UNIVERSIDADE FEDERAL DO PAMPA

GABRIEL DA ROSA HENTSCHKE

TÉCNICAS DE PROCESSAMENTO DE IMAGENS PARA AUXILIAR NA GERAÇÃO DE MÁSCARAS PARA INPAINTING

Alegrete
2023

GABRIEL DA ROSA HENTSCHKE

TÉCNICAS DE PROCESSAMENTO DE IMAGENS PARA AUXILIAR NA GERAÇÃO DE MÁSCARAS PARA INPAINTING

Trabalho de Conclusão de Curso (Graduação) apresentado ao curso de Bacharelado em Ciência da Computação da Universidade Federal do Pampa como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Marcelo Resende Thielo

Alegrete
2023

Gabriel da Rosa Hentschke

TÉCNICAS DE PROCESSAMENTO DE IMAGENS PARA AUXILIAR NA GERAÇÃO DE MÁSCARAS PARA INPAINTING

Trabalho de Conclusão de Curso apresentado ao Curso de Ciência da Computação da Universidade Federal do Pampa, como requisito parcial para obtenção do Título de Bacharel em Ciência da Computação.

Trabalho de Conclusão de Curso defendido e aprovado em 07/12/2023.

Banca examinadora:

_____

Prof. Dr. Marcelo Resende Thielo

Orientador

Unipampa

_____

Prof. Dr. Eliezer Soares Flores

Unipampa

_____

Prof. Dr. Fábio Paulo Basso

Unipampa

Assinado eletronicamente por **MARCELO RESENDE THIELO**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 10/12/2023, às 02:34, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Assinado eletronicamente por **ELIEZER SOARES FLORES**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 18/12/2023, às 06:00, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Assinado eletronicamente por **FABIO PAULO BASSO**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 18/12/2023, às 07:35, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

A autenticidade deste documento pode ser conferida no site https://sei.unipampa.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **1317927** e o código CRC **4DBFDE74**.

Dedico esse trabalho para meus pais e irmã por todo apoio que me deram.

AGRADECIMENTO

Eu agradeço a toda a minha família e amigos por estarem presentes durante toda essa jornada que está sendo cursar o ensino superior. Agradecendo especialmente minha mãe e meu pai, Jiceli da Rosa Hentschke e João Carlos Hentschke, que me criaram com carinho e me permitiram cursar o ensino superior longe de casa, mesmo com os contratempos que isso gera. Agradeço também a minha irmã, Marina da Rosa Hentschke, por ser minha amiga e companheira, sempre me fazendo cair na gargalhada. Agradeço aos meus colegas de apartamento Bruno Scalabrin da Silva Cariolato, Jeanine Flores Correa e Luiz Rafael Jaques Caldeira por todas as conversas e diversão nesse último ano de curso. Agradeço os meus amigos, Felipe Bedinotto Fava, Uéslei Bervanger Brand, Leonardo Castro, Luan Preira Vargas, Wesley Ferreira de Ferreira, Douglas Vinicius Ledur Duillius e Guillermo Pando Teixeira, pelas horas de jogo (seja de RPG ou qualquer outro) e pelas conversas. Agradeço ao Prof. Cláudio Schepke por ter me dado acesso ao computador de seu laboratório de pesquisa, sem o qual não teria conseguido executar nenhum código de *inpainting*. Agradeço também o Prof. Marcelo Resende Thielo por ter aceitado a proposta deste trabalho e por ter respondido pacientemente minhas dúvidas (mesmo durante as folgas).

RESUMO

Restauração de imagens é um clássico problema de baixo nível de visão computacional. Métodos de restauração de imagens buscam desfazer, ou remediar, defeitos em imagens, restaurando imagens de baixa qualidade. Uma das maneiras de remediar esses efeitos através de software é utilizar um algoritmo de *inpainting*. Muitos destes algoritmos necessitam não só de uma imagem de entrada, mas também de uma máscara onde ficam marcadas as regiões que se deseja que o algoritmo realize o *inpainting*. A máscara deve ser feita a mão ou obtida através da utilização de um algoritmo como uma rede neural. Neste trabalho mostramos como técnicas de processamento de imagens digitais podem ser utilizadas como uma alternativa menos custosa para criação de máscaras para *inpainting*. Utilizamos técnicas para segmentação da imagem, como *thresholding* e clusterização, filtro de mediana para redução de ruído e operações morfológicas, tanto para a acentuação de detalhes, quanto para remoção de outros. As máscaras foram utilizadas com o método de *inpainting* RePaint, para testar sua qualidade e comparadas com máscaras geradas por uma rede de detecção de arranhões. Por fim, os resultados obtidos sugerem que a utilização de técnicas de processamento de imagens utilizadas conseguem gerar máscaras que acarretam em resultados visualmente satisfatórios de *inpainting*, comparando com a imagem original.

Palavras-chave: Processamento de imagem. Máscara. Inpainting.

ABSTRACT

Image restoration is a classic low-level problem in computer vision. Image restoration methods aim to undo or remedy defects in images, restoring low-quality images. One way to address these effects through software is by using an inpainting algorithm. Many of these algorithms require not only an input image, but also a mask that marks the regions where the algorithm should perform the inpainting. The mask can be created manually or obtained through the use of an algorithm such as a neural network. In this work, we demonstrate how digital image processing techniques can be used as a less costly alternative for creating masks for inpainting. We use techniques for image segmentation, such as thresholding and clustering, median filtering for noise reduction, and morphological operations, both for detail enhancement and removal of others. The masks were used with the RePaint inpainting method to test their quality and compared with masks generated by a scratch detection network. Ultimately, the results obtained suggest that the use of image processing techniques can generate masks that lead to good inpainting results.

Keywords: Image processing. Mask. Inpainting.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

UNIPAMPA – Universidade Federal do Pampa

DDPM – Modelos Probabilísticos de Difusão de Redução de Ruídos (DDPM do inglês *Denoising Diffusion Porbabilistic Models*)

HSV – Luminosidade, Saturação e Tonalidade (HSV do inglês *Hue, Saturation e Value*)

PIL – Biblioteca de Imagens do Python (PIL do inglês *Python Image Library*)

Sumário

1 INTRODUÇÃO

Restauração de imagem é um clássico problema de baixo nível de visão computacional. Métodos de restauração de imagens buscam desfazer, ou remediar, defeitos em imagens. Esses defeitos podem aparecer de diversas formas e por diversos motivos como motion blur, compressão JPEG, desfoque da câmera, ruído, etc. Alguns tipos de métodos de restauração de imagens são os de super resolução, redução de artefatos de compressão, *denoising*, *inpainting*, *deblurring*, etc (GU et al., 2020).

Hoje em dia, mesmo considerando os avanços tecnológicos, os danos em imagens ainda são bastante presentes na área de imagens digitais. No entanto, sempre haverá mídia impressa, como capas de livros, capas de vídeo games, caixas de jogos de tabuleiro. Também temos ainda mídia impressa de itens mais antigos como os encartes de discos de vinil, capas de CD, encarte de fitas cassete e VHS. Tanto as mídias que ainda existem impressas quanto as que já existiram estão sujeitas ao tempo e a acidentes que podem as danificar.

Muitas vezes, possuidores de itens danificados têm interesse em restaurá-los, seja por razões históricas, porque o item está decorando algum local ou mesmo por razões sentimentais. Considerando que não é difícil procurar imagens na internet, seria suficiente encontrar esta imagem e imprimir um novo encarte para o seu disco, VHS ou outro item de interesse. A questão é que nem sempre imagens digitais de boa qualidade estão disponíveis na internet ou correspondem à exata edição do item. Além disso, uma digitalização em alta resolução poderia ser realizada na atualidade com um exemplar disponível, no entanto por se tratar de um item antigo, este provavelmente já sustentará algum tipo de degradação por envelhecimento natural ou ainda dano acidental ou por manuseio.

As Figuras1, 2 e 3 são um exemplo do fato de acharmos as imagens na internet mas que nem sempre se apresentam do jeito que desejamos. A Figura 1 mostra duas imagens da capa do encarte de um disco de vinil, estando a primeira imagem danificada em várias partes e a outra em bom estado, porém com uma resolução menor do que a primeira e com uma parte coberta por um plástico.

A Figura 2 mostra duas imagens da parte de trás do encarte do mesmo disco da Figura 1. A primeira imagem está bastante danificada e a segunda em bom estado, mas, também tem uma resolução menor que a da esquerda e o pedaço de plástico. As imagens não danificadas escolhidas para parear com as imagens danificadas nas Figuras 1 e 2 são imagens da mesma edição do encarte danificado, o que filtra bastante a busca por imagens, deixando com poucas opções para escolher.

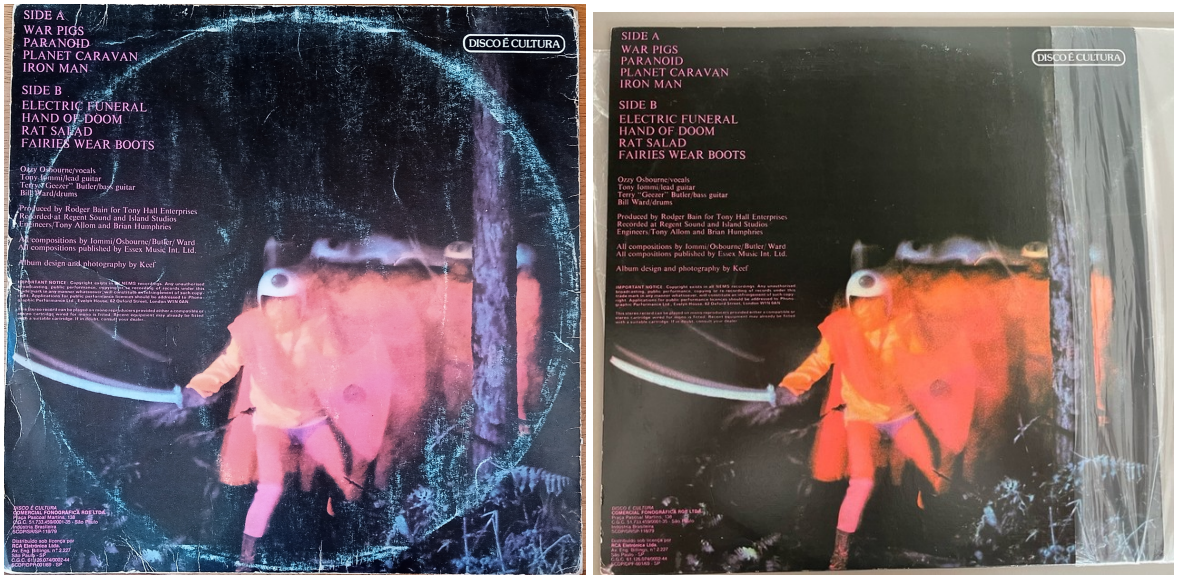
A Figura 3 mostra duas imagens de uma revista, a da esquerda danificada e a da direita em bom estado. Diferente das Figuras 1 e 2, a Figura 3 não tem os danos facilmente identificáveis. Ela está danificada nas laterais e tem um risco que cruza a imagem de cima a baixo diagonalmente. A imagem em bom estado tem uma resolução bem menor do que a danificada e poucos exemplos na internet.

Figura 1 – Exemplo de imagem danificada: capa do disco Paranoid.

(a) Encarte de disco danificado (b) Encarte em bom estado

Fonte: O autor, 2023

Figura 2 – Exemplo de imagem danificada: parte de trás do encarte do disco Paranoid.



(a) Encarte de disco danificado (b) Encarte em bom estado

Fonte: O autor, 2023

Considerando os danos presentes nas imagens, uma operação de *inpainting* com uma máscara produzida a mão marcando as áreas danificadas, seria o suficiente. Apesar disso, ao observar a Figura 4, pode-se notar que com a máscara feita à mão perdemos alguns detalhes importantes, como as letras no lado esquerdo da imagem e algumas plantas ao pé do poste na direita. Além disso, essa máscara fez com que o inpainting criasse um artefato indesejado na parte superior direita da imagem.

Figura 3 – Exemplo de imagem danificada: resvista Super Interessante: O Livro das Mitologias.

(a) Revista danificada (b) Revista em bom estado

Fonte: O autor, 2023

A partir disso, é possível concluir que fazer a máscara para uma imagem, cuidando todos os detalhes, pode acabar sendo muito trabalhoso.

Pensando nisso, este trabalho tem como proposta apresentar processos já muito presentes na área de processamento de imagens digitais para auxiliar na criação de máscaras para inpainting, utilizando algoritmos já estabelecidos, como *thresholding*, operações morfológicas, segmentação por clusterização, filtro de mediana, evitando assim a necessidade de alto poder computacional.

