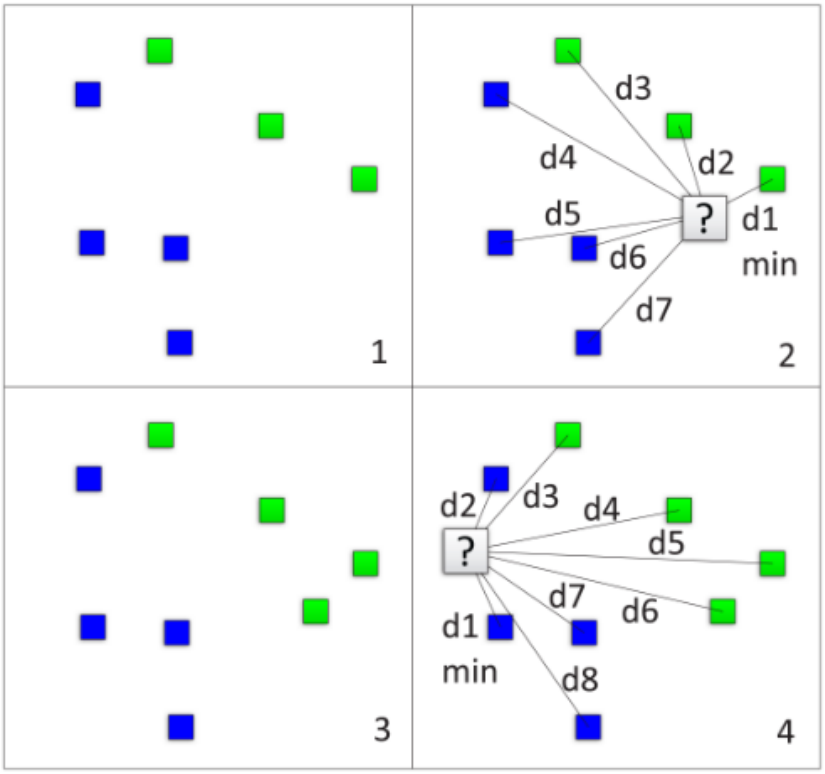
Fonte: (BONNIN, 2017)

Cada passo do processo é descrito de forma simplificada abaixo:

a) toma-se as amostras previamente cuja classe é previamente conhecida;
b) lê-se a próxima amostra a ser classificada e então calcula-se a distância euclidiana a partir da nova amostra para todos os outras do conjunto.
c) determina-se a classe da amostra com base na classe da amostra mais próxima. O método requer levar em conta a contribuição de todas *k* amostras;
d) o processo é repetido até não sobrarem amostras não classificadas (BONNIN, 2017).

Na Figura 10 é apresentado o processo de aplicação de um *loop* do algoritmo de classificação KNN.

Figura 10: Exemplo de aplicação de um *loop* KNN.



Fonte: (BONNIN, 2017)

1.7.1 K-Nearest Neighbours Background Subtraction

Imagens de uma cena filmada por exemplo por uma câmera estática apresentam um comportamento regular, que pode ser descrito por meio de um modelo estatístico. Com um modelo estatístico da cena algum objeto intruso pode ser facilmente detectado apenas determinando as partes da imagem que não se enquadram no modelo estabelecido. Isso é basicamente um processo de subtração de fundo (ZIVKOVIC; HEIJDEN, 2006). Para uma imagem de uma cena estática um modelo que pode ser utilizado é a própria imagem da cena livre da presença de objetos intrusos.

No presente trabalho o modelo utilizado para a subtração do fundo é baseado no modelo de estimativa de densidade de kernel proposto por (ELGAMMAL; HARWOOD; DAVIS, 2000) e melhorado por (ZIVKOVIC; HEIJDEN, 2006).

A subtração do fundo baseada em pixel envolve um processo de tomada de decisão para determinar se o pixel pertence ao fundo ou ao primeiro plano da imagem. Um dado pixel tem maior probabilidade de pertencer ao fundo da imagem se a Relação 2 for maior que 1 (ZIVKOVIC; HEIJDEN, 2006).

$$\frac{p(BG|\overrightarrow{x_t})}{p(FG|\overrightarrow{x_t})} = \frac{p(\overrightarrow{x_t}|BG)p(BG)}{p(\overrightarrow{x_t}|FG)p(FG)} \quad (3)$$

O valor $\overrightarrow{x_t}$ na Relação 2 é o valor de um pixel em um dado tempo t. Os parâmetros $p(BG)$ e $p(FG)$ representam o modelo estimado do fundo e do primeiro plano da imagem respectivamente. A classificação de um pixel como fazendo parte do fundo ou do primeiro plano da imagem é feita com o auxílio da Equação 3 (ZIVKOVIC; HEIJDEN, 2006).

$$p(\overrightarrow{x_t}|BG) > c_{thr} \quad (4)$$

Onde $c_{thr} = p(\overrightarrow{x_t}|FG)\frac{p(FG)}{p(BG)}$ é o valor de *threshold*. O modelo para a imagem de fundo é estimado a partir de um conjunto de treino

O processo para estimar a densidade de *kernel* começa por meio da contagem do número de amostras *k* do conjunto de dados X_T que se encontra dentro do volume V do $kernel$. Onde esse volume V é uma hiperesfera de diâmetro D (ZIVKOVIC; HEIJDEN, 2006). Com essas informações podemos estimar a densidade de $kernel$ através da Equação 5.

$$\hat{p}((\overrightarrow{x_t}|X_T,\ BG + FG) = \frac{1}{TV}\sum_{m=t-T}^{t} K\left(\frac{\|\overrightarrow{x_m} - \vec{x}\|}{D}\right) = \frac{k}{TV} \quad (5)$$

Onde $K(u)$ é a função *kernel* e é igual 1 se $u < 1/2$ e 0 para quaisquer outros casos e T é o período de adaptação.

O método de estimação de *kernel* utilizado no presente trabalho é chamado de *ballon estimation*, nessa abordagem o *kernel* é adaptado a cada ponto $\vec{x}$ e o valor de D é incrementado à cada novo ponto $\vec{x}$ até que todas as amostras *k* sejam cobertas (ZIVKOVIC; HEIJDEN, 2006).

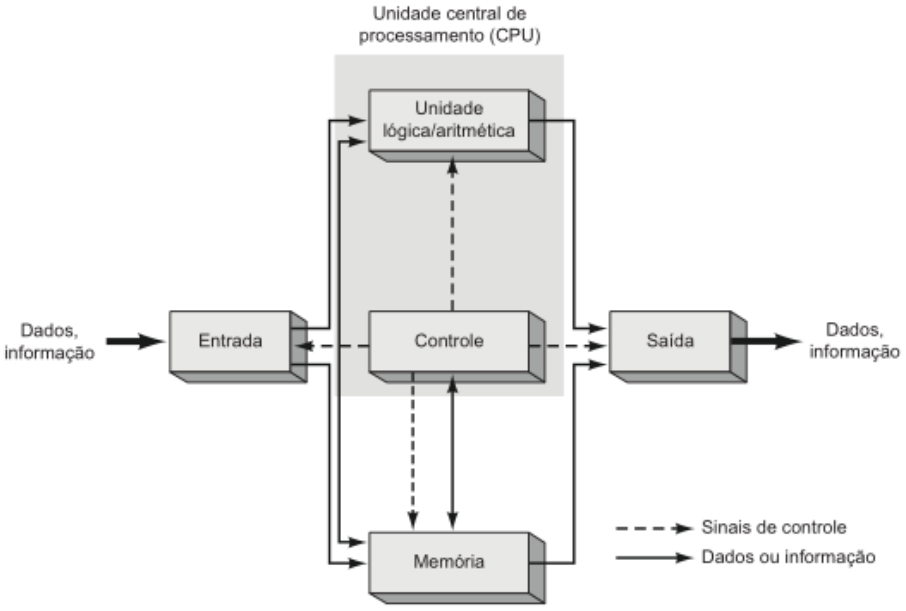
1.8 *Hardware* para Processamento de Imagens

1.8.1 Microcomputador

O Microcomputador (Ver Figura 11) é um dispositivo de uso genérico, pode executar uma vasta quantidade de aplicações de acordo com os programas que o usuário desejar utilizar. Um Microcomputador é constituído de:

a) unidade de Entrada que é responsável por armazenar temporariamente as informações que forem introduzidas pelo usuário até o momento de sua utilização;
b) unidade de Memória armazena as instruções e os dados que foram recebidos pelos dispositivos de entrada;
c) unidade de Controle tem a função de buscar as instruções armazenadas na Unidade de Memória e interpretá-las e então enviar sinais para as outras unidades de acordo com as instruções específicas a serem, executadas;
d) unidade de Unidade Lógica Aritmética (ULA) que é onde são realizados todos os cálculos aritméticos e decisões lógicas;
e) unidade de Saída recebe os dados vindos da Unidade de Memória e os mostra para o usuário em forma de uma imagem no display, ou mesmo impressa (TOCCI; WIDMER; MOSS, 2011).

Figura 11: Diagrama funcional de um computador de um microcomputador



Fonte: (TOCCI; WIDMER; MOSS, 2011)

As aplicações em processamento de imagens demandam uma quantidade significativa de poder computacional. Essas aplicações são desenvolvidas em diferentes arquiteturas tais como *Digitial Signal Processor* (DSP), *hardware* de uso geral, *Application Specific Integrated Circuits* (ASIC) e *Field Programmable Gate Array* (FPGA) (SIVASHANMUGAM, 2008).

O microprocessador (processador de uso geral) é o principal componente de um microcomputador. Sua arquitetura é fixa e seu conjunto de instruções é limitado e pré-definido. O microprocessador executa programas que são conjuntos de instruções que ficam armazenadas em uma memória externa. Porém o microcomputador pode ser lento na execução de tarefas como as que envolvem operação de ponto flutuante ou mesmo funções matemáticas complexas, (SIVASHANMUGAM, 2008) entretanto nos últimos anos a tecnologia dos processadores evoluiu a ponto de atenuar essas dificuldades com a utilização de vários núcleos de processamento em um único *chip*.

Atualmente a maior parte dos processadores disponíveis no mercado são processadores multinúcleo, que possuem desde dois núcleos até quantidades como 16 núcleos. Essa abordagem possibilita o aumento de performance sem a necessidade de um sistema complexo e sem aumento de requisitos de energia (KIM; KIM; LEE, 2011).

Outra solução para aumentar o desempenho dos microcomputadores é tirar vantagem do paralelismo dos *softwares* ao invés de depender de tecnologias de *hardware* (KIM; KIM; LEE, 2011). Os processadores Intel possuem uma tecnologia chamada *Hyper-Threading Technology* que possibilita um simples processador ter (pela perspectiva do software) múltiplos processadores lógicos.

