Relatório do EP de MAC0447-MAC5749 - Análise de Formas

Sabrina Araújo

18 de dezembro de 2023

Resumo

O vídeo de apresentação do EP está disponível em https://youtu.be/_oJ7eIKL2YA.

O repositório do EP está disponível em https://github.com/sabrizzs/MAC0447_EP

O presente relatório descreve o desenvolvimento de um projeto da disciplina MAC0447-MAC5749 - Análise de Formas, que tem como objetivo explorar os principais conceitos de análise e reconhecimento de formas em problemas de visão computacional. O projeto consiste na implementação de um aplicativo classificador, dividido em três partes: aquisição de dataset, preparação e processamento do dataset no Jupyter Notebook, e desenvolvimento do aplicativo classificador.

Conteúdo

1 Introdução

O objetivo da disciplina é apresentar os principais conceitos envolvidos na análise e reconhecimento de formas em problemas de visão computacional. Neste contexto, este projeto tem como objetivo central a implementação de um aplicativo classificador que utiliza técnicas de visão computacional.

A primeira parte do projeto envolve um aplicativo para aquisição de imagens e a formação de um dataset para posteriormente ser usado no aplicativo classificador. O dataset é composto por diferentes imagens, armazenado na plataforma Google Drive, e acompanhado por um arquivo de metadados.

Em um Jupyter Notebook, foram aplicadas diversas tarefas importantes de visão computacional, funções e técnicas para processamento e preparação do dataset. Isso incluiu a leitura das imagens, a extração de características relevantes e o treinamento de um modelo classificador para reconhecimento de objetos.

A parte final desse projeto consiste na criação de um aplicativo classificador, que representa a aplicação do que foi estudado e explorado durante a disciplina. O aplicativo é desenvolvido em Python, utilizando a biblioteca Kivy para a construção da interface gráfica.

O aplicativo obtém imagens por meio da câmera integrada, permitindo ao usuário apontar para um objeto de interesse e executar o processo de classificação. Ao pressionar o botão "Classificar", a imagem capturada é submetida ao modelo treinado com o algoritmo K-Nearest Neighbors (KNN). E, então, o resultado da classificação da classe do objeto é exibido ao usuário.

1.1 Cronograma

O cronograma do projeto está disponível no formato Gantt Chartt: https://docs.google.com/spreadsheets/d/1p7GvkiMFoUXSKuhSjeIs-9z69lHGfGmW/edit?usp=drive_link&ouid=106099876110466241223&rtpof=true&sd=true

2 Objetivos

Este projeto tem como objetivo compreender e aprofundar os conceitos importantes de análise e reconhecimento de formas vistos na disciplina. Inicialmente, temos a criação de um aplicativo de aquisição de dataset e a organização desse dataset. Esse conjunto de dados constitui uma base fundamental para a criação do aplicativo de classificação. No Jupyter Notebook, diversas tarefas importantes de visão computacional foram feitas para o processamento do dataset e otimizar a performance do classificador. Após todas as técnicas de visão computacional usadas, temos o aplicativo de classificador final, que proporciona em tempo real uma demonstração prática do que foi estudado.

3 Dados e métodos

3.1 Aplicativo de aquisição do dataset

O aplicativo para aquisição de dados foi implementado utilizando o Python3IDE para iOS, que utiliza Python e bibliotecas comuns de ciência de dados, como scikit-learn, OpenCV (cv2) e pandas. A construção da interface do aplicativo foi realizada com a biblioteca Kivy. No aplicativo, o usuário pode selecionar a classe e o fundo desejados por meio de checkboxes na interface do aplicativo. Após capturar a imagem, ela é salva em um diretório chamado "dataset", organizado em pastas conforme o nome da classe.

3.2 Formação do dataset

Para a formação do dataset, foram escolhidas as seguintes 5 classes de objetos: lápis, mouse, livro, capinha de celular e garfo. Para cada classe, foram selecionados três objetos distintos, e para cada objeto, foram capturadas três fotos em fundos claro e escuro. As fotos foram armazenadas na plataforma Google Drive.

O nome dos arquivos segue o formato "classe_objeto_obj-id_cor-do-fundo".

