TÉCNICAS DE VISÃO COMPUTACIONAL PARA CLASSIFICAÇÃO DE PEÇAS

Pedro Henrique Benigno Rodrigues
Alexandre Lasthaus
Universidade Presbiteriana Mackenzie (UPM)

Resumo

Visão computacional é uma área da inteligência artificial que tem grande impacto em vários setores da sociedade. A capacidade da máquina de enxergar objetos de interesse em uma imagem e produzir uma resposta de classificação ou detecção de elementos é de suma importância no contexto de automação. As técnicas de visão computacional permitem, por meio de etapas de tratamento de imagem e do uso de classificadores, oferecer respostas a diversos problemas que se apresentam. O objetivo deste trabalho é analisar a capacidade de resposta de um classificador em específico, o *K-nearest neighbors* (KNN), para um problema de classificação de um grupo e analisar seu desempenho por meio de parâmetros de taxa de acerto e precisão. Primeiramente, foram criados quatro grupos de peças em um *software* de desenho em três dimensões (3D) e cortadas numa máquina de corte a *laser*, depois foram tiradas fotos de cada uma das peças individualmente. Posteriormente, fazendo uso das bibliotecas Open Source Computer Vision Library (OpenCV), pudemos realizar o processamento das imagens como mudança do padrão *red-green-blue* (RGB) para tons de cinza, binarização, correção de ruídos e obtenção de momentos invariantes; já utilizando a biblioteca Scikit-Learn, pudemos realizar o treinamento do classificador e os testes. Pôde-se concluir que o classificador, nos testes 1 e 2, foi capaz de classificar de forma adequada as classes das peças tendo *recall* e precisão acima de 65 e 80%, enquanto no teste 3 há um *recall* de 42% e precisão de 55%, mostrando que, com mais classes analisadas, o classificador diminui sua eficiência.

Palavras-chave: Visão computacional. KNN. Peças.

1 INTRODUÇÃO

Visão computacional é um ramo da área de inteligência artificial (IA) que tem como foco a emulação da visão humana por meio de computadores, utilizando conceitos e técnicas de aprendizado de máquina (*machine learning*) e processamento digital de imagens, incluindo a capacidade de fazer associações e agir com base em informações visuais (GONZALEZ; WOODS, 2010).

A presença de algoritmos de *machine learning*, tais como os K-vizinhos mais próximos ponderados (*K-nearest neighbors* [KNN]), as redes neurais artificiais (RNA) e a máquina de vetores de suporte (*support vector machine* [SVM)], serve de base para o desenvolvimento dessa tecnologia, visto que contribuem para abstrair informações e buscar fatores de correlação entre cada elemento da informação. Para isso, os dados precisam ser estruturados e os elementos importantes apresentados de forma manual para o algoritmo (RODRIGUES, 2020). Sendo assim, essa abordagem é capaz de generalizar determinado conhecimento adquirido durante um processo de treinamento, indo ao encontro do comportamento esperado para um sistema de visão computacional que irá fazer reconhecimento a partir de um conjunto de imagens e então classificar os elementos semelhantes (MIRANDA, 2011).

Essa tecnologia é fundamental por ter inúmeras aplicações a diversas atividades. Com a criação e o aperfeiçoamento das ferramentas de detecção e análise, a visão computacional criou sistemas capazes de interagir e tomar decisões a partir de elementos visuais, possibilitando a realização de tarefas que antes acreditava-se que só o ser humano poderia realizar, por exemplo: por meio de veículos autônomos, é possível obter localização para produzir mapas do ambiente e detectar obstáculos (MACHADO, 2017); além da possibilidade de detecção de anomalias em exames de imagem, como tomografia computadorizada, ressonância magnética e ultrassonografia (BARELLI, 2018).

No contexto de revolução industrial atual, embasado pela Indústria 4.0, no qual há uma necessidade de aumento na produtividade, melhorando o processo produtivo e oferecendo serviços de ponta, fazem-se imperativos o uso e a análise de técnicas focadas na abordagem de visão de máquina que promovam maior automação de serviços e facilidade no reconhecimento de padrões e classificações de objetos de interesses.

Kumar (2008) demonstra que há um aumento na qualidade do produto final devido à robustez do processamento digital de imagens para classificar defeitos quando ocorre a automação da detecção de defeitos em processos de fabricação de tecidos utilizando visão computacional.

Ferreira e Jaimes (2018) introduzem um sistema de automatização de reconhecimento de números de série de placas de aço bruto por meio de visão computacional com precisão de 93% em ambientes controlados.

Rodrigues (2020) promove um sistema de classificação de peças mecânicas por meio da geração de imagens sintéticas utilizando algoritmos de *machine learning*, através de redes neurais e técnicas de visão computacional, obtendo 89% de precisão.

Nesse contexto de implementação e análise, o presente trabalho tem como objetivo resolver um problema de classificação de imagens de peças por meio da manipulação das ferramentas de visão computacional utilizando o classificador KNN e, como objetivos específicos, analisar seu desempenho considerando como medida de análise o *recall*, a precisão e a matriz de confusão, bem como utilizar os métodos de processamento digital de imagens para obtenção de áreas de interesse nas imagens, por meio da biblioteca Open Source Computer Vision Library (OpenCV), e a manipulação do classificador por meio da biblioteca Scikit-Learn, ambas manipuláveis em Python.

2 REVISÃO DA LITERATURA

Em um sistema de visão computacional, é necessário passar por etapas de processamento digital de imagens e fazer o uso de algoritmos de *machine learning* para identificar e classificar os objetos. Nessas etapas, o processo tomará como base obter imagens com objetos de interesses; tratar os possíveis erros ou ruídos por meio de filtros; segmentar os objetos de interesses; e extrair características intrínsecas de modo a uniformizar as informações obtidas, criando, assim, um banco de dados para promover o reconhecimento ou a classificação da imagem.

2.1 Aquisição de imagens

A aquisição de uma imagem é feita por meio de sensores que recebem as ondas de luz tridimensionais e geram um sinal elétrico proporcional à intensidade de energia. Em seguida, há a conversão para um sinal digital bidimensional. Uma imagem pode ser definida, segundo Vieira Neto e Marques Filho (1999), como o produto de dois fatores, sendo eles a intensidade de luz que incide sobre um objeto e a fração de luz incidente que o objeto transmite.