Na programação para processamento de imagens a característica de paralelismo dos processadores pode ser utilizada para extrair os melhores resultados das aplicações. Primeiramente a execução divide os algoritmos em estágios e então cada um deles é atribuída a uma *thread* diferente. Na execução simultânea os dados são divididos e cada *thread* processa parte que lhe cabe dos dados de entrada ao longo de todo o algoritmo (KIM; KIM; LEE, 2011).

Intel Integrated Performance Primitives (Intel IPP) é uma biblioteca que oferece suporte para aplicações *multi-thread* para processadores Intel e que pode ser usada para processamento tanto de imagem como de vídeo. A biblioteca OpenCV a partir da versão 3.0 passou a ter incorporado ao se escopo um subconjunto do Intel IPP chamado IPPICV, que acelera o OpenCV por padrão (KAEHLER; BRADSKI, 2017).

1.9 *Software* para Processamento de Imagens

1.9.1 Python

Python é uma linguagem de programação que foi criada pelo matemático e programador holandês Guido van Rossum em 1990. Python é uma linguagem de programação de propósito geral, ou seja, suporta todos os paradigmas de programação (estrutural, imperativo, funcional e orientado a objetos) (LIANG, 2013), isso implica que a linguagem Python pode ser utilizada para resolver uma grande gama de problemas.

Python é também uma linguagem interpretada, o que significa que o código escrito em Python é traduzido e então executado por um programa interpretador chamado CPython, uma instrução por vez.

Python é atualmente uma das linguagens de programação mais utilizadas no mundo, isso devido não só a robustez da linguagem mais também pelo fato ser uma linguagem com uma sintaxe clara, concisa e de fácil entendimento, adiciona-se também ao fato de que o Python é uma linguagem *open source*, logo, pode ser utilizada por qualquer pessoa sem a necessidade de obtenção de licenças para uso, assim milhares de desenvolvedores ao redor do mundo podem contribuir com novas *features* para a linguagem. Ao contrário do MATLAB (linguagem muito usada para aplicações de cunho científico) que está sob licença proprietária.

O Python também facilita a implementação de aplicações em qualquer área, assim é muito usada para a criação e novos recursos. Em relação a área de processamento de imagens a linguagem Python dispõe de várias bibliotecas dedicadas como Numpy, OpenCV entre outras. Essa linguagem ainda traz suporte para análise de dados, plotagem de gráficos, criação de interfaces, aprendizado de máquina e tantos outros.

Assim a utilização de Python para aplicações relacionadas a análise de imagens é válida não deixando em nada a desejar se comparada com o MATLAB que é uma linguagem bastante utilizada para implementar aplicações nessa área.

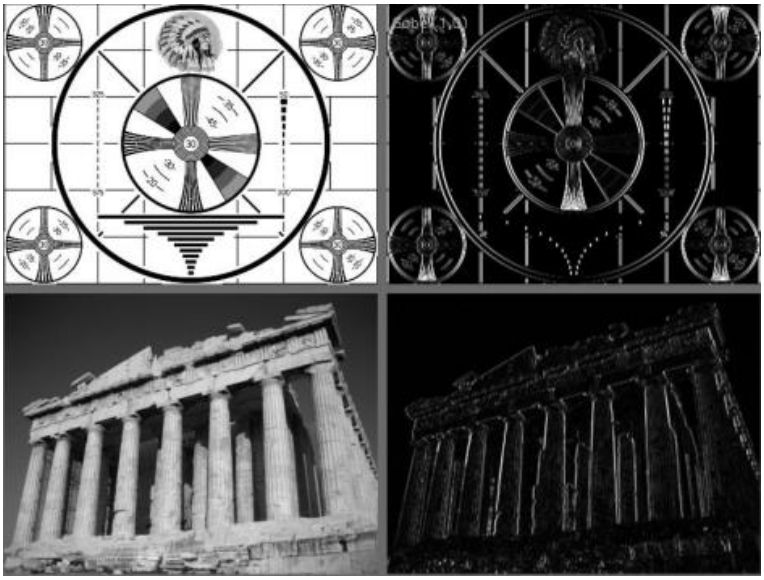
1.9.2 OpenCV

Open Source Computer Vision Library (OpenCV) é uma biblioteca *open source* de visão computacional que teve o início de seu desenvolvimento em 1999 através de Gary Bradsky na Intel Comporation. O OpeCV foi criado com o objetivo de acelerar tarefas de visão computacional e inteligência artificial (KAEHLER; BRADSKI, 2017).

Toda biblioteca OpenCV foi escrita em C/C++ e pode ser utilizada em Linux, Windowns e Mac OS, tendo suporte para outras linguagens como Python, Java entre outras.

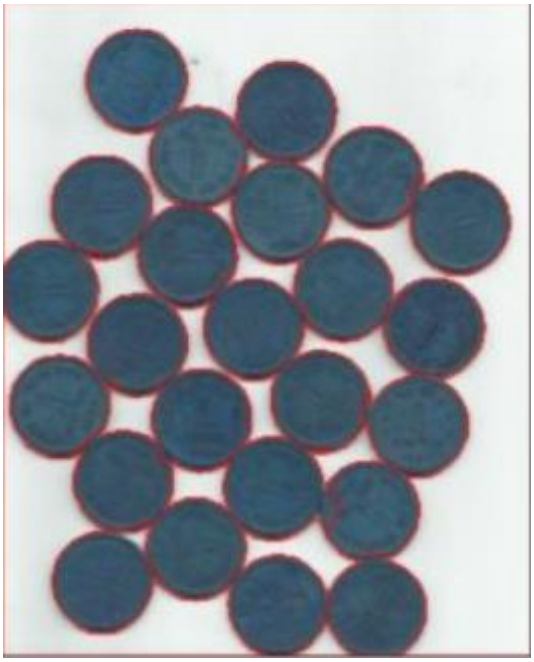
Nas Figuras 12 e 13 abaixo são mostrados resultados da utilização da biblioteca OpenCV para algumas tarefas como detecção de bordas, segmentação.

Figura 12: Detecção de bordas utilizando filtro Sobel através do OpenCV



Fonte: (KAEHLER; BRADSKI, 2017)

Figura 13: Segmentação utilizando o Algoritmo de Watershed com o OpenCV



Fonte: (IMAGE..., 2019)

2 METODOLOGIA

O Trabalho apresentado foi uma Pesquisa Aplicada, e teve como objetivo a realização de Pesquisa Exploratória sobre o material bibliográfico adquirido sobre o assunto. O procedimento técnico de pesquisa bibliográfica foi utilizado. A abordagem foi realizada através do método hipotético-dedutivo e o método de elaboração foi o monográfico. Para coleta de dados foi utilizada documentação indireta, com auxílio de documentos primários e secundários, e a análise e interpretação de seus dados foi realizada de forma quantitativa.

As pesquisas foram realizadas principalmente nas áreas de Processamento Digital de Imagens e Linguagem de Programação em Python, já a parte de implementação do projeto foi realizada em um computador equipado com sistema operacional *Windows 10*, processador *Intel® CoreTM i5-3210M 2.5GHz* e 8Gb de memória *RAM*. Para a avaliação do projeto foram utilizados 2 arquivos de vídeo em formato *.avi* além da linguagem de programação Python.

A maior parte da pesquisa foi realizada na área de Processamento Digital de Imagens, mais especificamente na parte de segmentação de imagens. Outra área muito importante na pesquisa é a área de Estatística, particularmente os métodos estatísticos não paramétricos. E por fim também foi realizada pesquisa sobre a linguagem de programação Python, no que diz respeito as bibliotecas voltadas para a área abordada. Na parte de programação é válido frisar a utilização da biblioteca para visão computacional OpenCV.

O desenvolvimento do projeto iniciou-se a partir da aquisição dos arquivos de vídeo, que feita por meio de uma base de dados, *site* disponibilizada gratuitamente. Os arquivos então tiveram seus nomes modificados com fim de facilitar sua utilização. Então foi iniciado o desenvolvimento do algoritmo em Python. O sistema é composto por três programas.

Após a separação dos vídeos em *frames* é então feita a subtração do fundo e a detecção das regiões de sombra. Por fim ao sistema ainda remove as sombras e reconstrói a imagem sem a presença de sombras.

No que tange à validação dos resultados obtidos, foi feita a geração da Matriz de Confusão. A matriz de confusão é uma forma bastante eficiente de mensurar o grau de desempenho da classificação, nela reunimos dois conjuntos de dados, um deles gerado pelo classificador e outro classificado de forma manual (BONNIN, 2017). Os dados são comparados e colocados dentro da matriz e partir daí pode-se extrair o número de Falsos Positivos, Falsos Negativos, Verdadeiros Positivos e Verdadeiros Negativos. Esses valores são importantes para calcular as métricas de avaliação apresentadas na seção 4.1.

2.1 Etapas de Desenvolvimento

2.1.1 Programa Principal

O programa principal foi nomeado de *main.py* e está no apêndice A. Primeiramente definiu-se uma variável chamada video_or_images_path ela é responsável por armazenar o nome do arquivo de vídeo ou o caminho caso o arquivo esteja em outro diretório. Após isso temos a definição da variável background_subtractor, essa variável recebe como valor o retorno do método cv2.createBackgroundSubtractorKNN, método que cria um subtrator de fundo levando em conta o modelo KNN, esse método possui um parâmetro booleano opcional chamado detectShadows que pode então receber valor verdadeiro ou falso. Esse parâmetro habilita ou não a detecção de sombras nos frames, no caso seu valor foi colocado como *True*.