- **classe:** refere-se ao objeto (lápis, mouse, livro, capinha, garfo)

- **objeto:** identifica o objeto dentro de uma classe (A, B, C, por exemplo)
- **obj-id:** indica a foto da sequência de fotos tiradas do objeto (1, 2, 3, por exemplo se foram tiradas 3 fotos)
- **cor-do-fundo:** indica se a imagem foi capturada em um fundo claro ou escuro

Considerando o exemplo: lapis_A_2_escuro. Este nome representa uma imagem de um lápis (classe) do objeto A (objet) na segunda foto (numero-da-foto) com fundo escuro (cor-do-fundo).

Para cada imagem, um arquivo de texto foi gerado com o seguinte formato: "Classe: Objeto: ID do objeto: Cor do Fundo:". Seguindo o exemplo acima, o arquivo conteria: "Classe: lápis Objeto: A ID do objeto: 2 Cor do Fundo: escuro".

3.3 Metadados

Os metadados foram organizados em um arquivo xlsx, contendo as colunas: DIR (indica a pasta da classe e o nome do arquivo), CLASS_NAME, OBJ, OBJ_ID e BACKGROUND. Essas informações foram extraídas do nome do arquivo de cada imagem.

3.4 Tabela sumária dos dados

A tabela sumária incluiu informações gerais sobre o dataset.

Tabela 1: Tabela Sumária dos Dados

Descrição	**Valor**
Número de classes	5
Número de Imagens	90
Tamanho da Base	5.11 MB
Resolução das Imagens	724 x 724
Classe lápis	18 imagens
Classe mouse	18 imagens
Classe livro	18 imagens
Classe capinha	18 imagens
Classe garfo	18 imagens

3.5 Jupyter Notebook - Processamento, segmentação e aprendizado do classificador

Após a obtenção do dataset, foi desenvolvido um Jupyter Notebook com as funções de visão computacional. O código importa as imagens do dataset do Google Drive e realiza o processamento de imagem, usando funções criadas como RGB2GRAY(), gradiente(), logaritmo(), exponencial() e media_convolução().

As imagens geradas por essas funções são normalizadas, e as segmentações manual e automática são realizadas. A segmentação manual foi feita em 15% das imagens de cada classe usando o aplicativo GIMP.

Para avaliar o ground-truth, foi utilizado o Índice de Similaridade de Jaccard, também chamado de interseção sobre a união (IoU). O IoU mede a taxa de sobreposição entre a imagem prevista (segmentação automática) e a imagem base (segmentação manual). O cálculo do IoU foi realizado utilizando duas imagens de segmentação manual: uma em que o objeto possui cor branca com fundo preto e outra em que o objeto tem cor preta com fundo branco. Foi feito desse modo devido às variações na cor do objeto em algumas segmentações automáticas, alternando entre cores preta e branca. Para determinar a correspondência, foi considerado que imagens de segmentação automática com mais de 70% de IoU em relação a uma das duas imagens de segmentação manual foram classificadas como correspondentes, enquanto aquelas com uma taxa de IoU inferior a 70% foram consideradas discrepantes.

Em seguida, foi feita a extração da Feret box, da região de interesse, e as características das imagens. A extração de características foi realizada com base nos pixels de cada imagem, resultando no vetor de características.

Para o classificador, foram testados aproximadamente 10 modelos, e os três que apresentaram as melhores acurácias e tempo de execução, em ordem decrescente, foram: Linear SVM, Neural Net e Nearest Neighbours.

Para o uso do classificador no aplicativo de classificação, a biblioteca "pickle"foi usada para exportar e importar o modelo de classificação como arquivo. Apesar do Linear SVM demonstrar o melhor desempenho entre os classificadores avaliados, o modelo Nearest Neighbours foi usado. Devido ao seu desempenho, por ser leve e pela compatibilidade com as versões do Python e da biblioteca scikit, garantindo um bom uso do aplicativo Python3IDE no iOS.

3.6 Aplicativo de classificação

Assim como usado no aplicativo de aquisição do dataset, o Python3IDE também foi usado para desenvolver o aplicativo de classificação. O aplicativo criado concede ao usuário acesso à câmera, e ao pressionar o botão de classificação, a imagem capturada é processada e classificada utilizando o modelo de classificação treinado. E então, é apresentado ao usuário a classe do objeto como resultado da classificação.