Atualmente os aparelhos de captura de imagem utilizam os sensores do tipo *charge coupled device* (CCD) e *complementary metal-oxide semicondutor* (CMOS). Os sensores CCD são usados em diversos equipamentos digitais, tais como *tablets*, *smartphones* e *webcams* e utilizam uma matriz de capacitores fotossensíveis para capturar a carga de energia correspondente à intensidade luminosa e depois convertê-la por meio de conversor analógico-digital (AD). Os CMOS têm o mesmo princípio, porém não precisam de um conversor AD (BARELLI, 2018).

Para a representação de cores de uma imagem, é muito comum o uso do padrão *red-green-blue* (RGB). Esse padrão foi elaborado segundo a teoria tricromática de Young-Helmholtz, segundo a qual, para a percepção de cores pelo olho humano, há três cores primárias – vermelho, verde e azul (PEDRINI; SCHWARTZ, 2008). Cada cor é capturada por um CCD, que tem um conjunto de prismas e filtros responsáveis pela decomposição da imagem colorida em componentes específicos (BARELLI, 2018).

2.2 Pré-processamento

A etapa de pré-processamento busca melhorar as imagens adquiridas com o intuito de destacá-las na etapa de segmentação. Entretanto, algumas vezes, as imagens encontram-se com problemas que dificultam sua leitura, seja quando são obtidas seja ao longo das etapas, e elas podem apresentar defeitos por meio de ruídos ou imperfeições relacionadas à descontinuidade. Para corrigir isso, faz-se necessário o uso de filtros, que são matrizes de tamanhos variados que irão percorrer a imagem, realçando detalhes como bordas e contornos, eliminação de ruídos, bem como irão realizar operações morfológicas que corrigem vazios e descontinuidades (BARELLI, 2018).

2.3 Segmentação

A segmentação é um dos problemas mais antigos e amplamente estudados em processamento digital de imagens (SZELISKI, 2010). Consiste em separar do plano da imagem que se estuda o objeto em questão. Essa etapa é de grande importância, pois ela qualifica os diferentes elementos para, na etapa seguinte, extrair suas características definidas. Segundo Pedrini e Schwartz (2008), esse procedimento é necessário para que as informações resultantes do sistema de análise sejam confiáveis.

Em termos de abordagem, a segmentação procura tratar de duas propriedades básicas dos níveis de cinza de uma imagem: a descontinuidade e a similaridade. Na primeira, as ferramentas buscam separar a imagem com base em variações abruptas

dos níveis de cinza, como bordas; na segunda, as ferramentas procuram agrupar pontos da imagem que trazem valores similares para um determinado conjunto de características, como no caso da limiarização da imagem (PEDRINI; SCHWARTZ, 2008). Ela acontece por meio da definição de um limiar de intensidade de *pixels*, cujo resultado será uma imagem binária. Em uma imagem com um objeto de interesse, estabelece-se um valor do qual os *pixels* do objeto estarão acima e, desse modo, destacar-se-ão como a imagem a ser obtida ou o primeiro plano, geralmente na cor branca. Do contrário, os *pixels* que possuem valor abaixo do limiar serão considerados o segundo plano, sendo redefinidos para cor preta.

2.4 Extração de características

A extração de características é umas das etapas fundamentais de um sistema de visão computacional, visto que ela consiste na obtenção de informações que servirão de base para a etapa seguinte de classificação e identificação. Para isso, faz-se necessário estabelecer parâmetros quantificáveis do objeto de interesse. Esses parâmetros variam de acordo com o aspecto, dimensões, posição, orientação e até mesmo características da luz (CONCI; AZEVEDO; LETA, 2008).

Um dos principais métodos para a extração das propriedades do objeto de interesse é o de momentos da imagem, que descreve a distribuição espacial dos pontos em uma imagem ou região.

2.4.1 Momentos da imagem

Os momentos de uma imagem são definidos por funções matemáticas, com base estatística, que fornecerão valores mostrando as propriedades intrínsecas do elemento que se deseja destacar e classificar. Esses valores são calculados tendo como base imagens binárias de um objeto já segmentado. As propriedades que compõem os momentos podem ser área, orientação, centro geométrico ou vetores invariantes em relação à escala, rotação e translação da imagem (BARELLI, 2018).

A fórmula que representa o conceito matemático de um momento é:

$$m_{pq} = \sum_{1}^{nx} \sum_{1}^{ny} x^p y^q \, f(x, y) \tag{1}$$

O valor Mpq representa o momento de ordem (p + q) da função de intensidade f(x; y), em que nx e ny representam, respectivamente, a largura e a altura da imagem

digital. Como a imagem é binária, os valores da função *f*(x, y) serão iguais a 0 ou a 1 (CHACON *et al.*, 2011).

A partir dos momentos regulares, podemos definir algumas medidas importantes sobre os objetos de interesse, como os momentos de ordem 0 e 1, que são usados para o cálculo do baricentro do objeto por meio da função abaixo:

$$x_c = \frac{m_{10}}{m_{00}} \qquad y_c = \frac{m_{01}}{m_{00}} \tag{2}$$

O momento m_{00} representa a área do objeto, enquanto m_{10} e m_{01} mostram as projeções nos eixos *x* e *y* respectivamente.

Momentos centrais μ_{pq} são momentos geométricos da imagem calculada com relação ao baricentro x_c e y_c:

$$\mu_{pq} = \sum_{1}^{nx}\sum_{1}^{ny}(x - x_c)^p (y - y_c)^q \, f(x, y) \tag{3}$$

Finalmente, existem os momentos centrais normalizados representados por η_{pq} definidos pela seguinte equação:

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{pq}^{\gamma}}, \text{ em que } \gamma = \frac{p+q}{2} + 1, \ \forall \ p + q \geq 2 \tag{4}$$

Essa equação servirá de base para definir os momentos invariantes de Hu.

