

Relatório do EP de MAC0417-MAC5768

Lucas Prates Talles Viana Eduardo Kapp Henrique Damasceno
João Salles Ricardo Tolesano Tore Fossland

2 de setembro de 2022

Conteúdo

1	Introdução	3
1.1	Cronograma	3
2	Objetivos	3
3	Preparação do conjunto de dados	4
3.1	Aquisição e descrição dos dados	4
3.1.1	Padronização de nomenclatura	4
3.2	Formatação dos dados	5
3.3	Aumento de dados	6
3.4	Normalização dos dados	7
3.5	Análise das classes	8
3.5.1	Protótipo médio	8
3.5.2	Histograma médio	8
3.5.3	Média e desvio padrão do histograma médio	10
4	Segmentação e extração da caixa delimitadora	12
4.1	Construção do Ground Truth	12
4.2	Métricas de avaliação	13
4.3	Modelos de segmentação	15
4.3.1	BETQF	15
4.3.2	Método utilizando Watershed	19
4.4	Desempenho geral	22
4.4.1	Resultados BETQF	22
4.4.2	Resultados pipeline com Watershed	28
5	Descrição e classificação de objetos	29
5.1	Estratégia de descrição	30
5.2	Split de dados e Cross Valuation	31
5.3	Estratégia de classificação	32
5.3.1	Métricas de avaliação do classificador	32
5.4	Classificadores utilizados	32
5.5	Resultados dos classificadores	33
5.5.1	PCA+SVM	33
5.5.2	LinearSVC	35
5.5.3	KNN	37
5.5.4	QDA	38
5.5.5	LDA	39
5.5.6	LogisticRegression	41
5.5.7	Rede Neural - MLP	42
6	Conclusão do trabalho	44

A Cronograma - Gannt Chart	45
B Visualização estilo MNIST sobre as operações no Dataset	46

1 Introdução

As revoluções tecnológicas do século XX e XXI mudaram completamente a forma como a sociedade produz e consome informação. O mundo encontra-se cada vez mais conectado através da internet, e cada interação faz com que existe fluxo de dados. Para além disso, uma vasta quantidade de sensores registram dados das mais diversas fontes. Assim, uma quantidade extraordinária de informação que é produzida e armazenada. Todos esses aspectos justificam o nome que muitos atribuem ao nosso período histórico: A era da informação.

Uma das fontes mais interessantes e úteis de informação são as imagens digitais. Interagimos diariamente com imagens digitais para fins recreativos e também para expandir nosso conhecimento acerca da natureza. De fato, criamos imagens utilizando de sensores sensíveis a fenômenos físicos não detectáveis pelo olho humano. Por diversas razões, por vezes é necessário transformar tais imagens para que fiquem adequadas para suas aplicações. As áreas do conhecimento que estudam a geração, processos e aplicações de imagens digitais são chamadas de visão computacional e processamento de imagens digitais.

A primeira aplicação vêm à mente está relacionada aos processamentos básicos feitos por programas editores de imagens. Essas técnicas permitem que os usuários filtrem e refinem as informações disponíveis como, por exemplo, na aplicação filtros para aumentar o contraste, destacar objetos e reduzir o ruído. Conforme descrito em [2], as imagens resultantes auxiliam em tomadas de decisão e compreensão em áreas como medicina, astronomia, meteorologia, monitoramento de qualidade de produtos, entre outras.

Para além da sua utilização como uma forma de melhorar a tomada de decisão humana, há também o interesse em utilizar suas técnicas em processos automáticos. Encontramos em [1] uma tabela descrevendo utilizações em diversas áreas, como por exemplo detecção automática de caracteres, biometria, classificação de objetos, detecção de casos clínicos a partir de imagens de imagens médicas, carros autônomos, entre outros. Muitos desses problemas são considerados problemas da área de visão computacional. Esta é uma área interdisciplinar e frutífera, sendo responsável pelo desenvolvimento e aprimoramento de diversas tecnologias modernas.

Considerando a diversidade e relevância da área, é importante entender, de forma precisa, como são definidos os conceitos fundamentais, o fluxo de trabalho em uma aplicação, como melhor utilizar as ferramentas, e os tipos de problemas que podemos resolver. Portanto, temos o interesse em desenvolver uma base sólida, tanto teórica quanto prática, dos principais conceitos e processos que compõem uma aplicação da área de visão computacional e processamento de imagens.

1.1 Cronograma

O cronograma encontra-se representado pelo Gantt chart no Apêndice A.

2 Objetivos

Nós temos três principais objetivos nesse trabalho sobre a análise de imagens digitais. O primeiro objetivo é adquirir e armazenar imagens de itens domésticos. Essas imagens devem preencher os requisitos para formar um conjunto de dados adequado. Os requisitos devem ser decididos em grupo, baseados nas diretrizes providenciadas pelo professor. Depois, queremos organizar o conjunto de dados no Google Drive, em pastas e de maneira hierárquica. Incluiremos duas tabelas de metadados das imagens coletadas. A primeira tabela conterá informação geral sobre os arquivos, e a segunda descreverá cada classe escolhida em detalhe.

O segundo objetivo se constitui no processamento e normalização do conjunto de dados obtido no item 1. Para conseguir isso, providenciaremos a lógica para resolver três tarefas essenciais de visão computacional: Data Augmentation, normalização e análise da variação das classes. Sobre Data Augmentation, o objetivo é converter o conjunto de dados para grayscale e criar as cinco funções de Data Augmentation especificadas na descrição do projeto. Para a análise da variação das classes nós precisamos gerar um conjunto de dados normalizado utilizando o método de equalização de histogramas. Teremos assim três conjuntos de dados: o original, o aumentado proveniente de Data Augmentation, e o normalizado. Então, utilizaremos funções de análise nos conjuntos de dados obtidos. Um exemplo de função a ser utilizada é uma que calcula o protótipo médio de cada classe.

O terceiro objetivo é realizar segmentação e classificação nos dados processados. Inicialmente, queremos segmentar os objetos de interesse tanto manualmente quanto automaticamente. Então, desenvolveremos uma função que permite obter uma feretbox (bounding box). Feito isso, queremos utilizar a feretbox para extraír features da imagem. Finalmente, utilizaremos um algoritmo para classificar os objetos com base nessas features.

Durante o projeto, utilizaremos Jupyter Notebooks para apresentar, processar e visualizar os dados. Também explicaremos cada parte do trabalho, nossas decisões de implementações, e providenciaremos uma conclusão. Jupyter Notebooks e o Github permitirão que o grupo trabalhe nas mesmas tarefas em paralelo.

3 Preparação do conjunto de dados

3.1 Aquisição e descrição dos dados

Foram coletadas imagens relativos à 10 classes de objetos diferentes. As classes de objetos selecionadas foram: caneta, celular, chave, chinelo, garrafa, livro, portacopo, prato, sapato e tesoura. Selecioneamos objetos propositalmente pequenos para que fosse possível deitar o objeto e tirar fotos de cima.

Para cada classe, foram selecionados 4 objetos distintos, e foram tiradas fotos considerando 3 planos de fundo diferentes e 4 iluminações diferentes. As iluminações foram: dia interior, dia exterior, noite interior e noite exterior. Com relação ao fundo, exigiu-se que dois dos 3 fossem monocromáticos de cores diferentes, e que o terceiro fosse não monocromático. Um total de 1440 imagens foram obtidas.

As imagens foram capturadas utilizando os celulares dos membros do grupo. Como há uma diferença no modelo dos celulares e, consequentemente, nas câmeras, variação na qualidade das imagens era esperado. Para tirar a foto, o objeto objeto foi posicionado "deitado" contra o fundo e capturou-se a foto por cima.

O sumário do conjunto de dados é definido pela Tabela 1.

Tabela 1: Sumário do Dataset

Descrição	Valor
Nº de Classes	10
Nº de Imagens	1440
Tamanho da Base (MB)	601.89
Resolução das Imagens	640x480

O sumário por classe é definido pela Tabela 2.

Tabela 2: Sumário por Classe

Descrição	Valor
Nº de Objetos	4
Nº de Poses	3
Nº de Variações de Iluminação	4
Nº de Fundos	3

Uma amostra de 36 imagens do Dataset é apresentada na Figura 1. Demais visualizações do Dataset sob diferentes transformações, a serem introduzidas nas Subseções a seguir, são apresentadas no Apêndice deste trabalho.

3.1.1 Padronização de nomenclatura

Para facilitar o rótulo das imagens e os processos de preparação de dados, foi definida uma nomenclatura padrão para as imagens. A nomenclatura descreve as principais características da imagem coletada em termos de fundo, iluminação e demais configurações, conforme descrito na Subseção 3.1.

A nomenclatura padrão é definida como: (classe)-(objeto)-(pose)-(iluminação)-(fundo). Dessa forma, foi possível verificar, de forma sistemática, a presença de todas as configurações necessárias para cada classe de objeto, garantindo o cumprimento das restrições do projeto.

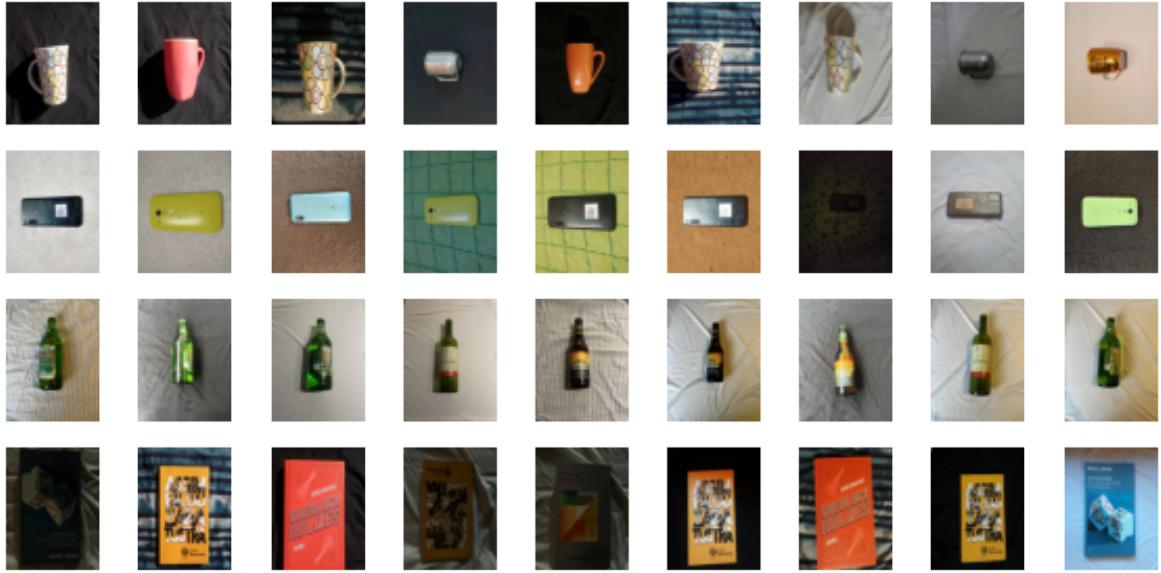


Figura 1: Conjunto de 36 imagens obtidas do Dataset, com visualização estilo [MNIST](#).

3.2 Formatação dos dados

Foram aplicadas algumas técnicas de processamento de imagens para padronizar o formato de cada uma das imagens coletadas. O intuito desse processo foi simplificar os dados para as aplicações de segmentação e classificação.



Figura 2: Pipeline de preparação dos dados. A imagem original é redimensionada para uma imagem 256x256 utilizando a função [resize](#). Em seguida, aplica-se a função [rgb2gray](#). As imagens finais foram salvas em uint8.

A Figura 2 descreve o pipeline inicial de transformação aplicado em cada imagem. Foi utilizada a função [resize](#) e [rgb2gray](#) do módulo `skimage`^[7] para redimensionar as imagens para 256x256 e transformá-las para a escala de cinza, respectivamente. As imagens finais foram salvadas no formato uint8.

Uma inspeção visual indicou uma boa qualidade geral das imagens, sem deformações drásticas ou perda de informação. É possível, entretanto, notar que algumas imagens ficaram demasiada escuras.

A Figura 3, ilustra uma imagem na qual é difícil identificar o objeto em questão por conta da baixa iluminação. A imagem original já apresenta o problema referido, e a imagem em cinza é quase idêntica.

Assim, ponderou-se em inserir, no final do pipeline, uma equalização por histograma ou transformação γ com a finalidade de clarear a imagens em que as fotos foram tiradas no exterior no período da noite. A figura mostra o resultado da aplicação para a imagem discutida acima.

A Figura 4 mostra que tanto a equalização por histograma como a transformação γ inserem "artefatos" indesejados, que seriam os diversos pontos claros e escuros. Isso pode vir a dificultar a tarefa de segmentação, além de viesar os histogramas à serem computados. Assim, preferiu-se manter as imagens obtidas pelo pipeline 2.



Figura 3: (a) Imagem original de uma garrafa marrom em um fundo escuro no exterior de noite. É difícil distinguir visualmente o entorno do objeto devido à iluminação; (b) Imagem em cinza 256x256 apresentando o mesmo problema da original.

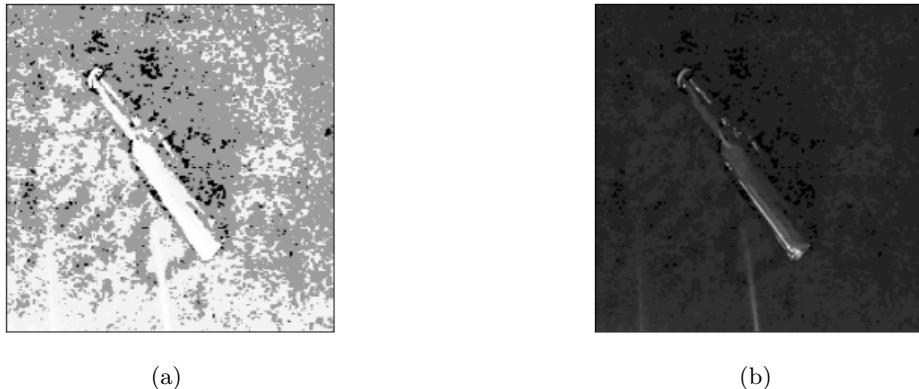


Figura 4: Imagem resultante após a aplicação da (a) equalização por histograma; (b) transformação γ com $\gamma = 0.4$.

3.3 Aumento de dados

Para expandir o conjunto de dados, foram aplicadas transformações de mudança de intensidade. Isto é, a posição relativa do objeto na imagem não muda, o que muda é apenas a intensidade dos pixels. Foram aplicadas as transformações **logarítmica**, a **exponencial**, a **convolução com filtro da média** utilizando um kernel $K 3 \times 3$ e a soma de fundo com gradiente.

Antes de aplicar cada uma das transformações, as imagens foram convertidas para float64 a fim de evitar saturações indesejadas. A conversão se deu dividindo cada um dos valores de intensidade por 255. A imagem resultante após a aplicação também tinha formato float64. Para converter novamente para uint8, foi aplicado um reescalonamento na imagem. Seja f a imagem após a transformação, a imagem final g ficou dada por

$$g(x, y) = 255 \left(\frac{f(x, y) - \min_{x, y \in \{0, \dots, 255\}^2} f(x, y)}{\max_{x, y \in \{0, \dots, 255\}^2} f(x, y) - \min_{x, y \in \{0, \dots, 255\}^2} f(x, y)} \right), \quad \forall x, y \in \{0, \dots, 255\}^2 .$$

A Figura 5 mostra a diferença da imagem original para a imagem final a fim de mostrar como cada transformação alterou a intensidade dos pixels.

O kernel utilizado para a convolução com filtro da média foi definido como

$$K = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} . \quad (1)$$

Para a soma de fundo com gradiente, foi construída uma matriz $G 256 \times 256$ dada por

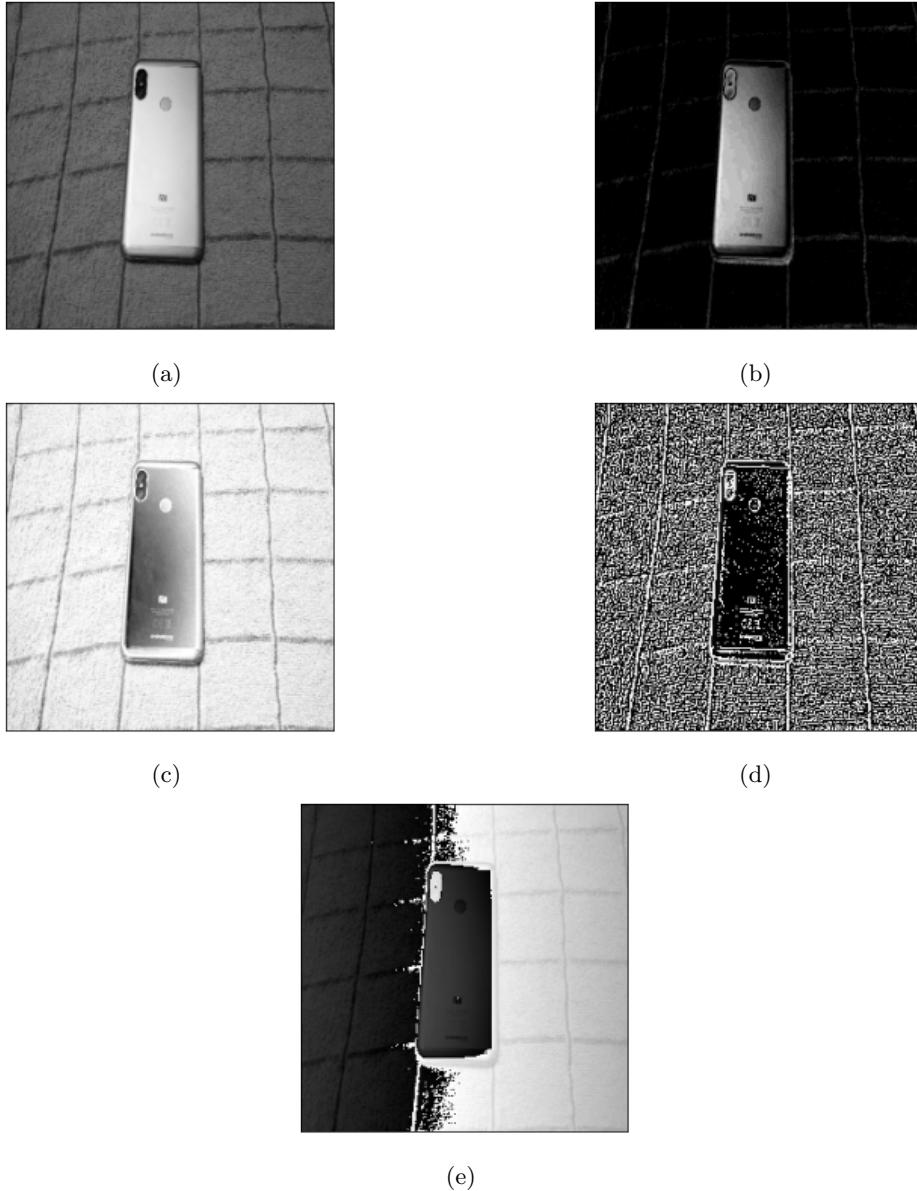


Figura 5: (a) Imagem original de um celular, de noite, no interior, com um fundo escuro; diferença da imagem original com a imagem após a transformação (b) logarítmica; (c) exponencial (d) convolução com o filtro da média utilizando um kernel $K 3 \times 3$ explicitado em 1; (e) soma de fundo com gradiente.

$$G(x, y) = x \quad , \forall x, y \in \{0, \dots, 255\} \quad .$$

Assim, fixada uma coluna, a intensidade dos pixels é constante, e a imagem vai clareando conforme crescemos as colunas. A Figura 6 mostra a imagem do fundo com gradiente.