4 Resultados experimentais

Apresente os resultados obtidos, Explore tabelas e gráficos ilustrativos.

Tabela 2: Resultados dos Classificadores

Classificador	**Acurácia**	**Tempo de Treinamento (segundos)**
Linear SVM	0.9167	99.06
Neural Net	0.8722	412.71
Nearest Neighbors	0.8056	3.11
Decision Tree	0.6556	16.64
Random Forest	0.6556	0.39
AdaBoost	0.5167	270.55
Naive Bayes	0.4611	6.46
QDA	0.4167	62.24
Gaussian Process	0.2278	1713.81
RBF SVM	0.1944	186.57

É possível observar na tabela 2 que o classificador Linear SVM alcançou a maior acurácia entre os modelos avaliados, atingindo 91,67%. Além disso, o Neural Net também demonstrou um desempenho semelhante, obtendo uma acurácia de 87,22%. Porém, ambos os modelos apresentaram um tempo de treinamento significativamente longo, com o Linear SVM exigindo 99,06 segundos e o Neural Net, 412,71 segundos. Esses tempos de treinamento longos resultaram em modelos mais pesados, o que impactou uma implementação eficiente no aplicativo para smartphone.

Considerando a necessidade construir um aplicativo para smartphone que seja leve e eficiente, o Nearest Neighbours, que obteve uma acurácia de 80,56% foi escolhido como modelo de classificação. Pois, além de apresentar uma boa acurácia, também teve um tempo de treinamento curto (3,11 segundos).

Na tabela 3, as amostras de IoU apresentam porcentagens sobre a sobreposição entre as segmentações automáticas e manuais. Observando os resultados, é possível notar que a média dos IoU ficou em torno de 50%.

5 Discussão

Interprete os resultados e apresente uma visão crítica.

Não se esqueça de incluir as referências bibliográficas e citá-las no texto. Use o Bibtex.

O projeto proporcionou uma compreensão mais aprofundada dos conceitos de análise e reconhecimento de formas, aplicando esses conhecimentos na prática com a construção do aplicativo classificador.

Ao observar os resultados obtidos sobre os classificadores, é evidente que o classificador selecionado é rápido e eficiente para o aplicativo desenvolvido. No entanto, embora apresente uma acurácia de

Tabela 3: Resultados do IoU para Segmentação Automática

Imagem	**Classe**	**IoU (%)**	**Observação**
garfo_A_2_claro.jpg	Garfo	53.74	Sobreposição insuficiente
lapis_C_2_claro.jpg	Lápis	54.63	Sobreposição insuficiente
garfo_C_3_claro.jpg	Garfo	52.78	Sobreposição insuficiente
capinha_A_1_escuro.jpg	Capinha	51.93	Sobreposição insuficiente
livro_C_3_escuro.jpg	Livro	86.38	Sobreposição suficientemente alta
mouse_B_3_claro.jpg	Mouse	62.37	Sobreposição insuficiente
livro_B_2_escuro.jpg	Livro	91.24	Sobreposição suficientemente alta
mouse_C_2_claro.jpg	Mouse	64.51	Sobreposição insuficiente
lapis_C_1_claro.jpg	Lápis	55.23	Sobreposição insuficiente
capinha_C_2_claro.jpg	Capinha	72.92	Sobreposição suficientemente alta

80%, não é considerado o melhor, pois o linear SVM alcança uma acurácia de 90%, marcando uma diferença em termos de eficácia. Mas, ainda assim o modelo Nearest Neighbors demonstra desempenho adequado para os objetivos do projeto.

Também foi usado o Índice de Sobreposição (IoU) para avaliar a segmentação automática no projeto. Pelos resultados mostrados, é possível observar que a média do IoU está em torno de 50%. Essa porcentagem, no entanto, não está de acordo com a métrica desejada, que considera uma sobreposição alta para IoU $\geq 0{,}7$. Diante disso, é notável que o desempenho da segmentação automática não atingiu o nível desejado.

A partir dessas observações, embora o classificador não tenha sido otimizado da melhor forma possível para ter previsões mais precisas, ainda possui resultados bons para classificação.