2.4.1.1 Momentos invariantes de Hu

Um bom sistema de classificação precisa identificar os objetos para o qual foi treinado independentemente da disposição desse, ou seja, as características devem ser invariantes à translação, à rotação e à escala das imagens. Hu (1962) propôs um conjunto de sete momentos que atendesse a esses requisitos chamados por Chacon *et al.* (2011) de momentos de Hu, ou momentos invariantes.

As equações abaixo representam os setes momentos calculados a partir dos momentos normalizados até a terceira ordem:

$$M1 = \eta_{20} + \eta_{02}, \tag{5}$$

$$M2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}{}^2, \tag{6}$$

$$M3 = (\eta_{30} - 3\eta_{12})^2 + 3(\eta_{03} - 3\eta_{21})^2, \tag{7}$$

$$M4 = (\eta_{30} + \eta_{12})^2 + (\eta_{03} + \eta_{21})^2, \tag{8}$$

$$M5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{03} + \eta_{21})^2] + (3\eta_{21} - \eta_{03})(\eta_{03} + \eta_{21})[3(\eta_{30} + \eta_{12})^2 - (\eta_{03} + \eta_{12})^2], \tag{9}$$

$$M6 = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - 7(\eta_{03} + \eta_{21})^2] + 4\eta_{11}[(\eta_{30} + \eta_{12})(\eta_{03} + \eta_{21})], \tag{10}$$

$$M7 = 3(\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{03} + \eta_{21})^2] + (\eta_{30} - 3\eta_{12})(\eta_{03} + \eta_{21})[3(\eta_{30} + \eta_{12})^2 - (\eta_{03} + \eta_{21})^2] \tag{11}$$

2.5 Classificação e reconhecimento

Classificação e reconhecimento são as etapas finais, em que são feitas as comparações necessárias entre os objetos de interesse de modo a organizar um conjunto com propriedade em comum. Segundo Pedrini e Schwartz (2008), esse procedimento visa a determinar um mapeamento que relacione as características das amostras com um conjunto de rótulos, de modo que amostras com propriedades semelhantes são mapeadas no mesmo rótulo.

Para estabelecer o mapeamento entre as características das amostras e o con junto de rótulos, são usados os classificadores. Os classificadores buscam atribuir uma classe específica a cada amostra. O tipo de trabalho que eles realizam pode ser dividido em classificação supervisionada ou não supervisionada (PEDRINI; SCHWARTZ, 2008).

A classificação supervisionada consiste no treinamento de um algoritmo para que ele consiga reconhecer padrões a partir de objetos já conhecidos, ou seja, um conjunto de características de objetos e suas respectivas classes são mostrados ao classificador. Desse modo, torna-se capaz de classificar automaticamente novos objetos (BARELLI, 2018). Existem várias técnicas para realizar a classificação de objetos, desde as mais simples, baseadas em KNN, até as mais complexas, como redes neurais e SVM (CONCI; AZEVEDO; LETA, 2008).

Neste trabalho, como exposto anteriormente, será realizada a abordagem supervisionada com o uso do algoritmo KNN para avaliação da classificação na definição dos grupos de peças.

A classificação não supervisionada é semelhante à supervisionada no que tange a receber uma lista de características de diversos objetos para treinamento, porém, o ponto central de diferença é o fato de que não é necessário apresentar classificações para os objetos, descartando a necessidade de um supervisor. Os classificadores desse tipo são capazes de identificar classes apenas analisando quão semelhantes são as propriedades dos objetos (BARELLI, 2018).

2.5.1 O classificador KNN

O classificador KNN é um dos métodos mais simples e mais antigos para reconhecimento de padrões supervisionado. Estudado por Cover e Hart em 1967 para valores fixos de *k*, até ser estabelecido por Patrick e Fischer em 1970 como um classificador generalizado para múltiplas classes (SILVA, 2014).

O funcionamento do classificador tem como base a analogia. Em um conjunto de dados, cada amostra corresponde a um ponto em um espaço de dimensão N, em que N é o número de características de cada amostra. A fim de definir a classe de uma nova amostra, o classificador procura *k* elementos do conjunto de dados que estejam mais próximos dessa amostra, ou seja, que apresentem a menor distância. Verifica-se quais são as classes mais frequentes desses elementos mais próximos e, então, a classe mais frequente será atribuída ao elemento desconhecido (SILVA, 2014).

Para definir a distância de uma amostra para outra, utiliza-se o cálculo da distância euclidiana (SILVA, 2014):

$$d_e = \sqrt{\sum_{i=1}^{p} (X_i - Y_i)^2} \tag{12}$$

Nessa equação, d_e representa a distância entre os pontos X_i e Y_i, enquanto p, o número total de características de cada ponto.

Para determinar a classe, faz-se uso da equação abaixo (MITCHELL, 1997):

$$f(x_q) \leftarrow argmax_{v \epsilon V} \sum_{i=1}^{k} \delta(v, f(x_i)) \tag{13}$$

Aqui, $f(x_q)$ recebe o valor final da classificação da amostra x_q. Esse valor é obtido da função *argmax*, que retorna o valor da classe (v) obtido a partir de um conjunto de V classes possíveis, levando em conta todos os k vizinhos a serem analisados. A função $f(x_i)$ representa a classe da amostra x_i e sigma tem o valor um, caso $f(x_i)$ for igual à classe v que está sendo considerada – caso contrário, será zero.

2.6 OpenCV

OpenCV é uma biblioteca de *software* desenvolvida pela Intel em 2000, com foco em aplicações de visão computacional e *machine learning* para processamento digital de imagens (PDI) em código aberto.

A biblioteca tem mais de 2.500 algoritmos otimizados, o que inclui um conjunto abrangente de algoritmos de visão computacional e de *machine learning* clássicos e de última geração, os quais podem ser usados para detectar e reconhecer rostos, identificar objetos, classificar ações humanas em vídeos, rastrear movimentos de câmera, objetos em movimento, extrair modelos 3D de objetos, produzir nuvens de pontos a partir de câmeras estereoscópicas, unir imagens para produzir alta resolução, acompanhar movimentos dos olhos, reconhecer paisagens, estabelecer marcadores para sobreposição com realidade aumentada, entre outros (BRADSKI, 2000).