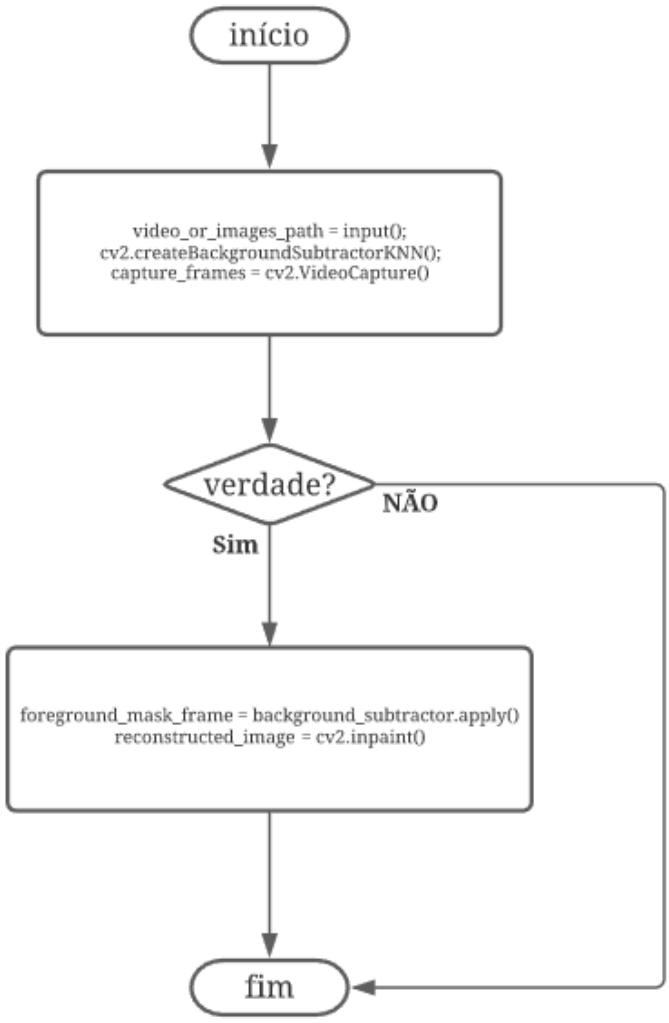
Para definir o número de vizinhos próximos utilizado usou-se o método setkNNSamples, que recebe como parâmetro um número inteiro, foi utilizado 5 como valor no projeto.

Os quadros das imagens foram capturados com o auxílio do método cv2.VideoCapture, os quadros então são armazenados na variável capture_frames.

O núcleo do programa encontrasse dento de um laço *while* que é executado até que todos os quadros do arquivo de vídeo sejam processados. O método apply então é usado para gerar as máscaras binárias, que são imagens segmentadas que indicam as regiões que representam o fundo, o primeiro plano e as regiões de sombra. Esse método recebe como parâmetro quadros do arquivo de vídeo, todas as máscaras geradas são então salvas em um diretório a parte.

Para remover as sombras dos quadros, são criadas máscaras onde todos os pixels que pertencem as regiões do fundo e primeiro plano tem seu valor mudado para zero, deixando apenas as regiões de sombra demarcadas. Então aplicamos a função cv2.inpaint para reconstruir as regiões de sombra, e por fim os quadros sem as sombras são salvos em um diretório a parte. Na Figura 14 está um fluxograma resumido do programa principal.

Figura 14: Fluxograma do programa principal



Fonte: Elaborada pelo autor

2.1.2 Módulo de Quantificação dos Resultados

Para a validação dos resultados obtidos foi escrito um segundo programa chamado *Validation.py* em Python onde foram implementadas as métricas de validação utilizadas: precisão, revocação e Medida F.

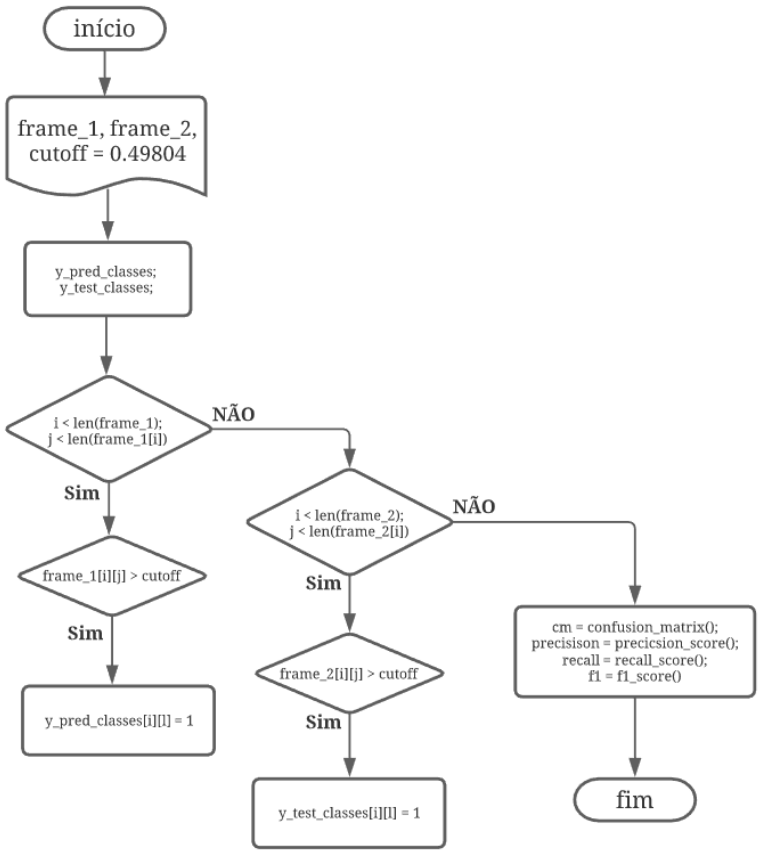
O programa então tem como entrada dois arquivos frame1 e frame2, ambos os arquivos são imagens binárias em que também estão segmentadas as áreas de sombra. Essas imagens representam a imagem obtida pelo algoritmo e uma imagem que foi segmentada manualmente e serve como referência. Com essas duas imagens podemos mensurar o quanto o algoritmo teve sucesso em classificar as regiões dos frames que compõe o vídeo.

Após a leitura dos arquivos de imagem são criadas duas matrizes y_pred_classes e y_test_classes a primeira matriz será utilizada para armazenar os valores preditos e a segunda será utilizada para armazenar os valores detectados. Descobriu-se por meio da observação da forma matricial das imagens que os valores que representam regiões de sombra na imagem possuem um valor de 0.49804, esse valor foi então atribuído a uma variável chamada cutoff.

Por meio de dois laços for salvamos nas matrizes y_pred_classes e y_test_classes com valor 1 as regiões se sombra e com 0 as regiões de fundo e primeiro plano. Com isso temos duas matrizes apenas formadas por zeros e uns, esse procedimento é necessário pelo fato de que as métricas utilizadas para a validação do algoritmo de classificação necessitam de que as amostras estudadas sejam compostas apenas de zeros e uns.

Por fim são calculadas as métricas de validação. Utilizando-se da biblioteca sklearn que é voltada para análise estatística gerou-se então a matriz de confusão por meio da função confusion_matrix, com a matriz de confusão em mãos foi possível calcular a precisão, revocação e Medida F por meio das funções precision_score, recall_score e f1_score respectivamente. A Figura 15 apresenta o fluxograma do algoritmo de validação dos resultados.

Figura 15: Fluxograma do programa de validação de resultados



Fonte: Elaborada pelo autor

3 IMPLEMENTAÇÃO

3.1 Obtenção dos Arquivos de Imagem e Vídeo

Os arquivos de vídeo e imagens foram de forma gratuita obtidos no endereço *http://arma.sourceforge.net/shadows/*, é possível fazer o download tanto dos vídeos como da sequência de frames. Existem diversos tipos de vídeos tanto em ambientes externos como internos. Para o projeto foram selecionados dois arquivos de vídeo que mostram ambientes internos, uma cena que se passa em um laboratório e uma cena que se passa em uma sala. Os nomes dos arquivos são os seguintes:

a) *aton_lab.AVI* (laboratório);

b) *aton_room.AVI* (sala)

Estes dois arquivos tiveram seus nomes alterados para sample_1 e sample_2 respectivamente essa mudança foi feita para padronizá-los para melhor manipulação por parte do programa, como foi dito anteriormente.

3.2 Programa Principal: Parte de Subtração de Fundo

Para realizar a subtração de fundo foi utilizada a biblioteca OpenCV, mais precisamente a classe cv::BackgroundSubtractorKNN onde é implementada o método KNN, com isso criou-se o modelo para a subtração do fundo do *frame* através do método cv2.createBackgroundSubtractorKNN(). A subtração de fundo é propriamente executada dentro do laço *while* com a ajuda do método apply(), após esse passo a máscara binária gerada foi então salva em um diretório específico.

3.3 Programa Principal: Obtenção dos Frames

Os vídeos utilizados tiveram seus quadros separados para manipulação dentro do programa, isso foi realizado por meio do método read() da biblioteca OpenCV. Com os quadros já salvos foi utilizado a função apply() em cada quadro do vídeo, ela então gerou como saída uma imagem em preto e branco e cinza, onde a cor preta indica o fundo, a cor branca indica o primeiro plano e a cor cinza indica as regiões de sombra.

3.4 Programa Principal: Remoção das Sombras

Para a realização da remoção das sombras das imagens foram gerados frames auxiliares, nestes frames as áreas em branco que representam o primeiro plano da imagem tiveram o valor dos seus pixels alterados para zero, dessa forma tanto fundo como primeiro plano estavam representados pela cor preta na imagem resultante, enquanto que as regiões de sombra

permaneceram com a cor cinza. Após esse passo aplicamos a função cv2.cvtColor() e geramos novas imagens binárias, mais desta vez as regiões de sombra estão marcadas com a cor branca, enquanto o fundo e o primeiro plano estão com a cor preta.

A remoção das sombras é feita com o auxílio da função cv2.inpaint(). Essa função é aplicada em todos os quadros gerados, removendo as regiões de sombra e em seguida reconstruindo-as, gerando assim imagens livres de sombra.

3.5 Módulo de Quantificação dos Resultados: Geração da Matriz de Confusão

A parte de avaliação de desempenho na classificação das imagens no que diz respeito a detecção de sombras e feita em outro programa chamado de *Validation.py*. Nesse módulo duas imagens são carregadas, uma é a máscara binária gerada pelo programa principal e a outra é uma imagem binária segmentada previamente e é utilizada como referência para a análise do quão bom foi o algoritmo em classificar os quadros.

Para a geração da matriz de confusão os dados de entrada necessitam ser apenas 0 ou 1, para contornar esse problema foi criada duas matrizes e nelas foram armazenados os valores dos pixels tanto da imagem gerada pelo programa principal tanto da imagem de referência, entretanto nessas duas matrizes os pixels de representam o fundo e o primeiro plano são salvos com valor 1 e as regiões de sombra com valor 0. Com isso duas matrizes compostas de zeros foram geradas e então foi utilizada as funções confusion_matrix(), precision_score(), recall_score() e f1_score() para a geração da matriz de confusão e o cálculo da precisão, revocação e medida F respectivamente.

4 VALIDAÇÃO DOS RESULTADOS

4.1 Métricas de Avaliação

Para avaliar o desempenho do sistema foram feitos testes com o intuito de obter medidas da taxa de acerto na detecção das sombras, e então apresentar gráficos que mostrem de forma visual a performance do sistema.

Tarefas de classificação requerem o uso de diferentes regras para estimar os erros. É possível determinar exatamente se os resultados estão errados ou não de maneira binária (BONNIN, 2017). Isso nos levas aos principais indicadores: Precisão, Revocação e Medida-F.