Neste capítulo ainda serão apresentadas as motivações na seção 1.1, os objetivos na seção 1.2 e, por fim, será mostrada a organização deste trabalho a partir do próximo capítulo na seção 1.4.

1.1 Motivação

Uma máscara é uma imagem que pode ser utilizada como guia para a aplicação de efeitos em uma imagem digital, desde que a máscara e a imagem tenham as mesmas dimensões.

Eliminar defeitos indesejados de uma imagem através de *inpainting* e obter um bom resultado final pode ser desafiador dependendo da complexidade da máscara necessária. Máscaras feitas à mão podem acabar afetando detalhes inportantes e, dependendo da complexidade dos

Figura 4 – Etapas do *Inpainting* com máscara feita a mão.

(a) Encarte danificado

(b) Região de *inpainting*

(c) Resultado do *inpainting*

Fonte: O autor, 2023

objetos que se deseja eliminar, sendo trabalhosas demais para elaborar e consequentemente demorar tempo demais para serem obtidas.

Uma solução seria utilizar uma rede neural profunda para detectar os defeitos na imagens e construir a máscara. No entanto, a rede necessitaria de treinamento para reconhecer arranhões, manchas e artefatos que não pertencem a imagem, como assinaturas, adesivos, etc. Contudo, uma rede neural tão completa demanda bastante poder computacional e um conjunto de treinamento complexo, que englobe todos esses problemas. Mesmo que uma rede neural já treinada não exija muito poder computacional, nem sempre a generalização que ela faz será eficiente para todos os casos.

Dessa forma, neste trabalho buscamos apresentar técnicas de processamento de imagens digitais já estabelecidas, que não dependam de redes neurais, e que possam ser aplicadas a imagens de maneira que uma máscara suficientemente detalhada seja gerada.

1.2 Objetivos

O objetivo deste trabalho é demonstrar técnicas de processamento de imagens digitais que possam ser utilizadas para gerar uma máscara. A máscara que buscamos neste trabalho deve demarcar alguma região de interesse. A partir das máscaras geradas será feito o *inpainting* da imagem que deu origem a máscara para verificar sua eficácia.

1.3 Trabalhos relacionados

No nosso trabalho procuramos comparar as máscaras geradas com outras, geradas por uma rede neural, escolhemos o trabalho de (WAN et al., 2020), cujo a proposta é uma rede para a restauração de fotografias antigas com múltiplos tipos de degradação, pois ele utiliza de uma rede neura de detecção de arranhões para gerar suas máscaras. Fotos antigas tem degradações muito complexas, o trabalho divide elas em não estruturais e estruturais. As primeiras sendo coisas como ruído no filme fotográfico, desfoque e desbotamento, essas podem ser resolvidas com filtros homogêneos que usam os pixels na vizinhança como referência. As degradações estruturais são arranhões e manchas, que podem ser resolvidas com *inpainting*. As máscaras para as degradações estruturais são geradas através de uma rede treinada com arranhões sintéticos e reais.

O método de *inpainting* utilizado no trabalho para os testes com as máscaras geradas foi o RePaint. Proposto por (LUGMAYR et al., 2022), é um método de *inpainting* que se aproveita de um modelo probabilísticos de difusão de redução de ruídos (DDPM) treinado previamente. Ao invés de aprender um modelo generativo de máscara condicional, o método condiciona o processo de geração por amostragem através dos pixels fornecidos durante as iterações do processo de difusão reversa. O modelo não é treinado para *inpainting*, e isso traz duas vantagens. Primeiro, permite à rede generalizar qualquer máscara durante a inferência. Segundo, permite à rede aprender mais recursos de geração semântica, pois o algoritmo possui uma forte capacidade de síntese de imagens DDPM. No entanto, esse modelo probabilístico é mais lento do que métodos de Redes Adversárias Generativas e Autorregressivos. Além disso, para o caso de máscaras mais extremas, o RePaint consegue gerar imagens realistas baseadas na entrada, mas não consegue reproduzir a imagem original. O algoritmo produz imagens foto-realísticas independentemente do tipo de máscaras aplicadas sobre a imagem.

A Figura 5 mostra o resultado uma imagem em que o RePaint foi aplicado. Podemos ver que o algortimo preenche toda a área da máscara formando o resto do rosto usado de input. Como comentado anteriormente, é uma reprodução realista do rosto, porém, não é rosto original por debaixo da máscara.

Figura 5 – Amostra do RePaint.

(a) Imagem original

(b) Região de *inpainting*

(c) Resultado do *inpainting*

Fonte: (LUGMAYR et al., 2022)

1.4 Organização do trabalho

- **Capítulo 2 Fundamentação Teórica:** conceitos e técnicas abordadas neste trabalho;
- **Capítulo 3 Desenvolvimento do Trabalho:** é mostrado o uso das técnicas de processamento de imagens no trabalho;
- **Capítulo 4 Resultados :** são mostrados resultados de testes realizados;
- **Capítulo 5 Considerações Finais:** considerações feitas sobre o que foi realizado no trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Processamento de Imagens Digitais

Existem vários objetivos ao se manipular uma imagem digital, sendo que a área de processamento de imagens digitais se preocupa em ter imagens digitais tanto como entrada quanto como saída (YOUNG; GERBRANDS; VLIET, 1998). Os processos geralmente tem como objetivo melhorar a visibilidade dos detalhes ou das caraterísticas da imagem, garantir sua qualidade para impressão ou transmissão ou preparar a imagem para um análise (RUSS; RUSS, 2017). Nesta seção apresentaremos as definições de imagem digital e máscara. Também serão mostradas as técnicas de processamento de imagens utilizadas para alcançar os resultados obtidos. Estas sendo, filtro de mediana, um filtro utilizado para atenuar ruído, técnicas de segmentação de imagens, que tem como destacar os elementos que nos interessam em uma imagem e operações morfológicas, que são operações de conjutos que podem servir para destacar ou reduzir elementos na imagem.

2.1.1 Imagens Digitais

Segundo (GONZALEZ; WOODS, 2018) uma imagem é uma função $f(x, y)$ onde x e y são coordenadas em um plano e a amplitude da função para cada coordenada é chamada de intensidade ou nível de cinza. Uma imagem digital tem como características: número finito de valores para x, y e intensidades de f. Também é composta por um número finito de elementos, cada um com valores e coordenadas próprias. Esses elementos são chamados de pixels[1].

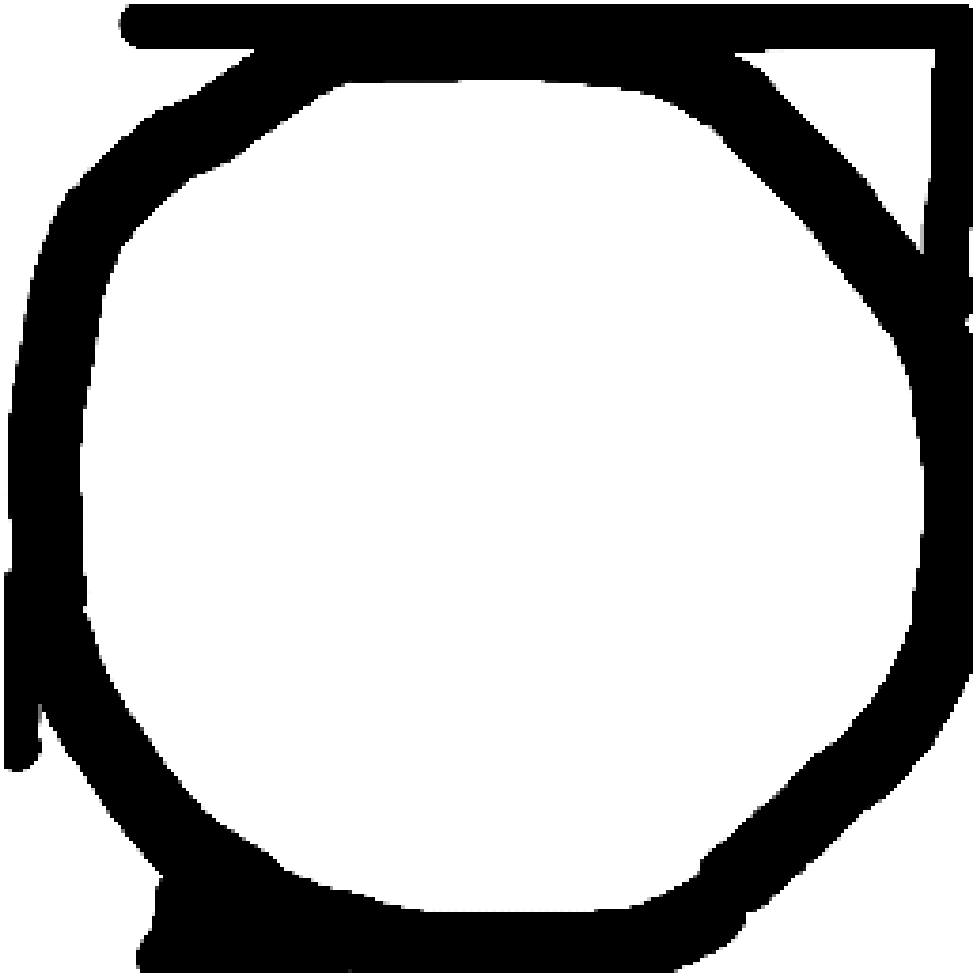
Cada pixel guarda os valores de cor presentes em sua coordenada na imagem. Esse valor pode ser um canal de 0 a 255, representando escalas de cinza ou, o mais utilizado para imagens coloridas, três canais com valores de 0 a 255 para cada componente vermelha, verde e azul. Uma maneira de caracterizar cores de forma mais natural para seres humanos é observando três propriedades distintas: a luminosidade, saturação e a tonalidade (HSV) (SHIRLEY; ASHIKHMIN; MARSCHNER, 2005).

2.1.2 Máscaras

Segundo (FISHER et al., 1996), uma imagem digital composta de elementos binários, sendo eles ou zero ou não zero, é uma máscara. Se aplicada a outra imagem binária de mesmo tamanho, os pixels que são zero na máscara viram zero na imagem de saída. Normalmente utilizada para limitar operações, como por exemplo, operações aritméticas, à área definida pela máscara. Na Figura 6 temos um exemplo de máscara. Neste caso, a área em que a máscara delimita é a com os pixels pretos.

[1] *Picture element*

Figura 6 – Exemplode máscara.



Fonte: O autor, 2023

2.1.3 Filtro de mediana

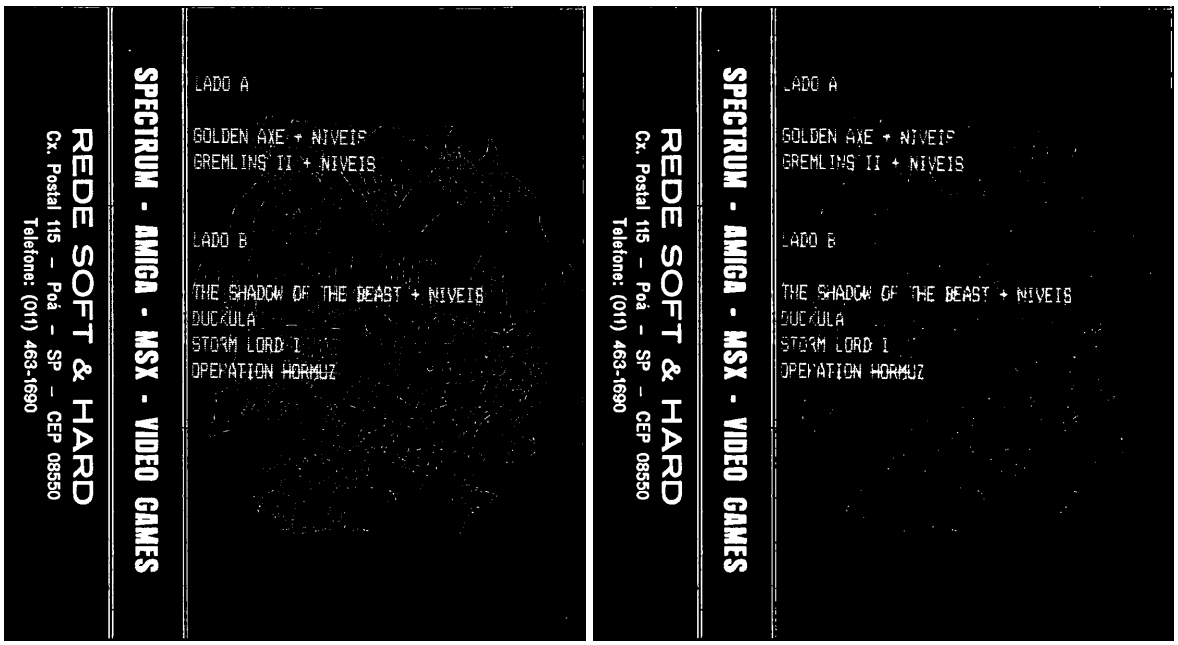
Filtro de mediana é um filtro de passa baixa não linear. Um filtro passa baixa é um tipo filtro espacial utilizado para reduzir transições bruscas de intensidade. Como ruído aleatório é comumente constituído de transições bruscas, estes filtros são utilizados como forma de atenuá-lo (GONZALEZ; WOODS, 2018).