As transformações acima indicadas foram aplicadas à cada imagem. Assim, o conjunto de dados aumentou em quatro vezes, finalizando com um total de $1440 \times 5 = 7200$ imagens.

Além de utilizar o módulo skimage já citado, também foi amplamente utilizado o módulo numpy [3].

3.4 Normalização dos dados

Partindo do conjunto de dados aumentado explicado anteriormente, aplicou-se uma transformação de [equalização de histograma](#) para criar um conjunto de dados normalizado. Essa transformação altera

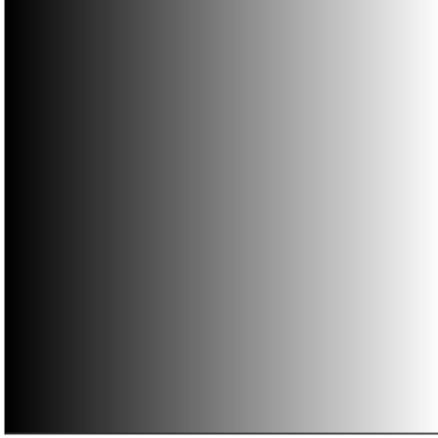


Figura 6: Fundo com gradiente considerado na transformação.

o histograma da imagem a partir de uma função de distribuição acumulada do próprio histograma. A saída é uma imagem normalizada de alto contraste, otimizando o uso dos níveis de cinza disponíveis.

3.5 Análise das classes

Para entender melhor as características de cada uma das classes, foram utilizadas três diferentes técnicas: construção do protótipo médio de cada classe, construção do histograma médio de cada classe e, por fim, o cálculo da média, variância e desvio-padrão do histograma médio de cada uma das classes.

3.5.1 Protótipo médio

Foi construído o protótipo médio de cada classe para cada os conjuntos de dados cinza, aumentando e normalizado. Isto nos permite identificar quais características são visualmente semelhantes aos objetos de uma mesma classe.

Para se obter tal protótipo, computou-se a média de todas as imagens de uma mesma classe, estratificando apenas pela posição um, dois ou três na qual o objeto se encontrava. A estratificação foi feita para se ter uma imagem média mais nítida do objeto.

A Figura 7 exibe os protótipos para as classes na posição 1. Para algumas classes, como chinelo, prato e celular, obteve-se um protótipo médio relativamente nítido, possivelmente pela semelhança entre os objetos selecionados. Para classes como livro, garrafa, caneca e livro, percebe-se um contorno comum aos objetos, porém o protótipo está borrado. Para o sapato, note que há duas formas diferentes, uma mais opaca que a outra. Isso provavelmente se deve ao fato de algumas imagens terem sido rotacionadas ao se capturar a foto. Finalmente, para as classes chave e tesoura, o ”contorno característico” da classe não fica nítido no protótipo médio. No geral, não se nota grandes diferenças entre os protótipos do conjunto de dados cinza, aumentado e normalizado.

3.5.2 Histograma médio

O histograma médio nos ajuda a entender a variabilidade da intensidade dentro das classes. Objetos com tons mais claros contra um fundo mais escuro terão histogramas mais concentrados em valores de 255 para o conjunto de dados cinza e aumentado. Para o conjunto normalizado, espera-se um histograma próximo a um uniforme.

Para computar o histograma médio, calculou-se a frequência cada nível de intensidade (codificado como um uint8) em todas as imagens. Após estas frequências serem computadas, dividiu-se cada valor pela soma total das frequências, obtendo-se assim as frequências relativas para cada nível de intensidade.

A Figura 8 mostra os histogramas para as classes garrafa, chinelo, portacopo, prato e livro. A Figura 9 mostra os histogramas para as classes tesoura, celular, sapato, prato e chave. Os histogramas

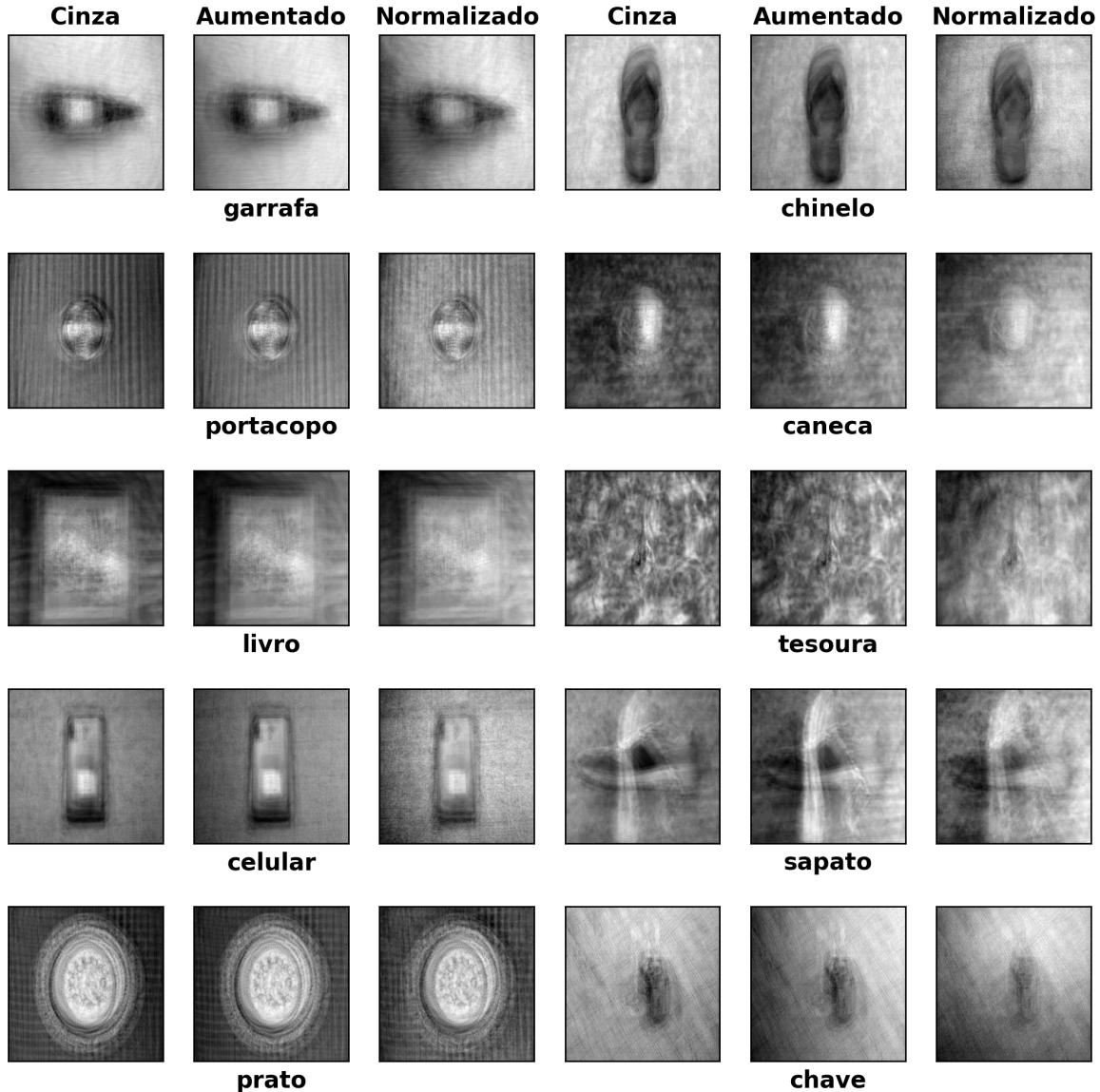


Figura 7: Figuras estilo MNIST para os protótipos das classes em cada um dos conjuntos de dados. Selecionou-se apenas os objetos que estavam na posição 1 para exibição, dado que a conclusão é semelhante para as outras posições.

de cada objeto referido estão em uma mesma linha, enquanto as colunas exibem os histogramas desses objetos no conjunto de dados cinza, aumentado e normalizado, respectivamente.

Fica evidente que os histogramas para a classe normalizada estão próximos de histogramas uniformes para todas as classes, conforme esperado teoricamente. As variações se devem ao fato de estarmos trabalhando com quantidades digitais.

Também é possível notar que os histogramas para o conjunto cinza é suave, significando que não há uma discrepância abrupta entre valores de intensidades próximos. Já para o conjunto de dados aumentado, notamos que ele segue a mesma forma histograma associado ao conjunto de dados cinza, porém é um versão não suave com vários saltos e discrepâncias entre valores próximos.

Considerando o conjunto de dados cinza, nota-se, em geral, uma saturação dos valores no 0 (preto) e poucos valores próximos de 255 (branco), possivelmente por conta das figuras tiradas durante a noite.

Desconsiderando os saturados, alguns dos histogramas possuem valores de intensidade mais dispersos e semelhantes a uma distribuição uniforme, como para a classe garrafa, livro, sapato e chinelo. Já outras classes, como caneca, chave e tesoura, o histograma assemelha-se à uma distribuição bimodal.

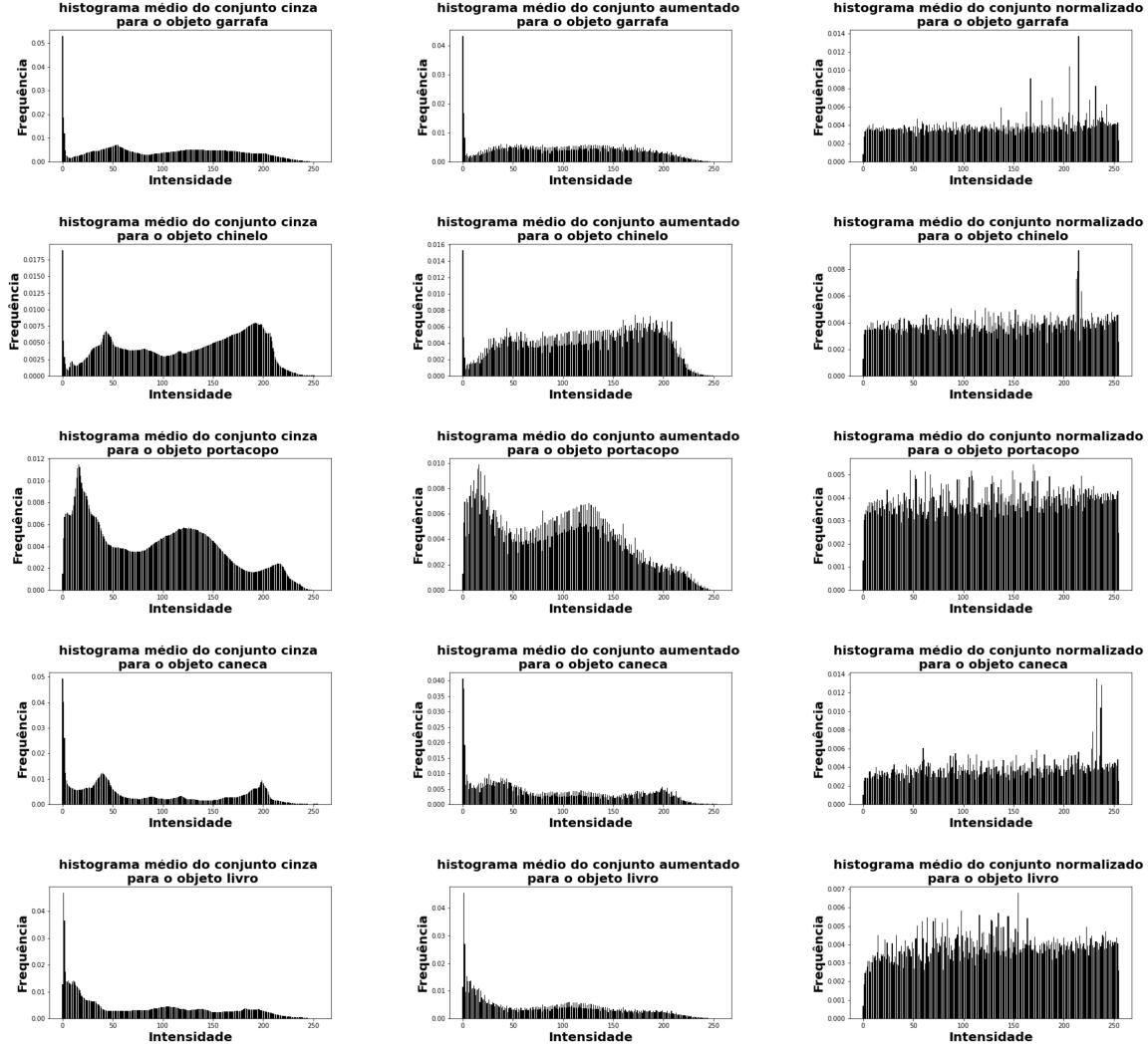


Figura 8: Histograma, por linha, para as classes garrafa, chinelo, portacopo, prato e livro. As colunas exibem os histogramas para os conjuntos de dados cinza, aumentado e normalizado, respectivamente.

Para as classes restantes, não fica tão nítido a forma da distribuição.

3.5.3 Média e desvio padrão do histograma médio

Por vezes, é mais simples analisar a distribuição dos dados considerando apenas algumas medidas resumo. Assim, selecionamos resumir a distribuição apresentando a média e o desvio padrão. A média provém informação de localização da distribuição, enquanto o desvio padrão nos fornece uma ideia da variabilidade. Vale ressaltar que, conforme analisado anteriormente, os dados não são unimodais, e portanto a média do histograma pode passar uma impressão incorreta da localização da distribuição.

Essas quantidades foram calculadas a partir do histograma de cada objeto em cada um dos conjuntos de dados. Recorde que os histogramas médios construídos foram normalizados de forma que a soma das frequências relativas soma 1. Assim, o histograma é, na verdade, a distribuição de probabilidade discreta dos valores de intensidade dos pixels. Daí, computamos essas quantidades utilizando as fórmulas teóricas de teoria da probabilidade. Seja p_i a frequência relativa (probabilidade de observar) do pixel com intensidade i , temos

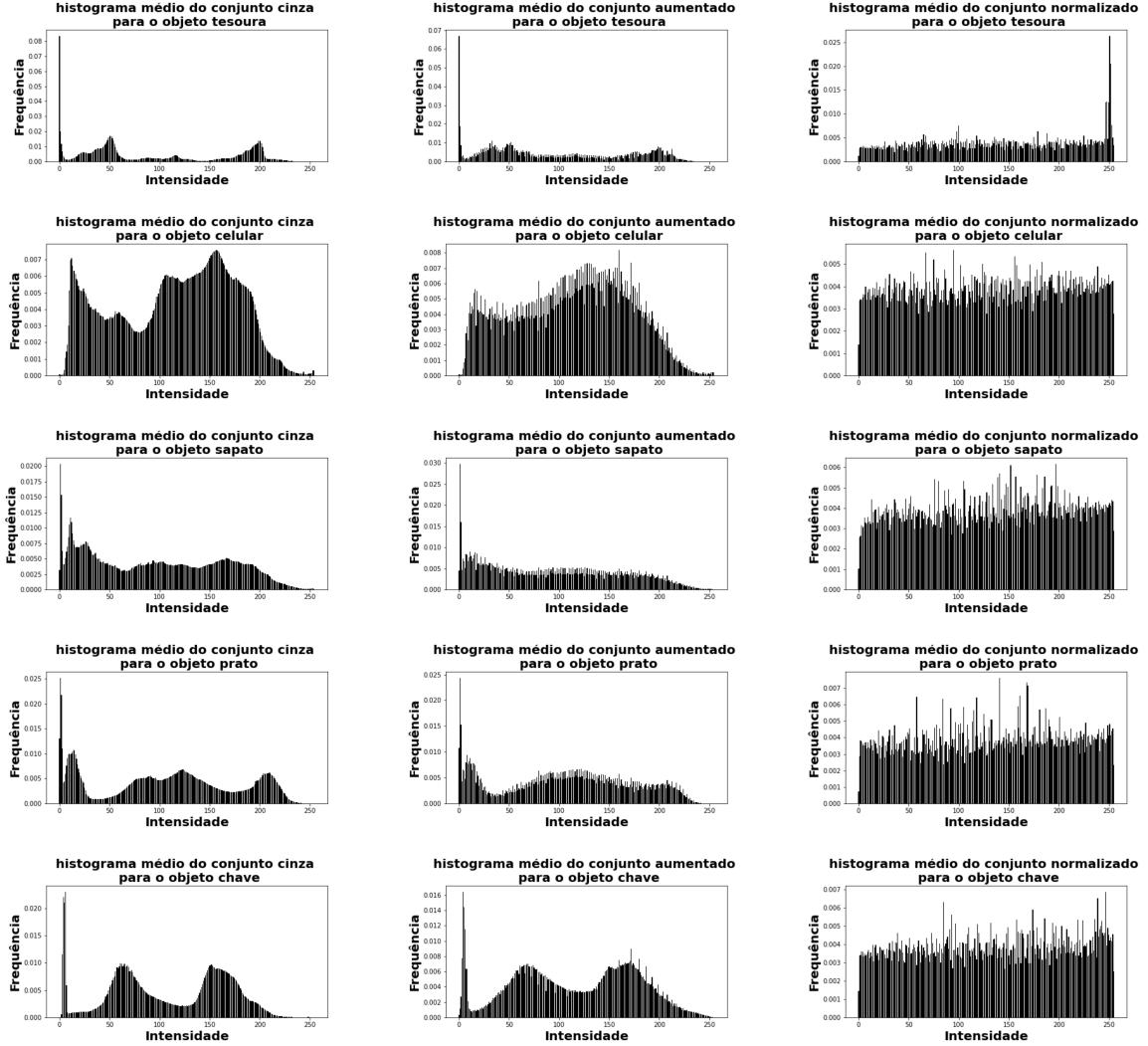


Figura 9: Histograma, por linha, para as classes tesoura, celular, sapato, prato e chave. As colunas exibem os histogramas para os conjuntos de dados cinza, aumentado e normalizado, respectivamente.

$$\begin{aligned}\mu &= \sum_{i=0}^{255} p_i \times i \\ \sigma &= \left[\left(\sum_{i=0}^{255} p_i \times i^2 \right) - \mu^2 \right]^{\frac{1}{2}}.\end{aligned}$$

Obteve-se assim: 1) a média e o desvio padrão para o conjunto de dados cinza μ_c e σ_c ; 2) a média e o desvio padrão para o conjunto de dados aumentado μ_a e σ_a ; 3) a média e o desvio padrão para o conjunto de dados normalizado μ_n e σ_n .

A Tabela 3 exibe essas medidas resumo. Note que a média e o desvio padrão do conjunto de dados cinza e a média do conjunto de dados aumentado estão muito próximas. Para todas as classes, a média está abaixo do 127, indicando que as figuras provavelmente há uma concentração de pixels mais próximos do preto. Uma justificativa plausível deste fato é o que foi observado nos histogramas: parece haver uma saturação no preto.