Essa biblioteca tem interface para várias plataformas, como Python, Java, matrix laboratory (MatLab), C++ e suporta Windows, Linux, Android e MacOS.

2.7 Scikit-Learn

Scikit-Learn é uma biblioteca da linguagem Python que trabalha com conceitos de IA por meio de algoritmos e técnicas de *machine learning*. Essa biblioteca dispõe de métodos para ajudar na criação, no treinamento e na validação de classificadores de visão computacional, como o KNN (BARELLI, 2018).

2.8 Python

Python é uma linguagem de programação de alto nível, com código aberto, voltada para orientação a objeto. Foi criada em 1991 pelo holandês Guido Van Rossum visando à fácil utilização e escrita linguagem.

Muitas linguagens, como C/C++ ou Java, têm uma sintaxe de construção mais extensa ou detalhada, que acaba criando certas barreiras ao entendimento ou ampliando

o código de forma demasiada e Python surge para solucionar essas questões. Por ser uma linguagem robusta e com muitas funcionalidades, Python teve um crescimento acentuado nos últimos anos, sendo usada em áreas de ciências de dados, mercado financeiro, IA, automação, entre outras (DIDÁTICA.TECH, 2022).

Além de sua praticidade, há uma grande variedade de bibliotecas e pacotes que ampliaram a capacidade da linguagem facilitando a elaboração de códigos e suas aplicações, como a biblioteca OpenCV, que auxilia em atividades de visão computacional.

3 METODOLOGIA

Nesta fase do trabalho, foram realizadas pesquisas em documentação das bibliotecas, livros, artigos e trabalhos correlatos para auxiliar no bom uso das ferramentas e melhor estruturar o classificador e os métodos de avaliação. Além disso, explicam-se as peças criadas para servir de treinamento e teste de classificador e os detalhes de cada etapa do processamento digital das imagens obtidas.

3.1 Peças

As análises poderiam ter sido executadas utilizando-se apenas as fotos da tela de computador das peças virtuais criadas no aplicativo Inventor, um *software* de desenho 3D da empresa Autodesk, mas se optou por ter em mãos as peças físicas. Assim, os desenhos criados foram salvos em arquivos DXF e depois exportados para serem utilizados em uma máquina de corte a laser, para gerar cada uma das peças em placa de fibra de média densidade (*medium density fiberboard* [MDF]). A máquina utilizada foi uma Golden Layer, modelo 1080, cujo acesso se deu nas dependências do laboratório de mecânica da Escola de Engenharia da Universidade Presbiteriana Mackenzie, assim como o MDF para a fabricação das peças.

As peças definidas para este trabalho foram divididas em quatro classes e cada uma tem cinco peças de treinamento e três peças para o teste propriamente dito. Elas foram elaboradas no Inventor. A Figura 1 mostra as peças de treinamento e de teste com suas respectivas classes.

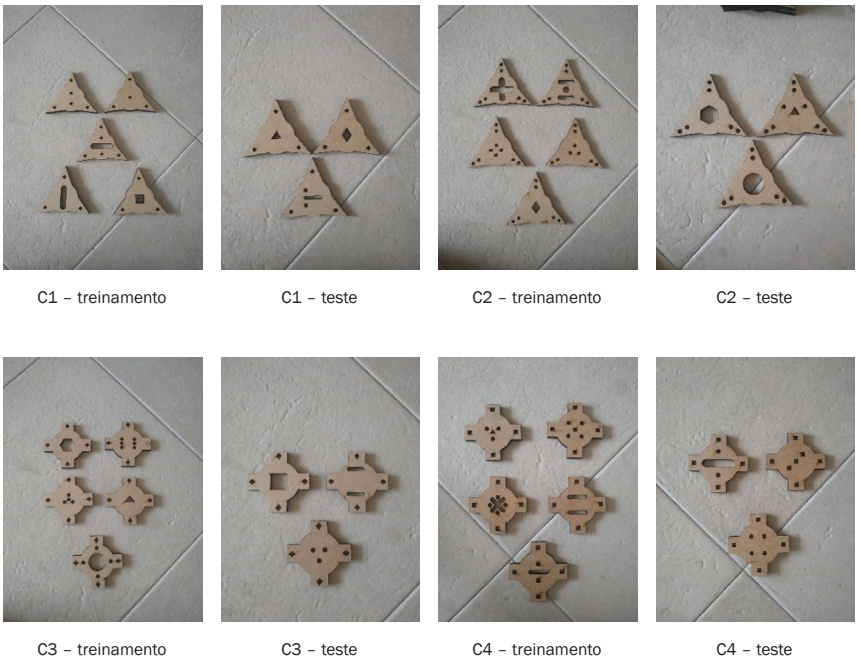


Figura 1 Fotos das peças de treinamento e teste utilizadas no experimento

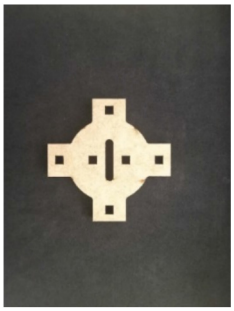
Fonte: Elaborada pelos autores.

3.2 Etapas e descrição do processamento

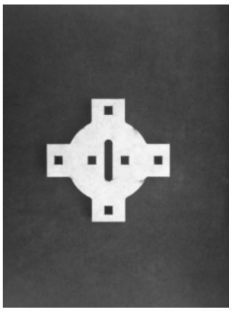
Tendo definido as peças que serão trabalhadas na passagem anterior, segue-se para a etapa de processamento digital das imagens, fazendo uso dos recursos de visão computacional a partir das bibliotecas OpenCV e Scikit-Learn. Todas as estágios descritos a seguir foram realizados utilizando linguagem Python em um ambiente de desenvolvimento chamado Pycharm. O computador utilizado para as operações foi um notebook HP com Intel Core i7-6500 2.5GHz, 6 GB de memória RAM e disco rígido de 1 TB.