Filtro de mediana é um filtro espacial normalmente utilizado para atenuar ruído em imagens. O filtro passa por cada pixel da imagem, observando o pixel e sua vizinhança. Ele encontra a mediana dos valores da vizinhança e substitui o valor do pixel por ele. A vizinhança é determinada por uma matriz quadrada (FISHER et al., 1996). Isso pode ser melhor observado no Algoritmo 1.

Algoritmo 1 Filtro de Mediana

$I \leftarrow$ imagem $x \times y$
$N \leftarrow$ lista com os valores da vizinhança
$n \leftarrow$ tamanho da vizinhança
$k \leftarrow \lfloor n/2 \rfloor$
for each linha i, indo de k até $x - k$ **do**
  for each coluna j , indo de k até $y - k$ **do**
    for each inteiro l, de 0 a n **do**
      for each inteiro c, de 0 a n **do**
        $N \leftarrow$ adiciona $I_{i+(l-k),j+(c-k)}$ na lista
      end for
    end for
    $I_{i,j} \leftarrow mediana(N)$
    $esvazia\_lista(N)$
  end for
end for

Figura 7 – Suavização utilizando filtro de mediana com vizinhança 3×3.



(a) Antes do filtro (b) Depois do filtro

Fonte: O autor, 2023

Podemos ver na Figura 7, onde foi aplicado um filtro de mediana com vizinhança 3×3, que há menos pontos brancos na imagem pós filtro.

2.1.4 Segmentação de Imagens

São técnicas utilizadas para destacar objetos de interesse na imagem (ou um objeto de primeiro plano) e separá-los do segundo plano (YOUNG; GERBRANDS; VLIET, 1998). As técnicas de segmentação de interesse para esse trabalho são as de *thresholding* e de clusterização.

Nas técnicas de *thresholding* é escolhido um valor T para ser o valor de thresholding. Depois verificamos se cada pixel da imagem tem um valor maior que T e, se sim, o pixel é um pixel de primeiro plano, senão é um de segundo plano (GONZALEZ; WOODS, 2018). Na Figura 9 temos a aplicação de uma operação de thresholding na Figura 8.

Figura 8 – Imagem original do encarte dos Garotos da Rua.

Fonte: O autor, 2023

Outra maneira de segmentar a imagem é através da clusterização das cores na imagem. Utilizando um algoritmo de k-médias para clusterização é possível segmentar a imagem em k *clusters* (GONZALEZ; WOODS, 2018). Ele é um algoritmo não-determinístico, não supervisionado iterativo e numérico. No algoritmo, os *clusters* são representados pelo valor médio, chamado de centróide, dos elementos que o compõem. Divide-se os n elementos entre k *clusters* de uma forma que a similaridade entre os *clusters* seja baixa e dentro deles seja alta. Essa similaridade é calculada com a média dos elementos no cluster. Na inicialização do algoritmo k centróides são selecionados aleatóreamente. Após isso, cada elemento é associado ao centróide mais próximo. O algoritmo itera, calculando os centróides e atualizando quais elementos pertencem a cada *cluster*, através do cálculo da distância euclidiana entre elemento e o contróide. Ele itéra até os centróides não mudarem mais (YADAV; SHARMA, 2013).

Figura 9 – *Thresholding* do encarte dos Garotos da Rua.

Fonte: O autor, 2023

A Figura 10 mostra a aplicação do algoritmo de k-médias na Figura dos Garotos, utilizando k igual a 6 e igual a 12. Observando-a percebe-se o grande impacto gerado pelo valor de k, principalmente em imagens mais coloridas.

2.1.5 Operações Morfológicas

Operações morfológicas se tratam de uma série de técnicas que se importam com a forma de características das imagens. Geralmente, essas operações são utilizadas para remover imperfeições presentes em uma imagem que passou por segmentação. Suas operações básicas são dilatação e erosão, e ambas são produzidas com uma operação de conjuntos, onde um conjunto é formado por pixels de interesse da imagem e o outro é formado por um elemento estruturante (GOYAL, 2011).

Um elemento estruturante é composto de um padrão de uma quantidade discreta de pontos relativos a uma origem (FISHER et al., 1996). Na Figura 11 vemos um exemplo visual de elemento estruturante, onde a origem é o ponto contornado. A origem não precisa ser no centro do elemento estruturante, e seu tamanho e forma pode variar.

Figura 10 – Segmentação com k-médias.

(a) $k = 6$ (b) $k = 12$

Fonte: O autor, 2023

2.1.5.1 Dilatação

A operação de dilatação faz com que objetos de interesse "cresçam" em tamanho. Um algoritmo de dilatação para imagens binárias funciona da seguinte forma:

1. Encontra um objeto de interesse;
2. Considera esse objeto sendo a origem do elemento estruturante;
3. Todo pixel na área do elemento estruturante que não seja branco se torna branco.

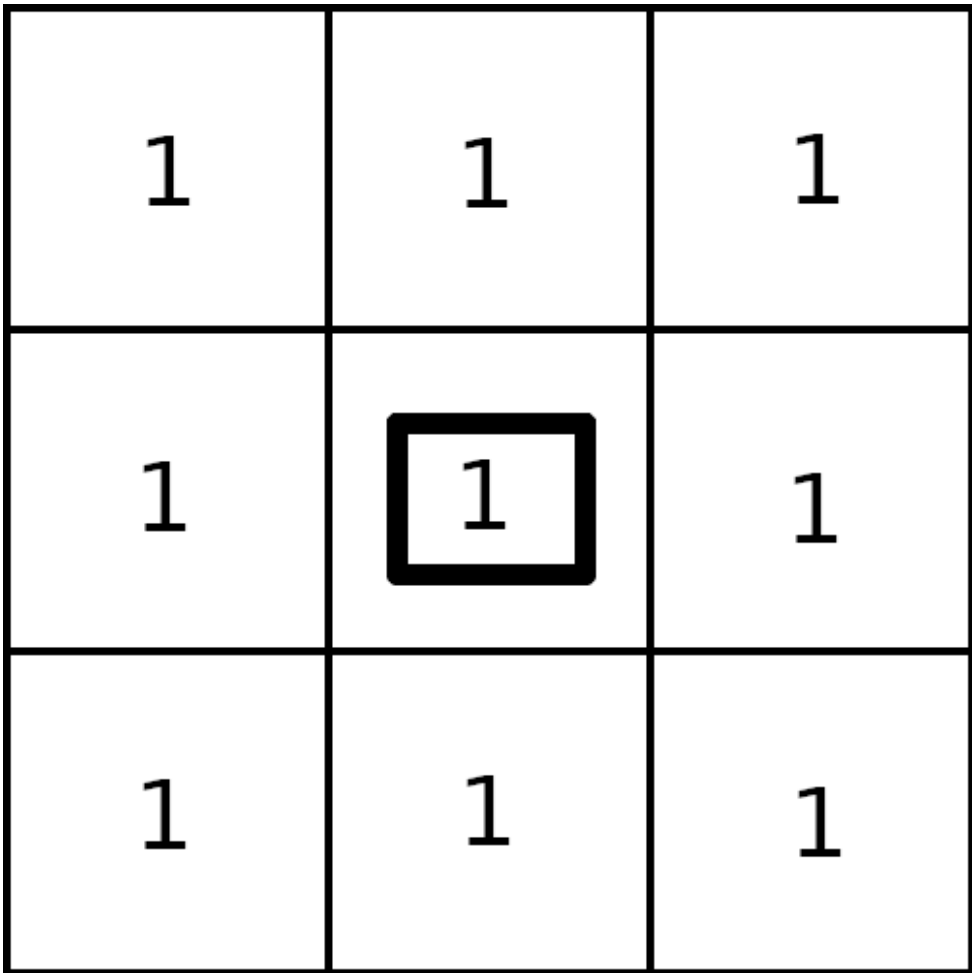
A Figura 13 é o resultado da operação de dilatação com um elemento estruturante 5×5, com origem na coordenada central da matriz, sobre a Figura 12. Podemos ver que todas as áreas brancas da figura original estão maiores, engrossando assim as linhas pontos e letras.

2.1.5.2 Erosão

A operação de Erosão faz com que objetos "diminuam" de tamanho. Um algoritmo de erosão para imagens binárias funciona da seguinte forma:

1. Encontra um objeto de interesse;
2. Considera esse objeto sendo a origem do elemento estruturante;
3. Se houver algum pixel preto na área do elemento estruturante, o objeto de interesse muda seu valor para preto.

Figura 11 – Exemplo visual de um elemento estruturante.



Fonte: O autor, 2023

A Figura 14 é o resultado da operação de erosão com um elemento estruturante 5×5, com origem na coordenada central da matriz, sobre a Figura 12. Nesse caso as letras pequenas pontos e linhas finas foram eliminados da imagem.

2.1.5.3 Abertura

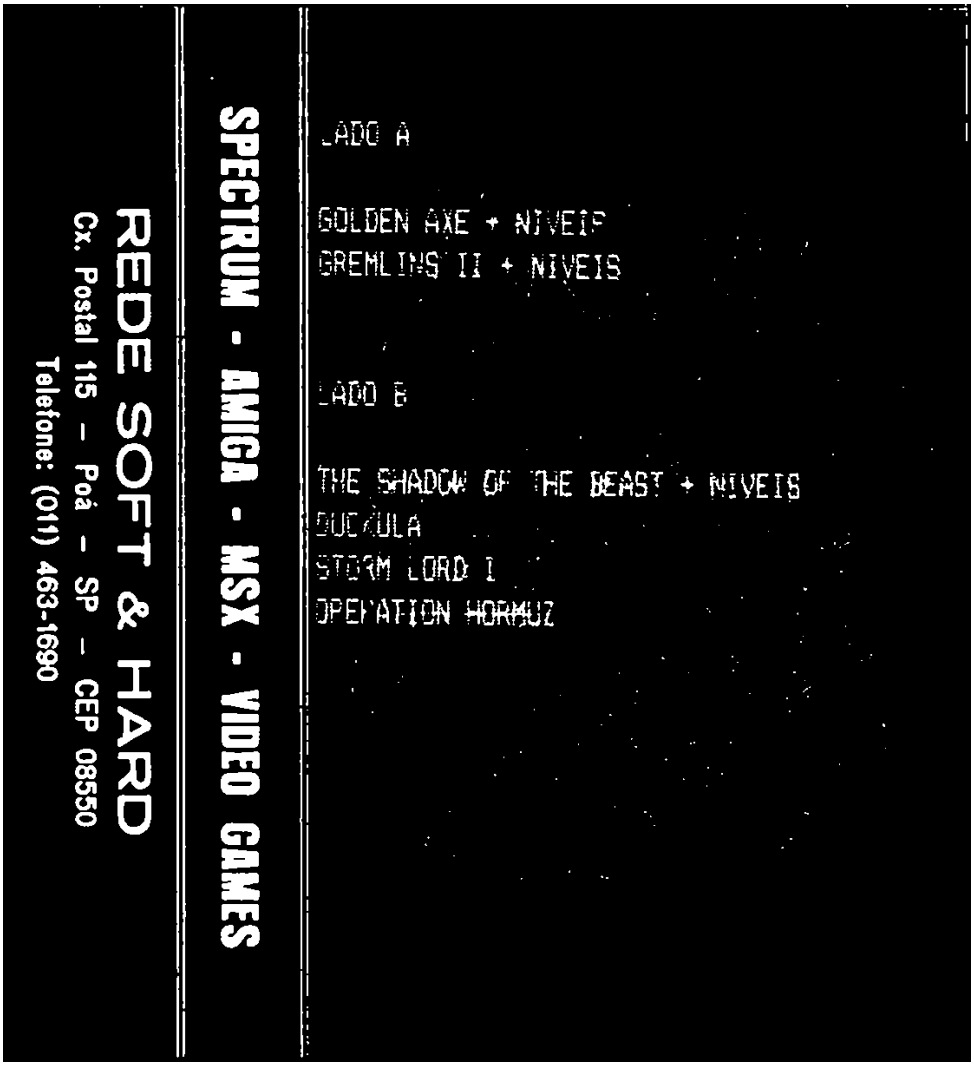
Abertura é erosão seguida de dilatação. Essa operação suaviza contornos de um objeto, quebra uniões estreitas entre objetos e elimina saliências finas(GONZALEZ; WOODS, 2018).