Para o conjunto de dados normalizado, a média está relativamente próxima, porém um pouco maior, da média de uma distribuição uniforme em $\{0, \dots, 255\}$, que teria média 127.5. Observa-se o mesmo no desvio padrão, em que o desvio padrão teórico de uma distribuição uniforme em $\{0, \dots, 255\}$

Objeto	μ_c	μ_a	μ_n	Objeto	σ_c	σ_a	σ_n
garrafa	102.51	103.12	133.52	garrafa	65.68	63.5	74.5
chinelo	121.97	120.31	130.95	chinelo	64.04	61.93	74.22
portacopo	91.17	93.12	129.87	portacopo	63.04	60.68	73.47
caneca	83.69	86.61	135.57	caneca	72.9	69.79	74.15
livro	77.74	81.3	130.96	livro	68.63	66.3	72.14
tesoura	93.4	95.22	140.47	tesoura	74.19	70.55	76.47
celular	114.96	112.59	129.69	celular	58.2	56.59	73.79
sapato	95.97	93.11	130.99	sapato	66.59	66.21	73.03
prato	105.63	105.96	130.71	prato	69.68	66.15	72.95
chave	107.6	114.85	131.32	chave	58.27	61.2	74.13

Tabela 3: Média e desvio padrão por objeto em cada um dos conjuntos de dados.

seria $\sqrt{\frac{255 \times 257}{12}} \approx 73,9$.

4 Segmentação e extração da caixa delimitadora

Conforme descrito nos objetivos, utilizamos técnicas clássicas de visão computacional para a segmentação e extração da caixa delimitadora do conjunto de dados aumentado. A tarefa de segmentação consiste em encontrar a região que define um determinado objeto, enquanto a extração da caixa delimitadora fornece uma região na qual se encontra o objeto. Assumiremos que a tarefa é detectar um único objeto contra um fundo.

Explicaremos em detalhes como foram construídos os algoritmos, compararemos seus resultados tanto por inspeção visual quanto por métricas específicas para essa tarefa.

4.1 Construção do Ground Truth

Para que fosse possível quantificar os resultados dos algoritmos foi necessário a construção do Ground Truth, um conjunto de dados o qual contém as imagens binárias dos objetos.

Foi feita uma segmentação manual utilizando a ferramenta "free selection tool" e "elipse selector" do software GIMP [6] para definir o contorno do objeto, e então se preencheu o interior de branco e o a parte exterior de preto. A Figura 10 mostra a comparação da imagem em cinza com o Ground Truth de dois objetos usando as duas ferramentas.

Selecionou-se cerca de 15% das amostras do conjunto de dados cinza, estratificadas por classe. A aleatorização dos objetos de uma mesma classe garante que não haverá viés induzidos pela escolha dos objetos. A estratificação por classes garante que todas elas estarão igualmente representadas. A Tabela 4 mostra o número de objetos rotulados no Ground Truth estratificado por classe e iluminação/ambiente.

Objeto	dia-exterior	dia-interior	noite-exterior	noite-interior	Diferentes	TOTAL
tesoura	5	6	6	4	4	21
garrafa	6	5	7	4	4	22
chave	8	5	2	7	4	22
prato	6	8	2	5	4	21
livro	7	8	7	5	4	27
sapato	4	13	4	5	4	26
chinelo	4	7	9	4	4	24
celular	10	7	4	3	4	24
portacopo	3	8	5	6	3	22
caneca	9	9	4	5	4	27
TOTAL	62	76	50	48	39	236

Tabela 4: Número de objetos rotulados no Ground Truth por classe e iluminação-ambiente.

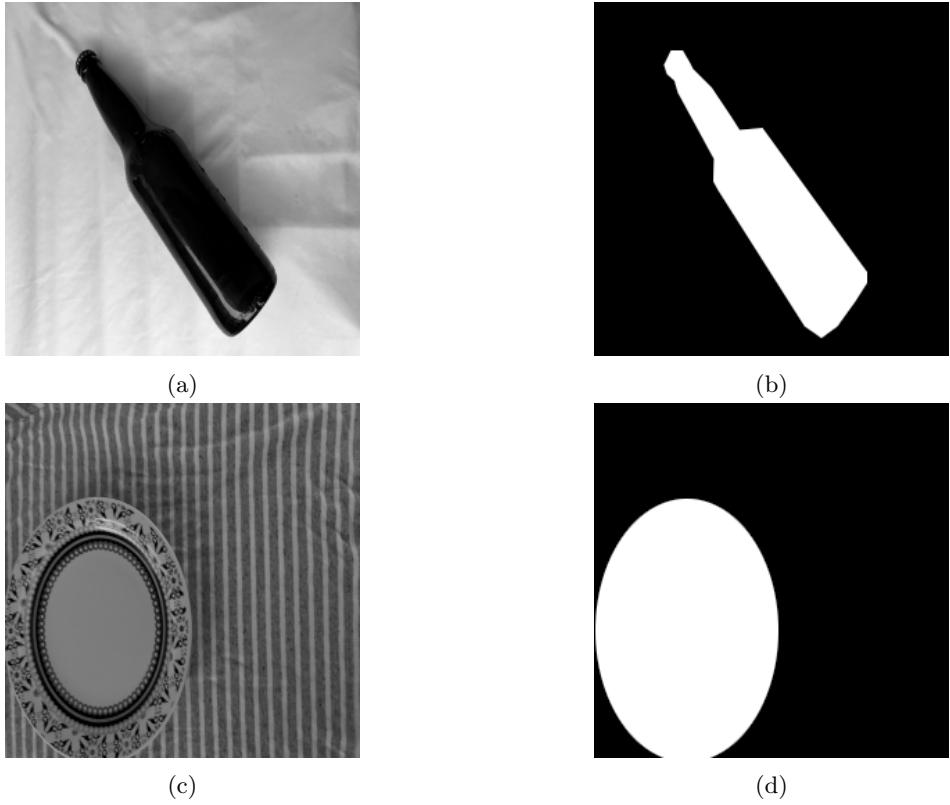


Figura 10: (a) Imagem cinza de uma garrafa (b) Ground Truth manual da garrafa utilizando a ferramenta free selection tool; (c) imagem em cinza de um prato (d) Ground Truth manual do prato utilizando a ferramenta elipse selector.

4.2 Métricas de avaliação

A utilização de métricas adequadas para avaliação é crucial para garantir que os resultados das tarefas de segmentação e detecção de objetos (caixas de Feret) são, de fato, úteis. Selecionamos algumas das métricas propostas na recente revisão de [4]: Índice de Jaccard e Coeficiente de Dice. Definiremos, a seguir, cada um destes índices.

Sejam A e B dois sub-conjuntos de um conjunto X , o **Índice de Jaccard** $J(A, B)$ é definido como

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} .$$

Pode-se demonstrar que $1 - J(A, B)$ é uma métrica no conjunto das partes de X caso este seja finito, e portanto pode ser vista como uma medida de similaridade entre os conjuntos A e B .

O **Coeficiente de Dice** $D(A, B)$ é definido por

$$D(A, B) = \frac{2|A \cap B|}{|A| + |B|} .$$

Assim como anteriormente, $D(A, B)$ é, de certa forma, uma medida de similaridade entre A e B , mesmo não sendo uma métrica propriamente dita. É simples de se demonstrar que ambos os índices variam de 0 até 1, 0 significando dissimilaridade total e 1 similaridade total.

As definições anteriores foram dadas para conjuntos A , B e X genéricos. No caso de segmentação imagens digitais, o conjunto X seria o conjunto de todos os pixels da imagem. Teríamos então duas imagens binárias, I_A e I_B , das quais definimos A e B como o conjunto dos pixels que assumem valor 1 (VERDADEIRO) nas imagens I_A e I_B , respectivamente.

Na tarefa de detecção de objetos, concentrarmos-nos na extração das caixas de Feret, também conhecidas como caixas delimitadores. As caixas de Feret do conjunto A em que os pixels assumem o valor 1 pode ser descritas por dois pontos: o ponto superior esquerdo e o ponto inferior direito.

Esses pontos são encontrados de forma que a caixa é o menor retângulo que contém A e cujos lados são alinhada com o eixo das abscissas e ordenadas. Assim, para essa tarefa, avaliamos as métricas nas caixas de Feret do conjunto A e B .

A Figura 11 mostra o valor do Índice de Jaccard para a tarefa de segmentação e extração de caixas, respectivamente. Note na Figura 11 que o Índice de Jaccard tem um valor alto para ambas as tarefas no caso da segmentação dessa chave. Os resultados da tarefa são o output do algoritmo BETQF, explicado na próxima seção.

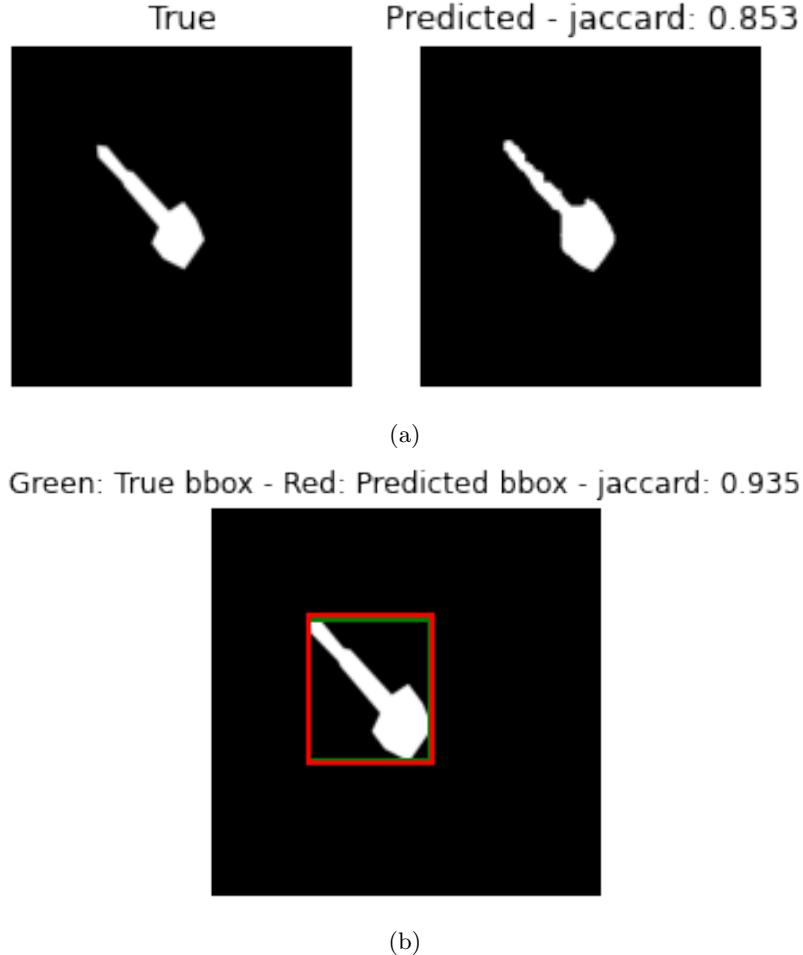


Figura 11: Avaliação do BETQF usando o Índice de Jaccard. Temos (a) ground truth a esquerda e segmentação da chave. (b) caixas delimitadoras verdadeira (verde) e predita (vermelha).

A Figura 12 mostra o valor do Coeficiente de Dice no caso de uma caneca. Note que o valor na tarefa de segmentação é um pouco mais baixo pelo fato de o algoritmo ter perdido parte do interior da caneca. A caixa delimitadora foi subestimada, mas mesmo assim ainda tem uma boa interseção com a caixa delimitadora verdadeira.

Por fim, mostramos um exemplo em que o valor da métrica é alto para uma tarefa, mas não para a outra.

A discrepancia dos valores do índice em 13 é clara: o algoritmo consegue segmentar bem a chave, porém fornece uma caixa delimitadora horrível. Isso se deve pelo fato de um pequeno ruído ter sido detectado como parte do objeto na parte direita da imagem. A situação reversa pode ocorrer, assim como o resultado ser ruim para ambas as tarefas.

Exemplos como esse motivaram tentativas de robustecer os algoritmos de segmentação.

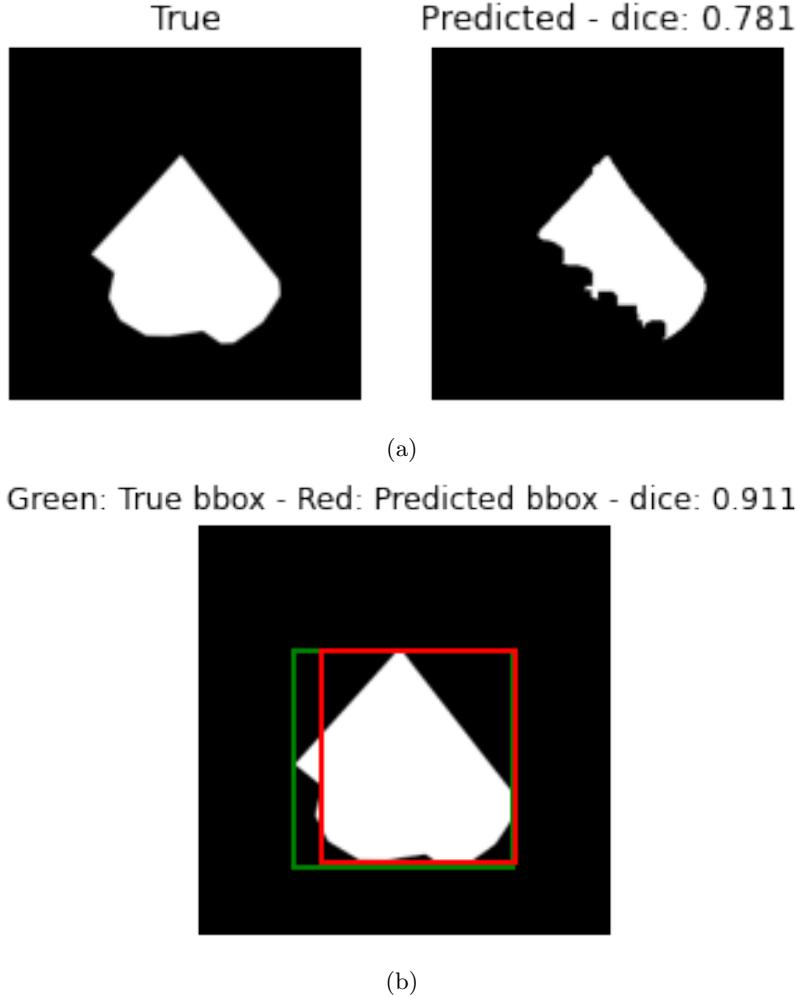


Figura 12: Avaliação do BETQF usando o Coeficiente de Dice. Temos (a) ground truth a esquerda e segmentação da caneca. (b) caixas delimitadoras verdadeira (verde) e predita (vermelha).

4.3 Modelos de segmentação

Foram testadas diferentes técnicas tanto na tarefa de segmentação quanto na detecção de objetos. Descreveremos os detalhes de cada técnica e algumas das imagens resultantes da sua aplicação.

4.3.1 BETQF

Um dos modelos construídos foi o BETQF, um acrônimo de Blur-Edge-Treshold-Quantile-Flood, que descreve a sequência de técnicas utilizadas para se fazer a segmentação e detecção.

O pseudo algoritmo para o BETQF é exibido abaixo.

- (1) Aplica-se uma técnica de borramento na imagem de input I , por exemplo uma convolução com um [filtro uniforme](#) 3×3 , obtendo uma imagem borrada B ;
- (2) Aplica-se um filtro de detecção de borda, como por exemplo o filtro de [Sobel](#), na imagem B , obtendo uma imagem E ;
- (3) Aplica-se um algoritmo de limiarização global, como o [algoritmo de Otsu](#), na imagem E , obtendo uma imagem T ;
- (4) Ainda na imagem E , aplica-se um filtro quantílico, por exemplo considerando o quantil 99.9%, obtendo um conjunto de pixels Q com valores maiores que os desse quantil;

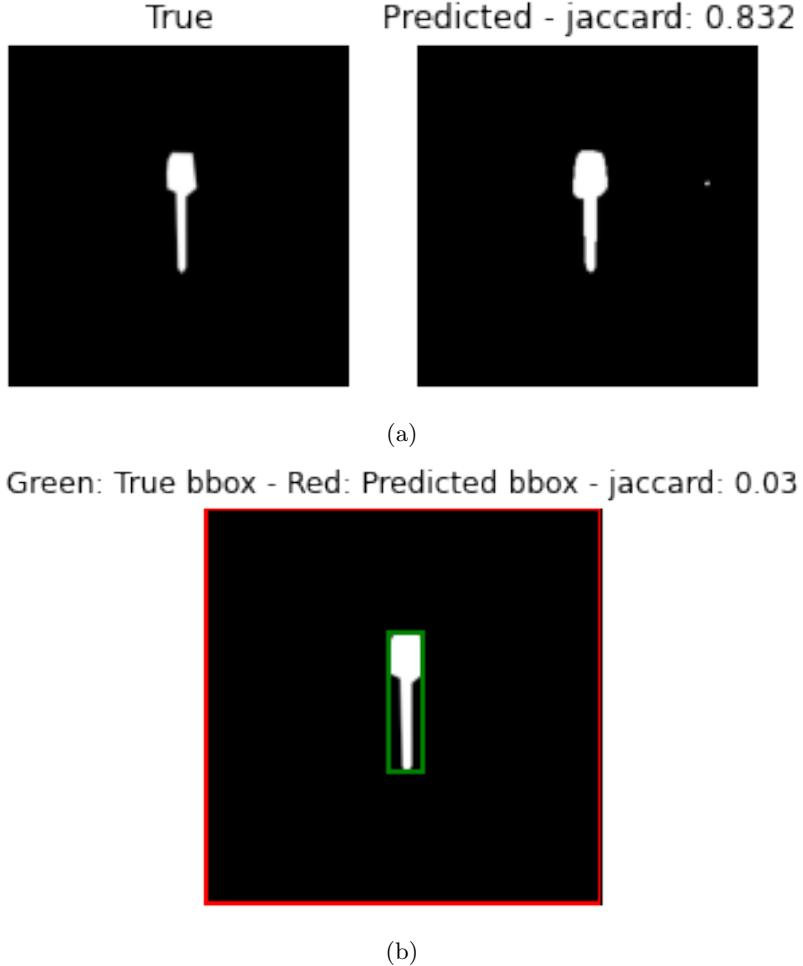


Figura 13: Exemplo em que o Índice de Jaccard é alto para uma tarefa mas muito ruim para a outra. Temos (a) ground truth a esquerda e segmentação da caneca. (b) caixas delimitadoras verdadeira (verde) e predita (vermelha).

- (5) Por fim, amostra-se alguns elementos de Q para serem sementes iniciais em algoritmo de [flooding](#). O flooding é aplicado na imagem T , e assim obtém-se uma imagem final F .

Explicamos, agora, a ideia por trás de cada uma das etapas. O borramento em (1) é feito para fazer com que ruídos no fundo da imagem (pequenas dobras nas roupas/panos usadas como fundo, detalhes das roupas/panos) tivessem uma menor intensidade quando se aplicasse a técnica de detecção de borda. A etapa (2) é feita para detectar as bordas, elemento crucial na segmentação. A etapa (3) é feita para tornar a imagem binária, tentando escolher um limiar que consegue detectar as bordas, mas possivelmente também incorpora um pouco do ruído do fundo. A etapa (4) tenta fazer um corte quantílico extremo para que apenas pontos da borda sobrem na imagem. Note que podem sobrar pouquíssimos pontos da borda, perdendo assim a maior parte da segmentação. A etapa (5) é feita para recuperar a borda partindo do poucos pontos que restaram em (4).

Assim, as "hipóteses" chaves para que o algoritmo funcione bem são as seguintes: (1) os quantis mais extremos da intensidade na imagem E (após detecção de borda) estão na fronteira do objeto; (2) Todos os pontos da fronteira ficam após a limiarização global T ; (3) a fronteira do objeto é conexa.