Na etapa de aquisição, foram geradas imagens individuais das peças utilizando a câmera de um celular Motorola G7 Plus com sensor do tipo CMOS. Cada peça foi colocada por cima de uma base escura de modo a destacar o fundo do objeto de interesse em questão. Como essas imagens são geradas segundo o padrão RGB, é necessário que haja a conversão desse padrão para a escala de cinza, para posterior uso

na etapa de segmentação. A conversão é feita utilizando uma função chamada *cv2.cvtColor*, a qual recebe dois parâmetros, sendo a primeira a imagem e a segunda, o padrão de transformação – nesse caso, cv2.COLOR_RGB2GRAY. A Figura 2 ilustra o resultado para uma amostra.



Padrão RGB



Padrão escala de cinza

Figura 2 Fotografias da amostra

Fonte: Elaborada pelos autores.

Além disso, optou-se por redimensionar o tamanho das imagens de 1200 x 1600 *pixels* para 400 x 533 *pixels*, visto que, desse modo, haveria uma melhor visualização das amostras como um todo, enquanto na outra resolução a imagem ficava com os seus detalhes comprometidos, pois o programa não a mostrava em totalidade.

Na etapa seguinte, foi feita a segmentação do objeto de interesse, ou seja, a substituição da escala de cinza pelo padrão binário. Para isso, utilizou-se a função cv2.threshold, que recebe quatro parâmetros: a imagem em questão; o limiar de intensidade que corresponde à intensidade do valor do *pixel* que se quer destacar (neste trabalho, o número 100); a intensidade do *pixel* na imagem binarizada (ilustrada na Figura 3), cujo valor foi de 255; e, por fim, a cor de destaque do objeto de interesse, que, no caso, foi branca por meio da função cv2.THRESH_BINARY.

Figura 3 Imagem binarizada da amostra

Fonte: Elaborada pelos autores.

Salienta-se que a etapa de pré-processamento não foi utilizada, porque a imagem em escala de cinza não apresentava distorções ou ruídos no seu contexto. No entanto, ao fim do processo de segmentação, notou-se que, nas classes C3 e C4, havia alguns ruídos, como pontos brancos e descontinuidades. Para corrigir esses problemas, fez-se uso de uma operação morfológica de abertura e de dilatação da etapa de pré-processameto.

A operação de abertura consiste em erodir a imagem e depois dilatá-la de modo que ruídos fora do objeto de interesse desapareçam. Para isso, utiliza-se a função cv2.morphologyEx, tendo como parâmetros a imagem em questão, a função da operação de abertura cv2.MORPH_OPEN e a matriz de tamanho 3 x 3, que irá perpassar a imagem.

Na operação de dilatação, as descontinuidades presentes na classe C4 foram corrigidas de modo a deixar a imagem homogênea. Para isso, utilizamos a função cv2.dilate, que recebe a imagem, a matriz 3 x 3 que irá perpassar a imagem e o número de iterações suficientes para retirar as descontinuidades. Nesse caso, encontrou-se o valor de duas iterações como satisfatório.

Na etapa de extração de características, foram utilizados os sete momentos invariantes de Hu, conforme explicado na seção 2.4.1.1. A biblioteca OpenCV é utilizada primeiramente para o cálculo de todos os momentos da imagem por meio da função cv2.moments. Em seguida, os resultados obtidos servirão de parâmetros para a função cv2.HuMoments, que irá listar os sete momentos invariantes. Por fim, é necessário normalizar os valores obtidos para uma escala logarítmica, pois, como estamos trabalhando com figuras planas, assim os valores transformados terão melhor desempenho (BARELLI, 2018). A função utilizada foi -np.sign(cv2.HuMoments) * np.log10(np.abs(cv2.HuMoments). A Tabela 1 mostra os valores de momentos normalizados obtidos das amostras de cada classe.

TABELA 1

Momentos das amostras estudadas

	1	2	3	4	5	6	7	8
	2,95	2,97	3,06	3,07	3,01	3,05	3,03	3,03
	10,34	9,29	9,39	9,65	9,34	9,71	9,75	9,81
	8,94	8,99	9,26	9,28	9,12	9,24	9,18	9,16
C1	14,12	13,34	13,29	13,34	13,5	13,8	13,58	12,86
	-25,69	25,74	-24,58	-24,67	-24,82	-25,64	25,82	24,47
	19,65	-17,99	-17,99	-18,17	-18,17	-18,71	-19,05	17,92
	26,12	-24,51	25,3	25,27	25,51	-25,37	24,97	-23,89
	3,03	3,05	2,95	3,06	2,9	2,96	2,93	3,051
	10,62	9,37	8,79	10,01	9,76	10,38	9,25	9,78
	9,2	9,25	8,95	9,26	8,82	9,01	8,92	9,23
C2	13,88	12,81	13,07	13,41	12,95	13,7	12,69	12,37
	25,57	24,08	24,12	25,23	-24,56	26,03	-23,66	23,27
	19,28	17,71	17,51	18,71	18,25	18,96	-17,51	17,48
	-25,59	-23,94	-24,46	-24,77	-23,85	-25,06	-23,64	-23,41
	3,1	3,072	3,09	3,11	3,044	3,02	3,11	3,11
	11,33	10,29	10,9	11,23	10,73	9,35	10,42	10,64
	13,56	13,17	14,49	13,16	12,56	12,82	12,84	13,6
C3	14,78	14,44	15,89	14,55	15,11	14,44	13,96	15,48
	-28,98	-28,32	31,36	28,42	29,84	-28,12	-27,37	-30,39
	20,65	-20,05	21,37	21,5	-20,63	-19,72	19,46	20,81
	-29,47	-28,55	-31,16	29,13	-28,95	28,47	-28,37	30,07
	3	3,13	3,13	3,05	3,14	3,04	3,12	3,14
	10,41	10,47	10,52	11,1	12,52	13,29	9,9	12,27
	12,47	13,18	15,37	13,76	13,89	13,69	13,43	13,57
C4	14,96	14,85	15,26	15,1	15,17	15,46	13,83	15,34
	29,1	29,73	-30,8	29,55	29,76	30,46	27,52	29,95
	20,18	-20,17	20,52	20,66	22,66	-22,17	18,83	-21,69
	-28,71	28,88	30,67	-30,21	-29,99	-30,07	-27,75	29,95

Fonte: Elaborada pelos autores.