Precisão é definida pela Equação 6.

$$precisão = \frac{tp}{tp+fp} \tag{6}$$

Na Equação 6 tp representa o número de verdadeiros positivos e fp representa o número de falsos positivos. A precisão é a habilidade do sistema tem de não classificar como positiva uma amostra negativa (BONNIN, 2017).

Revocação é definida pela Equação 7.

$$revocação = \frac{tp}{tp+fn} \tag{7}$$

Na Equação 7 fn representa o número de falsos negativos. A revocação é a capacidade do sistema em encontrar todas as amostras positivas (BONNIN, 2017).

Medida-F é definida na Equação 8;

$$F = \frac{2(precisão \cdot revocação)}{precisão+revocação} \tag{8}$$

A Medida-F pode ser descrita como um tipo especial de média (uma média harmônica ponderada) dos valores de Precisão e Revocação (BONNIN, 2017).

4.2 Análise dos Resultados

Os arquivos de vídeo utilizados no projeto têm o formato *.AVI* sigla para *Audio Video Interleave* que é um formato criado pela Microsoft. Os quadros obtidos através da função cv2.imwrite são em formato *.png* sigla para *Portable Network Graphics* de dimensões 320x240.

As amostras de imagens utilizadas estão no Apêndice D, foram utilizadas duas amostras de cada cena, para a análise dos resultados. No Apêndice E estão os resultados gerados pela execução do programa de validação dos resultados *Validation.py* para o arquivo de vídeo sample_1. Estes resultados são as métricas apresentadas na seção 4.1, além da imagem reconstruída. Os resultados para o arquivo sample_2 estão no Apêndice F.

Para a análise dos resultados foram retirados os quadros 281 e 291 do arquivo sample_1 e os quadros 175 e 235 do arquivo sample_2, estes quadros foram comparados com seus equivalentes previamente segmentados, ou seja, temos uma imagem gerada pelo classificador e outra imagem de referência onde a segmentação das regiões da imagem foi feita de forma manual.

Como pode ser visto nas Tabelas 2 e 3, os valores das métricas são os mesmos e isso decorre do fato de que o problema de classificação neste projeto é um problema de classificação de múltiplas classes. Nestes casos as métricas, Precisão, Revocação e Medida F, são aplicadas a cada classe de forma independente, se todos os rótulos forem considerados a média "micro" para uma configuração com múltiplas classes irá produzir valores idênticos para as métricas acima citadas (SCIKIT-LEARN, 2020).

Tabela 2: Métricas para o quadro 235 do arquivo sammple_2.AVI

Métrica	**Resultado**
Precisão	0.7625
Revocação	0.7625
Medida-F	0.7625

Fonte: Elaborada pelo autor

Tabela 3: Métricas para o quadro 281 do arquivo sammple_1.AVI

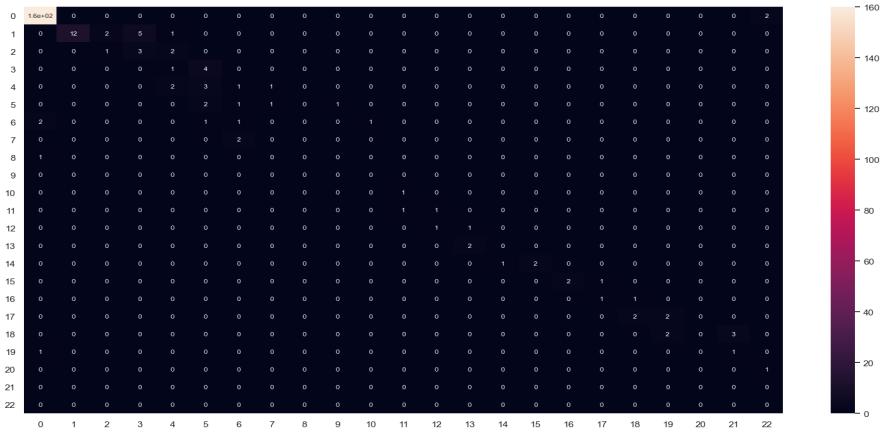
Métrica	**Resultado**
Precisão	0.6291
Revocação	0.6291
Medida-F	0.6291

Fonte: Elaborada pelo autor

De forma mais simples para uma configuração com múltiplas classes sempre que houver um valor falso positivo haverá um valor falso negativo e vice-versa, ou seja, o número de falsos negativos e falsos positivos é sempre igual quando se utiliza a noção de média micro para o cálculo das métricas (HESSNER, 2018). Na Figura 16 é mostrada a matriz de confusão para o quadro 235 do arquivo sample_2.AVI, nela pode-se observar a quantidade de classes nas quais os pixels podem ser classificados para esse quadro têm-se o valor de 23 classes diferentes, já

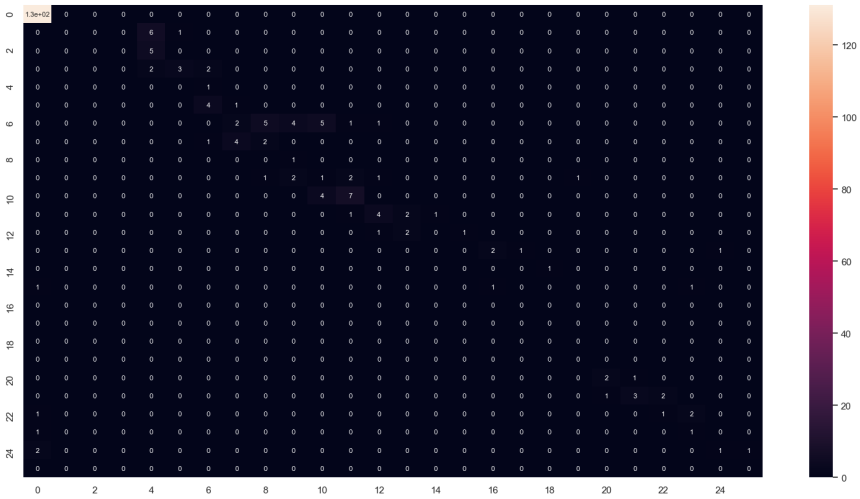
para o quadro 281 do arquivo sammple_1.AVI, cuja matriz de confusão é apresentada na Figura 17, existem 25 classes.

Figura 16: Matriz de confusão para o quadro 235 do arquivo sample_2.AVI



Fonte: Elaborada pelo autor

Figura 17: Matriz de confusão para o quadro 281 do arquivo sample_1.AVI



Fonte: Elaborada pelo autor

Nos resultados apresentados observa-se que os valores das métricas variam mesmo para quadros extraídos do mesmo arquivo. Esses valores podem alterar se for modificado o número de vizinhos que são levados em conta no processo de classificação do Modelo KNN. Outro fator que interfere na diferença nos valores das métricas é a intensidade das regiões de sombra e também nível de ruído.

No que diz respeito a fase de remoção das sombras foi utilizada a função cv2.inpaint, com a finalidade de reparar imagens degradadas pelo tempo. Basicamente funciona refazendo

as partes da imagem que estão degradas fazendo uso dos pixels vizinhos que se parecem com a área danificada.

Para a tarefa de remoção das sombras passamos o parâmetro cv2.INPAINT_NS a função. Esse parâmetro indica que para realizar a reconstrução das áreas afetadas pelas sombras o algoritmo proposto por (BERTALMIO; BERTOZZI; SAPIRO, 2001) será utilizado, além desse método têm-se a opção de utilizar o algoritmo proposto por (TELEA, 2004).

Figura 18: Reconstrução do quadro 235 do arquivo sample_2.AVI

Fonte: Elaborada pelo autor

Figura 19: Reconstrução do quadro 281 do arquivo sample_1.AVI

Fonte: Elaborada pelo autor

Como pode bem ser visto nas Figuras 18 e 19, o programa foi capaz de remover as regiões de sombra e reconstruí-las. Como a reconstrução faz uso dos pixels mais próximos para

as regiões de sombra pode ocorrer o caso em que as regiões vizinhas representem objetos de cor mais escura o que tende a tornar a região reconstruída também mais escura.

Os valores para as métricas obtidos foram em torno dos 63% para os quadros referentes ao arquivo sample_1.AVI e chegando a 85%, para quadros do arquivo sample_2.AVI. Para fim de comparação com a bibliografia existente, no artigo apresentado por (PRATI; MIKIC; TRIVEDI et. al, 2003) é mostrado uma comparação entre alguns métodos de detecção de sombras. Entre eles o método desenvolvido por (HORPRASET; HARWOOD; DAVIS, 1999), que é um método estatístico não paramétrico mais complexo que o apresentado neste trabalho. Na comparação os autores utilizam uma métrica chamada por eles de Taxa de Detecção de Sombra que é nada mais que a medida de Revocação apresentada na Equação 6. Os arquivos utilizados por (PRATI; MIKIC; TRIVEDI et. al) são os mesmos utilizados no presente trabalho e seus resultados para a medida de Revocação foi de 84% para o arquivo correspondente ao sample_1.AVI e de 76% para o arquivo correspondente ao sample_2.AVI.

CONCLUSÃO

O desenvolvimento do projeto requereu uma pesquisa a respeito de um subtópico da área de Processamento Digital de Imagens que é a parte de detecção de sombras em imagens. Para tanto foi necessário revisão em assuntos chave como: segmentação de imagens, estatística e linguagem de programação assuntos estes que estão desenvolvidos no referencial teórico do presente trabalho.

O algoritmo utilizado para a detecção de sombras foi o algoritmo KNN, um método que utiliza uma abordagem estatística não paramétrica para realizar o processo de segmentação da imagem. Com o uso desse modelo foi possível segmentar as sequências de quadros extraídas de um vídeo em regiões que correspondem ao fundo, primeiro plano e regiões de sombra, essas tarefas foram possíveis através da utilização da biblioteca OpenCV.

Depois da obtenção das imagens segmentadas foi aplicado o processo de remoção das regiões de sombra além da reconstrução das áreas afetadas pelas mesmas, obtendo assim imagens livres de sombras.

Para mensurar o desempenho do sistema no que diz respeito a detecção das regiões de sombra, foi realizado o cálculo da matriz de confusão de onde se pode extrair as métricas apresentadas no Capítulo 4.

Os resultados obtidos pelo sistema estão no Apêndices E e F. Para a geração dos resultados foi utilizada como referência imagens que foram segmentadas manualmente, assim foi possível medir o quão bom foi o algoritmo em detectar as regiões de sombra, fazendo uso da matriz de confusão.

Como é de se observar o sistema conseguiu detectar boa parte das regiões de sombras presentes nas imagens.