A Figura 15 é o resultado da aplicação da operação de abertura na Figura 12. Um resultado muito parecido com o de erosão, porém com as letras mais grossas.

2.1.5.4 Fechamento

Fechamento é dilatação seguida de erosão. Também suaviza contornos, porém, fecha fendas estreitas, elimina buracos e preenche lacunas nos contornos(GONZALEZ; WOODS, 2018).

Figura 12 – Imagem com ruído.



Fonte: O autor, 2023

A Figura 16 é o resultado da aplicação da operação de fechamento da Figura 12. Resultado semelhante a dilatação na parte das letras grandes e nas linhas verticais a esquerda da imagem, porém os pontos brancos e as letras a direita estão bem menos destacados.

2.2 Image Inpainting

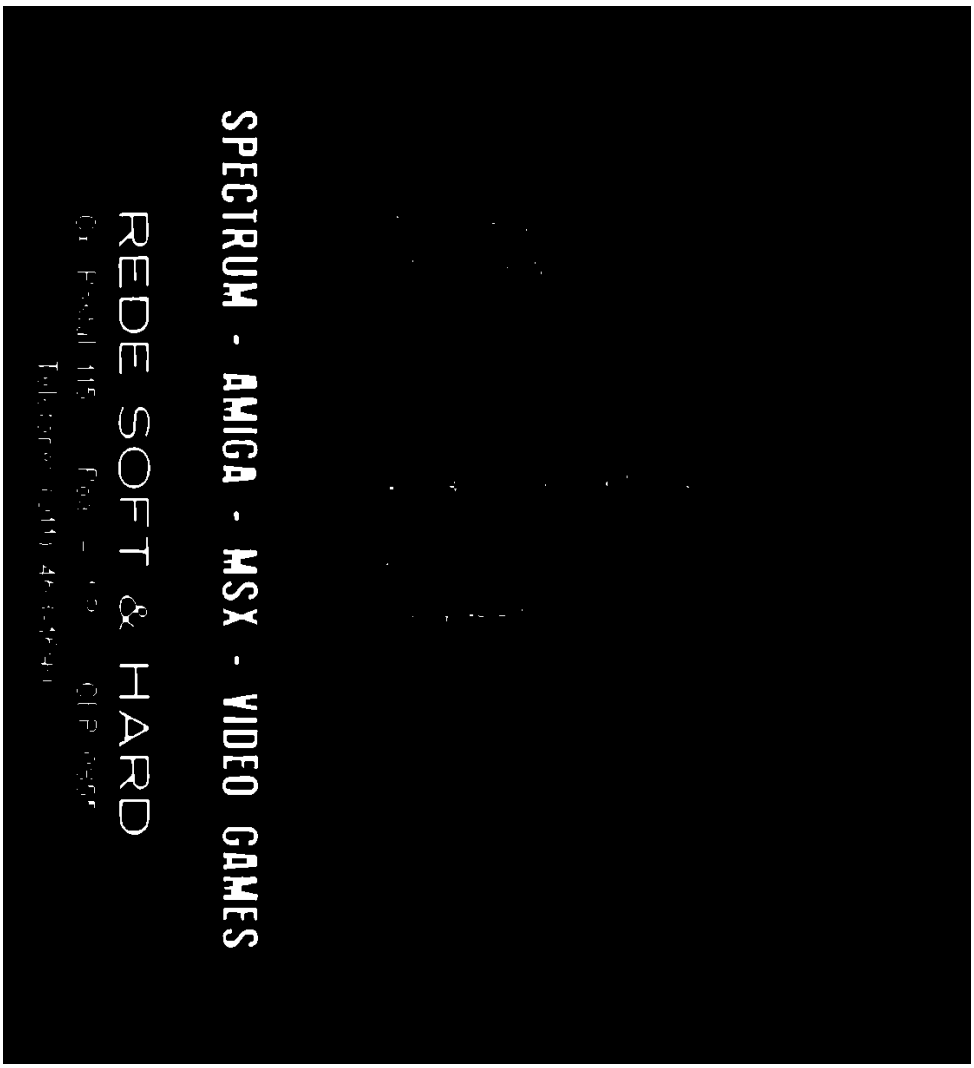
Image inpainting trata-se de preencher lacuna de uma imagem. Por exemplo, pode ser utilizado para remover conteúdo indesejado da imagem, enquanto preenche o espaço com imagens plausíveis (LIU et al., 2018). Uma máscara é posta por cima da área da imagem em que se quer fazer o inpainting. Essa região precisa harmonizar com o resto da imagem e ser semanticamente razoável (LUGMAYR et al., 2022). A Figura 17 mostra um exemplo de inpainting. Primeiro observa-se a imagem onde queremos aplicar o *inpainting*. Depois a máscara é aplicada nos pontos onde queremos que o inpainting ocorra, no caso dessa imagem são as grades atrás do casal e o texto no canto superior direito. Por fim, temos o resultado do

Figura 13 – Figura Dilatada.

Fonte: O autor, 2023

inpainting.

Figura 14 – Operação de Erosão.



Fonte: O autor, 2023

Figura 15 – Operação de Abertura.

Fonte: O autor, 2023

Figura 16 – Operação de Fechamento.

Fonte: O autor, 2023

Figura 17 – Exemplo de Image Inpainting.

(a) Imagem original

(b) Imagem com a máscara nos pontos indesejados

(c) Imagem após a operação de inpainting

Fonte: (YU et al., 2019)

3 DESENVOLVIMENTO DO TRABALHO

Neste capítulo são apresentados os passo-a-passos sugeridos para auxiliar na geração de máscaras para *inpainting* e algumas soluções que podem ser utilizadas pós *inpainting*.

Para fins de demonstração vamos utilizar a Figura 18. A parte que nos interessa nessa imagem é o centro, onde há algumas assinaturas. Começamos com uma operação de *thresholding*. Neste trabalho, estamos considerando que as áreas da máscara que possuem pixels pretos são as áreas onde será aplicado o *inpainting*.

Figura 18 – The Song Remains the Same danificado.

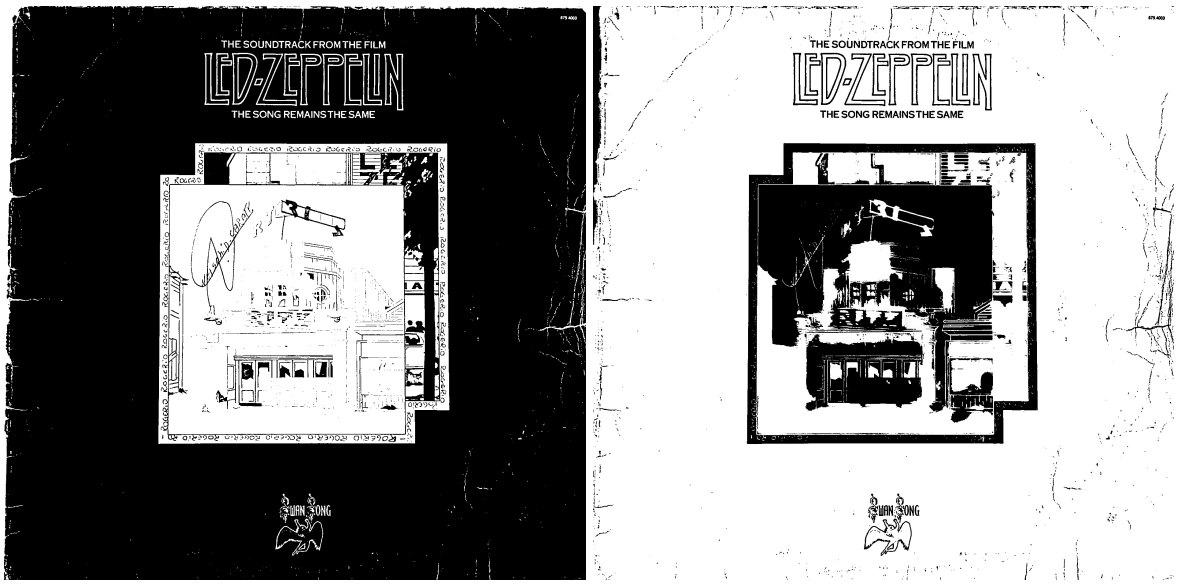
Fonte: O autor, 2023

O *thresholding* foi utilizado de duas maneiras. A primeira é convertendo a imagem para tons de cinza T sendo um valor de tom de cinza. A outra é para imagens coloridas, representadas pelo espectro HSV.

O *thresholding* com o espectro HSV foi realizado utilizando a abordagem disponível no OpenCV. A implementação não utiliza uma medida de distância contínua entre as cores, apenas verificando se todos os componentes individuais do pixel se encontra dentro do limite estabelecido, definido na chamada da função. Se o pixel estiver dentro do limite, ele é marcado

como um pixel de interesse na imagem de saída.

Figura 19 – Comparação de *Thresholding*.



(a) Tons de cinza (b) HSV da assinatura

Fonte: O autor, 2023

Na Figura 19 podemos ver as diferenças entre os dois tipos de T utilizados, sendo que nosso objeto alvo é a assinatura grande. O *threshold* na imagem em tons de cinza marcou as assinaturas a caneta, porém elas estão finas e com as cores pouco definidas. No *threshold* com HSV, com a assinatura grande como fonte dos valores, a imagem marcou todo o contorno da imagem central, onde haviam assinaturas e o céu de fundo. O problema é que como todo o contorno está marcado isso acarreta na perda dessa região pós *inpainting*. Outro aspecto é que a assinatura que está dentro da parte central da figura ficou um pouco marcada como branca, ou seja, nessa região não será feito o *inpainting*.

Algo que pode ser feito antes do *threshold* com HSV é quantizar as cores da imagem com um algoritmo de k-médias. Dessa forma as cores ficam mais uniformes e o *thresholding* por sua vez mais preciso. Observando a Figura 20 é possível notar que as assinaturas estão agora com a mesma cor e o resultado do *thresholding* com a imagem segmentada.

Caso julgue necessário, pode-se usar um filtro de mediana para atenuar ruído ou operações morfológicas, tanto para enfatizar as áreas onde se deseja o *inpainting* quanto para tirar elementos indesejados, como pontos pretos nas imagens. A Figura 21 é o resultado da operação de dilatação do *threshold* presente na Figura 20. Nota-se que as assinaturas ficaram mais espessas, o que garante o inpainting completo sobre elas.

A Figura 21 é uma ótima máscara para as assinaturas, contudo, grande parte da imagem está desnecessariamente coberta. A solução para isso pode ser simplesmente pintar as partes desnecessárias da máscara em um editor de imagens. Contudo, essa solução, dependendo da máscara gerada, pode acabar não sendo o suficiente, ou ainda, trabalhoso para o usuário.

Figura 20 – Imagem segmentada e *threshold* dela.

(a) Segmentada com k= 6 (b) *Threshold* com T sendo HSV das assinaturas

Fonte: O autor, 2023

Outra coisa que se deve observar é a imagem resultante do *inpainting* utilizando essa máscara. Por muitas vezes o resultado final será ótimo, mas apenas na exata área da máscara. O resultado em volta pode acabar não sendo tão satisfatório, ou algum artefato foi criado na imagem.