Da forma que foi implementado, o algoritmo possui diversos parâmetros: o tamanho da janela de borramento uniforme, o operador de detecção de bordas, o operador de limiarização, o valor do quantil e o número de pontos passados como fontes iniciais. Na seção seguinte, vamos mostrar como foi feita a otimização dos parâmetros.

A Figura 14 mostra como fica a imagem após a aplicação de cada uma das etapas.

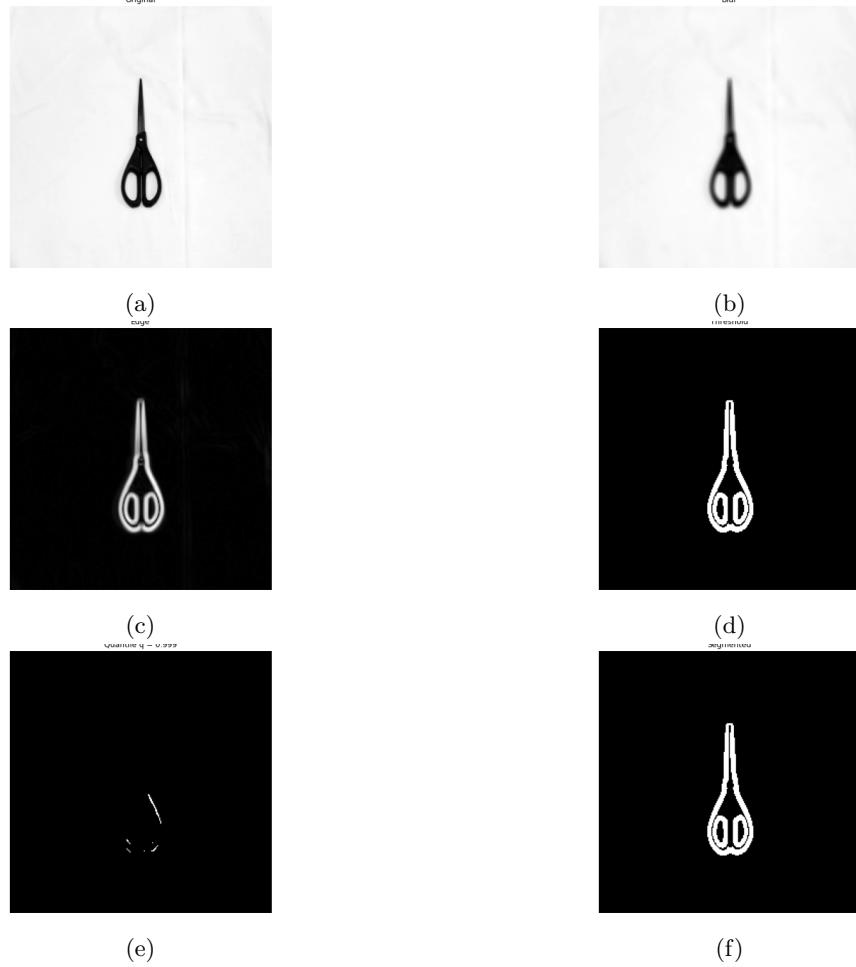


Figura 14: Aplicação do BETQF (a) Imagem cinza de uma tesoura (b) Borramento da imagem por um Kernel uniforme 3×3 ; (c) imagem resultante após aplicar o filtro de Sobel (d) aplicação da limiarização de Otsu; (e) aplicação do filtro quantílico com $q = 0.999$; (f) imagem final após usar os pontos do filtro quantílico como input para um flood da imagem (d).

Após aplicar o BETQF, temos uma segmentação da fronteira do objeto. Se quisermos as caixas delimitadoras, basta extrair o mínimo e o máximo de cada uma das coordenadas. Se quisermos a segmentação do objeto (inclui o interior) aplicamos um algoritmo de [floodfill](#) para finalizar. A Figura 15 mostra os resultados finais para a tesoura.

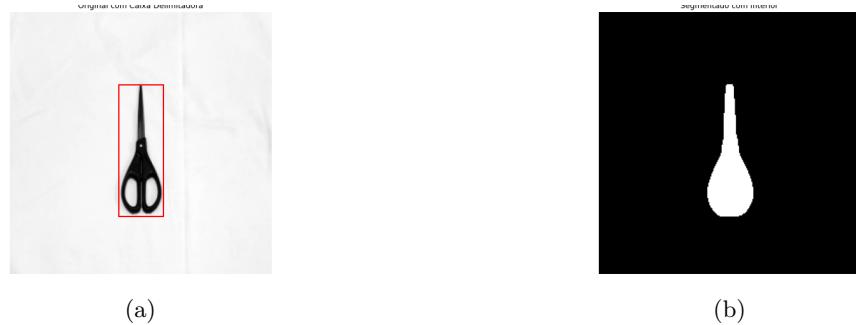


Figura 15: Aplicação do BETQF (a) Imagem cinza com caixa delimitadora; (b) imagem binária da segmentação do objeto após aplicar o floodfill.

A detecção da tesoura apresentada anteriormente é extremamente fácil, e de fato não foi essa

imagem que motivou um algoritmo com a aplicação de tantas técnicas. A Figura 16 mostra as etapas do algoritmo aplicado a um chinelo cujo fundo é relativamente ruidoso. Os outros algoritmos de segmentação testados não conseguiam detectar bem esse objeto, e foi isso que motivou a técnica BETQF. Note que, ao limiarizarmos pelos quantis, restam pouquíssimos pontos, porém todos da borda do objeto. Fazendo o fill, recuperamos todo o entorno do chinelo.

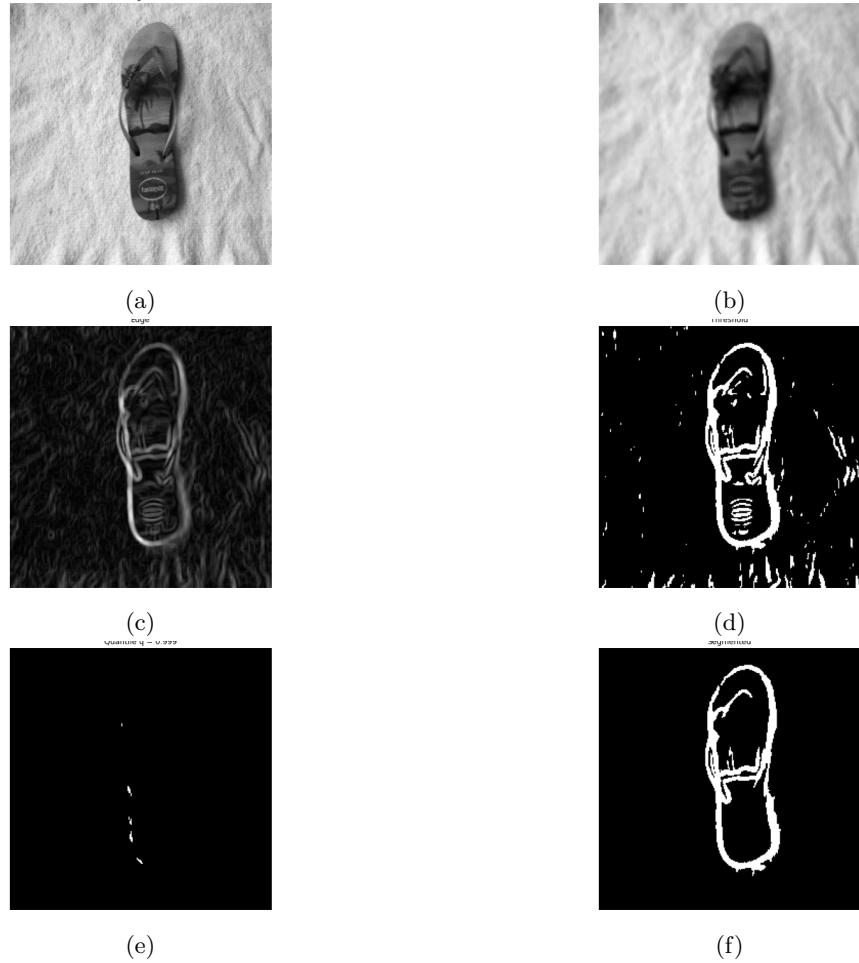


Figura 16: Aplicação do BETQF (a) Imagem cinza de uma tesoura (b) Borramento da imagem por um Kernel uniforme 3×3 ; (c) imagem resultante após aplicar o filtro de Sobel (d) aplicação da limiarização de Otsu; (e) aplicação do filtro quantílico com $q = 0.999$; (f) imagem final após usar os pontos do filtro quantílico como input para um flood da imagem (d).

A Figura 17 mostra os resultados finais para o chinelo.

Mesmo com um fundo não monocromático mas de variação leve, o algoritmo conseguiu segmentar alguns objetos corretamente, conforme mostra a Figura 18. Note, entretanto, que parte do rótulo da garrafa foi perdido.

Mostramos, agora, dois exemplos em que o BETQF não tem uma performance ideal. A Figura 19 que, como o livro contém muita variação na sua capa, o algoritmo efetivamente captura as palavras na capa do livro, e não o contorno correto do livro.

A Figura 20 mostra os resultados finais para o livro.

O algoritmo "falhou" anteriormente pois o objeto tinha muita variação no seu interior. Há casos em que há muita variação no fundo, e assim o algoritmo não consegue segmentar apenas o objeto. A Figura 21 mostra os resultados de cada etapa na tentativa de segmentar uma tesoura com um fundo extremamente complicado. De fato, é difícil detectar a tesoura na imagem original.

A Figura 22 mostra os resultados finais para essa tesoura. O output do algoritmo, nesse caso, é totalmente inútil.



Figura 17: Aplicação do BETQF (a) Imagem cinza com caixa delimitadora; (b) imagem binária da segmentação do objeto após aplicar o floodfill.



Figura 18: Aplicação do BETQF (a) Imagem cinza com caixa delimitadora; (b) imagem binária da segmentação do objeto após aplicar o floodfill.

4.3.2 Método utilizando Watershed

Outra alternativa de pipeline foi construída baseada no conceito de [Watershed](#) de forma a podermos comparar a solução anterior, baseada em bordas, com uma alternativa baseada em regiões. Os passos desse pipeline foram:

- (1) Aplica-se uma equalização de histograma para correção de luminosidade e contraste, como [CLAHE](#), obtendo uma imagem equalizada E ;
- (2) Aplica-se um filtro para borrar a imagem tentando preservar as bordas, como [Filtro bilateral](#), na imagem E , obtendo uma imagem B ;
- (3) Realce das bordas da imagem B utilizando a diferença dela com seu Laplaciano, obtendo assim a imagem L ;
- (4) Aplica-se um algoritmo de limiarização global, como o [algoritmo de Otsu](#), na imagem L , obtendo uma imagem T ;
- (5) Compara-se a média da intensidade de pixels no contorno da imagem T em relação à média completa, de forma a verificar se possivelmente o objeto é mais escuro que o fundo, e assim, inverter a binarização: branco vira preto, preto vira branco;
- (6) Na imagem T após o item 5, encontram-se as sementes pra aplicação do Watershed: a) Faz-se uma transformação de distância na imagem binarizada T , onde calcula-se para cada pixel a distância entre ele e o pixel de fundo mais próximo, b) Com essa imagem em níveis de cinza e utilizando apenas uma janela central, aplica-se uma limiarização utilizando 70% do valor máximo, assim, obtemos as sementes localização possivelmente no centro do objeto.
- (7) Finalizando, aplica-se o método Watershed numa versão borrada da imagem B (com filtro Gausiano) utilizando as sementes obtidas no item 6. Obtendo assim, as 'bacias' e intersecções do objeto com seu fundo.

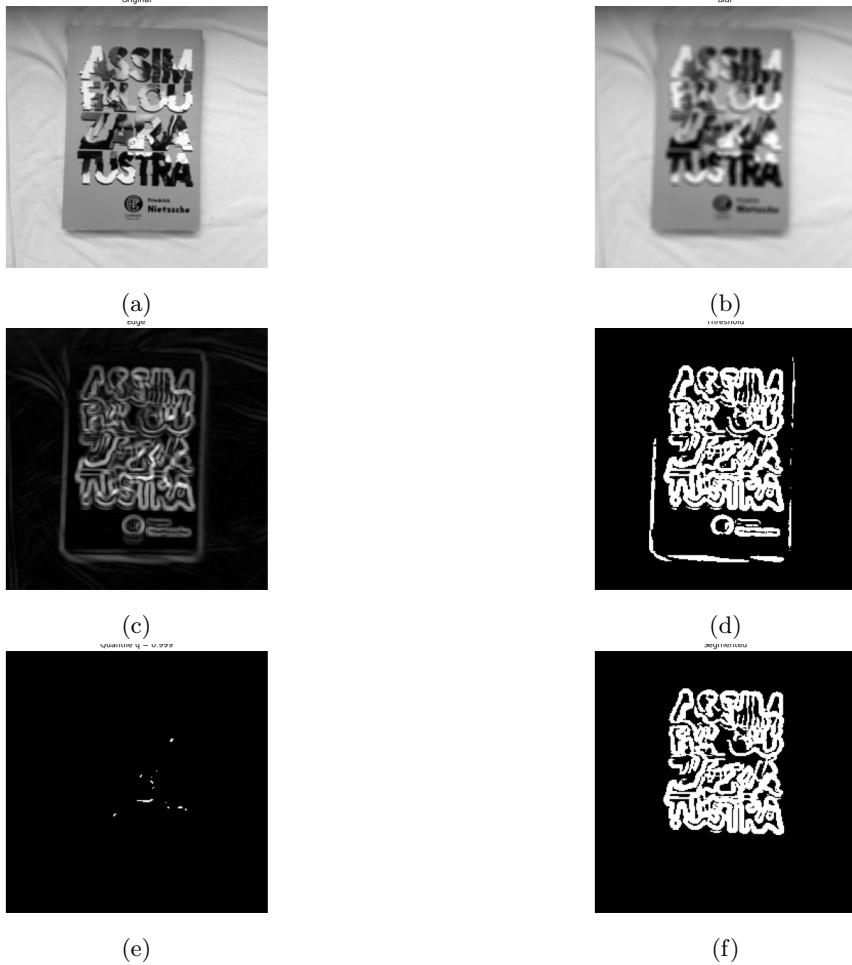


Figura 19: Aplicação do BETQF (a) Imagem cinza de uma tesoura (b) Borramento da imagem por um Kernel uniforme 3×3 ; (c) imagem resultante após aplicar o filtro de Sobel (d) aplicação da limiarização de Otsu; (e) aplicação do filtro quantilíco com $q = 0.999$; (f) imagem final após usar os pontos do filtro quantilíco como input para um flood da imagem (d).



Figura 20: Aplicação do BETQF (a) Imagem cinza com caixa delimitadora; (b) imagem binária da segmentação do objeto após aplicar o floodfill.

- (8) Como pós processamento, faz-se a união de todas as 'bacias' vizinhas, já que o próprio objeto pode ter padrões.

Primeiramente, (1) aplicou-se uma equalização de histograma local para correção de luminosidade e contraste da imagem. Depois, em (2) o borramento que tenta preservar as bordas é importante para a etapa de Watershed, já que não queremos que ruído internos no objeto sejam considerados. Na etapa

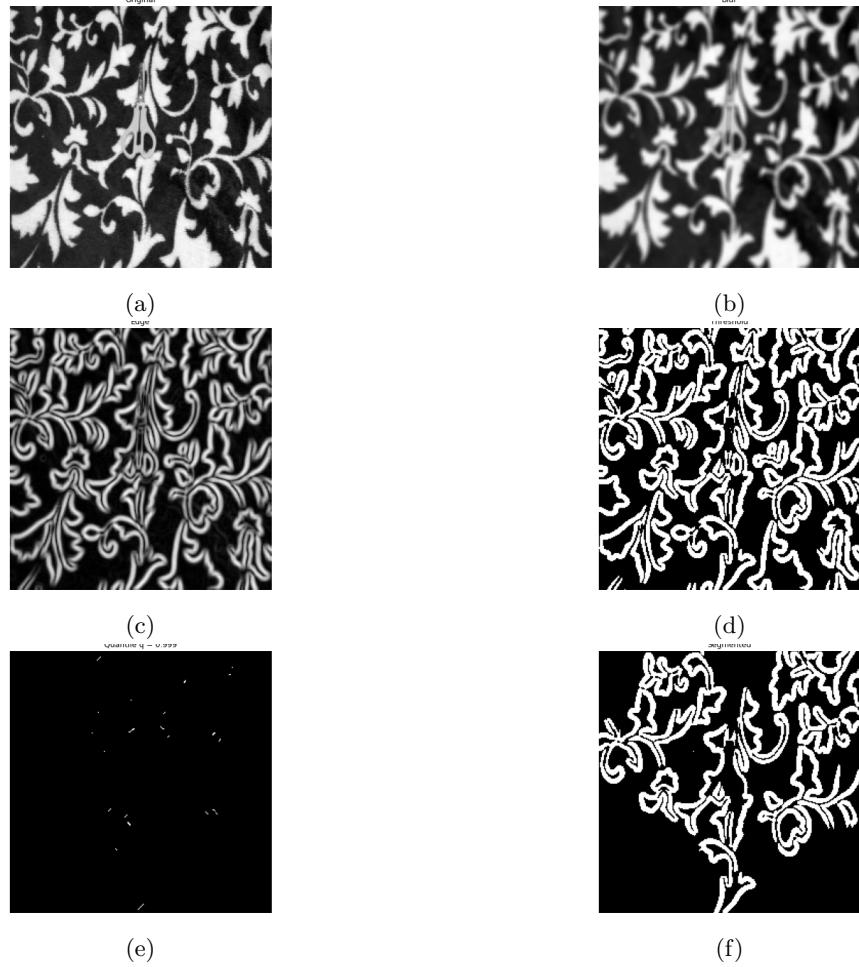


Figura 21: Aplicação do BETQF (a) Imagem cinza de uma tesoura (b) Borramento da imagem por um Kernel uniforme 3×3 ; (c) imagem resultante após aplicar o filtro de Sobel (d) aplicação da limiarização de Otsu; (e) aplicação do filtro quantílico com $q = 0.999$; (f) imagem final após usar os pontos do filtro quantílico como input para um flood da imagem (d).



Figura 22: Aplicação do BETQF (a) Imagem cinza com caixa delimitadora; (b) imagem binária da segmentação do objeto após aplicar o floodfill.

(3), ao fazer a diferença da imagem pelo seu Laplaciano, estamos realçando mais ainda as bordas, mais uma vez, preparando para a etapa de Watershed. Em (4), a limiarização global é o começo do processo para encontrarmos as sementes para o algoritmo de Watershed, seguidos do item (5), (6), visando o mesmo objetivo. E por fim, processamos o algoritmo (7) e realizamos o pós processamento de união das áreas encontradas via Watershed. A Figura 23 mostra algumas imagens do processo.

A principal hipótese dessa pipeline é de que seria possível, após do pré processamento, realizar o

Watershed e preencher apenas a 'bacia' do objeto.

O pipeline não obteve resultados ideais para objeto que internamente tem algum padrão de cor diferente, como o próprio chinelo analisado na Figura 23. E caso haja alguma variação intensa de luz no objeto devido a reflexão de alguma luz como vemos na Figura 24, onde a variação de luz no material externo do objeto atrapalhou o algoritmo. Assim como para fundos com padrões muito diferentes e que, possivelmente, tenha detalhes maiores do que o próprio objeto, como na figura 25, que nesse caso existe um desenho grande no fundo e assim, o pipeline definiu a semente no centro desse objeto.