No que tange à obtenção da matriz de confusão dos quadros pode-se notar que a quantidade de classes que os pixels são classificados é grande e varia conforme o quadro analisado, por isso o fato das matrizes de confusão geradas terem muitas classes.

Acerca da hipótese apresentada na introdução do presente trabalho, pode-se concluir que é possível desenvolver um sistema para detectar e remover sombras baseado em uma abordagem não paramétrica, voltado para ambientes internos, e mesmo em ambientes internos o desempenho do sistema varia conforme as peculiaridades da cena em questão.

Portanto, como sugestão de melhoria do sistema, pode-se buscar outros métodos não paramétricos, além do fato de que existem muitas linhas de pesquisa utilizado outras abordagens estatísticas e até mesmo técnicas de Aprendizado de Máquina como: Redes Neurais, Redes Generativas entre outras.

REFERÊNCIAS BIBLIOGRÁFICAS

AJANKI, Antti. https://commons.wikimedia.org. ***Example of K-Nearest Neighbour Classification***, 2007. Disponível em: https://commons.wikimedia.org/wiki/File:KnnClassification.svg. Acesso em: 22 set. 2020.

AMATO, Ariel. et al. ***Moving Cast Shadows Detection Methods for Video Surveillance Applications***, *2014*. Disponível em: https://www.researchgate.net/publication/283503034_Moving_Cast_Shadows_Detection_Methods_for_Video_Surveillance_Applications. Acesso: 03 nov. 2019.

BERTALMIO, Marcelo.; BERTOZZI, Andrea L.; SAPIRO, Guillermo. ***Navier-stokes, fluid dynamics, and image and video inpainting***, 2001. Disponível em: https://ieeexplore.ieee.org/document/990497. Acesso em: 15 set. 2020.

BONNIN, Rodolfo. ***Machine Learning for Developers: Uplift your regular applications with the power of statistics, analytics and machine learning***. 1.ed. Birmingham: Packt Publishing, 2017.

CHAUHAN, Nagesh S. https://towardsdatascience.com/. ***Introduction to Image Segmentation with K-Means clustering***, 2019. Disponível em: https://towardsdatascience.com/introduction-to-image-segmentation-with-k-means-clustering-83fd0a9e2fc3. Acesso em: 18 nov. 2019.

CHONDAGAR, Vijay. et al. ***A Review: Shadow Detection and Removal***, 2015. Disponível em: https://www.researchgate.net/publication/296396435_A_Review_Shadow_Detection_and_Removal. Acesso: 14 set. 2019.

CORDER, Gregory W.; FOREMAN, Dale I. ***Nonparametric Statistics for Non-Statisticians: A Step-by-Step Approach***. 1.ed. New Jersey: Wiley, 2009.

DAS.; Rakesh K. et al. ***A Survey on Shadow Detection and Removal in Images***, 2017. Disponível em: https://ieeexplore.ieee.org/document/8378149. Acesso: 08 set. 2019.

ECINS, Aleksandrs.; FERMÜLLER, Cornelia.; ALOIMONOS, Yiannis. ***Shadow free segmentation in still images using local density measure***, 2014. Disponível em: https://ieeexplore.ieee.org/document/6831803. Acesso: 12 nov. 2019.

ELGAMMAL, Ahmed.; HARWOOD, David.; DAVIS, Larry. ***Non-parametric Model for Background Subtraction***, 2000. Disponível em: https://www.researchgate.net/publication/243778086_Non-parametric_model_for_background_subtraction. Acesso em: 08 set. 2019.

FURHT, Borko.; AKAR, Esad.; ANDREWS, Whitney A. ***Digital Image Processing: Practical Approach***. 1.ed. Berlim: Springer, 2018.

GONZALEZ, Rafael C.; WOODS, Richard E. ***Processamento Digital de Imagens***. 3.ed. São Paulo: Pearson, 2010.

HASTIE, Trevor.; TIBSHIRANI, Robert.; FRIEDMAN, Jerome. ***The Elements of Learning: Data Mining, Inference, and Prediction***. 2. ed. New York: Springer, 2009.

HESSNER, Simon. ***Why are precision, recall and F1 score equal when using micro averaging in a multi-class problem?***, 2018. Disponível em: https://simonhessner.de/why-are-precision-recall-and-f1-score-equal-when-using-micro-averaging-in-a-multi-class-problem/. Acesso em: 03 out. 2020.

HORPRASET, Thanara.; HARWOOD, David.; DAVIS, Larry S. ***A Statistical Approach for Real-time Robust Background Subtraction and Shadow Detection***, 1999. Disponível em: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwiE69qes5XtAhWCCrkGHcBHDk4QFjABegQIAhAC&url=https%3A%2F%2Fwww.researchgate.net%2Fpublication%2F228609042_A_statistical_approach_for_real-time_robust_background_subtraction_and_shadow_detection&usg=AOvVaw3H8_wOcEiZbLNIz6PZHQRC. Acesso em: 02 set. 2019.

IMAGE Segmentation with Watershed Algorithm, 2019. https://docs.opencv.org. Disponível em: https://docs.opencv.org/3.4/d3/db4/tutorial_py_watershed.html. Acesso em: 12 nov. 2019.

JAIN, Anil K. ***Fundamentals of Digital Image Processing***. 1.ed. Englewood Cliffs: Prentice Hall, 1989.

KAEHLER, Adrian; BRADSKI, Gary. ***Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library***. 1.ed. Sebastopol: O'Reilly, 2017.

KIM, Cheong G.; KIM Jeom G.; LEE, Do H. ***Optimizing image processing on multi-core CPUs with Intel parallel programming Technologies***, 2011 Disponível em: https://link.springer.com/article/10.1007/s11042-011-0906-y. Acesso em: 16 nov. 2019.

LIANG, Daniel. ***Introduction to Programming Using Python***. 1.ed. Upper Saddle River: Pearson, 2013.

PETERS, James F. ***Foundations of Computer Vision: Computational Geometry, Visual Image Structures and Object Shape Detection***. 1.ed. Berlim: Springer, 2017.

PETROU, Maria.; PETROU Costas. ***Image Processing: The Fundamentals***. 2.ed. Hoboken: Wiley, 2010.

PRATI, Andrea.; MIKIC, Ivana.; TRIVEDI, Mohan M. et. al. ***Detecting Moving Shadows: Algorithms and Evaluation***, 2003. Disponível em: https://ieeexplore.ieee.org/document/1206520. Acesso em: 02 set. 2019.

PRINCE, Simon J. D. ***Computer Vision: Models, Learning and Inference***. 1.ed. Cambridge: Cambridge University Press, 2012.

RESOURCE Files 2020. http://arma.sourceforge.net/shadows/. Disponível em: http://cvrr.ucsd.edu/aton/shadow/. Acesso em: 20 jan, 2020.

SINGH, Namrita.; MAXTON A. A. ***A Survey on Shadow Detection Methods***, 2014. Disponível em: http://ijarcet.org/wp-content/uploads/IJARCET-VOL-3-ISSUE-4-1220-1224.pdf. Acesso: 14 set. 2019.

SIVASHANMUGAM, Kaarthik. ***Special Purpose Hardware for Image Processing***, 2008. Disponível em: https://www.researchgate.net/publication/241536792_SPECIAL_PURPOSE_HARDWARE_FOR_IMAGE_PROCESSING. Acesso: 16 nov. 2019.

SCIKIT-LEARN Documentation. 2020. https://scikit-learn.org/stable/user_guide.html. Disponível em: https://scikit-learn.org/stable/modules/model_evaluation.html#multiclass-and-multilabel-classification. Acesso em: 25 set. 2020.

STRANG, Senior A. J. https://commons.wikimedia.org. ***The Aurora Borealis, or Northern Lights, shines above Bear Lake***, 2005. Disponível em: https://commons.wikimedia.org/wiki/File:Polarlicht_2_kmeans_16_large.png. Acesso: 04 nov. 2019.

TELEA, Alexandru. ***An Image Inpainting Technique Based on the Fast Marching Method***, 2004. Disponível em: https://www.researchgate.net/publication/238183352_An_Image_Inpainting_Technique_Based_on_the_Fast_Marching_Method. Acesso em 15 set. 2020.

TOCCI, Ronald J.; WIDMER, Neal S.; MOSS, Gregory L. ***Sistemas Digitais: Princípios e Aplicações***. 11.ed. São Paulo: Pearson, 2011.

TYAGI, ***Vipin. Understand Digital Image Processing***. 1.ed. Boca Raton: CRC Press, 2018.

ZIVKOVIC, Zoran.; HEIJDEN, Ferdinand van der Heijden. ***Efficient adaptive density estimation per image pixel for the task of background subtraction***, 2006. Disponível em: https://www.sciencedirect.com/science/article/abs/pii/S0167865505003521. Acesso em: 10 ago. 2020.

APÊNDICE A – PROGRAMA PRINCIPAL

```
import cv2
import argparse
import numpy
import Utils
import time

Utils.clear_files(r'C:\Users\cassio\PycharmProjects\FinalProject\Frames\ima
ge_frames')
Utils.clear_files(r'C:\Users\cassio\PycharmProjects\FinalProject\Frames\mas
k_frames')
Utils.get_frames('sample_1.AVI',
r'C:\Users\cassio\PycharmProjects\FinalProject\Frames\image_frames')

video_or_images_path = input()

parser = argparse.ArgumentParser(description='Program that implements a
background subtraction using the KNN method')
parser.add_argument('--input', type=str, help='Path to a video or a
sequence of image',
                    default=f'{video_or_images_path}')
parser.add_argument('--algo', type=str, help='Background subtraction method
KNN', default='KNN')
args = parser.parse_args()

background_subtractor =
cv2.createBackgroundSubtractorKNN(detectShadows=True)
background_subtractor.setkNNSamples(7)

capture_frames = cv2.VideoCapture(cv2.samples.findFileOrKeep(args.input))

counter = 0

if not capture_frames.isOpened:
    print('Unable to open: ' + args.input)
    exit(0)

while True:
    _, frame = capture_frames.read()
    if frame is None:
        break

    foreground_mask_frame = background_subtractor.apply(frame)
    Utils.save_frame(foreground_mask_frame,
r'C:\Users\cassio\PycharmProjects\FinalProject\Frames\mask_frames',
counter)

Utils.apply_filter(r'C:\Users\cassio\PycharmProjects\FinalProject\Frames\ma
sk_frames',
r'C:\Users\cassio\PycharmProjects\FinalProject\Frames\filtered_mask',
counter)

    auxiliary_mask = cv2.imread(

Utils.path_name(r'C:\Users\cassio\PycharmProjects\FinalProject\Frames\mask_
frames', counter))

    white_pixels = numpy.where(
        (auxiliary_mask[:, :, 0] == 255) & (auxiliary_mask[:, :, 1] == 255)
& (auxiliary_mask[:, :, 2] == 255))
```

```
    auxiliary_mask[white_pixels] = (0, 0, 0)
    auxiliary_mask = cv2.cvtColor(auxiliary_mask, cv2.COLOR_BGR2GRAY)
    Utils.save_frame(auxiliary_mask,
r'C:\Users\cassio\PycharmProjects\FinalProject\Frames\modified_mask_frames'
,
                     counter)

    reconstructed_image = cv2.inpaint(frame, auxiliary_mask, 3,
cv2.INPAINT_NS)
    Utils.save_frame(reconstructed_image,
r'C:\Users\cassio\PycharmProjects\FinalProject\Frames\reconstructed-
frames',
                     counter)

    _, threshold_frame = cv2.threshold(foreground_mask_frame, 127, 255,
cv2.THRESH_OTSU)

    counter += 1
    '''
    cv2.imshow('frame', frame)
    cv2.imshow('foreground mask frame', foreground_mask_frame)
    cv2.imshow('modified foreground mask', auxiliary_mask)
    cv2.imshow('threshold frame', threshold_frame)
    cv2.imshow('reconstructed image', reconstructed_image)
    '''
    keyboard = cv2.waitKey(30)
    if keyboard == 'q' or keyboard == 27:
        break
```