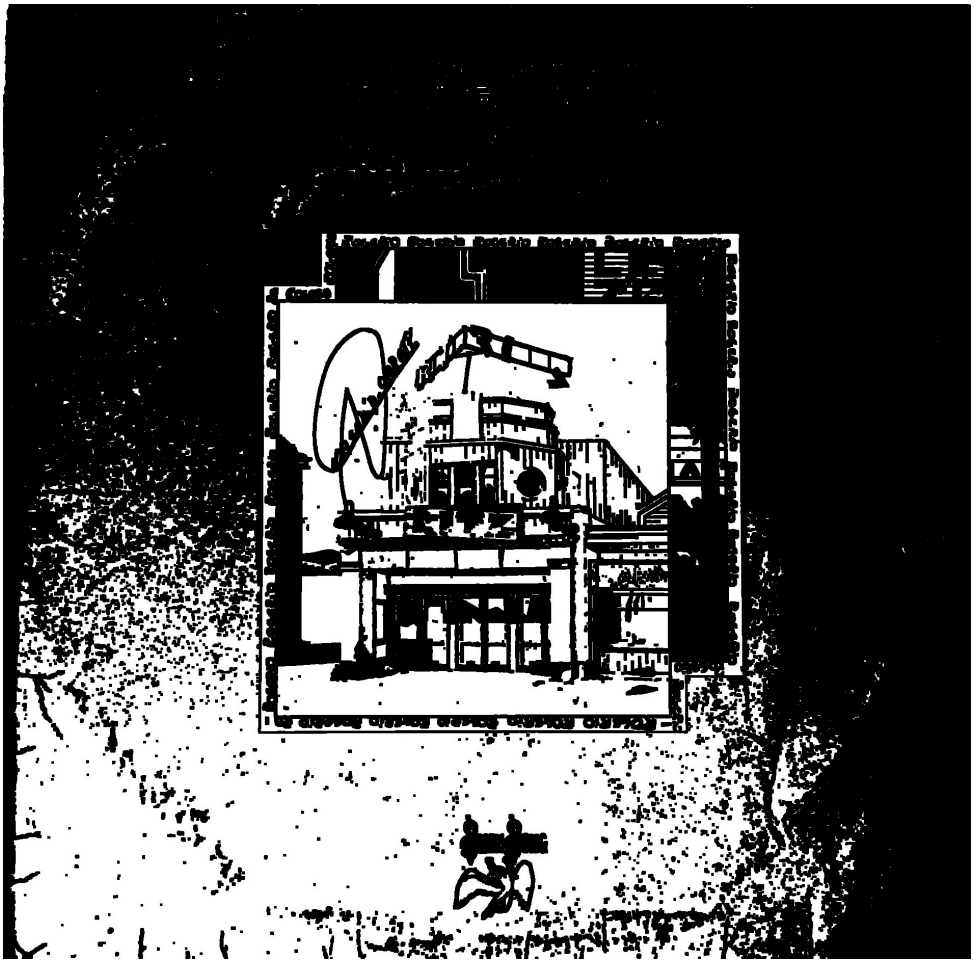
Neste trabalho fornecemos duas sugestões para lidar com esse tipo de problema. A primeira é recortar a área da imagem onde se deseja fazer o *inpainting* e fazê-lo apenas nela, e após isso, substituimos o resultado obtido na imagem. A outra é transferir apenas os pixels da área de *inpainting* que interessa na imagem original.

Para essa última sugestão, este trabalho oferece uma solução algorítmica. Primeiro, é preciso marcar na máscara que foi utilizada para o *inpainting* as áreas desejadas, podemos ver na Figura 22 que aqui utilizamos um retângulo vermelho. É importante que as linhas do retângulo tenham a espessura de um pixel e, caso haja mais de um, que os retângulos não se tangenciem. A máscara, a imagem original e a imagem pós *inpainting* precisam ter a mesma resolução.

O Algoritmo 2 demonstra de maneira genérica como encontrar cada retângulo vermelho e então transferir a área de inpainting dentro deles para a imagem original. A primeira parte do algoritmo é um *loop* que percorrerá a imagem da máscara inteira, garantindo que iremos encontrar todos os retângulos presentes. Ele então verifica se o pixel em questão é vermelho e se está na ponta esquerda superior do retângulo. Ou seja, se os pixels abaixo e a direita dele são vermelhos também.

Se forem, ele percorre a largura inteira do retângulo e por fim a altura, se guiando pelos pixels vermelhos. Dessa forma obtemos tanto as coordenadas iniciais do retângulo quanto

Figura 21 – Máscara dilatada gerada com o *threshold* da Figura 20.



Fonte: O autor, 2023

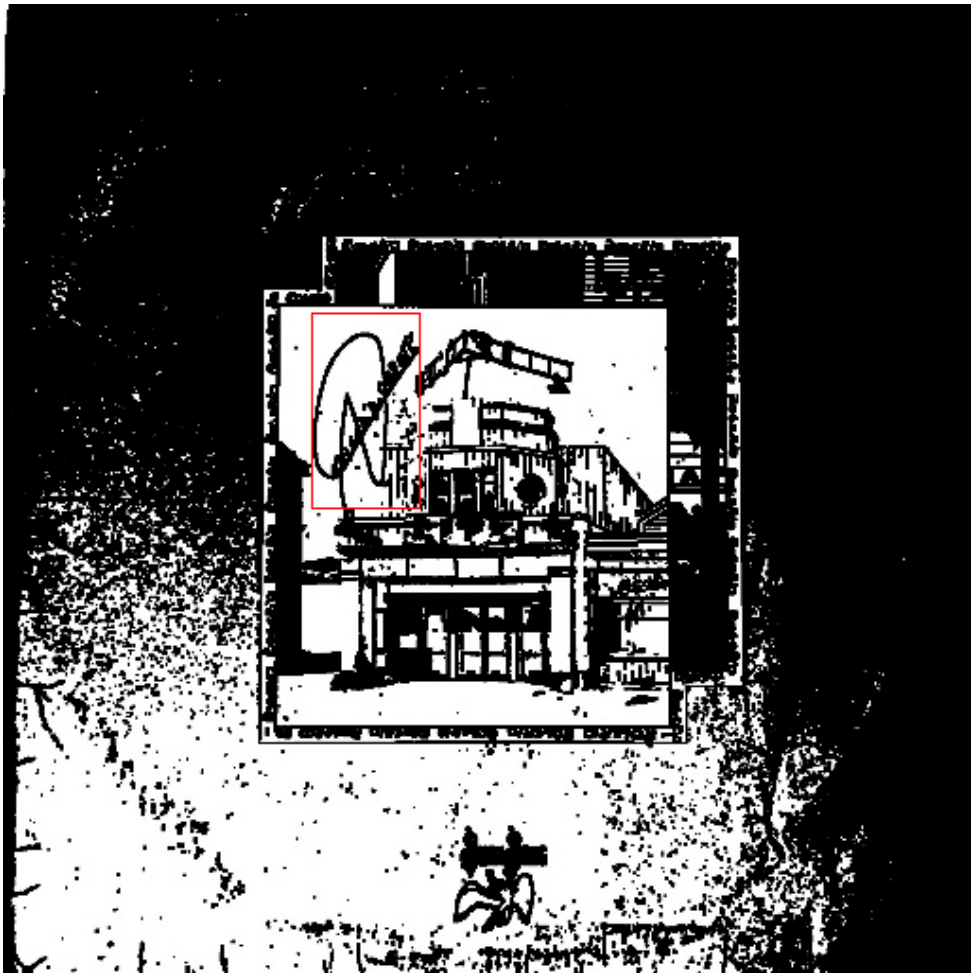
as finais, o que é suficiente para conseguirmos percorrer apenas as partes internas de cada retângulo.

Na segunda parte do Algoritmo 2, para cada retângulo vermelho encontrado, fazemos dois *loops* aninhados, dessa forma para cada pixel dentro do retângulo verificamos se na máscara ele era preto. Se sim mudamos o valor do pixel na imagem original mesma coordenada pelo valor que consta na imagem pós *inpainting*. Na Figura 23 temos a imagem antes do *inpainting* e após o *inpainting*, utilizando a máscara apresentada na Figura 24 sem o retângulo vermelho. Essas imagens, sendo que a máscara neste caso tem o retângulo vermelho, foram utilizadas como entrada do algrotimo. Como saída obtivemos a Figura 25.

Os testes com as máscaras foram feitos com o algoritmo de *inpainting* RePaint. Por limitações de hardware as imagens e máscaras tiverem que ser reduzidas para as resoluções de 256×256 e 512×512.

Tanto o algoritmo sugerido quanto os de processamento de imagens foram implementados na linguagem de programação Python, utilizando as bilbiotecas OpenCV, NumPy e PIL.

Figura 22 – Figura 21 demarcada com retângulo vermelho.



Fonte: O autor, 2023

Figura 23 – Imagem original e *inpainted* da luva da fita.

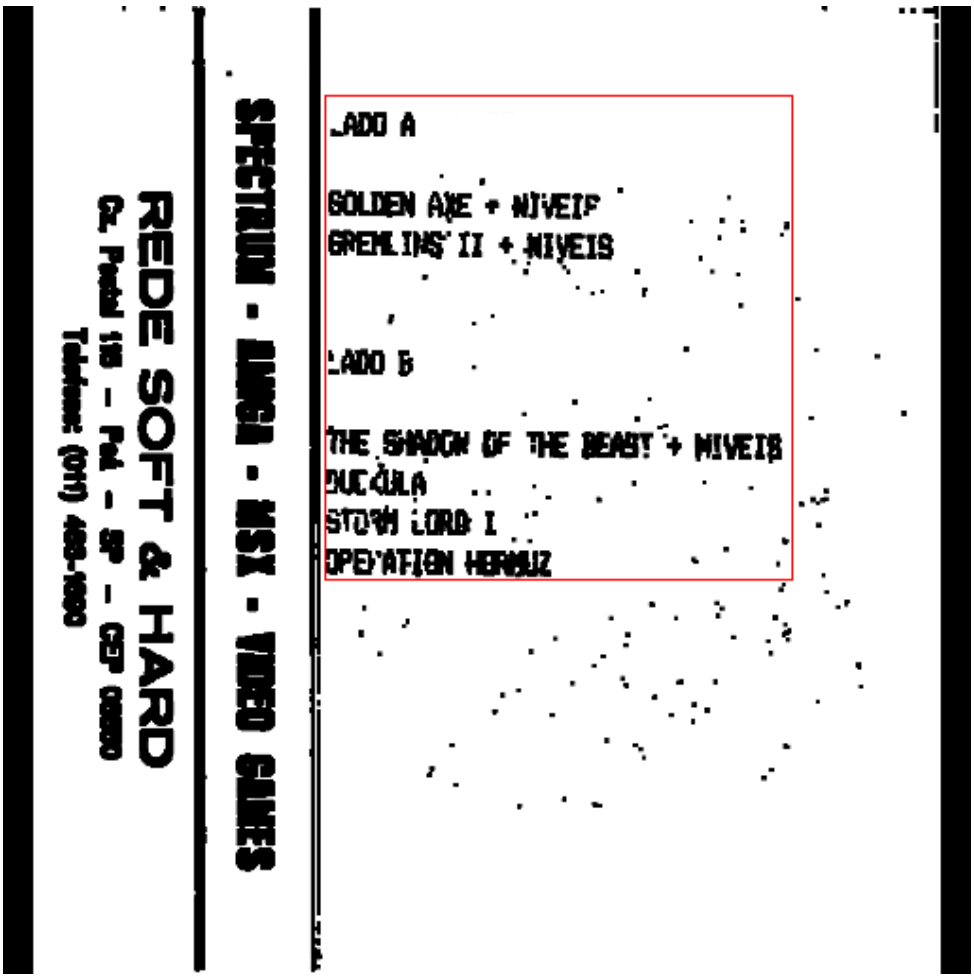
(a) Luva da fita (b) Luva da fita pós *inpainting*

Fonte: O autor, 2023

Algoritmo 2 Transfere área de uma imagem para outra.

```
M ← máscara
O ← imagem original
Inp ← imagem pós inpainting
Ret ← lista de tuplas de coordenadas x,y
for each pixel p da máscara do
    if p é vermelho e os pixels abaixo e a direita também são then
        for each pixel rp a direita de p do
            if o pixel a direita de rp não for vermelho e o abaixo for then
                for each pixel dp abaixo de rp do
                    if pixel abaixo de dp não for vermelho then
                        Ret ← (coordenadas de p, coordenadas de dp)
                        BREAK
                    end if
                end for
                BREAK
            end if
        end for
    end if
end for
for each retângulo r em Ret do
    for each linha i entre r(0,0) e r(1,0) do
        for each coluna j entre r(0,1) e r(1,1) do
            if o pixel em M(i, j) for preto then
                O(i, j) ← Inp(i, j)
            end if
        end for
    end for
end for
```

Figura 24 – Máscara utilizada no inpainting da Figura 23 demarcada por um retângulo.



Fonte: O autor, 2023

Figura 25 – Resultado do Algoritmo 2 utilizando as Figuras 23 e 24.