Por fim, na etapa de classificação, realiza-se o treinamento do KNN e os testes. Nessa fase, é utilizada a biblioteca Scikit-Learn, visto que nela é onde se encontra o algoritmo. Para isso, cria-se uma lista para o treinamento com os momentos das amostras 1 até 5 e uma outra para teste com os momentos das amostras 6 até 8. Outra lista é criada para estabelecer as respectivas classes das amostras da etapa de treinamento. A função utilizada para o treinamento é knn.fit, a qual recebe a lista com amostras de treinamento e a lista com as respectivas classes dessas amostras. Para o teste, a função utilizada é a knn.predict, que recebe a lista com os momentos das amostras de teste.

Para analisar e avaliar o desempenho do KNN, realizamos três experiências de classificação, tendo como parâmetros o *recall* e a precisão e como representação numérica das classificações, a matriz de confusão. O *recall* analisa quão eficiente é o classificador para prever os valores positivos, chamados de verdadeiros positivos. A relação consiste em fazer a divisão dos verdadeiros positivos (TP) pela soma de verdadeiros positivos (TP) e falsos positivos (FP). Assim, Precisão = TP/(TP+FP). A precisão irá focar qual proporção de identificações positivas foi realmente correta. Para isso, a relação será de verdadeiros positivos dividido pela soma de verdadeiros positivos mais falsos positivos. Já a matriz de confusão trata da frequência de classificação para cada classe do algoritmo estudado (MACHINE LEARNING COURSE, 2020).

Além disso, é necessário ressaltar que diferentemente de alguns trabalhos analisados na bibliografia, a quantidade de amostras usadas para treinamento é bastante pequena, visto que as peças utilizadas foram fabricadas uma a uma e fotografadas individualmente para uso neste estudo, em vez de retiradas de repositórios disponíveis na internet, como no caso de Chacon *et al*. (2011), que utilizaram 28 imagens por elemento. No entanto, como o intuito é estudar a resposta do classificador a um problema de classificação e avaliar seu desempenho, as conclusões puderam oferecer implicações satisfatórias ao presente estudo.

4 RESULTADOS E DISCUSSÃO

Segundo os resultados obtidos no teste 1, pode-se perceber que o KNN manteve um *recall* (Figura 3) constante de 67% para todos K-vizinhos analisados e com uma precisão de 80% quando analisado o K sendo igual a 1 e 9, conforme a Figura 4. A matriz de confusão da Tabela 2 mostra as classificações nesses dois casos.

No teste 2, nota-se que houve uma variação no *recall* conforme ilustrado na Figura 5, e o melhor resultado foi obtido com K igual 5 (83%). Na precisão ilustrada na

Figura 6, o valor de K também é igual a 5 (88%). A matriz de confusão da Tabela 3 ilustra a classificação para esse K.

Por fim, no teste 3, ilustrado na Figura 7, é perceptível a queda acentuada no *recall*, obtendo-se, no melhor valor, 42% para K igual a 1, 3, 5 e 6. Na análise de precisão, a Figura 8 mostra que somente com K igual a 3 é notado o maior valor, de 55%, porém demonstrando queda igualmente quando comparado aos dois últimos testes. A matriz de confusão da Tabela 4 mostra a classificação para o K igual a 3, sendo esse teste o que teve melhor desempenho.

Figura 3 *Recall versus* valores de K-vizinhos para o teste 1 (C1 *versus* C2)

Fonte: Elaborada pelos autores.

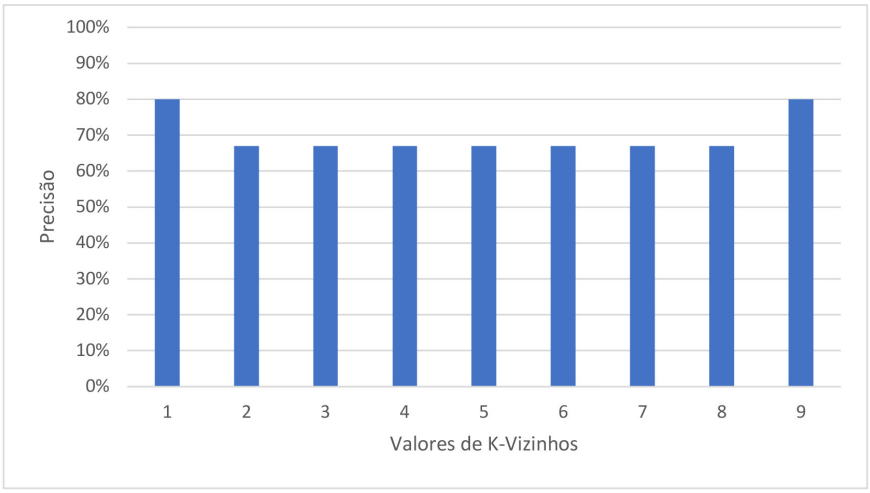


Figura 4 Precisão *versus* valores de K-vizinhos para o teste 1 (C1 *versus* C2)

Fonte: Elaborada pelos autores.

TABELA 2

Matriz de confusão C1 *versus* C2 para o teste 1

		Valores preditos	
		C1	C2
Valores reais	C1	1	2
	C2	0	3

Fonte: Elaborada pelos autores.