APÊNDICE B – PROGRAMA DE VALIDAÇÃO DOS RESULTADOS

```
import numpy as np
from matplotlib.image import imread
from sklearn.metrics import confusion_matrix, precision_score,
recall_score, f1_score
import matplotlib.pyplot as plt
import seaborn as sn
import pandas as pd

# leitura dos arquvos de imagens
frame1 =
imread(r'C:\Users\cassio\PycharmProjects\FinalProject\Frames\mask_frames\im
age235.png')
# frame2 = imread(r'D:\aton_lab\shadows\aton_lab_0281.png')
frame2 = imread(r'D:\aton_room\shadows\aton_room_0235.png')

cutoff = 0.49804    # valor de corte para as sombras

y_pred_classes = np.zeros_like(frame1, dtype=int)   # matriz de valores
pretidos
y_test_classes = np.zeros_like(frame2, dtype=int)   # matriz de valores
detectados

for i in range(len(frame1)):
    for j in range(len(frame1[i])):
        if frame1[i][j] > cutoff:
            y_pred_classes[i][j] = 1

for i in range(len(frame2)):
    for j in range(len(frame2[i])):
        if frame2[i][j] > cutoff:
            y_test_classes[i][j] = 1

cm = confusion_matrix(y_test_classes.argmax(axis=1),
y_pred_classes.argmax(axis=1)) #matriz de confusão

precision = precision_score(y_test_classes.argmax(axis=1),
y_pred_classes.argmax(axis=1), average='micro')  # precisão
recall = recall_score(y_test_classes.argmax(axis=1),
y_pred_classes.argmax(axis=1), average='micro') # revocação
f1 = f1_score(y_test_classes.argmax(axis=1), y_pred_classes.argmax(axis=1),
average='micro') # medida F

print(precision)
print(recall)
print(f1)


df_cm = pd.DataFrame(cm, range(21), range(21))
sn.set(font_scale=1.0)  # for label size
sn.heatmap(df_cm, annot=True, annot_kws={"size": 8})  # font size

plt.show()
```

APÊNDICE C – FUNÇÕES UTILITÁRIAS

```
import cv2
import os


def clear_files(path):
    files = [file for file in os.listdir(path) if file.endswith('.png')]

    for file in files:
        os.remove(os.path.join(path, file))


def get_frames(video, path):
    video = cv2.VideoCapture(video)
    success, image = video.read()
    counter = 0

    while success:
        if counter < 10:
            cv2.imwrite(fr'{path}\frame00{counter}.png', image)  # Save
frame as image file
            success, image = video.read()
            counter += 1
        elif 10 <= counter < 100:
            cv2.imwrite(fr'{path}\frame0{counter}.png', image)  # Save
frame as image file
            success, image = video.read()
            counter += 1
        else:
            cv2.imwrite(fr'{path}\frame{counter}.png', image)  # Save frame
as image file
            success, image = video.read()
            counter += 1

    print('finish')


def save_frame(frame, path, counter):
    if counter < 10:
        cv2.imwrite(fr'{path}\image00{counter}.png', frame)
    elif 10 <= counter < 100:
        cv2.imwrite(fr'{path}\image0{counter}.png', frame)
    else:
        cv2.imwrite(fr'{path}\image{counter}.png', frame)


def path_name(path, counter):
    if counter < 10:
        return fr'{path}\image00{counter}.png'
    elif 10 <= counter < 100:
        return fr'{path}\image0{counter}.png'
    else:
        return fr'{path}\image{counter}.png'


def apply_filter(path_source, path_target, counter):
    if counter < 10:
        mask_frame = cv2.imread(fr'{path_source}\image00{counter}.png')
        blur = cv2.medianBlur(mask_frame, 7)
        filtered_frame = cv2.cvtColor(blur, cv2.COLOR_BGR2GRAY)
```

```
        cv2.imwrite(fr'{path_target}\image00{counter}.png', filtered_frame)

    elif 10 <= counter < 100:
        mask_frame = cv2.imread(fr'{path_source}\image0{counter}.png')
        blur = cv2.medianBlur(mask_frame, 7)
        filtered_frame = cv2.cvtColor(blur, cv2.COLOR_BGR2GRAY)
        cv2.imwrite(fr'{path_target}\image0{counter}.png', filtered_frame)

    else:
        mask_frame = cv2.imread(fr'{path_source}\image{counter}.png')
        blur = cv2.medianBlur(mask_frame, 7)
        filtered_frame = cv2.cvtColor(blur, cv2.COLOR_BGR2GRAY)
        cv2.imwrite(fr'{path_target}\image{counter}.png', filtered_frame)
```

APÊNDICE D – QUADROS UTILIZADOS

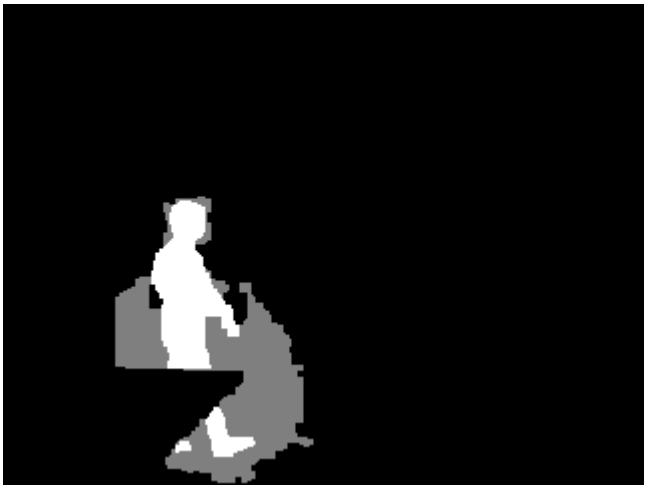
Figura 20: Sample_1 quadro 281

Fonte: Elaborada pelo autor

Figura 21: Sample_1 quadro 291

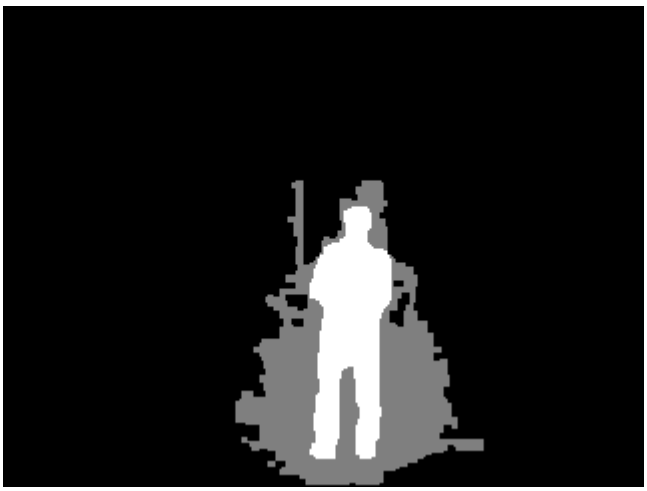
Fonte: Elaborada pelo autor

Figura 22: Segmentação manual: Sample_1 quadro 281



Fonte: http://arma.sourceforge.net/shadows/

Figura 23: Segmentação manual: Sample_1 quadro 301



Fonte: http://arma.sourceforge.net/shadows/

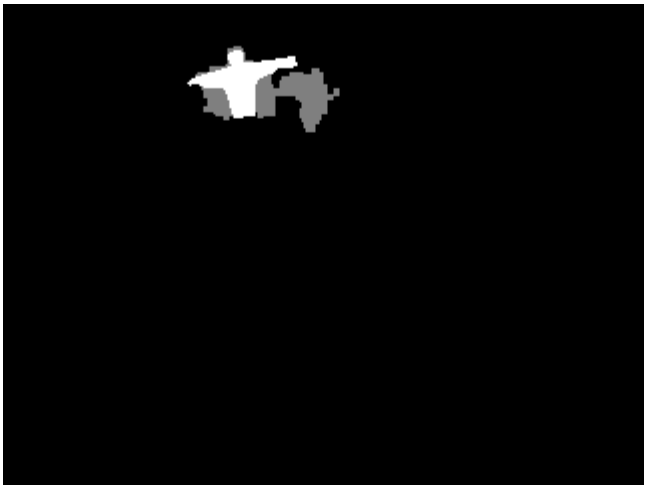
Figura 24: Sample_2 quadro 175

Fonte: Elaborada pelo autor

Figura 25: Sample_2 quadro 235

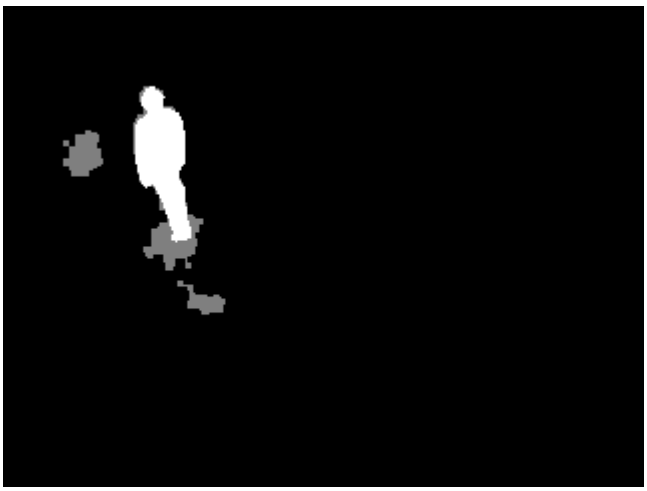
Fonte: Elaborada pelo autor

Figura 26: Segmentação manual: Sample_2: quadro 175



Fonte: http://arma.sourceforge.net/shadows/

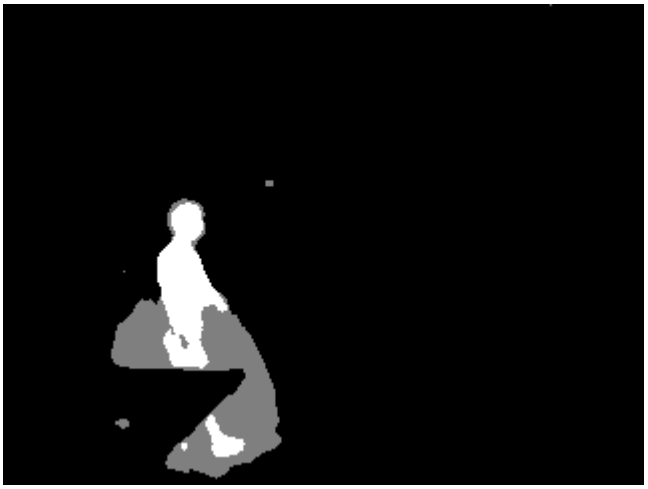
Figura 27: Segmentação manual: Sample_2: quadro 235