Fonte: O autor, 2023

4 RESULTADOS

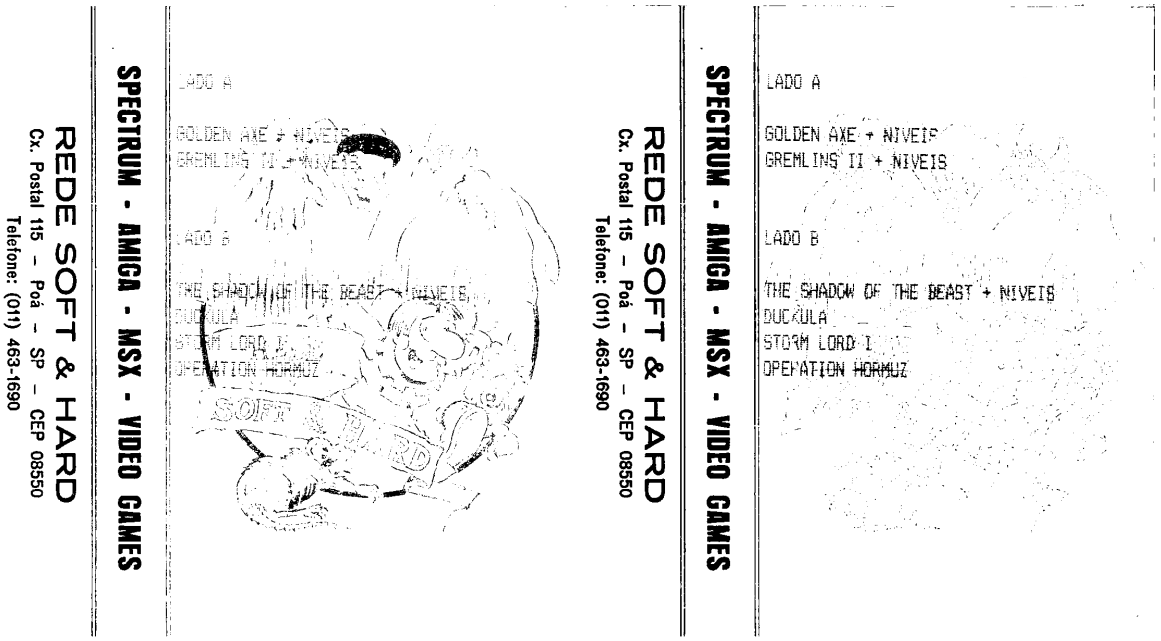
Neste capítulo serão apresentados os resultados obtidos ao utilizar as técnicas de processamento de imagens para gerar máscaras e os resultados das aplicações dessas máscaras em operações de *inpainting*. Também serão apresentados os resultados das soluções sugeridas para pós *inpainting* e para manter a resolução original da imagem.

4.1 Gerando máscaras

Nesta seção apresentaremos máscaras feitas com as técnicas apresentadas no capítulo de Desenvolvimento do Trabalho e as compararemos com máscaras geradas com uma rede neural de detecção de arranhões.

Os *thresholdings* foram todos realizados considerando um valor de HSV. Dessa maneira, torna-se mais fácil destacar apenas os elementos que temos como alvo para a máscara. Podemos observar uma comparação lado a lado na Figura 26, onde o nosso alvo são as letras na imagem e queremos que a gravura de fundo não seja marcada. O resultado com HSV não só destacou melhor as letras como ignorou muitos pixels da gravura.

Figura 26 – Comparação entre *threshold*.



(a) Escala de cinza (b) HSV

Fonte: O autor, 2023

Comparamos as máscaras geradas utilizando técnicas de processamento de imagens com as máscaras geradas pela rede neural de detecção de arranhões utilizada por (WAN et al., 2020) na Figura 27. É importante salientar que a rede não foi feita para detectar nada além de arranhões e manchas notáveis.

A rede de detecção de arranhões demonstra ótimos resultados para marcar arranhões e manchas mais agravadas. Outro ponto positivo da máscara, é o fato dela vir com as áreas marcadas um pouco mais grossas que os defeitos em si, reduzindo a probabilidade de precisar pós processamento.

Contudo, a rede por vezes detecta letras e alguns outros detalhes como se fossem arranhões. Além disso, manchas mais sutis não são detectadas pela rede. Outra consideração é que as máscaras geradas não têm a exata mesma resolução que a imagem de entrada.

Gerando máscaras utilizando *thresholding* conseguimos capturar detalhes diferentes. Além dos arranhões, manchas, suaves ou intensas, letras e outros detalhes são destacados. Os objetos segmentados são bastante fidedignos com os presentes na imagem de entrada.

Uma máscara gerada com muitos detalhes é algo que pode parecer ideal, mas também pode trazendo uma quantidade excessiva de elementos. Mesmo possibilitando a marcação dos mais variados elementos presentes na imagem, também pode acabar marcando muitos detalhes, o que faz aumentar a necessidade de pós processamento para a máscara.

4.2 Inpainting com as máscaras geradas

Nesta seção apresentaremos os resultados de *inpainting* utilizando as máscaras geradas com as técnicas de processamento de imagens sugeridas. Primeiramente mostraremos os resultados com máscara feita com *thresholding* com T sendo HSV e sem pré ou pós processamento. A imagem que buscamos restaurar é o encarte denificado da Figura 2.

A Figura 28 demonstra que um *thresholding*, mesmo sem retoques posteriores, já pode oferecer uma máscara que resulta em um *inpainting* mais satisfatório do que uma máscara feita a mão, como visto na Figura 4. A máscara foi gerada com um *thresholding* utilizando o valor de HSV dos pixels na região do halo que envolve boa parte do encarte.

Porém, a Figura 28, por conta de sua resolução, não gera um bom inpainting com as letras. Comparando com o resultado demonstrado na Figura 29, vemos que a resolução da máscara e da imagem faz bastante diferença. O inpainting da imagem 512×512 elimina mais defeitos e mantém a parte superior direita da imagem, onde diz “Disco é cultura” em boa qualidade.

No entanto, há casos que apenas um *threshold* para marcar as partes indesejadas não é o suficiente para garantir a geração de uma boa máscara. Observa-se na Figura 30, que mesmo com a máscara cobrindo exatamente as letras, não conseguimos eliminar completamente sua exisência. A máscara foi gerada segmentando as cores da imagem com k-médias, com k = 6, depois fazendo o *thresholding* utilizando o valor de HSV dos pixels na região das letras que ficam por cima da ilustração e então passando um filtro de mediana sobre ela.

Para obtenção de um resultado satisfatório para esse caso se mostrou necessário realizar o espessamento da máscara. Dessa maneira, foi nela aplicada a operação morfológica de dilatação. Observa-se na Figura 31 que as letras da imagem foram retiradas, ficando apenas a ilustração.

Outra situação é quando a máscara gerada é boa para a região de *inpainting* mas existem elementos suavemente marcados que não precisam de *inpainting*. Como exemplo temos a máscara presente da Figura 33, a máscara da Figura 32, que foi gerada com um *thresholding* utilizando o valor de HSV dos pixels na região marrom da imagem.

Os elementos que não queremos que sejam pintados são as letras e as linhas dos tijolos. Utilizando uma operação de abertura, como demonstrado na Figura 34, eliminamos essas partes indesejadas da máscara, garantindo um bom resultado de inpainting.

4.3 Soluções sugeridas pós inpainting

Nesta seção apresentaremos os resultados das soluções sugeridas para o pós *inpainting* e para preservar a resolução imagem alvo de *inpainting*.

Quando as operações morfológicas não são o suficiente para tirar partes indesejadas da máscara, mas ela ainda assim se encontra satisfatória para os elementos em que desejamos que o *inpainting* seja aplicado e não quisermos editar a máscara, podemos utilizar o Algortimo 2, desde que o inpainting na região desejada esteja adequado. Como exemplo, se submetermos a Figura 18 ao *inpainting* utilizando a Figura 21 como máscara teremos a Figura 35. Essa imagem não é igual à original, muito por conta da máscara cobrir grandes regiões com preto. Porém os nossos alvos eram as assinaturas na moldura da imagem central e no céu ao lado do prédio. Nessas regiões o inpainting foi adequado.

O resultado mostrado na Figura 36 é bastante satisfatório, mesmo com a ocasião de muitos retângulos se encostarem. Por conta dessa situação, nota-se que existem algumas partes na moldura central que não foram completamente transferidas da imagem *inpainted* para a imagem original.

Até agora os resultados de inpainting foram todos com imagens com a resolução reduzida. Na seção de Desenvolvimento do trabalho também mencionamos a possibilidade de recortar especificamente a área de interesse na imagem para que seja possível manter a resolução original intacta. Outra razão para justificar a seleção e recorte de porções específicas é que eventualmente a imagem total pode ser muito complexa, com muitas cores ou muitos tons da mesma cor, o que pode acabar gerando máscaras com detalhes excessivos que frequentemente são indesejáveis.

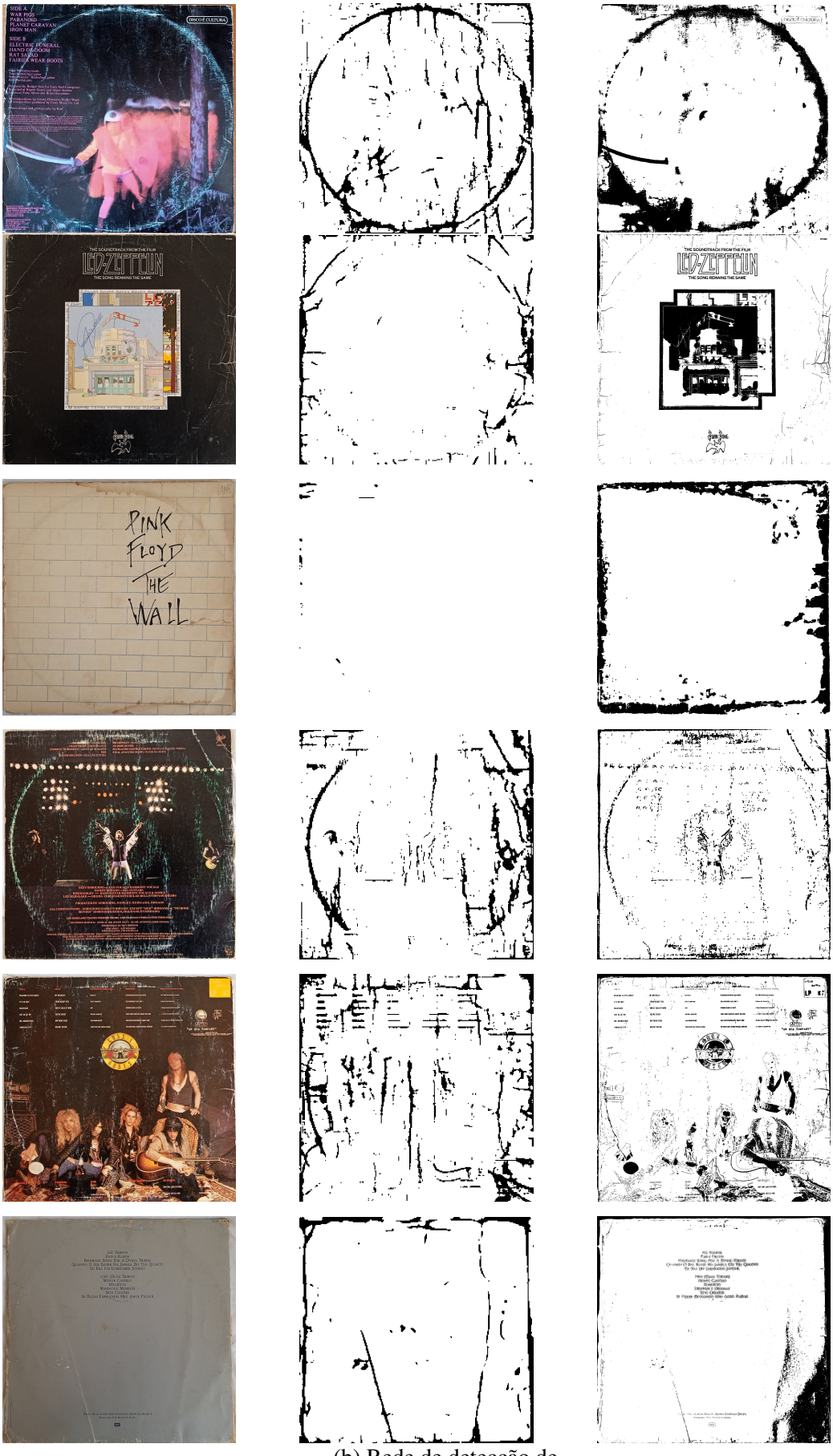
Um exemplo de imagens com essas características está presente na Figura 37. O que desejamos remover das imagens, nesse caso, são as assinaturas presentes nelas exclusivamente. Recortando as assinaturas, reduzimos a quantidade de elementos presentes nas imagens antes de gerar as máscaras, conseguindo preservar elementos que não queremos que sejam removidos e também reduzindo o custo computacional de geração das máscaras.

As imagens, antes de serem utilizadas para gerar as máscaras, foram segmentadas com k-médias, sendo a da Figura 38 com k = 4, a Figura 40 com k = 3 e as Figuras 39 e 41 com k = 6. Depois disso foi aplicado o *threshold* com HSV, considerando os valores dos pixels das assinaturas de cada imagem. Como pós-processamento foi feita a operação de dilatação nas

máscaras e uma edição manual para eliminar elementos indesejados nas máscaras.

Após a inserção dos recortes *inpainted* de volta nas imagens, como na Figura 42, o processo está concluído. A etapa de inserção dos recortes ainda envolve um processo manual e necessita que se guarde a posição das áreas recortadas para que se possa colocá-las de volta no exato local depois de serem submetidas ao *inpainting*. Apesar disso, observamos que, com essa pequena intervenção manual, além da resolução original da imagem ser mantida, as assinaturas foram apagadas com sucesso da imagem.