4.4 Desempenho geral

4.4.1 Resultados BETQF

Como discutido na seção anterior, o método BETQF possui alguns hiperparâmetros. Exploramos esses parâmetros tanto de forma visual quanto por avaliação das métricas. O operador de filtro foi fixado como o Sobel, pois testamos com o operador de Robertt, Scharr e Prewitt e não observamos quase nenhuma diferença no valor das métricas. Fixamos o operador de limiarização como o de Otsu, pois foi o que conseguia melhor segmentar a imagem.

Os hiperparâmetros selecionados para otimização foram: (1) $K \in \{0, 5, 10, 20\}$ tamanho da janela do algoritmo de borramento; (2) $q \in \{0.95, 0.99, 0.999\}$ quantil de corte; e (3) $ns \in \{1, 5, 10\}$ o número de sementes aleatórias passadas para o algoritmo de flood. Selecionamos aleatoriamente 70% do Ground Truth para otimizar os hiperparâmetros, e avaliamos a performance utilizando apenas o índice de Jaccard.

Otimização-caixa delimitadora: A Tabela 5 mostra o resultado da otimização para a tarefa de detecção de caixas delimitadoras.

K	q=0.95	q=0.99	q=0.999
0	0.402/0.395/0.388	0.443/0.475/0.496	0.512/0.526/0.536
5	0.492/0.482/0.474	0.518/0.544/0.562	0.569/0.574/0.58
10	0.541/0.535/0.528	0.553/0.567/0.578	0.582/0.586/0.589
20	0.548/0.547/0.544	0.554/0.561/0.565	0.567/0.568/0.569

Tabela 5: Valores do Índice de Jaccard para a otimização dos hiperparâmetros do BETQF para a tarefa de extração de caixas delimitadoras. As linhas indicam o valor da janela de otimização, as colunas o valor do quantil de corte. Dentro de cada célula temos três valores separados por "/" que correspondem ao uso de 1, 5 e 10 sementes passadas para o algoritmo de flooding, respectivamente.

Podemos ver que, para a tarefa de extração de caixas delimitadoras, temos uma variação considerável da performance alterando os hiperparâmetros. De fato, os algoritmos com $K = 0$ e $K = 5$ tem uma performance inferior no geral, assim como os que usam $q = 0.95$. A melhor combinação de hiperparâmetros foi $K = 10$, $q = 0.999$ e $ns = 10$.

Otimização-segmentação: A mesma otimização foi feita para a tarefa de segmentação do objeto. A Tabela 6 mostra o resultado da otimização para essa tarefa.

K	q=0.95	q=0.99	q=0.999
0	0.377/0.347/0.328	0.35/0.375/0.39	0.392/0.398/0.404
5	0.41/0.405/0.401	0.406/0.418/0.427	0.427/0.43/0.433
10	0.401/0.404/0.404	0.409/0.419/0.427	0.427/0.429/0.432
20	0.437/0.438/0.44	0.441/0.443/0.445	0.446/0.446/0.447

Tabela 6: Valores do Índice de Jaccard para a otimização dos hiperparâmetros do BETQF para a tarefa de segmentação. As linhas indicam o valor da janela de otimização, as colunas o valor do quantil de corte. Dentro de cada célula temos três valores separados por "/" que correspondem ao uso de 1, 5 e 10 sementes passadas para o algoritmo de flooding, respectivamente.

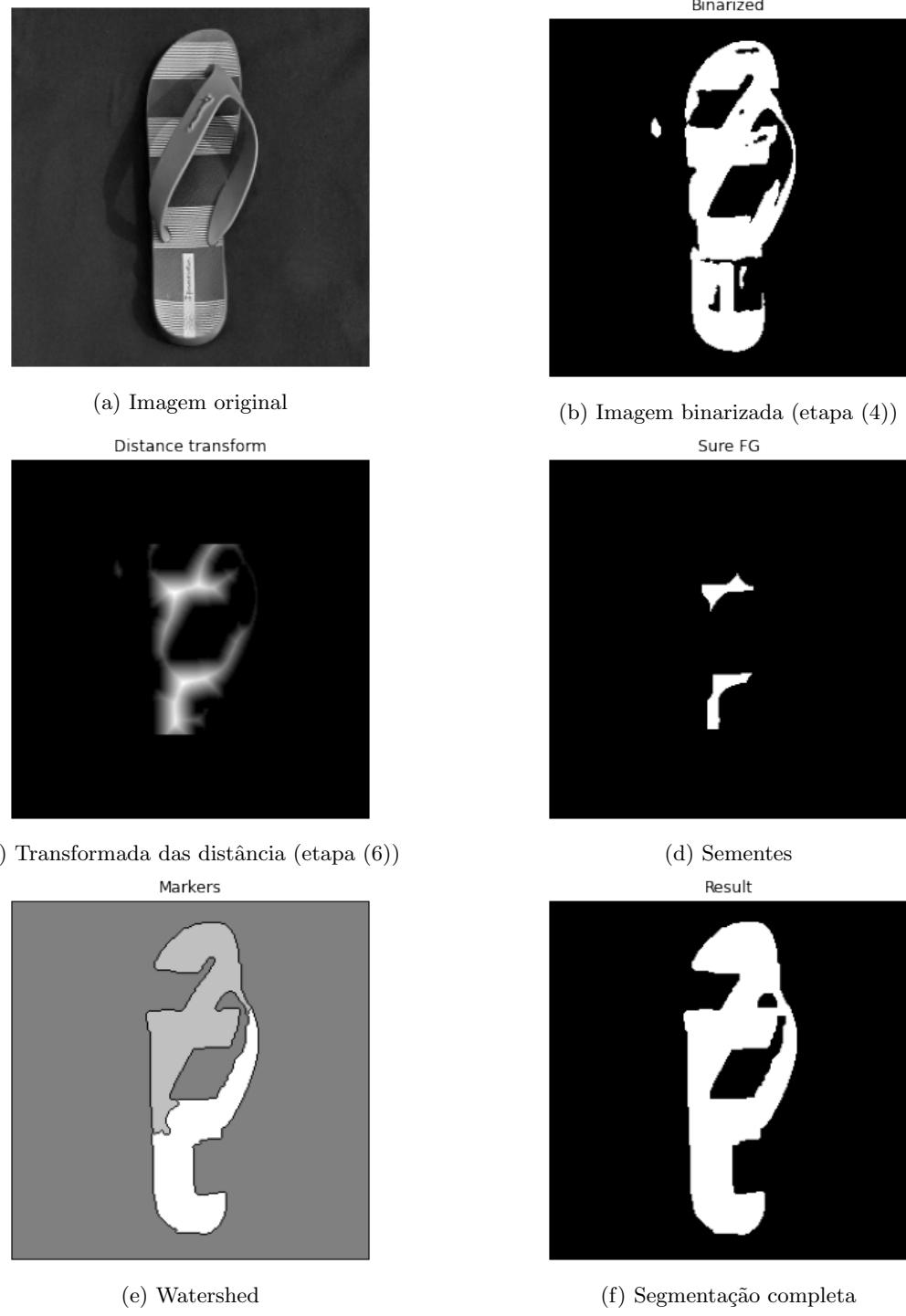
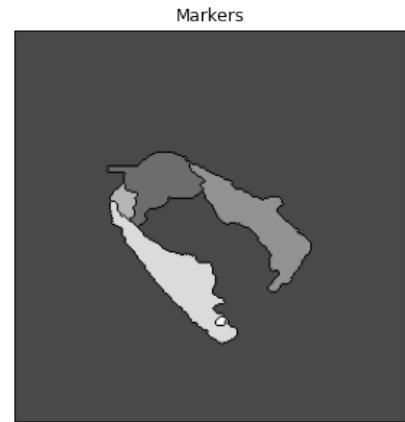


Figura 23: Aplicação do Watershed (a) Imagem cinza do chinelo (b) Após binarização; (c) imagem resultante após aplicar transformada das distâncias (d) Sementes obtidas; (e) Bacias obtidas após watershed com as sementes; (f) Imagem segmentada após pós processamento

Notamos que os valores do Índice de Jaccard para a tarefa de segmentação são, em geral, menores que os valores para a tarefa de extração de caixas delimitadoras. Há, novamente, uma variação considerável da performance variando os hiperparâmetros. A melhor combinação para a tarefa de segmentação foi $K = 20$, $q = 0.999$ e $ns = 10$.



(a) Imagem original

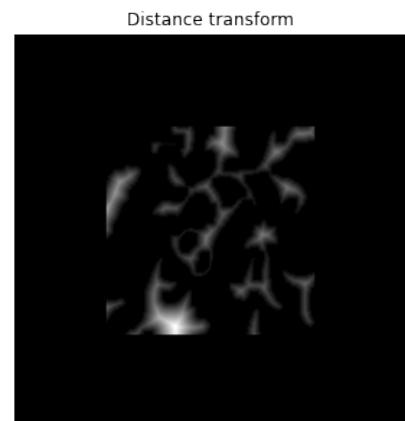


(b) Watershed no objeto com iluminação não uniforme

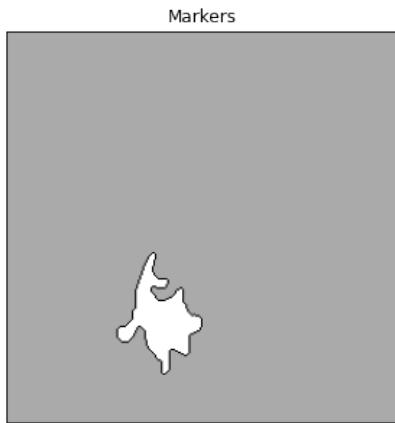
Figura 24: Aplicação do Watershed (a) Imagem cinza do caneca (b) Watershed deixou de fora a parte onde houve uma variação intensa de luminosidade



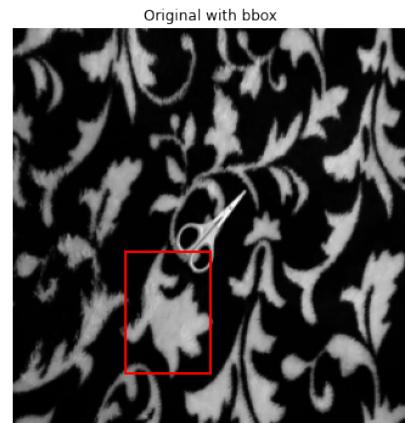
(a) Imagem original



(b) Transformada das distâncias na imagem binarizada



(c) Após preenchimento com watershed



(d) Objeto detectado após segmentação

Figura 25: Aplicação do Watershed (a) Imagem cinza da tesoura (b) Transformadas das distâncias (c) Utilização do watershed nas sementes obtidas apóis limiarização da imagem (b) e por fim, (d) Objeto detectado

Observação: Note que o melhor conjunto de hiperparâmetros foi encontrado na borda da malha de busca para ambos casos. Assim, seria interessante estender a malha e avaliar novamente a otimização.

Jaccard-caixa delimitadora: Exibimos, agora, os resultados dos métodos em cada uma das métricas. A Tabela 7 mostra a performance do algoritmo BETQF na tarefa de extração de caixas delimitadoras, avaliada pelo Índice de Jaccard estratificado por classe e iluminação-ambiente. A Tabela 8 mostra estratificado por classe e fundo. Foram utilizamos os valores ótimos do hiperparâmetros encontrados na etapa anterior.

Objeto	dia-exterior	dia-interior	noite-exterior	noite-interior	TOTAL
tesoura	0.581	0.381	0.621	0.423	0.505
garrafa	0.852	0.719	0.44	0.687	0.661
chave	0.545	0.695	0.328	0.577	0.57
prato	0.71	0.797	0.774	0.776	0.765
livro	0.831	0.762	0.51	0.723	0.707
sapato	0.757	0.716	0.548	0.757	0.705
chinelo	0.753	0.636	0.594	0.672	0.646
celular	0.453	0.408	0.44	0.878	0.491
portacopo	0.338	0.556	0.726	0.468	0.541
caneca	0.624	0.612	0.642	0.612	0.621
TOTAL	0.64	0.637	0.562	0.645	0.623

Tabela 7: Índice de Jaccard do BETQF na tarefa de extração de caixas delimitadoras estratificado por objeto e iluminação-ambiente.

Objeto	fundo1	fundo2	fundo3	TOTAL
tesoura	0.706	0.659	0.061	0.504
garrafa	0.84	0.51	0.678	0.661
chave	0.699	0.718	0.278	0.57
prato	0.729	0.916	0.704	0.78
livro	0.693	0.82	0.599	0.705
sapato	0.757	0.834	0.556	0.717
chinelo	0.81	0.504	0.629	0.648
celular	0.561	0.401	0.51	0.491
portacopo	0.801	0.801	0.326	0.542
caneca	0.632	0.87	0.278	0.624
TOTAL	0.714	0.71	0.477	0.626

Tabela 8: Índice de Jaccard do BETQF na tarefa de extração de caixas delimitadoras estratificado por objeto e fundo.

Jaccard-segmentação: A Tabela 9 mostra a performance do algoritmo BETQF na tarefa de segmentação, avaliada pelo Índice de Jaccard estratificado por classe e iluminação-ambiente. A Tabela 10 mostra estratificado por classe e fundo. Foram utilizamos os valores ótimos do hiperparâmetros encontrados na etapa anterior.

Novamente, percebemos que os valores da métrica são menores para a tarefa de segmentação, e que há uma variação considerável considerando classes, iluminação-ambiente e fundos diferentes.

Dice-caixa delimitadora: A Tabela 11 mostra a performance do algoritmo BETQF na tarefa de extração de caixas delimitadoras, avaliada pelo Coeficiente de Dice estratificado por classe e iluminação-ambiente. A Tabela 12 mostra estratificado por classe e fundo. Foram utilizamos os valores ótimos do hiperparâmetros encontrados na etapa anterior.

Dice-segmentação: A Tabela 13 mostra a performance do algoritmo BETQF na tarefa de segmentação, avaliada pelo Coeficiente de Dice estratificado por classe e iluminação-ambiente. A Tabela 14 mostra estratificado por classe e fundo. Foram utilizamos os valores ótimos do hiperparâmetros encontrados na etapa anterior.

Objeto	dia-exterior	dia-interior	noite-exterior	noite-interior	TOTAL
tesoura	0.286	0.186	0.231	0.116	0.209
garrafa	0.531	0.665	0.473	0.614	0.558
chave	0.381	0.384	0.324	0.344	0.365
prato	0.523	0.592	0.518	0.448	0.531
livro	0.394	0.462	0.365	0.608	0.446
sapato	0.451	0.504	0.316	0.489	0.464
chinelo	0.583	0.591	0.477	0.515	0.534
celular	0.414	0.449	0.235	0.761	0.438
portacopo	0.318	0.501	0.659	0.377	0.478
caneca	0.43	0.41	0.475	0.529	0.448
TOTAL	0.43	0.478	0.413	0.465	0.449

Tabela 9: Índice de Jaccard do BETQF na tarefa de segmentação estratificado por objeto e iluminação-ambiente.

Objeto	fundo1	fundo2	fundo3	TOTAL
tesoura	0.332	0.251	0.01	0.209
garrafa	0.569	0.533	0.574	0.558
chave	0.472	0.403	0.195	0.365
prato	0.481	0.652	0.466	0.531
livro	0.356	0.625	0.331	0.446
sapato	0.493	0.538	0.361	0.464
chinelo	0.615	0.55	0.438	0.534
celular	0.378	0.329	0.617	0.441
portacopo	0.688	0.703	0.286	0.473
caneca	0.42	0.676	0.163	0.448
TOTAL	0.469	0.531	0.354	0.449

Tabela 10: Índice de Jaccard do BETQF na tarefa de segmentação estratificado por objeto e fundo.

Objeto	dia-exterior	dia-interior	noite-exterior	noite-interior	TOTAL
tesoura	0.712	0.478	0.718	0.483	0.603
garrafa	0.918	0.824	0.583	0.793	0.767
chave	0.653	0.818	0.442	0.661	0.674
prato	0.88	0.872	0.868	0.832	0.864
livro	0.89	0.821	0.629	0.817	0.788
sapato	0.859	0.841	0.691	0.859	0.824
chinelo	0.858	0.746	0.713	0.776	0.757
celular	0.547	0.502	0.558	0.935	0.584
portacopo	0.505	0.683	0.827	0.61	0.671
caneca	0.745	0.709	0.732	0.71	0.725
TOTAL	0.748	0.739	0.678	0.737	0.728

Tabela 11: Coeficiente de Dice do BETQF na tarefa de extração de caixas delimitadoras estratificado por objeto e iluminação-ambiente.

Discussão BETQF: Os resultados finais das tabelas 7, 9, 11 e 13 nos permite tirar conclusões importantes. Notamos uma variação considerável na performance quando consideramos diferentes classes e iluminações. Por exemplo, pratos são os objetos com a melhor performance para a extração das caixas, com valores semelhantes por diferentes iluminações. Em contrapartida, celulares exibem o pior resultado, com uma variação muito grande dependendo da iluminação. De fato, quando é de noite e interior, o algoritmo tem uma ótima performance no caso das tesouras.

A iluminação-ambiente tem grande impacto na performance do algoritmo. Para todas as tarefas e

Objeto	fundo1	fundo2	fundo3	TOTAL
tesoura	0.826	0.774	0.118	0.604
garrafa	0.911	0.635	0.792	0.767
chave	0.82	0.826	0.352	0.673
prato	0.794	0.936	0.788	0.838
livro	0.777	0.874	0.715	0.79
sapato	0.815	0.905	0.704	0.811
chinelo	0.894	0.621	0.757	0.757
celular	0.602	0.512	0.609	0.575
portacopo	0.889	0.889	0.494	0.673
caneca	0.748	0.929	0.397	0.718
TOTAL	0.801	0.795	0.592	0.723

Tabela 12: Coeficiente de Dice do BETQF na tarefa de extração de caixas delimitadoras estratificado por objeto e fundo.

Objeto	dia-exterior	dia-interior	noite-exterior	noite-interior	TOTAL
tesoura	0.55	0.344	0.558	0.404	0.465
garrafa	0.594	0.752	0.578	0.82	0.666
chave	0.554	0.712	0.463	0.543	0.578
prato	0.692	0.672	0.545	0.611	0.651
livro	0.541	0.569	0.449	0.671	0.55
sapato	0.465	0.591	0.46	0.578	0.549
chinelo	0.606	0.694	0.441	0.668	0.58
celular	0.438	0.411	0.326	0.923	0.472
portacopo	0.493	0.691	0.844	0.661	0.691
caneca	0.471	0.656	0.458	0.554	0.546
TOTAL	0.533	0.607	0.514	0.629	0.572

Tabela 13: Dice do BETQF na tarefa de segmentação estratificado por objeto e iluminação-ambiente.

Objeto	fundo1	fundo2	fundo3	TOTAL
tesoura	0.675	0.603	0.039	0.466
garrafa	0.65	0.677	0.666	0.666
chave	0.736	0.619	0.341	0.578
prato	0.388	0.784	0.663	0.651
livro	0.478	0.739	0.411	0.55
sapato	0.514	0.682	0.431	0.549
chinelo	0.73	0.54	0.471	0.58
celular	0.428	0.387	0.601	0.472
portacopo	0.892	0.912	0.511	0.69
caneca	0.659	0.725	0.189	0.546
TOTAL	0.615	0.665	0.451	0.573

Tabela 14: Coeficiente de Dice do BETQF na tarefa de segmentação estratificado por objeto e fundo.

todas as métricas vemos que o algoritmo tem uma performance inferior na noite e exterior.