Fonte: http://arma.sourceforge.net/shadows/

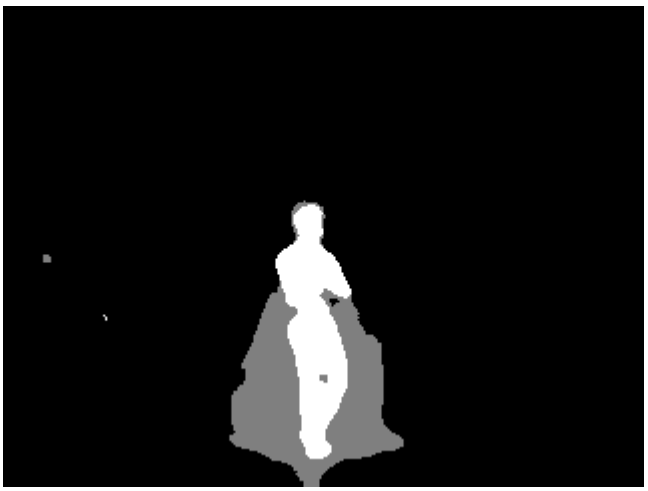
APÊNDICE E – RESULTADOS PARA O ARQUIVO *SAMPLE_1.AVI*

Figura 28: Resultado obtido pelo programa: quadro 281



Fonte: Elaborada pelo autor

Figura 29: Resultado obtido pelo programa: quadro 291



Fonte: Elaborada pelo autor

Figura 30: Resultado da remoção das regiões de sombra: quadro 281

Fonte: Elaborada pelo autor

Figura 31: Resultado da remoção das regiões de sombra: quadro 291

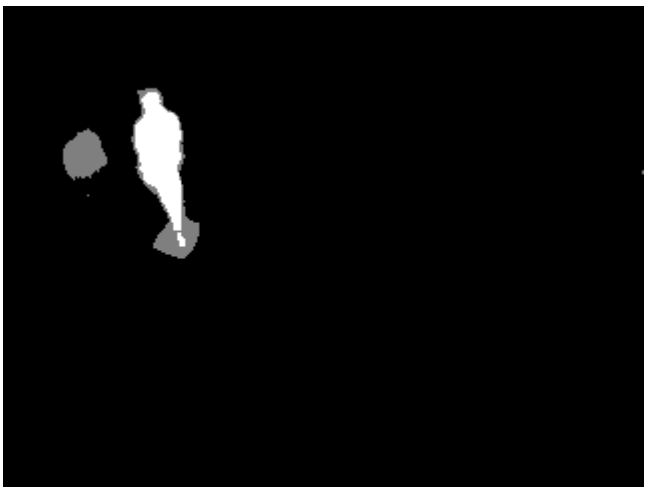
Fonte: Elaborada pelo autor

APÊNDICE F – RESULTADOS PARA O ARQUIVO *SAMPLE_2.AVI*

Figura 32: Resultado obtido pelo programa: quadro 175

Fonte: Elaborada pelo autor

Figura 33: Resultado obtido pelo programa: quadro 235



Fonte: Elaborada pelo autor

Figura 34: Resultado da remoção das regiões de sombra: quadro 175

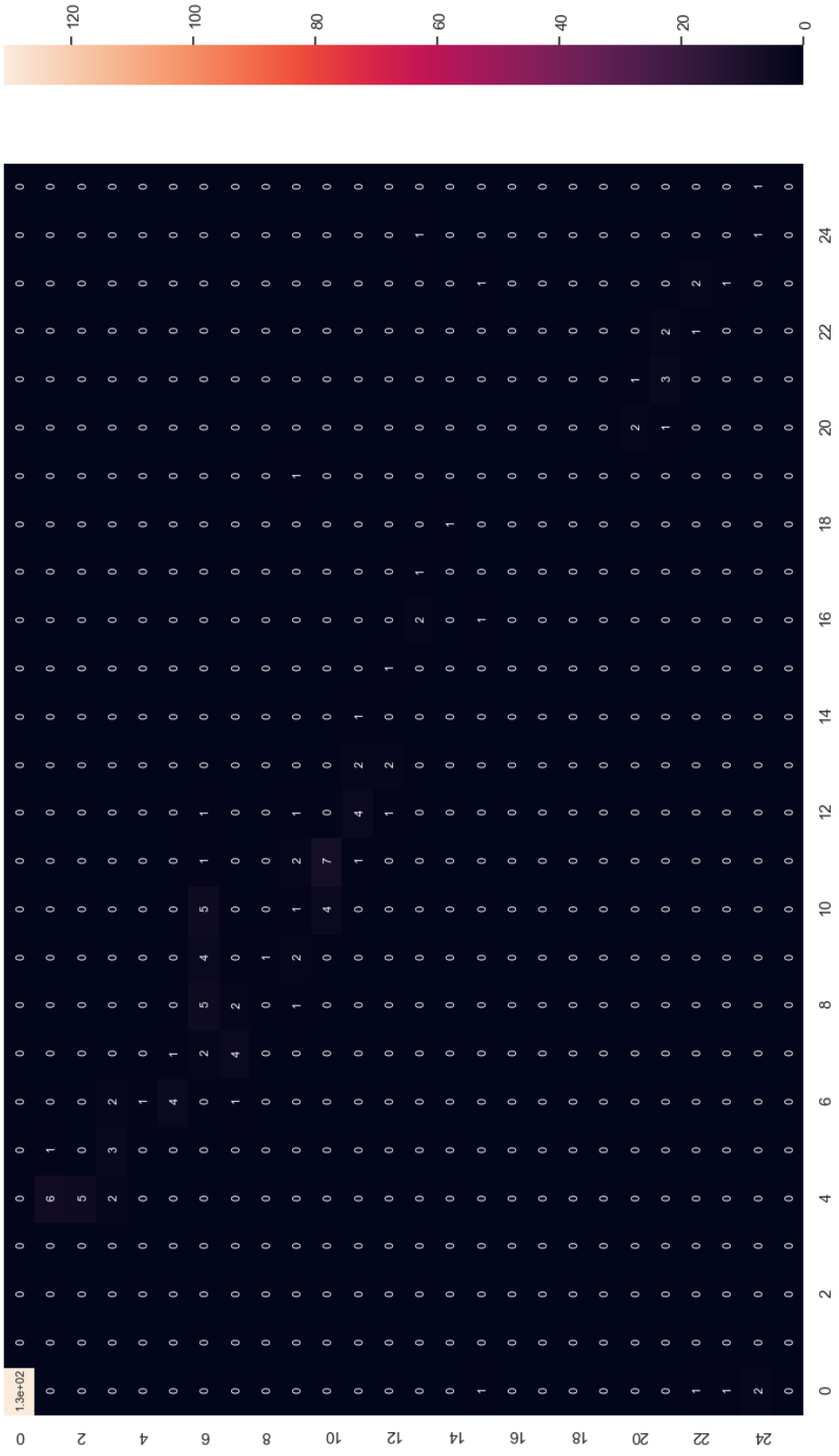
Fonte: Elaborada pelo autor

Figura 35: Resultado da remoção das regiões de sombra: quadro 235

Fonte: Elaborada pelo autor

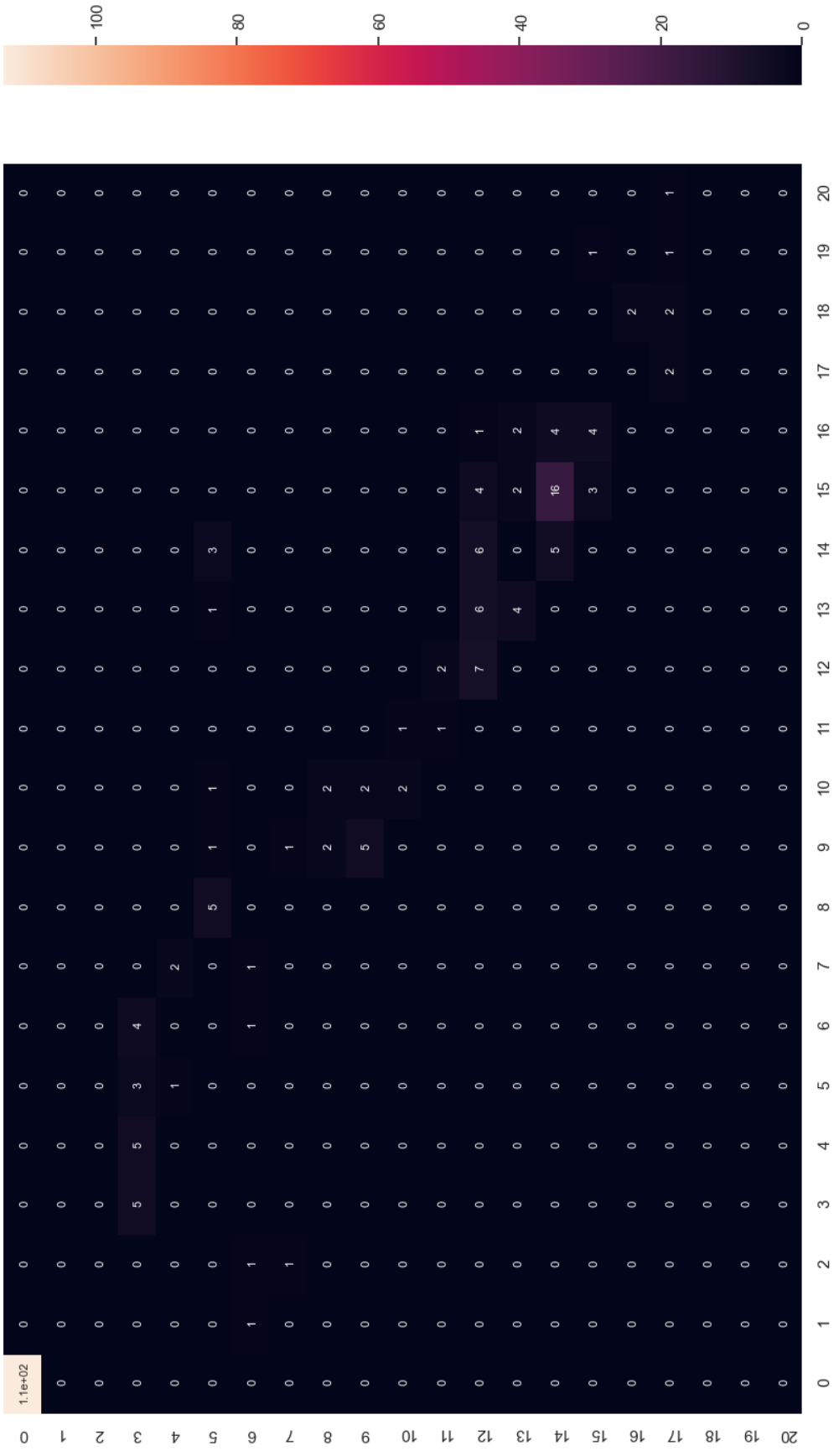
APÊNDICE G – MÉTRICAS PARA O ARQUIVO *SAMPLE_1.AVI*