Figura 27 – Comparação entre máscaras



(a) Imagem danificada

(b) Rede de detecção de arranhões

(c) *Threshold*

Fonte: O autor, 2023

Figura 28 – Etapas do *Inpainting* do Paranoid 256×256.

(a) Encarte danificada

(b) Máscara

(c) Resultado do *inpainting*

Fonte: O autor, 2023

Figura 29 – Etapas do *Inpainting* do Paranoid 512×512.

(a) Encarte danificado

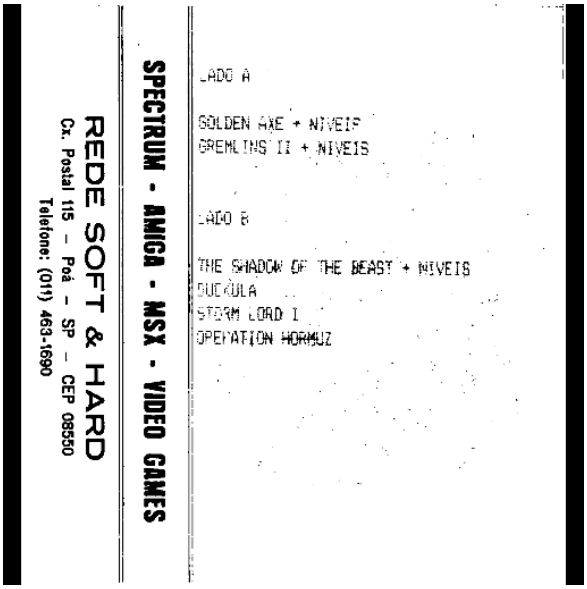
(b) Máscara

(c) Resultado do *inpainting*

Fonte: O autor, 2023

Figura 30 – Etapas do *Inpainting* da luva de fita 512×512.

(a) Imagem original da luva



(b) Máscara

(c) Resultado do *inpainting*

Fonte: O autor, 2023

Figura 31 – Etapas do *Inpainting* da luva de fita 512×512 com máscara dilatada.

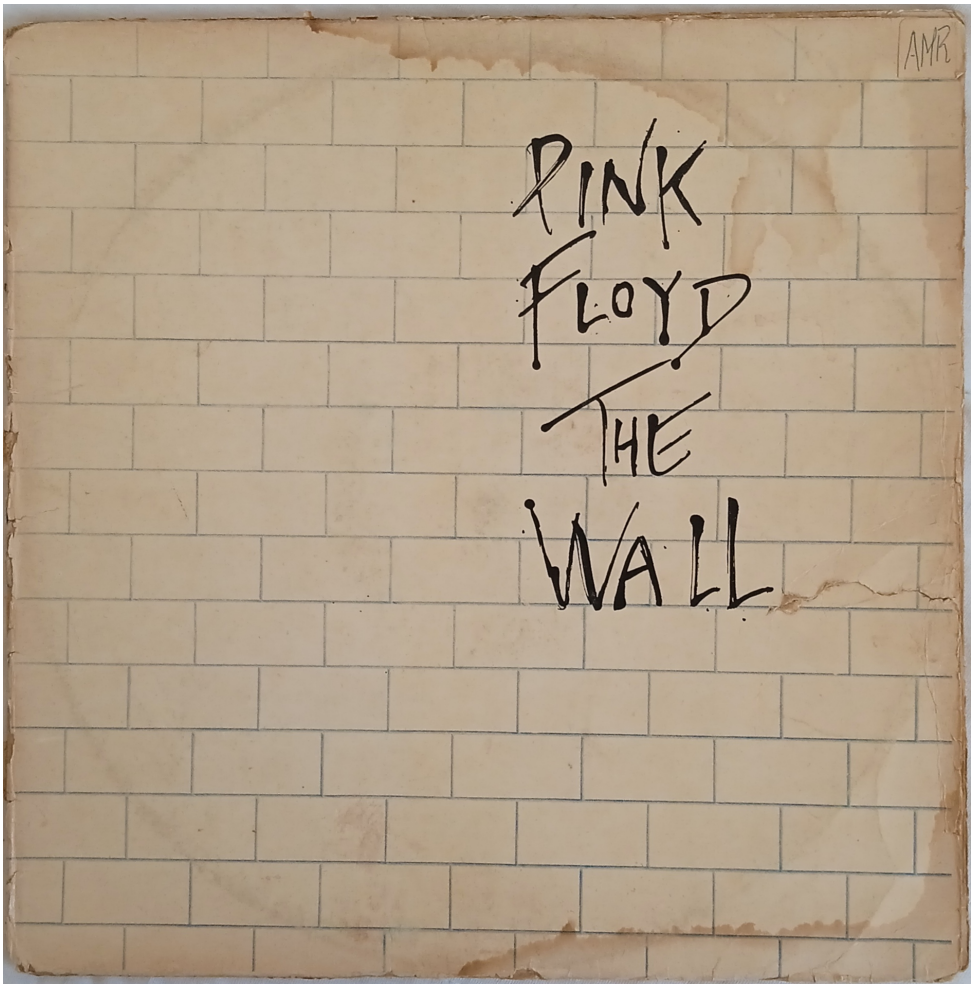
(a) Imagem original da luva

(b) Máscara dilatada

(c) Resultado do *inpainting*

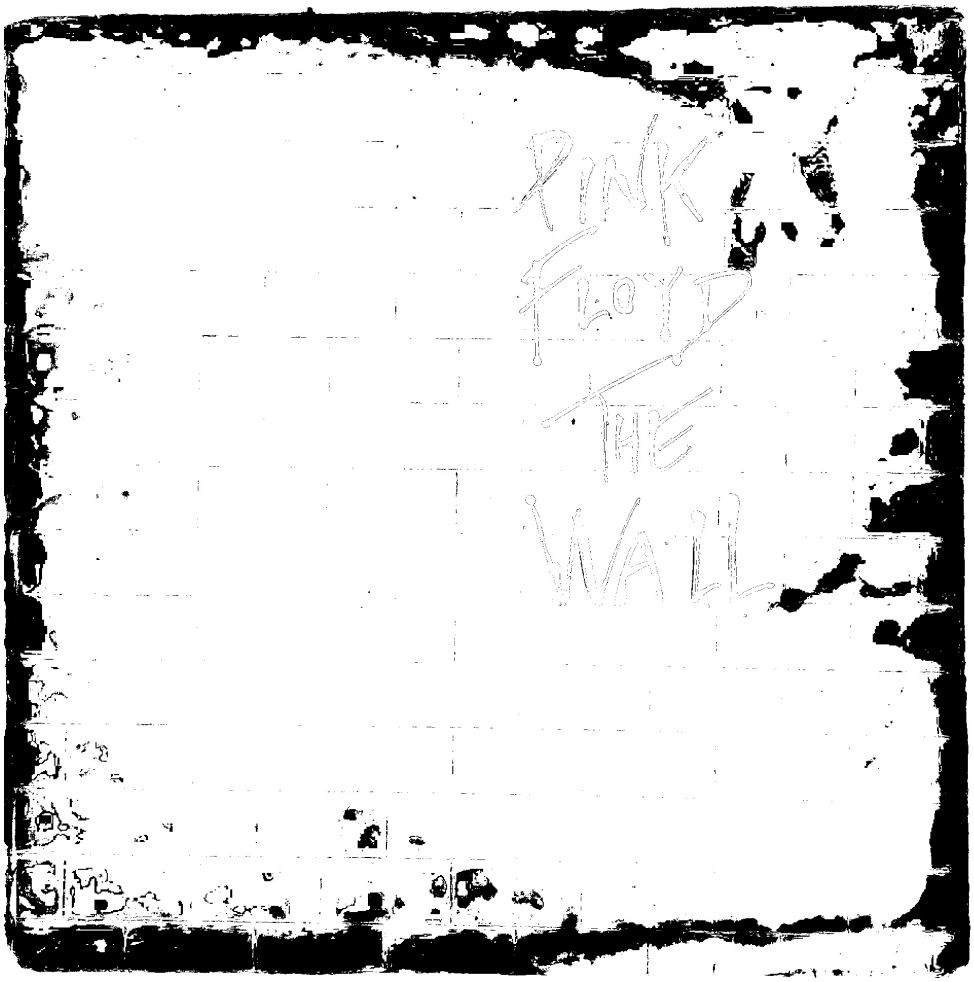
Fonte: O autor, 2023

Figura 32 – Disco The Wall danificado.



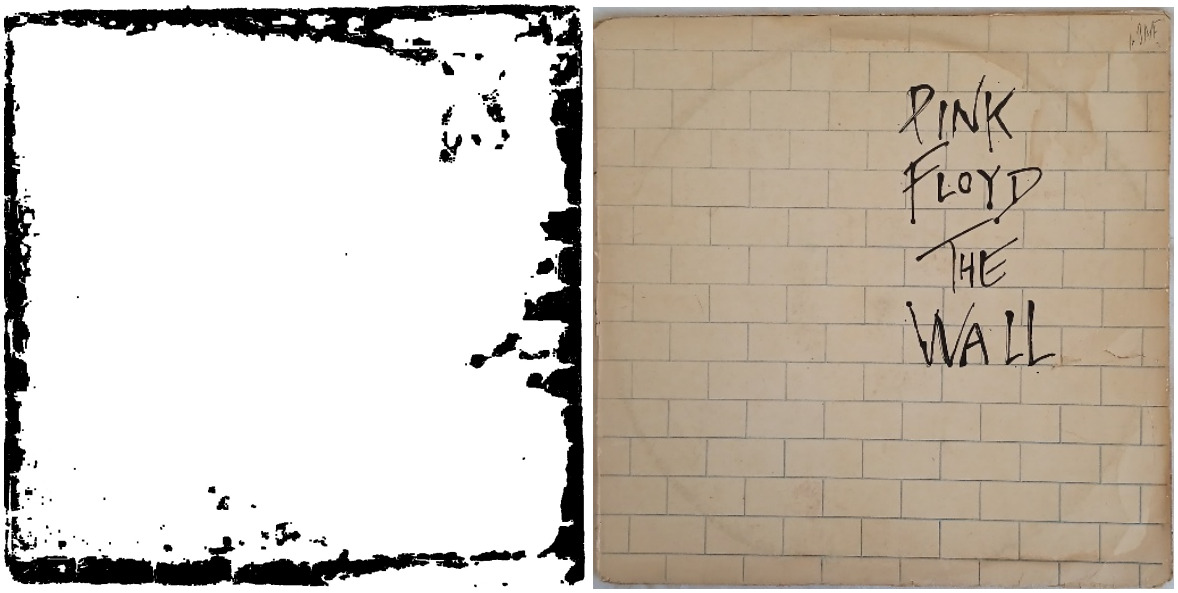
Fonte: O autor, 2023

Figura 33 – Máscara Disco The Wall.



Fonte: O autor, 2023

Figura 34 – Etapas do *Inpainting* do The Wall 512×512.



(a) Máscara pós abertura (b) Resultado do *inpainting*

Fonte: O autor, 2023

Figura 35 – Resultado do *inpainting* aplicado na Figura 18 com a Figura 21 como máscara.

Fonte: O autor, 2023

Figura 36 – Resultado do Algoritmo 2 512×512 utilizando as Figuras 18 e 35.

(a) Máscara demarcada com retângulos (b) Resultado do algoritmo

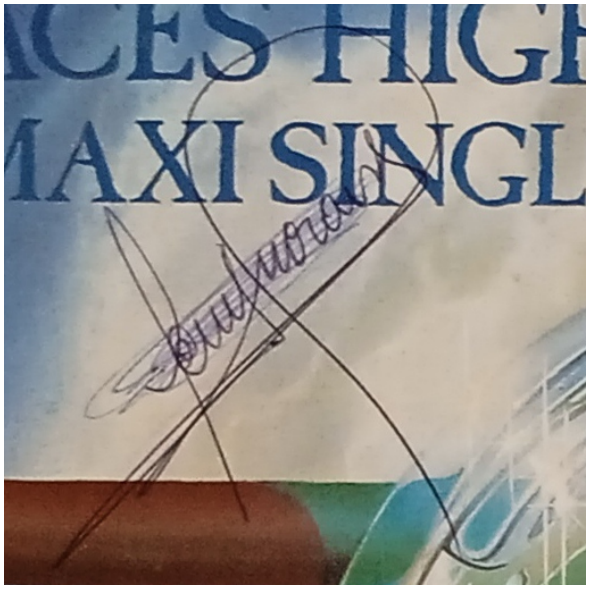
Fonte: O autor, 2023

Figura 37 – Aces High danificado na resolução original.