Outro fator que tem impacto drástico na performance do algoritmo é o fundo. Fundos monocromáticos apresentam boa performance, enquanto um fundo mais complexo dificulta a tarefa de segmentação. De fato, note que a segmentação e extração da tesoura no "fundo3", que corresponde à um fundo florido preto e branco, tem uma performance horrível.

4.4.2 Resultados pipeline com Watershed

Também, foram calculados o coeficiente de Dice e índice de Jaccard para fins de comparação, tanto na tarefa de segmentação, quanto na de extração de caixas delimitadoras. As tabelas 15, 16, 17, 18, 19, 20, 21, 22.

Discussão do pipeline com Watershed: observando a Tabela 17 e comparando com os resultados obtidos utilizando BETQF na Tabela 9, vemos que o pipeline utilizando Watershed teve uma performance bastante inferior nos objetos: *tesoura*, *portacopo* e *garrafa*. Pelas nossas observações, isso ocorreu devido a proposta do algoritmo em si, que à partir de algumas sementes que são expandidas, os contornos internos do objeto são definidos. Porém, os objetos podem ter materiais que afetam a iluminação deles, como exemplo, a *garrafa* ou mesmo a *caneca*, podem ter padrões, texturas, cores internas diferentes, ou seja, várias regiões com diferentes sementes, e por fim, para objetos pequenos (*tesoura*), fica difícil definir o local exato da semente.

Por outro lado, obteve-se uma melhoria na performance para objetos com superfícies mais uniformes como *livro*, *prato* e *celular*. Melhoria que possivelmente também é explicada pelo jeito como o algoritmo de Watershed é processado.

O algoritmo em si parece ser mais utilizável caso desejássemos segmentar todos os itens de alguma imagem, e não só encontrar apenas um único objeto, e também para objetos que tenha superfícies mais homogêneas em relação a luz, cor ou padrões.

Objeto	dia-exterior	dia-interior	noite-exterior	noite-interior	TOTAL
tesoura	0.061	0.148	0.116	0.356	0.158
garrafa	0.678	0.291	0.261	0.609	0.445
chave	0.456	0.502	0.118	0.5	0.45
prato	0.699	0.595	0.49	0.642	0.626
livro	0.663	0.872	0.563	0.713	0.708
sapato	0.713	0.656	0.375	0.765	0.643
chinelo	0.823	0.713	0.48	0.604	0.626
celular	0.688	0.711	0.497	0.725	0.668
portacopo	0.113	0.401	0.409	0.386	0.36
caneca	0.614	0.801	0.475	0.633	0.659
TOTAL	0.576	0.599	0.389	0.584	0.545

Tabela 15: Jaccard do Watershed na tarefa de extração de caixas delimitadoras.

Objeto	fundo1	fundo2	fundo3	TOTAL
tesoura	0.118	0.232	0.105	0.158
garrafa	0.731	0.314	0.36	0.445
chave	0.453	0.507	0.396	0.45
prato	0.496	0.822	0.541	0.626
livro	0.718	0.721	0.689	0.708
sapato	0.697	0.668	0.571	0.643
chinelo	0.783	0.685	0.409	0.626
celular	0.655	0.507	0.84	0.668
portacopo	0.484	0.782	0.107	0.36
caneca	0.665	0.838	0.408	0.659
TOTAL	0.587	0.616	0.445	0.545

Tabela 16: Índice de Jaccard do Watershed na tarefa de extração de caixas delimitadoras estratificado por objeto e fundo.

Objeto	dia-exterior	dia-interior	noite-exterior	noite-interior	TOTAL
tesoura	0.003	0.179	0.129	0.29	0.144
garrafa	0.616	0.224	0.172	0.662	0.394
chave	0.485	0.511	0.039	0.483	0.45
prato	0.592	0.614	0.446	0.646	0.599
livro	0.635	0.772	0.417	0.549	0.603
sapato	0.377	0.498	0.228	0.439	0.426
chinelo	0.616	0.675	0.467	0.521	0.561
celular	0.531	0.695	0.208	0.598	0.533
portacopo	0.113	0.326	0.386	0.382	0.326
caneca	0.51	0.764	0.394	0.608	0.596
TOTAL	0.481	0.545	0.307	0.512	0.471

Tabela 17: Jaccard do Watershed na tarefa de segmentação.

Objeto	fundo1	fundo2	fundo3	TOTAL
tesoura	0.125	0.185	0.112	0.144
garrafa	0.671	0.289	0.292	0.394
chave	0.474	0.413	0.45	0.45
prato	0.349	0.966	0.443	0.599
livro	0.497	0.738	0.543	0.603
sapato	0.456	0.413	0.418	0.426
chinelo	0.761	0.687	0.236	0.561
celular	0.545	0.323	0.732	0.533
portacopo	0.366	0.776	0.088	0.326
caneca	0.58	0.755	0.393	0.596
TOTAL	0.497	0.556	0.369	0.471

Tabela 18: Índice de Jaccard do Watershed na tarefa de segmentação estratificado por objeto e fundo.

Objeto	dia-exterior	dia-interior	noite-exterior	noite-interior	TOTAL
tesoura	0.113	0.222	0.176	0.469	0.23
garrafa	0.774	0.442	0.393	0.717	0.567
chave	0.569	0.557	0.21	0.56	0.531
prato	0.786	0.709	0.651	0.759	0.737
livro	0.757	0.93	0.625	0.816	0.785
sapato	0.824	0.782	0.537	0.857	0.765
chinelo	0.901	0.809	0.627	0.702	0.738
celular	0.741	0.789	0.636	0.774	0.742
portacopo	0.203	0.472	0.512	0.448	0.438
caneca	0.72	0.871	0.572	0.763	0.757
TOTAL	0.664	0.69	0.502	0.676	0.64

Tabela 19: Dice do Watershed na tarefa de extração de caixas delimitadoras.

5 Descrição e classificação de objetos

Para que seja possível definir a qual classe a imagem segmentada pertence, é necessária a utilização de um classificador, ou seja, um algoritmo capaz de discernir entre diferentes classes a partir de uma estratégia qualquer. Por sua vez, é conveniente representar a imagem segmentada como um conjunto de descritores, ou *features*. Dessa forma, a entrada do classificador torna-se um vetor de descritores unidimensional, não sendo necessário lidar com a imagem diretamente.

Objeto	fundo1	fundo2	fundo3	TOTAL
tesoura	0.176	0.316	0.178	0.23
garrafa	0.809	0.471	0.481	0.567
chave	0.535	0.573	0.49	0.531
prato	0.627	0.877	0.683	0.737
livro	0.802	0.784	0.774	0.785
sapato	0.812	0.792	0.699	0.765
chinelo	0.856	0.794	0.564	0.738
celular	0.728	0.605	0.892	0.742
portacopo	0.559	0.847	0.193	0.438
caneca	0.741	0.902	0.572	0.757
TOTAL	0.67	0.705	0.555	0.64

Tabela 20: Coeficiente de Dice do Watershed na tarefa de extração de caixas delimitadoras estratificado por objeto e fundo.

Objeto	dia-exterior	dia-interior	noite-exterior	noite-interior	TOTAL
tesoura	0.006	0.224	0.154	0.418	0.189
garrafa	0.741	0.325	0.252	0.782	0.498
chave	0.574	0.552	0.076	0.529	0.509
prato	0.698	0.712	0.613	0.755	0.709
livro	0.739	0.845	0.521	0.625	0.693
sapato	0.507	0.624	0.341	0.584	0.555
chinelo	0.727	0.762	0.604	0.605	0.671
celular	0.625	0.788	0.325	0.629	0.623
portacopo	0.197	0.399	0.483	0.435	0.4
caneca	0.621	0.85	0.511	0.735	0.702
TOTAL	0.577	0.631	0.405	0.602	0.563

Tabela 21: Dice do Watershed na tarefa de segmentação.

Objeto	fundo1	fundo2	fundo3	TOTAL
tesoura	0.144	0.24	0.174	0.189
garrafa	0.778	0.412	0.374	0.498
chave	0.518	0.486	0.519	0.509
prato	0.501	0.983	0.601	0.709
livro	0.597	0.79	0.662	0.693
sapato	0.584	0.541	0.549	0.555
chinelo	0.841	0.804	0.366	0.671
celular	0.65	0.43	0.789	0.623
portacopo	0.46	0.848	0.157	0.4
caneca	0.674	0.844	0.536	0.702
TOTAL	0.585	0.641	0.473	0.563

Tabela 22: Coeficiente de Dice do Watershed na tarefa de segmentação estratificado por objeto e fundo.

5.1 Estratégia de descrição

A partir da imagem segmentada, a estratégia adotada para descrição foi obter medidas que pudessem descrever o conteúdo da Feret Box associada. Especificamente, foi utilizado o método [regionpropstable](#), em que dado uma imagem previamente rotulada, são extraídas propriedades como: área, orientação, centroide, razão de pixels não nulos na feret box e assim por diante. As propriedades utilizadas são apresentadas pela Tabela ??, sendo que as descrições em maiores detalhes das propriedades podem ser acessadas [na documentação da ferramenta](#).

Propriedade	Descrição
1	Area
2	Extent
3	Perimeter
4	Solidity
5	Centroid
6	Orientation
7	Intensity max
8	Intensity min
9	Intensity mean
10	Eccentricity
11	Inertia tensor
12	Inertia tensor eigvals
13	Moments normalized
14	Moments weighted normalized
15	Inertia tensor eigvals
16	Eccentricity
17	Centroid weighted
18	Centroid weighted local
19	Equivalent diameter area
20	Euler number
21	Feret diameter max
22	Perimeter crofton
23	Axis minor length
24	Axis major length

Tabela 23: Descritores utilizados

5.2 Split de dados e Cross Valuation

Os dados que usamos são divididos em dados de treinamento e dados de teste. O conjunto de treinamento contém uma saída conhecida e o modelo aprende com esses dados para ser generalizado para outros dados posteriormente. Temos o conjunto de dados de teste para testar a previsão do nosso modelo nesse subconjunto. Para fazer isso em Python usamos a biblioteca Scikit-Learn e especificamente o método `train_test_split`.

```
: X_train, X_test, y_train, y_test = train_test_split(X, y,
train_size = .7, test_size = .3, stratify=y)
#Sanity check
print(X_train.shape, y_train.shape)
print(X_test.shape, y_test.shape)

(5008, 2) (5008,)
(2147, 2) (2147,)
```

Figura 26: Usando `train_test_split` Com um 70/30 ratio

O `train_size = 0.7` e `test_size= 0.3` na função indica a porcentagem dos dados que devem ser mantidos para teste. No nosso caso, é um split de 70/30 para treinamento e teste, respectivamente.

Então, até agora, temos um modelo que treina em um set aleatório específico de dados, valida em um set separado de dados e, finalmente, testa em um set de dados de teste. O problema com isso, especialmente quando o dataset é relativamente pequeno, é que você pode estar eliminando as observações que seriam cruciais para treinar um modelo ideal. Uma porcentagem relativamente pequena de dados, de 20 a 30%, pode ser os dados que tornariam nosso modelo de treinamento muito

mais eficaz.

Encontrar os parâmetros de ajuste ideais para um modelo pode ser um desafio. Podemos ter overfitting, o que faz com que nosso modelo seja treinado muito especificamente no subset de treinamento, o que causa menos transferibilidade e erros maiores quando aplicado ao nosso subset de teste. Por outro lado, podemos ter underfitting, o que significa que nosso modelo não treina especificamente o suficiente em nosso subset de treinamento. Isso também leva a erros maiores quando aplicado ao dataset.

Para combater esses problemas, usamos a Cross Valuation. Cross Valuation é feita dividindo nosso dataset em grupos aleatórios, usando um grupo como teste e treinando o modelo nos grupos restantes. Esse processo é repetido para usar cada grupo como o grupo de teste, então pegue a média dos modelos e use-a para o modelo resultante. Para fazer isso em Python usamos a biblioteca Scikit-Learn e especificamente o método GridSearchCV. A maior desvantagem de usar a Cross Valuation é que ela pode se tornar computacionalmente cara à medida que seu dataset aumenta e a quantidade de subsets aumenta.

5.3 Estratégia de classificação

O problema de classificação abordado é um problema do tipo multi-classe, dado que para cada instância ou observação, esta só pertence a uma classe, não podendo pertencer a mais de uma classe ao mesmo tempo. Além disso, o problema é do tipo supervisionado, pois existe um conjunto de dados em que a classe das observações presentes é conhecida a priori.

Para definir um classificador, foram utilizados alguns modelos disponíveis no módulo `scikit-learn`, sendo escolhido o que obteve melhor desempenho. As métricas utilizadas são descritas na Seção 5.3.1.

5.3.1 Métricas de avaliação do classificador

Para problemas de classificação, é comum utilizar métricas baseadas na matriz de confusão, como Exatidão (ou Acurácia), Taxa de Verdadeiro Positivo (TPR) e Taxa de Verdadeiro Negativo (TNR) para avaliar o desempenho de um classificador em uma determinada classe [5]. Além disso, existem métricas que summarizam o *trade-off* dos erros e acertos por classe, como a Precisão e o Score F1.

Por simplicidade, foi utilizada a função `classification_report` do módulo `sklearn.metrics` para visualizar o desempenho do classificador por classe, em que, para cada classe, obtemos as métricas

$$\begin{aligned} \text{Precision} &= \frac{TP}{TP + FP} \\ \text{Recall} &= \frac{TP}{TP + FN} \\ \text{F1-Score} &= \frac{2TP}{2TP + FP + FN} \end{aligned}$$

sendo que TP, FP, TN e FN são, respectivamente, a quantidade de verdadeiro positivos, falso positivos, verdadeiro negativos e falso negativos.

5.4 Classificadores utilizados

Para a classificação, adotamos alguns modelos simples, encontrados facilmente em bibliotecas como scikit-learn. Os modelos utilizados foram:

- **PCA+SVM** A primeira abordagem foi tentar reduzir o número de variáveis utilizando PCA. Após analisar quantas componentes principais eram relevantes, passamos essas variáveis para um classificador SVM.
- **LinearSVC**: É um classificador do tipo Support Vector Machine (SVM). SVMs são classificadores que transformam os dados para um domínio onde é possível classificá-los de forma mais simples, utilizando uma estratégia que diminui a influência de *outliers*. No caso do LinearSVC, a classificação é feita por hiperplanos. Essa implementação permite maior velocidade de treinamento que as outras implementações da biblioteca scikit-learn, embora possa levar a resultados inferiores.

- **KNN:** Algoritmo que realiza a classificação baseado na distância euclidiana multi dimensional do ponto a ser classificado em relação a grupos já rotulados conhecidos.
- **Regressão Logística:** Modelo linear que, utilizando uma função sigmoide, torna possível a tarefa de classificação. O modelo em si é uma combinação linear das variáveis de entrada.
- **MLP:** Rede Neural que consiste de camadas totalmente conectadas, funcionando como uma combinação linear de diferentes modelos de regressão logística.
- **LDA:** A análise de discriminante linear é uma técnica de classificação a qual assume que, condicional a classe (target) Y , as preditoras (features) X seguem uma distribuição Gaussiana multivariada com vetor de médias dependendo da classe e matriz de variância e covariância igual para todas as classes. Os parâmetros dos vetores de média das Gaussianas e da matriz de variância e covariância são ajustados por máxima verossimilhança. Após ajustado os parâmetros, é possível fazer a predição das classes comparando razão do logarítmico das probabilidades. Essa comparação gera fronteiras de decisão lineares.
- **QDA:** : A análise de discriminante quadrática é semelhante ao LDA, com a única diferença que cada classe tem sua própria matriz de variância e covariância.

5.5 Resultados dos classificadores

Nesta Subseção, são apresentados os principais resultados dos classificadores, utilizando as métricas propostas.

5.5.1 PCA+SVM

Aplicamos o `pca` do `sklearn` para obter as componentes principais, e então computamos o quanto da variância que cada uma das componentes explica, exibido na Figura 27

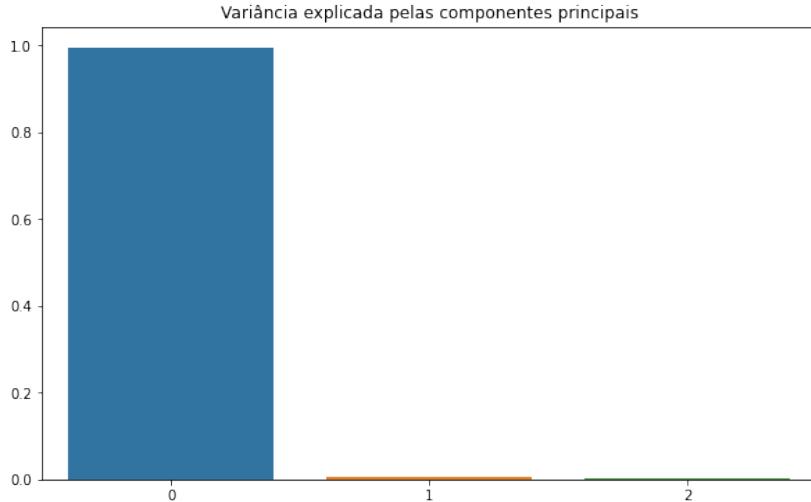


Figura 27: Variância explicada pelas componentes principais do PCA. Foi mostrado apenas as três primeiras componentes, o valor explicado pelas outras componentes foi ordens de magnitude menor.

A variância explicada pela primeira componente é muito maior que a explicada pelas outras. Exibimos apenas as três primeiras componentes pois as outras foram ordens de magnitude menor e não apareciam na figura. Assim, selecionamos as 3 primeiras componentes principais para passar como preditoras para o SVM.

Para ter uma noção da distribuição das classes, a Figura 28 mostra um gráfico de dispersão nos eixos das duas primeiras componentes principais, colorindo cada classe de uma cor

Note que as classes estão muito misturadas, apesar de algumas estarem mais concentradas em alguns pontos. Para tentar separar as classes, consideraremos diferentes kernels do SVM. De fato, utilizamos

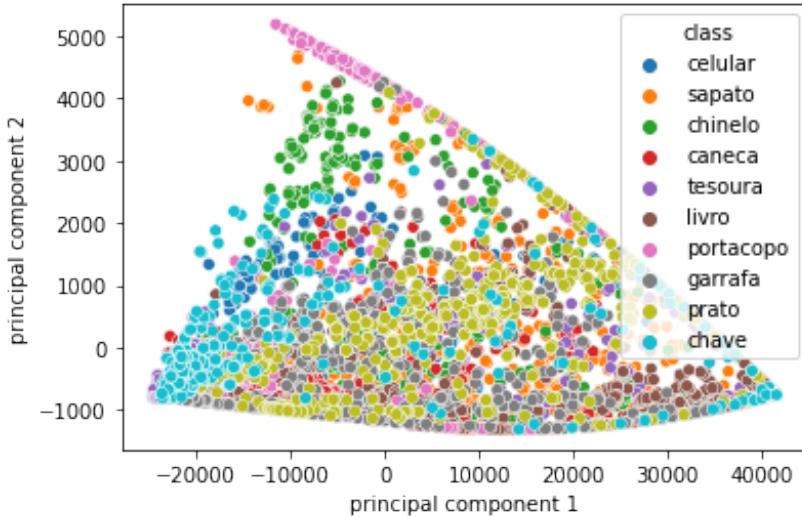


Figura 28: Gráfico de dispersão das classes considerando as duas primeiras componentes principais do PCA.

a `GridSearchCV` que implementa uma otimização de hiperparâmetros computando o valor da métrica por validações cruzadas para encontrar a melhor combinação de hiperparâmetros. A Tabela 24 os hiperparâmetros considerados para a malha.