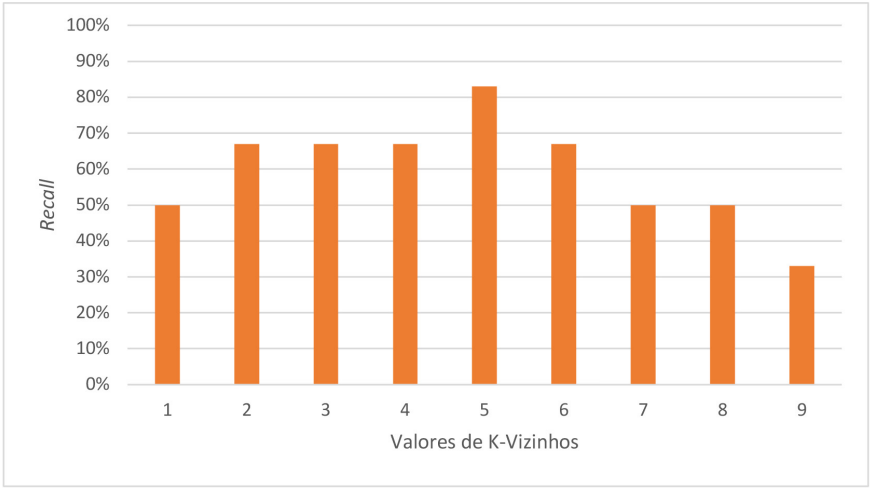


Figura 5 *Recall versus* valores de K-vizinhos para o teste 2 (C3 *versus* C4)

Fonte: Elaborada pelos autores.

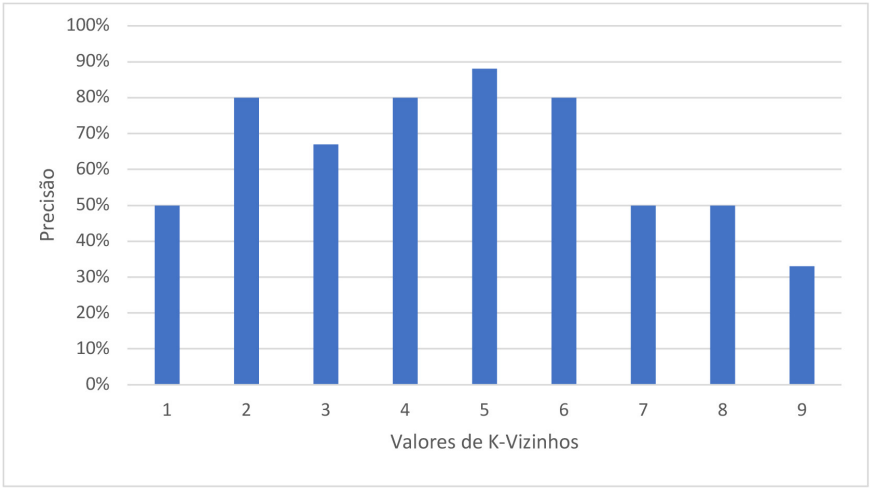


Figura 6 Precisão *versus* valores de K-vizinhos para o teste 2 (C3 *versus* C4)

Fonte: Elaborada pelos autores.

TABELA 3

Matriz de confusão C3 *versus* C4 para o teste 2

		Valores preditos	
		C3	C4
Valores reais	C3	2	1
	C4	0	3

Fonte: Elaborada pelos autores.

Figura 7 *Recall versus* valores de K-vizinhos para o teste 3 (todas as classes juntas)

Fonte: Elaborada pelos autores.

Figura 8 Precisão *versus* valores de K-vizinhos para o teste 3 (todas as classes juntas)

Fonte: Elaborada pelos autores.

TABELA 4

Matriz de confusão para o teste 3 (todas as classes juntas)

		Valores preditos			
		C1	C2	C3	C4
Valores reais	C1	1	1	1	0
	C2	0	2	1	0
	C3	2	0	1	0
	C4	2	0	0	1

Fonte: Elaborada pelos autores.

5 CONSIDERAÇÕES FINAIS

É importante salientar que, assim como foi dito na seção anterior, este estudo se trata de uma análise específica do classificador KNN na resolução de um problema com peças fabricadas para este trabalho, ou seja, os valores aqui são estabelecidos seguindo o padrão de imagens geradas na etapa de aquisição, bem como dos valores de momentos de Hu, responsáveis por alimentar o classificador na etapa de treinamento e teste. É possível que, considerando outros padrões geométricos de peças, novos tipos de classes e um aumento de amostras, os valores percentuais sofram alterações.

Conclui-se, então, que o KNN foi capaz de obter percentuais de classificação razoáveis para os dois primeiros testes quando considerados apenas dois grupos de classes. No entanto, no teste 3, quando analisado o conjunto completo e não apenas as duas classes, o algoritmo perdeu capacidade e não conseguiu oferecer valores satisfatórios de classificação, produzindo percentuais muito abaixo em relação aos dois primeiros. Isso demonstra que o algoritmo tende a piorar quanto mais classes ele considera na classificação. Sendo assim, em uma possível aplicação comercial ou industrial, o KNN teria que ser usado para comparação de classes específicas e não de forma conjunta.

Um ponto a destacar é que foram usados alguns parâmetros comuns na análise do KNN – os momentos de Hu e o cálculo da distância por meio do padrão euclidiano. No entanto, há outros extratores de características que podem ser utilizados além dos de Hu, como no trabalho de Silva (2014), em que o autor analisa outros seis tipos de momentos além dos de Hu para o reconhecimento de objetos invariantes à rotação, cujas imagens foram obtidas por meio de um sensor industrial 3D de baixa resolução. Outro destaque se refere ao tipo de distância utilizada: por padrão, utiliza-se a distância euclidiana, porém, segundo a documentação da biblioteca Scikit-Learn ([s. d.]), há outros parâmetros de distância que podem alterar o desempenho do classificador. Tendo isso em consideração, trabalhos futuros com foco nesses dois pontos poderiam ser conduzidos para análise do KNN. Outra sugestão seria comparar o desempenho do KNN com outros classificadores, tais como RNA, SVM e Naive Bayes, já que esses algoritmos também têm aplicações na área de visão computacional e sua execução é de maior complexidade.