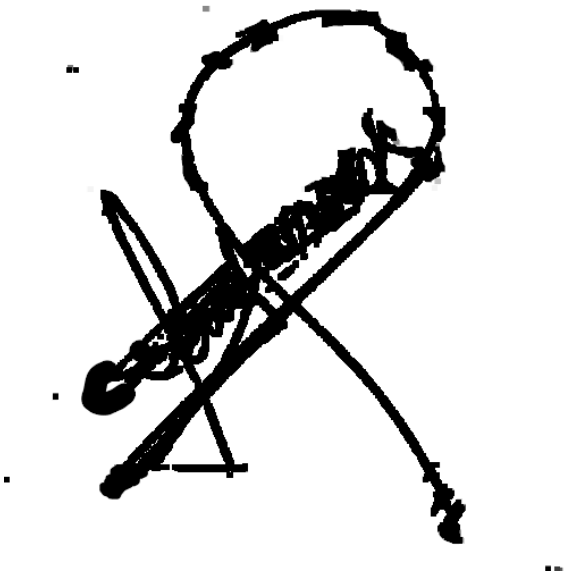
(a) Frente do encarte (b) Atrás do encarte

Fonte: O autor, 2023

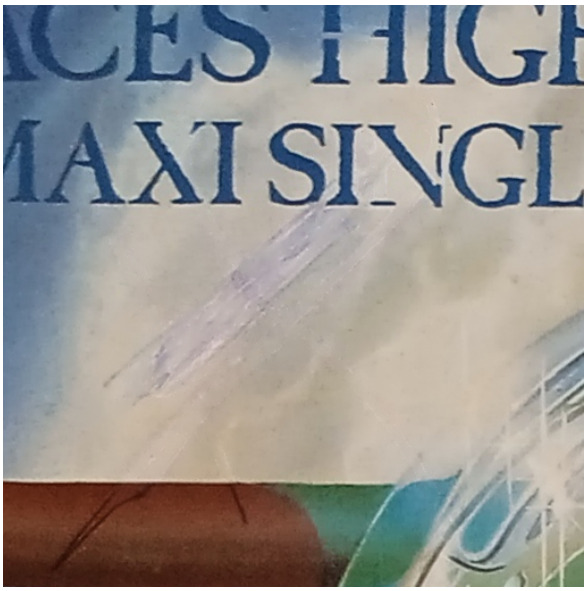
Figura 38 – Etapas do *inpainting* na assinatura da esquerda, frente do Aces High 512×512.



(a) Recorte da assinatura



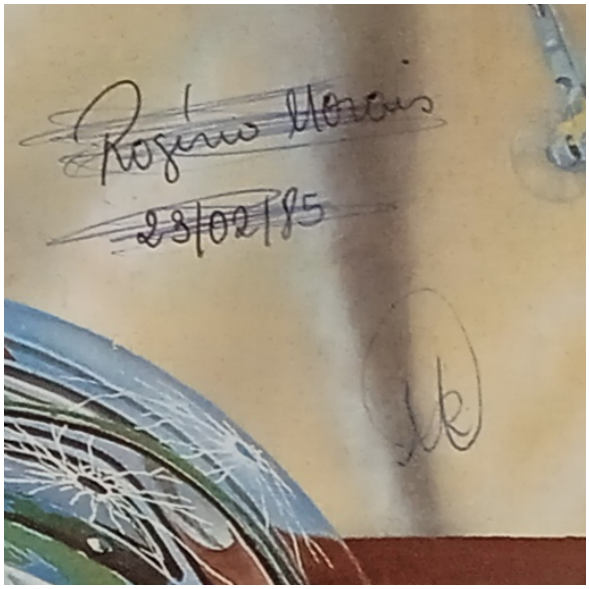
(b) Máscara



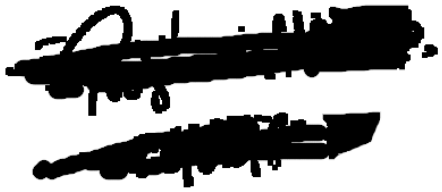
(c) Resultado do *inpainting*

Fonte: O autor, 2023

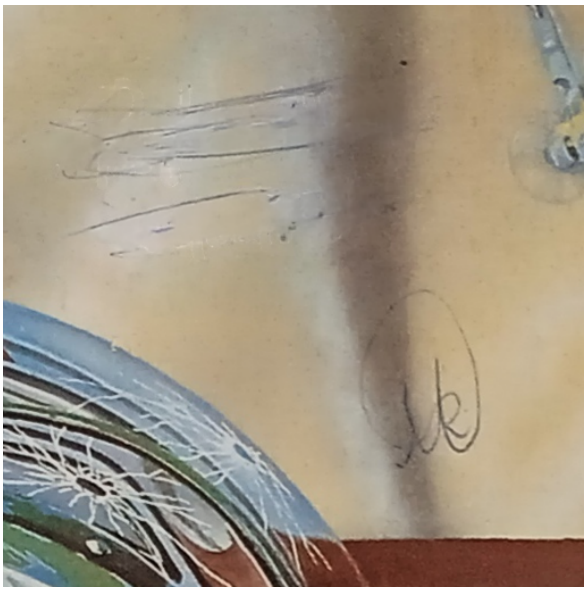
Figura 39 – Etapas do *inpainting* na assinatura da direita, frente do Aces High 512×512.



(a) Recorte da assinatura



(b) Máscara



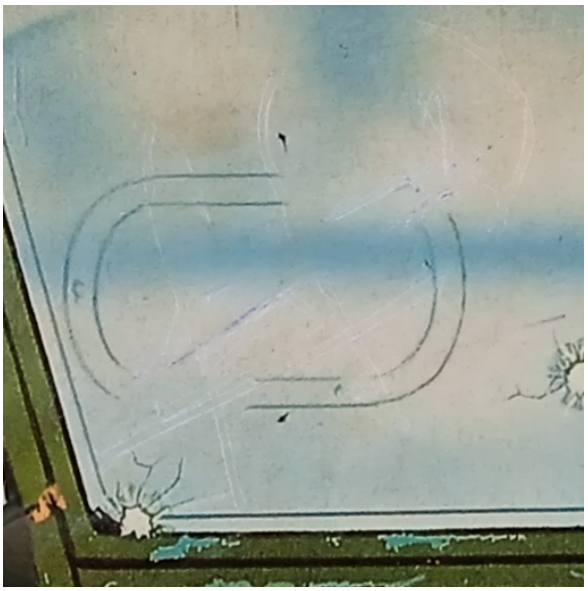
(c) Resultado do *inpainting*

Fonte: O autor, 2023

Figura 40 – Etapas do *inpainting* na assinatura da esquerda, atrás do Aces High 512×512.

(a) Recorte da assinatura

(b) Máscara

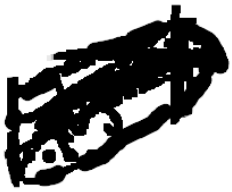


(c) Resultado do *inpainting*

Fonte: O autor, 2023

Figura 41 – Etapas do *inpainting* na assinatura da direita, atrás do Aces High 512×512.

(a) Recorte da assinatura



(b) Máscara

(c) Resultado do *inpainting*

Fonte: O autor, 2023

Figura 42 – Aces High após colagem dos resultados do *inpainting*.

(a) Frente do encarte (b) Atrás do encarte

Fonte: O autor, 2023

5 CONSIDERAÇÕES FINAIS

Este trabalho foi desenvolvido com o intuito de auxiliar a geração de máscaras para certos níveis de detalhes, como defeitos estruturais ou elementos indesejados nas imagens, e demonstrar que há a opção de fazê-lo sem ter que depender de algo como redes neurais profundas ou outros métodos computacionalmente exigentes. Para isso foram pesquisadas técnicas de processamento de imagens que pudessem ser usadas para esses fins.

Como a ideia da máscara para o *inpainting* é marcar a área onde se deseja que ele ocorra, logo foi considerada a ideia de utilizar técnicas de *thresholding* para segmentar elementos da imagem. Considerando que queriamos destacar partes específicas da imagem, foi utilizado, majoritariamente, métodos baseados em *thresholding* considerando o valor de HSV dos pixels do objeto desejado. Desta forma, a imagem da máscara produzida ao final do processamento já possui vários dos elementos indesejados eliminados.

A obtenção de um bom valor de HSV não é trivial, principalmente em imagens com muitas cores e tons, de forma que para isso foi utilizado o algoritmo de k-médias como pré-processamento da máscara, assim uniformizando as cores da imagem e diminuindo a quantidade de cores, baseando-se no valor de k. Desta forma, foi possível obter valores de HSV apropriados para a aplicação do *thresholding*.

Após a realização de testes com os passos anteriormente descritos, mostrou-se necessário, em alguns casos, realizar um pós-processamento da máscara. Para essa finalidade utilizamos operações morfológicas e filtro de mediana. Essas operações foram empregadas para acentuar os elementos da máscara ou atenuar outros elementos.

Ainda comparamos as máscaras geradas com *thresholding* com as geradas por uma rede neural de detecão de arranhões. Certamente, se tratando de sua especialidade, as máscaras da rede foram muito boas em detectar os arranhões e até manchas mais fortes. Em comparação, nossos resultados também foram capazes de detectar os arranhões, manchas, contornos e outros detalhes. Ambas acabaram, por vezes, identificando detalhes que, no caso da máscara ser utilizada em um *inpainting*, não seriam interessantes de permanecer na máscara final, como letras e silhuetas, sendo que as deste trabalho apresentam esses detalhes mais acentuados.

O trabalho ainda apresenta maneiras de aproveitar a imagem após o *inpainting*, ou ainda a imagem em sua resolução original, para obtermos uma imagem final com maior fidedignidade o possível.

Os resultados de *inpainting* usando as máscaras geradas e tratadas com as técnicas utilizadas foram bastante positivos. Claro que devemos lembrar que, em se tratando de imagens reais, existem muitos tipos de defeitos e só um *inpainting*, em diversos casos, pode acabar não sendo o suficiente, bem como o resultado de sua aplicação pode ser quase ótimo.

Em geral, a escolha e ordem em que os processos apresentados foram aplicados envolveu intervenção manual. Em se tratando de uma única imagem alvo, isso não é um grande impedimento, porém, não é viável para criar as máscaras de um grande *dataset* de maneira

sequencial.

Contudo, as técnicas apresentadas servem como um passo alternativo para a restauração de uma imagem. É necessário ressaltar que imagens reais, como as utilizadas neste trabalho, possuem muitos problemas estruturais diferentes, então apenas uma operação de *inpainting* frequentemente não será a única transformação que deverá ser feita para alcançar uma imagem completamente restaurada.

Para trabalhos futuros visamos investigar e obter maneiras de aplicar as técnicas de processamento de imagens mencionadas de maneira automática e eficiente, para que elas possam ser alternativas a soluções de redes neurais, ou até combinadas com estas, para grandes *datasets*. Outra linha é pensar em maneiras de construir um pipeline para restauração de imagens usando vários algoritmos de processamento de imagens e visão computacional do estado da arte.

Referências

FISHER, R. et al. Hypermedia image processing reference. **England: John Wiley & Sons Ltd**, p. 118–130, 1996.

GONZALEZ, R.; WOODS, R. **Digital Image Processing**. [S.l.]: Pearson, 2018. ISBN 9780133356724.

GOYAL, M. Morphological image processing. **IJCST**, Citeseer, v. 2, n. 4, p. 59, 2011.

GU, J. et al. **Image Quality Assessment for Perceptual Image Restoration: A New Dataset, Benchmark and Metric**. 2020. Disponível em: ⟨https://arxiv.org/abs/2011.15002⟩.

LIU, G. et al. Image inpainting for irregular holes using partial convolutions. In: **Proceedings of the European Conference on Computer Vision (ECCV)**. [S.l.: s.n.], 2018.

LUGMAYR, A. et al. Repaint: Inpainting using denoising diffusion probabilistic models. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2022. p. 11461–11471.

RUSS, J. C.; RUSS, J. C. **Introduction to image processing and analysis**. [S.l.]: CRC press, 2017.

SHIRLEY, P.; ASHIKHMIN, M.; MARSCHNER, S. **Fundamentals of Computer Graphics**. [S.l.]: Taylor & Francis, 2005. (Ak Peters Series). ISBN 9781568812694.

WAN, Z. et al. Bringing old photos back to life. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2020. p. 2747–2757.

YADAV, J.; SHARMA, M. A review of k-mean algorithm. **Int. J. Eng. Trends Technol**, Citeseer, v. 4, n. 7, p. 2972–2976, 2013.

YOUNG, I. T.; GERBRANDS, J. J.; VLIET, L. J. van. **Fundamentals of Image Processing**. [S.l.]: TU Delft, Faculty of Applied Physics, Pattern Recognition Group, 1998. ISBN 9789075691016.

YU, J. et al. Free-form image inpainting with gated convolution. In: **2019 IEEE/CVF International Conference on Computer Vision (ICCV)**. [S.l.: s.n.], 2019. p. 4470–4479.