Parâmetro	Valores da malha
C	1, 10, 100, 1000
kernel	linear, rbf, poly
degree	2, 3
gamma	1e-2, 1e-3, 1e-4

Tabela 24: Malha de hiperparâmetros do PCA+SVM

A melhor combinação de hiperparâmetros encontrada foi com $C = 1000$, $kernel = poly$, $degree = 2$ e $gamma = 1e - 2$.

A Tabela 25 mostra os resultados por classe obtido na amostra de teste.

Classe	Precision	Recall	F1-Score	Support
caneca	0.276	0.389	0.323	216
celular	0.424	0.369	0.395	203
chave	0.514	0.662	0.579	216
chinelo	0.584	0.208	0.307	216
garrafa	0.311	0.19	0.236	216
livro	0.347	0.468	0.398	216
portacopo	0.575	0.847	0.685	216
prato	0.307	0.639	0.414	216
sapato	0.196	0.046	0.075	216
tesoura	0.594	0.19	0.288	216
Estatística	Precision	Recall	F1-Score	Support
macro avg	0.413	0.401	0.37	2147
weighted avg	0.413	0.401	0.37	2147

Tabela 25: Resultados do classificador PCA+SVM estratificados por classe.

A abordagem por PCA+SVM apresenta uma baixa performance para a classificação, porém superior ao de um classificador aleatório. De fato, a acurácia foi de 40%, enquanto um classificador aleatório teria uma acurácia de 10%.

A Tabela 26 abaixo mostra a matriz de confusão para o modelo. O ideal é que tenhamos valores máximos na diagonal da matriz, indicando que os objetos de cada classe foram classificados corretamente. Podemos ver que o classificador tem uma baixa performance para a classe sapato. De fato, esses objetos são muito confundidos com canecas e livros pelo classificador. Note também que ele classifica os pratos corretamente (análise a linha dos pratos), porém ele tem uma tendência a classificar outros objetos como pratos também (análise a coluna dos pratos).

	Caneca	Celular	Chave	Chinelo	Garrafa	Livro	Portacopo	Prato	Sapato	Tesoura
Caneca	84	2	30	4	2	34	30	18	10	2
Celular	30	75	11	3	15	1	21	41	2	4
Chave	4	15	143	1	0	9	19	21	1	3
Chinelo	35	17	12	45	32	20	6	37	8	4
Garrafa	31	8	5	3	41	30	12	71	4	11
Livro	35	12	10	4	4	101	1	39	9	1
Portacopo	8	5	4	0	0	5	183	8	1	2
Prato	25	3	3	2	16	11	13	138	5	0
Sapato	49	9	3	14	10	56	14	50	10	1
Tesoura	3	31	57	1	12	24	19	27	1	41

Tabela 26: Matriz de confusão do PCA+SVM.

5.5.2 LinearSVC

Para treinar o classificador LinearSVC, foram fixados os hiperparâmetros de acordo com a Tabela 27:

Parâmetro	Valor Fixado
class_weight	'balanced'
random_state	12
max_iter	1500

Tabela 27: Parâmetros fixados para o treinamento do classificador LinearSVC.

Os melhores parâmetros encontrados podem ser visualizados na Tabela 28.

Parâmetro	Melhor Valor
penalty	'l1'
loss	'squared_hinge'
dual	False
tol	1e-4
C	1
fit_intercept	True
intercept_scaling	2

Tabela 28: Hiperparâmetros utilizados no modelo LinearSVC

Os melhores parâmetros levaram a uma precisão média de 41%. Na Tabela 29, é possível observar a precisão obtida para cada classe, sob diferentes métricas de avaliação.

O modelo possui precisão maior que o valor teórico de 10% ao escolher uma classe aleatória como predição. Entretanto, a performance do modelo está abaixo do esperado. Possíveis explicações para a baixa performance são:

- A dificuldade de segmentar as imagens utilizando somente métodos tradicionais. Essa dificuldade é exacerbada para as imagens com fundo não monocromático, que compõem um terço do dataset.
- As features extraídas com base na segmentação podem não ser suficientes para diferenciar os objetos com grande precisão.

Classe	Precision	Recall	F1-Score	Support
Celular	0.39	0.38	0.38	216
Sapato	0.32	0.16	0.21	203
Chinelo	0.43	0.74	0.54	216
Caneca	0.00	0.00	0.00	216
Tesoura	0.39	0.23	0.29	216
Livro	0.41	0.48	0.44	216
Portacopo	0.47	0.76	0.58	216
Garrafa	0.41	0.71	0.52	216
Prato	0.33	0.33	0.33	216
Chave	0.45	0.27	0.34	216
Estatística	Precision	Recall	F1-Score	Support
Macro Avg	0.36	0.41	0.36	2147
Weighted Avg	0.36	0.41	0.36	2147

Tabela 29: Resultados do classificador LinearSVC estratificados por classe.

	Celular	Sapato	Chinelo	Caneca	Tesoura	Livro	Portacopo	Garrafa	Prato	Chave
Celular	82	12	40	1	6	11	31	13	11	9
Sapato	17	32	35	3	15	7	21	30	13	30
Chinelo	7	2	160	0	0	10	5	17	0	15
Caneca	23	7	17	0	29	18	44	43	31	4
Tesoura	23	4	5	1	50	22	28	49	22	12
Livro	12	7	23	1	6	103	9	21	32	2
Portacopo	7	8	23	0	0	4	164	3	7	0
Garrafa	5	4	6	0	10	7	16	154	14	0
Prato	24	5	7	4	7	44	18	35	72	0
Chave	11	18	60	1	6	24	13	11	14	58

Tabela 30: Matriz de confusão do LinearSVC.

Classe	Fundo 1	Fundo 2	Fundo 3	Interior Noite	Interior Dia	Exterior Noite	Exterior Dia
Celular	0.39	0.22	0.48	0.34	0.44	0.40	0.35
Sapato	0.22	0.14	0.29	0.26	0.26	0.20	0.10
Chinelo	0.52	0.54	0.57	0.57	0.59	0.51	0.48
Caneca	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Tesoura	0.18	0.37	0.29	0.28	0.18	0.43	0.23
Livro	0.38	0.25	0.65	0.37	0.60	0.29	0.44
Portacopo	0.70	0.38	0.68	0.62	0.56	0.59	0.55
Garrafa	0.41	0.48	0.69	0.53	0.48	0.44	0.71
Prato	0.23	0.39	0.37	0.40	0.31	0.23	0.37
Chave	0.11	0.27	0.53	0.27	0.32	0.40	0.34
Accuracy	0.37	0.35	0.50	0.41	0.43	0.39	0.40
Macro Avg	0.31	0.30	0.46	0.36	0.37	0.35	0.36
Weighted Avg	0.31	0.31	0.45	0.36	0.38	0.35	0.37

Tabela 31: Precisão obtida pelo LinearSVC por classe discriminando pelo tipo de iluminação e pelo fundo utilizado.

- O fato de essa implementação de SVM utilizar hiperplanos para classificação. O uso de outros tipos de kernel tais como o kernel polinomial ou o kernel RBF talvez obteriam uma melhor performance.

5.5.3 KNN

Para o modelo KNN descobrimos que precisamos de mais clusters do que classes. Isso ocorre porque pode haver várias maneiras de representar cada classe. Cada classe tem uma variação de diferentes planos de fundo, iluminação e posições, e dá aos recursos de cada classe um alcance maior. Portanto, precisamos de mais de 1 cluster para representar as imagens de um determinado item. Ter o número de clusters igual a classes também levou a que vários rótulos de cluster fossem conectados a um cluster de imagens e nenhum dos rótulos de cluster fosse conectado a outros. Sem uma relação de 1 para 1 corremos o risco de perder a classificação de classes inteiras de imagens.

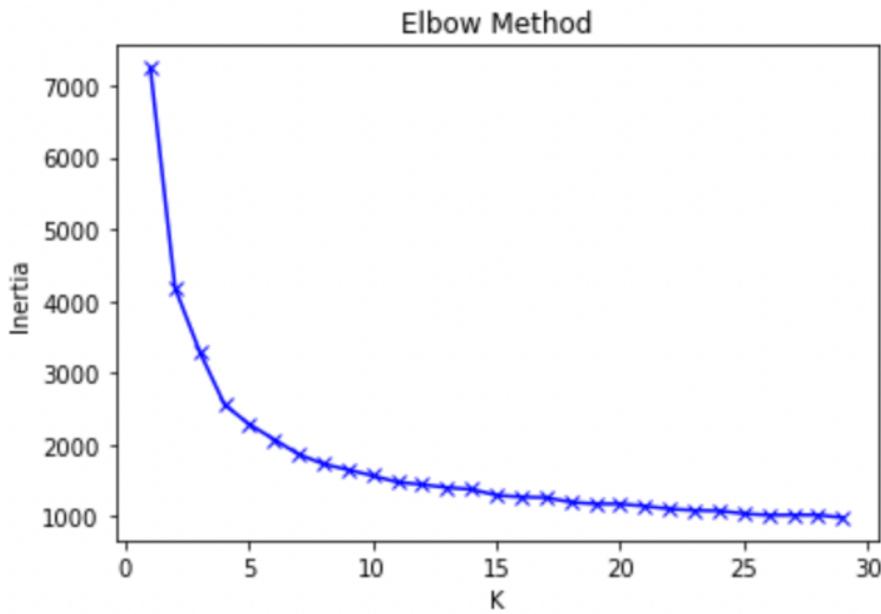


Figura 29: Usando o 'elbow method' para encontrar a quantidade ideal de clusters

Após $K = 20$, vemos que a redução da inércia diminui significativamente, e é, portanto, o que escolhemos como nosso número de clusters.

Para treinar o modelo KNN, primeiro tivemos que escolher os melhores parâmetros. Após normalizar todas as colunas para o intervalo $[0,1]$ usamos PCA para redução de dimensionalidade, mesmo como em SVM. Isso envolve zerar um ou mais dos menores componentes principais. Isso resulta em uma projeção de dimensão inferior dos dados que preserva a variância máxima dos dados. Optamos por manter os dois parâmetros de melhor desempenho e descartar o restante.

A Tabela 32 apresenta o desempenho por classe do grupo KNN:

Classe	Precision	Recall	F1-Score	Support
Celular	0.34	0.20	0.26	216
Sapato	0.51	0.36	0.43	203
Chinelo	0.53	0.63	0.58	216
Caneca	0.22	0.12	0.16	216
Tesoura	0.31	0.37	0.33	216
Livro	0.35	0.27	0.30	216
Portacopo	0.42	0.89	0.57	216
Garrafa	0.42	0.46	0.44	216
Prato	0.29	0.30	0.29	216
Chave	0.36	0.27	0.31	216
Estatística	Precision	Recall	F1-Score	Support
Macro avg.	0.38	0.39	0.37	2147
Weighted avg.	0.37	0.39	0.37	2147

Tabela 32: Desempenho por classe do classificador KNN

Para treinar o modelo KNN, primeiro tivemos que escolher os melhores parâmetros. Após normalizar todas as colunas para o intervalo $[0,1]$ usamos PCA para redução de dimensionalidade, mesmo como em SVM. Isso envolve zerar um ou mais dos menores componentes principais. Isso resulta em uma projeção de dimensão inferior dos dados que preserva a variância máxima dos dados. Optamos por manter os dois parâmetros de melhor desempenho e descartar o restante.

A precisão do classificador KNN por classe, discriminando por tipo de fundo e tipo de iluminação pode ser visualizada na Tabela 33

A precisão do classificador QDA por classe, discriminando por tipo de fundo e tipo de iluminação pode ser visualizada na Tabela 36

Classe	Fundo 1	Fundo 2	Fundo 3	Interior Noite	Interior Dia	Exterior Noite	Exterior Dia
Celular	0.22	0.17	0.11	0.36	0.0	0.09	0.175
Sapato	0.18	0.12	0.30	0.16	0.31	0.19	0.11
Chinelo	0.51	0.51	0.56	0.56	0.47	0.54	0.51
Caneca	0.23	0.35	0.28	0.25	0.29	0.47	0.14
Tesoura	0.31	0.18	0.11	0.21	0.20	0.29	0.09
Livro	0.42	0.29	0.31	0.35	0.32	0.42	0.20
Portacopo	0.63	0.46	0.79	0.65	0.54	0.68	0.59
Garrafa	0.46	0.41	0.21	0.33	0.41	0.22	0.44
Prato	0.24	0.35	0.21	0.29	0.36	0.29	0.12
Chave	0.32	0.46	0.31	0.45	0.44	0.31	0.21
Accuracy	0.39	0.35	0.33	0.39	0.36	0.38	0.30
Macro Avg	0.35	0.33	0.32	0.36	0.33	0.35	0.26
Weighted Avg	0.35	0.32	0.32	0.36	0.33	0.36	0.26

Tabela 33: Precisão obtida pelo KNN por classe discriminando pelo tipo de iluminação e pelo fundo utilizado.

5.5.4 QDA

O desempenho do classificador QDA por classe é visualizado na Tabela 34.

Classe	Precision	Recall	F1-Score	Support
Celular	0.44	0.38	0.41	216
Sapato	0.42	0.54	0.47	203
Chinelo	0.67	0.45	0.54	216
Caneca	0.79	0.12	0.22	216
Tesoura	0.64	0.51	0.57	216
Livro	0.56	0.36	0.44	216
Portacopo	0.47	0.90	0.62	216
Garrafa	0.30	0.94	0.46	216
Prato	0.49	0.21	0.30	216
Chave	0.69	0.11	0.19	216
Estatística	Precision	Recall	F1-Score	Support
Macro Avg	0.55	0.45	0.42	2147
Weighted Avg	0.55	0.45	0.42	2147

Tabela 34: Resultados do classificador QDA estratificados por classe.

A matriz de confusão do classificador QDA por classe é visualizada na Tabela 35.

Celular	Sapato	Chinelo	Caneca	Tesoura	Livro	Portacopo	Garrafa	Prato	Chave	
Celular	83	2	9	0	3	4	58	46	8	3
Sapato	2	109	1	0	1	2	29	56	3	0
Chinelo	3	19	97	0	0	2	20	69	1	5
Caneca	16	33	1	27	24	1	41	62	10	1
Tesoura	8	13	2	0	111	9	27	37	8	1
Livro	24	8	3	1	7	78	9	76	9	1
Portacopo	5	3	0	0	0	2	194	8	4	0
Garrafa	0	1	0	0	3	0	8	203	1	0
Prato	25	15	3	6	13	34	11	63	46	0
Chave	24	55	29	0	11	8	14	47	4	24

Tabela 35: Matriz de confusão do QDA.

A precisão do classificador QDA por classe, discriminando por tipo de fundo e tipo de iluminação pode ser visualizada na Tabela 36

Classe	Fundo 1	Fundo 2	Fundo 3	Interior Noite	Interior Dia	Exterior Noite	Exterior Dia
Celular	0.46	0.34	0.40	0.42	0.46	0.35	0.39
Sapato	0.55	0.41	0.47	0.48	0.45	0.47	0.50
Chinelo	0.13	0.67	0.68	0.58	0.43	0.58	0.55
Caneca	0.14	0.37	0.14	0.31	0.19	0.21	0.14
Tesoura	0.58	0.63	0.48	0.50	0.49	0.67	0.61
Livro	0.35	0.33	0.61	0.36	0.38	0.29	0.61
Portacopo	0.78	0.50	0.59	0.62	0.65	0.60	0.59
Garrafa	0.33	0.55	0.55	0.42	0.38	0.41	0.71
Prato	0.23	0.26	0.38	0.04	0.23	0.18	0.50
Chave	0.00	0.20	0.35	0.23	0.04	0.09	0.37
Accuracy	0.40	0.46	0.50	0.44	0.42	0.42	0.53
Macro Avg	0.36	0.43	0.47	0.40	0.37	0.38	0.50
Weighted Avg	0.35	0.43	0.46	0.40	0.37	0.38	0.51

Tabela 36: Precisão obtida pelo QDA por classe discriminando pelo tipo de iluminação e pelo fundo utilizado.

5.5.5 LDA

Os melhores parâmetros encontrados para o modelo LDA podem ser visualizados na Tabela 37.

Tabela 37: Melhores parâmetros encontrados após o treinamento do classificador LDA.

Parâmetro	Melhor Valor
solver	'svd'
tol	1e-06

A matriz de confusão do modelo LDA é definida pela Tabela 38.

	Celular	Sapato	Chinelo	Caneca	Tesoura	Livro	Portacopo	Garrafa	Prato	Chave
Celular	105	2	29	7	3	16	28	11	8	7
Sapato	20	54	14	17	21	8	3	21	15	30
Chinelo	5	1	161	2	0	6	9	17	0	15
Caneca	20	11	11	40	32	20	17	38	25	2
Tesoura	34	4	2	11	60	30	9	43	13	10
Livro	17	2	13	4	9	120	2	22	24	3
Portacopo	14	2	5	4	1	6	178	4	2	0
Garrafa	6	2	2	11	16	10	17	144	8	0
Prato	13	4	3	16	7	61	16	34	62	0
Chave	11	4	51	5	4	28	18	12	12	71

Tabela 38: Matriz de confusão do LDA.

Classe	Precision	Recall	F1-Score	Support
Celular	0.43	0.49	0.46	216
Sapato	0.63	0.27	0.37	203
Chinelo	0.55	0.75	0.64	216
Caneca	0.34	0.19	0.24	216
Tesoura	0.39	0.28	0.33	216
Livro	0.39	0.56	0.46	216
Portacopo	0.60	0.82	0.69	216
Garrafa	0.42	0.67	0.51	216
Prato	0.37	0.29	0.32	216
Chave	0.51	0.33	0.40	216
Estatística	Precision	Recall	F1-Score	Support
Macro Avg	0.46	0.46	0.44	2147
Weighted Avg	0.46	0.46	0.44	2147

Tabela 39: Desempenho por classe do classificador LDA

Classe	Fundo 1	Fundo 2	Fundo 3	Interior Noite	Interior Dia	Exterior Noite	Exterior Dia
Celular	0.42	0.33	0.58	0.45	0.54	0.45	0.37
Sapato	0.28	0.40	0.44	0.48	0.48	0.29	0.14
Chinelo	0.63	0.62	0.66	0.68	0.62	0.67	0.56
Caneca	0.21	0.28	0.22	0.22	0.33	0.16	0.24
Tesoura	0.22	0.38	0.37	0.32	0.09	0.43	0.38
Livro	0.41	0.33	0.63	0.36	0.56	0.39	0.49
Portacopo	0.79	0.49	0.81	0.71	0.72	0.70	0.63
Garrafa	0.37	0.49	0.69	0.53	0.44	0.42	0.69
Prato	0.26	0.31	0.40	0.25	0.33	0.24	0.44
Chave	0.00	0.49	0.58	0.27	0.41	0.51	0.39
Accuracy	0.40	0.42	0.57	0.46	0.48	0.45	0.46
Macro Avg	0.36	0.41	0.54	0.43	0.45	0.43	0.43
Weighted Avg	0.36	0.41	0.54	0.43	0.46	0.43	0.44

Tabela 40: Precisão obtida pelo LDA por classe discriminando pelo tipo de iluminação e pelo fundo utilizado.