Figura 36: Matriz de confusão para o quadro 281



Fonte: Elaborada pelo autor

Figura 37: Matriz de confusão para o quadro 291



Fonte: Elaborada pelo autor

Tabela 4: Métricas para o quadro 281

Métrica	**Resultado**
Precisão	0.6291
Revocação	0.6291
Medida-F	0.6291

Fonte: Elaborada pelo autor

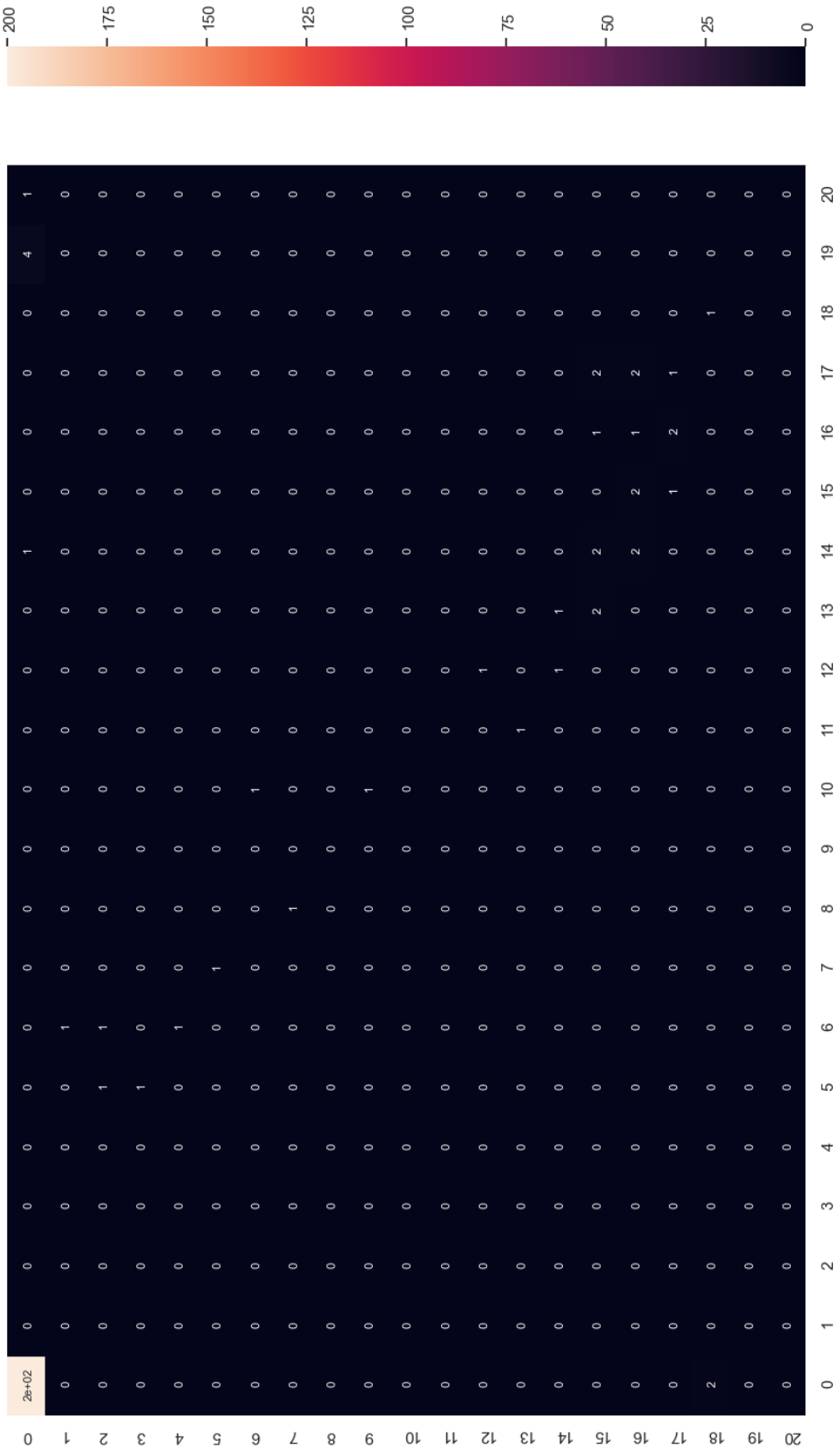
Tabela 5: Métricas para o quadro 291

Métrica	**Resultado**
Precisão	0.6166
Revocação	0.6166
Medida-F	0.6166

Fonte: Elaborada pelo autor

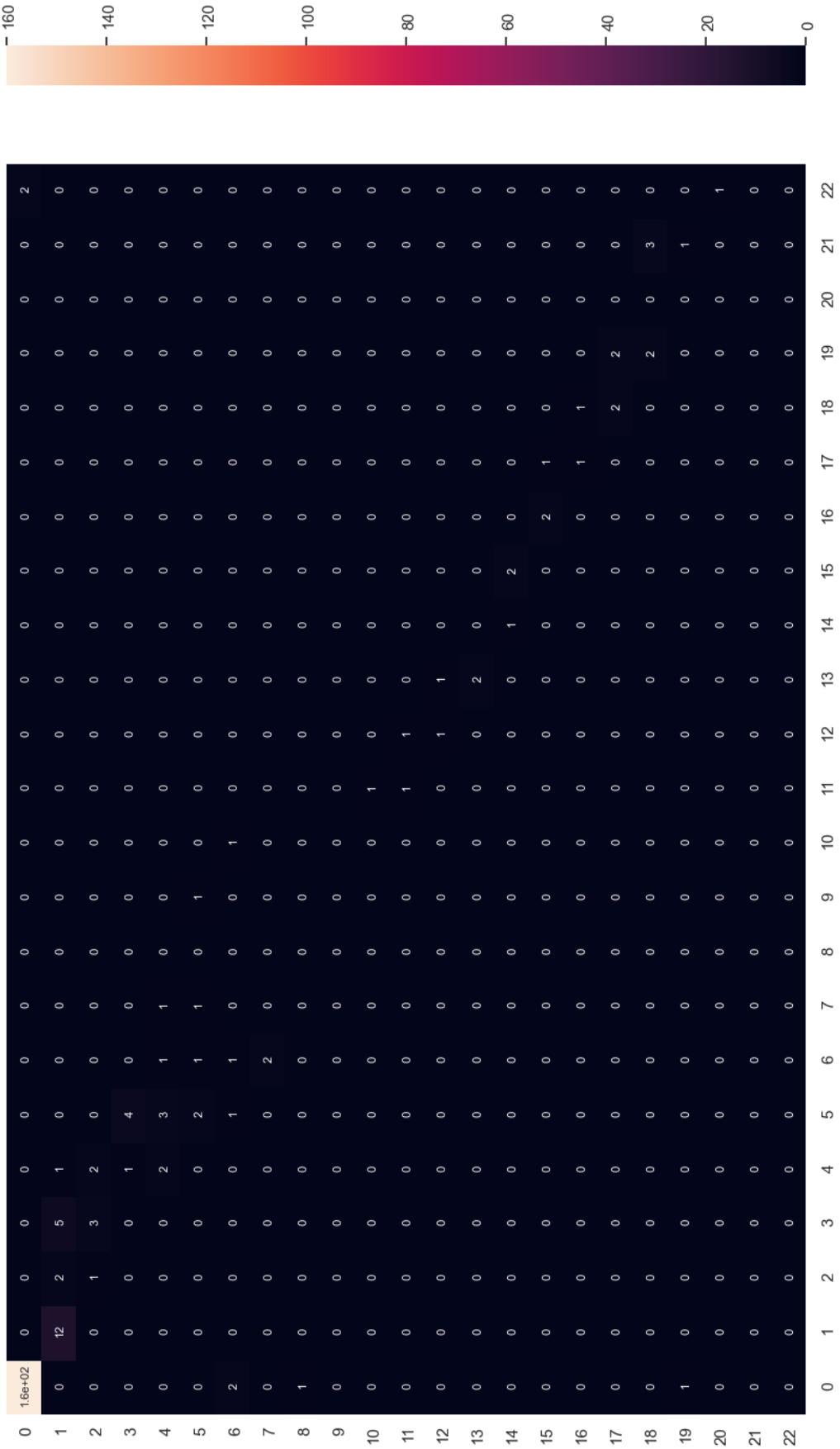
APÊNDICE H – MÉTRICAS PARA O ARQUIVO *SAMPLE_2.AVI*

Figura 38: Matriz de confusão para o quadro 175



Fonte: Elaborada pelo autor

Figura 39: Matriz de confusão para o quadro 235



Fonte: Elaborada pelo autor

Tabela 6: Métricas para o quadro 175

Métrica	**Resultado**
Precisão	0.85
Revocação	0.85
Medida-F	0.85

Fonte: Elaborada pelo autor

Tabela 7: Métricas para o quadro 235

Métrica	**Resultado**
Precisão	0.7625
Revocação	0.7625
Medida-F	0.7625

Fonte: Elaborada pelo autor