COMPUTER VISION TECHNIQUES FOR PART CLASSIFICATION

Abstract

Computer vision is an area of artificial intelligence that has a great impact on various sectors of society. The ability of the machine to see objects of interest in an image and produce a classification or detection response is very important in the context of automation. The computer vision techniques allow, through image processing steps and the use of classifiers, to provide answers to several problems that arise. The objective of this work is to analyze the responsiveness of a specific classifier, the K-nearest neighbors (KNN), for a group classification problem and analyze its performance through hit rate and accuracy parameters. First, four groups of parts were created in a three-dimension (3D) design software and cut on a laser cutting machine, then pictures were taken of each of the parts individually. Later, using the OpenCV libraries, we could carry out the image processing, such as changing the *red-green-blue* (RGB) pattern to grayscale, binarization, noise correction, and obtaining invariant moments; and in the Scikit-Learn library, the classifier was trained and tested. It was possible to conclude that the classifier, in tests 1 and 2, was able to classify properly the parts groups having recall and precision above 65 and 80%, while, in test 3, it had a recall of 42% and a precision of 55%, showing that, with more analyzed classes, the classifier decreases its efficiency.

Keywords: Computer vision. KNN. Parts.

REFERÊNCIAS

BARELLI, F. *Introdução à visão computacional*: uma abordagem prática com Python e OpenCV. São Paulo: Casa do Código, 2018.

BRADSKI, G. The OpenCV Library. Dr. Dobb's Journal of Software Tools. *Journal of Software Tools*, 2000.

CHACON, G. *et al. Aplicação da técnica de momentos invariantes no reconhecimento de padrões em imagens digitais*. Rio de Janeiro: Centro Brasileiro de Pesquisas Físicas, 2011. Disponível em: https://docplayer.com.br/12210596-Aplicacao-da-tecnica-de-momentos-invariantes-no-reconhecimento-de-padroes-em-imagens-digitais.html. Acesso em: 7 mar. 2022.

CONCI, A.; AZEVEDO, E.; LETA, F. R. *Computação gráfica*: teoria e prática. 2. ed. Rio de Janeiro: Campus, 2008.

DIDÁTICA.TECH. *A linguagem Python*. 2022. Disponível em: https://didatica.tech/a-linguagem-python/. Acesso em: 7 mar. 2022.

FERREIRA, L. S.; JAIMES, B. R. A. Aplicação de visão computacional para automatização do processo de reconhecimento de placas de aço bruto. *In*: BRASIL. *Prêmio Mercosul de Ciência e Tecnologia*. Brasília, DF: MCTI, 2018. p. 193-227. Disponível em: https://www.mbc.org.br/wp-content/uploads/2020/07/LIVRO_MERCOSUL_MONTADO_LOWRES_SINGLE-1.pdf. Acesso em: 7 mar. 2022.

GONZALEZ, R. C.; WOODS, R. E. *Processamento digital de imagens*. 3. ed. São Paulo: Prentice Hall, 2010.

HU, M. K. Visual pattern recognition by moment invariants. *IRE, Transaction on Information Theory*, v. 8, n. 2, p. 179-187, 1962. DOI 10.1109/TIT.1962.1057692

KUMAR, A. Computer-vision-based fabric defect detection: a survey. *IEEE Transactions on Industrial Electronics*, v. 55, n. 1, p. 348-363, 2008. DOI 10.1109/TIE.1930.896476.

MACHADO, R. C. *Sistema de visão computacional para mapeamento de obstáculos em uma mesa de provas de robôs*. 2017. Trabalho de conclusão de curso (Bacharelado em Engenharia Mecatrônica) – Faculdade de Engenharia Mecânica, Universidade Federal de Uberlândia, Uberlândia, 2017. Disponível em: https://repositorio.ufu.br/handle/123456789/22130. Acesso em: 7 mar. 2022.

MACHINE LEARNING CRASH COURSE. *Classification:* Precision and Recall. 2020. Disponível em: https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall. Acesso em: 7 mar. 2022.

MIRANDA, B. S. *Algoritmos clássicos de aprendizado de máquina aplicados ao problema do reconhecimento de imagens*. 2011. Trabalho de conclusão de curso (Bacharelado em Ciência da Computação) – Universidade Federal do Pampa, Alegrete, 2011. Disponível em: http://dspace.unipampa.edu.br/bitstream/riu/1547/1/Algoritmos%20cl%C3%A1ssicos%20de%20aprendizado%20de%20maquina%20aplicados%20ao%20problema%20do%20reconhecimento%20de%20imagens.pdf. Acesso em: 7 mar. 2022.

MITCHELL, T. M. *Machine learning*. New York: McGraw-Hill, 1997.

PEDRINI, H.; SCHWARTZ, W. R. *Análise de imagens digitais*: princípios, algoritmos e aplicações. São Paulo: Cengage Learning, 2008.

RODRIGUES, J. V. *Classificação de peças mecânicas a partir de visão computacional e aprendizado de máquina utilizando imagens sintéticas*. 2020. Trabalho de conclusão de curso (Bacharelado em Engenharia de Controle e Automação) – Escola de Engenharia, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2020. Disponível em: http://hdl.handle.net/10183/217435. Acesso em: 7 mar. 2022.

RODRIGUES, P. H. B. *Imagens e códigos*. 2021. Arquivo Word. Disponível em: https://drive.google.com/file/d/1VLYhQ-D06vLFomzQgSTL9Bkpi2FaC-qW/view?usp=sharing. Acesso em: 7 mar. 2022.

SCIKIT-LEARN. *Machine Learning in Python*. [s. d.]. Disponível em: https://scikit-learn.org/0.24/modules/generated/sklearn.neighbors.DistanceMetric.html. Acesso em: 7 mar. 2022.

SILVA, R. D. C. D. *Um estudo sobre a extração de características e a classificação de imagens invariantes à rotação extraídas de um sensor industrial 3D***.** Dissertação (Mestrado em Engenharia de Teleinformática) – Centro de Tecnologia, Universidade Federal do Ceará, Fortaleza, 2014. Disponível em: http://www.repositorio.ufc.br/bitstream/riufc/10852/1/2014_dis_rdcsilva.pdf. Acesso em: 7 mar. 2022.

SZELISKI, R. *Computer vision*: algorithms and applications. 2. ed. London: Springer-Verlag London Limited: Springer Science & Business Media, 2010.

VIEIRA NETO, H.; MARQUES FILHO, O. *Processamento digital de imagens*. Rio de Janeiro: Brasporte, 1999.

Contato

Alexandre Lasthaus
lasthaus@mackenzie.com.br

Tramitação

Recebido em julho de 2021.
Aprovado em novembro de 2021.