5.5.6 LogisticRegression

Para treinar o classificador LogisticRegression, fixamos os hiperparâmetros de acordo com a Tabela 41:

Tabela 41: Parâmetros fixados para o treinamento do classificador LogisticRegression.

Parâmetro	Valor Fixado
class_weight	'balanced'
max_iter	500
multi_class	'multinomial'

Os melhores parâmetros encontrados podem ser visualizados na Tabela 42.

Tabela 42: Melhores parâmetros encontrados após o treinamento do classificador LogisticRegression.

Parâmetro	Melhor Valor
tol	0.01
C	1.75
fit_intercept	True

A matriz de confusão do modelo de regressão logística pode ser vista na Tabela 43

Tabela 43: Matriz de confusão do modelo de Regressão Logística.

	Celular	Sapato	Chinelo	Caneca	Tesoura	Livro	Portacopo	Garrafa	Prato	Chave
Celular	63	45	19	4	23	13	19	21	4	5
Sapato	31	47	39	2	12	10	38	15	0	9
Chinelo	3	12	130	2	0	18	27	9	2	13
Caneca	30	17	13	5	30	27	45	32	16	1
Tesoura	35	16	7	9	39	29	14	48	14	5
Livro	18	7	19	3	8	108	6	22	21	4
Portacopo	14	43	20	6	0	7	91	1	7	27
Garrafa	14	20	7	4	16	17	1	121	13	3
Prato	9	4	5	16	27	83	8	41	20	3
Chave	14	36	63	0	0	29	14	16	12	32

Classe	Precision	Recall	F1-Score	Support
Celular	0.27	0.29	0.28	216
Sapato	0.19	0.23	0.21	203
Chinelo	0.40	0.60	0.48	216
Caneca	0.10	0.02	0.04	216
Tesoura	0.25	0.18	0.21	216
Livro	0.32	0.50	0.39	216
Portacopo	0.35	0.42	0.38	216
Garrafa	0.37	0.56	0.45	216
Prato	0.18	0.09	0.12	216
Chave	0.31	0.15	0.20	216
Estatística	Precision	Recall	F1-Score	Support
Macro Avg	0.27	0.31	0.28	2147
Weighted Avg	0.28	0.31	0.28	2147

Tabela 44: Desempenho por classe do classificador de Regressão Logística

Classe	Fundo 1	Fundo 2	Fundo 3	Interior Noite	Interior Dia	Exterior Noite	Exterior Dia
Celular	0.12	0.25	0.44	0.20	0.34	0.30	0.28
Sapato	0.28	0.13	0.24	0.18	0.24	0.29	0.03
Chinelo	0.47	0.46	0.52	0.51	0.43	0.56	0.41
Caneca	0.02	0.00	0.08	0.00	0.06	0.06	0.03
Tesoura	0.14	0.26	0.23	0.26	0.05	0.32	0.13
Livro	0.39	0.12	0.59	0.32	0.43	0.37	0.40
Portacopo	0.73	0.10	0.27	0.48	0.35	0.29	0.38
Garrafa	0.26	0.45	0.62	0.46	0.37	0.42	0.58
Prato	0.13	0.15	0.08	0.04	0.15	0.03	0.18
Chave	0.00	0.17	0.36	0.09	0.12	0.19	0.35
Accuracy	0.30	0.23	0.38	0.30	0.29	0.32	0.31
Macro Avg	0.26	0.21	0.34	0.25	0.25	0.28	0.28
Weighted Avg	0.26	0.22	0.34	0.25	0.26	0.28	0.28

Tabela 45: Precisão obtida pelo modelo de Regressão Logística por classe discriminando pelo tipo de iluminação e pelo fundo utilizado.

5.5.7 Rede Neural - MLP

Para treinar o classificador MLP, fixamos os parâmetros de acordo com a Tabela 46.

Tabela 46: Parâmetros fixados para o treinamento do classificador MLP.

Parâmetro	Valor Fixado
random_state	12
max_iter	100

Os melhores parâmetros encontrados podem ser visualizados na Tabela 47.

Tabela 47: Melhores parâmetros encontrados após o treinamento do classificador MLP.

Parâmetro	Melhor Valor
activation	'tanh'
alpha	0.0001
beta_1	0.8
beta_2	0.9
early_stopping	True
epsilon	1e-08
hidden_layer_sizes	(100 50 25)
learning_rate	'constant'
momentum	0.8
nesterovs_momentum	True
solver	'adam'
tol	0.1

A matriz de confusão para o MLP é visualizada na Tabela 48.

Tabela 48: Matriz de confusão do MLP.

	Celular	Sapato	Chinelo	Caneca	Tesoura	Livro	Portacopo	Garrafa	Prato	Chave
Celular	35	0	17	19	3	58	56	23	0	5
Sapato	37	28	20	0	4	54	6	27	0	27
Chinelo	4	4	108	0	0	64	6	17	0	13
Caneca	13	13	8	19	0	66	36	60	0	1
Tesoura	15	0	8	10	32	53	20	73	0	5
Livro	18	0	7	9	3	141	1	37	0	0
Portacopo	12	1	9	4	1	2	176	11	0	0
Garrafa	9	0	3	7	9	30	11	147	0	0
Prato	5	1	6	20	6	91	7	80	0	0
Chave	5	6	36	0	9	39	17	35	0	69

Classe	Precision	Recall	F1-Score	Support
Celular	0.23	0.16	0.19	216
Sapato	0.53	0.14	0.22	203
Chinelo	0.49	0.50	0.49	216
Caneca	0.22	0.09	0.13	216
Tesoura	0.48	0.15	0.23	216
Livro	0.24	0.65	0.35	216
Portacopo	0.52	0.81	0.64	216
Garrafa	0.29	0.68	0.40	216
Prato	0.00	0.00	0.00	216
Chave	0.57	0.32	0.41	216
Estatística	Precision	Recall	F1-Score	Support
Macro Avg	0.36	0.35	0.31	2147
Weighted Avg	0.35	0.35	0.31	2147

Tabela 49: Desempenho por classe do classificador MLP

Classe	Fundo 1	Fundo 2	Fundo 3	Interior Noite	Interior Dia	Exterior Noite	Exterior Dia
Celular	0.07	0.24	0.25	0.23	0.28	0.10	0.13
Sapato	0.03	0.42	0.20	0.27	0.27	0.25	0.07
Chinelo	0.00	0.71	0.60	0.58	0.49	0.60	0.33
Caneca	0.08	0.20	0.11	0.18	0.23	0.06	0.00
Tesoura	0.07	0.29	0.29	0.18	0.08	0.39	0.21
Livro	0.27	0.36	0.47	0.27	0.37	0.40	0.34
Portacopo	0.75	0.48	0.69	0.68	0.65	0.62	0.59
Garrafa	0.23	0.39	0.58	0.39	0.38	0.36	0.46
Prato	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Chave	0.00	0.57	0.56	0.36	0.42	0.48	0.37
Accuracy	0.21	0.40	0.45	0.35	0.36	0.37	0.32
Macro Avg	0.15	0.37	0.37	0.31	0.32	0.33	0.25
Weighted Avg	0.15	0.37	0.37	0.31	0.32	0.33	0.26

Tabela 50: Precisão obtida pelo modelo de Rede Neural por classe discriminando pelo tipo de iluminação e pelo fundo utilizado.

6 Conclusão do trabalho

Finalizamos com uma discussão geral sobre aquilo que pode ser aprendido durante o trabalho.

O trabalho começou pela aquisição das fotos, o qual já colocou um desafio inicial. Construímos um conjunto de dados diverso o suficiente para entender como as características da cena de aquisição da imagem afeta o algoritmo. Mesmo estabelecendo um protocolo de obtenção das imagens, tivemos várias imagens que continuaram complicadas de se trabalhar, como a garrafa em 3. Essas etapas, entretanto, são iniciais no trabalho, e portanto é crucial dar a atenção necessária pois, como vimos em 4, não é simples corrigir tais problemas posteriormente.

Os protótipos médios ficaram mais identificáveis para objetos muito semelhantes entre si, como por exemplo chinelo, e ruins para objetos difíceis de se distinguir contra fundos complexos, como por exemplo a tesoura. Vimos diferentes formas para os histogramas, e que ocorreu uma saturação geral no preto. Estudamos também como os histogramas se alteraram para cada um dos conjuntos de dados. Tabelas para os dois primeiros momentos dos dados foram construídas para completar as informações mostradas pelos histogramas.

Apresentamos dois algoritmos para segmentação e extração das caixas delimitadoras, o BETQF e outro baseado na aplicação do algoritmo de Watershed. Exibimos os pseudocódigos para os algoritmos, bem como a intuição por trás da combinação das técnicas. A imagem resultante após a aplicação de cada etapa foi apresentada figuras como 16, 17, 14 e 15. Mostramos exemplos em que a segmentação não performa bem em 21 e 22.

Utilizamos duas métricas para avaliação dos resultados da segmentação e extração das caixas delimitadoras: o Índice de Jaccard e o Coeficiente de Dice.

Apresentamos a otimização dos hiperparâmetros do BETQF nas tabelas 5 e 6, bem como o resultado final nas tabelas 9 e 7. Fatores como luz, ambientação e fundo afetam drasticamente a performance dos algoritmos, conforme discutido.

Na seção de classificação construímos algoritmos clássicos com características baseadas nas regiões das caixas delimitadoras estimadas. A performance desses algoritmos ficou muito abaixo daquelas que se encontra em artigos com técnicas mais modernas, porém são esperadas. A qualidade da segmentação reflete diretamente na qualidade das características extraídas e da precisão dos classificadores. Além disso, foram extraídas características simples já fornecidas pelas bibliotecas, e que talvez não sejam as ideais para o treinamento do modelo. Esses fatores colaboraram para que os classificadores treinados não atinjam um alto desempenho. A quantidade de modelos diferentes treinados, todos com baixo desempenho, deixa evidente o quanto a qualidade dos dados influenciou na qualidade final dos modelos.

Com uma segmentação mais precisa e extração de características mais relevantes, talvez seria obtido um modelo com uma precisão muito maior para a classificação dos objetos.

Apêndice A Cronograma - Gannt Chart

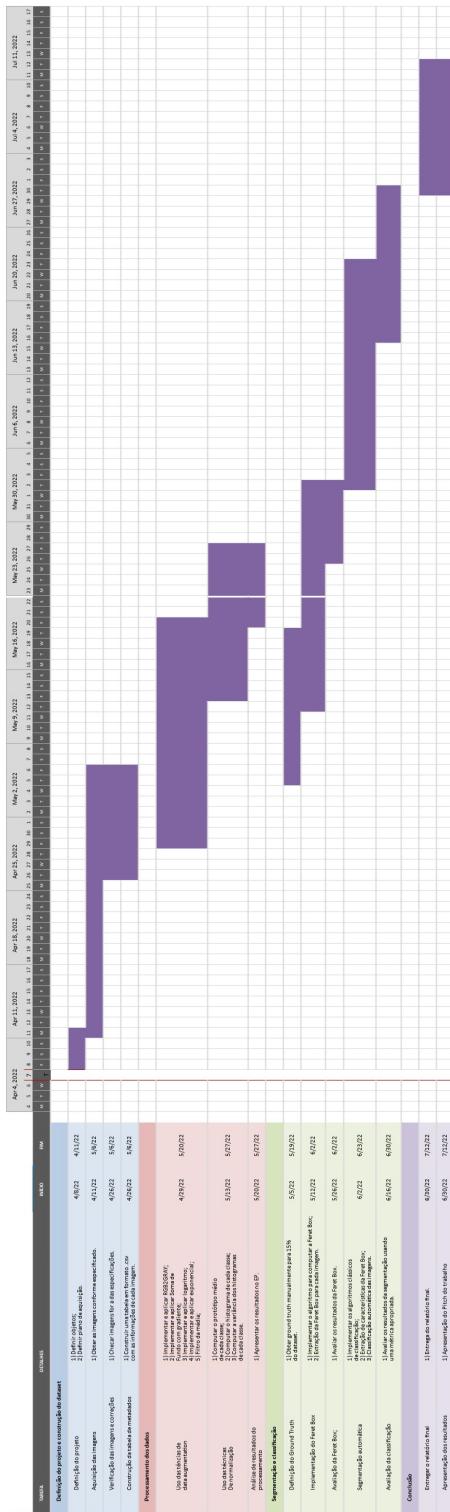


Figura 30: Gannt Chart

Apêndice B Visualização estilo MNIST sobre as operações no Dataset

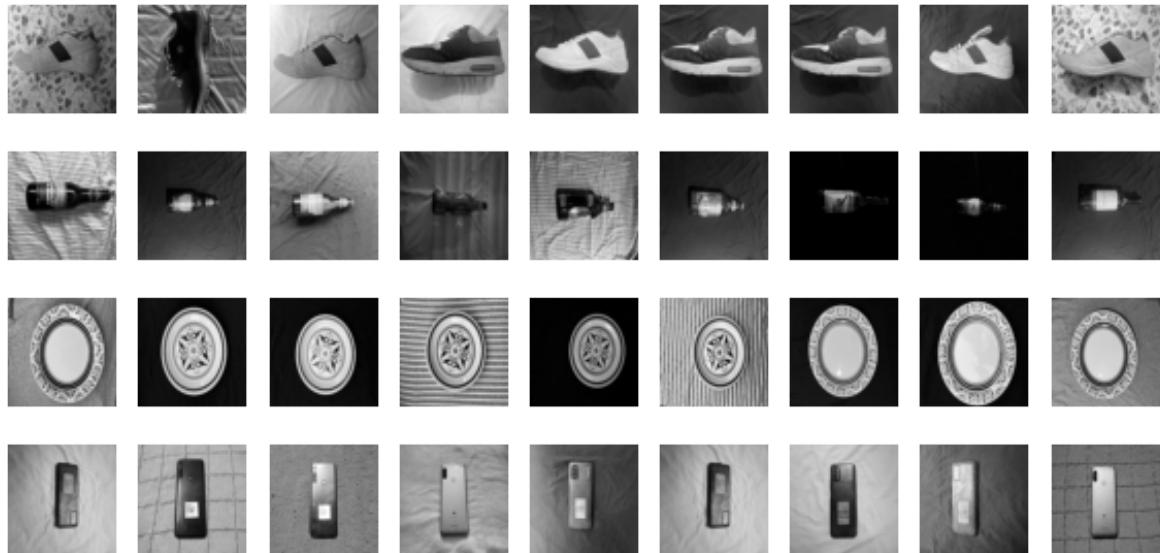


Figura 31: Dataset convertido para cinza

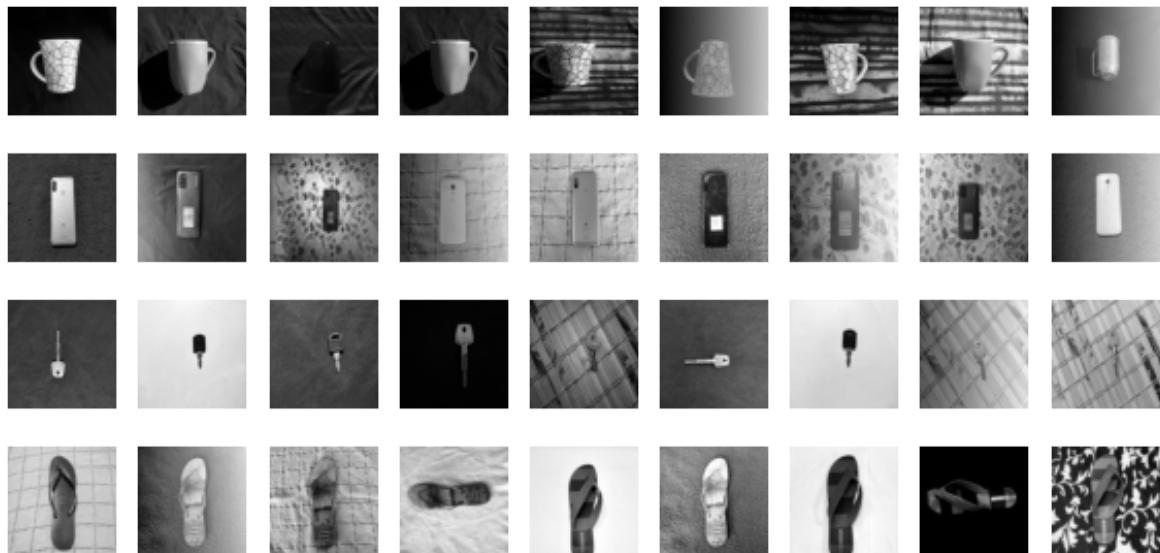


Figura 32: Dataset após Aumento de Dados

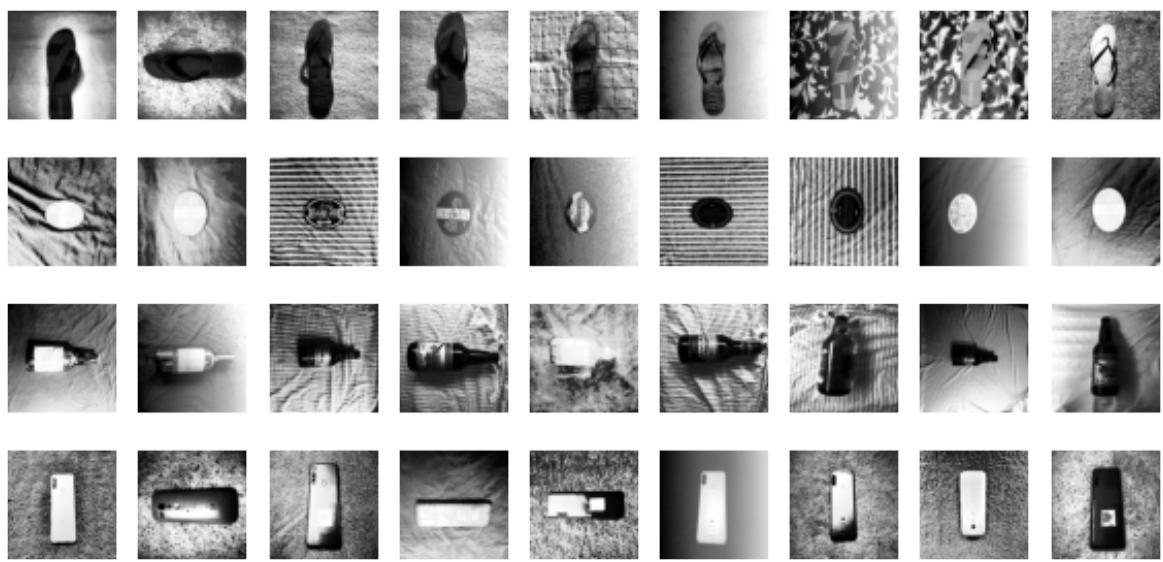


Figura 33: Dataset após Normalização de Histograma

Referências

- [1] L. da Fontoura Costa and R.M. Cesar. *Shape Analysis and Classification: Theory and Practice*. Image Processing Series. CRC Press, 2010.
- [2] R.C. Gonzalez and R.E. Woods. *Processamento de Imagens Digitais*. Editora Blucher, 2009.
- [3] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.
- [4] Shervin Minaee, Yuri Boykov, Fatih Porikli, Antonio Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. Image segmentation using deep learning: A survey. *CoRR*, abs/2001.05566, 2020.
- [5] Alaa Tharwat. Classification assessment methods. *Applied Computing and Informatics*, 17(1):168–192, July 2021.
- [6] The GIMP Development Team. Gimp.
- [7] Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu, and the scikit-image contributors. scikit-image: image processing in Python. *PeerJ*, 2:e453, 6 2014.