



Luiz Fernando Machado Silva

Reconhecimento de moedas de Real em imagens digitais para deficientes visuais em tempo real

São José dos Campos, SP

Luiz Fernando Machado Silva

Reconhecimento de moedas de Real em imagens digitais para deficientes visuais em tempo real

Trabalho de conclusão de curso apresentado ao Instituto de Ciência e Tecnologia – UNIFESP, como parte das atividades para obtenção do título de Bacharel em Ciência da Computação.

Universidade Federal de São Paulo – UNIFESP

Instituto de Ciência de Tecnologia

Bacharelado em Ciência da Computação

Orientador: Prof. Dr. Fábio Augusto Menocci Cappabianco

São José dos Campos, SP

Junho de 2015

Luiz Fernando Machado Silva

Reconhecimento de moedas de Real em imagens digitais para deficientes visuais em tempo real

Trabalho de conclusão de curso apresentado ao Instituto de Ciência e Tecnologia – UNIFESP, como parte das atividades para obtenção do título de Bacharel em Ciência da Computação.

Trabalho aprovado em 17 de junho de 2015:

**Prof. Dr. Fábio Augusto Menocci
Cappabianco**
Orientador

Prof. Dr. Jurandy Gomes de Almeida Junior
Convidado

Prof. Dr. Fábio Augusto Faria
Convidado

São José dos Campos, SP
Junho de 2015

*Este trabalho é dedicado ao pai da ciência da computação, Alan Turing.
E também à todos os outros gênios silenciados pelas limitações da ignorância humana.*

Agradecimentos

Aos meus pais e meu irmão, pelo amor incondicional.

Aos meus familiares, por sempre me lembrarem de quem eu sou.

Aos meus amigos, por me lembrarem que não estou sozinho.

Aos meus professores, pela inspiração, dentro e fora da sala de aula.

Aos meus colegas de sala, pela expiração, após duas greves e dúzias de provas.

Ao meu orientador, por ter sido peça fundamental no desenvolvimento desse trabalho.

Obrigado. Sem vocês, nada disso seria possível (e também não teria a menor graça!)

“We can only see a short distance ahead, but we can see plenty there that needs to be done.”

(Alan Turing)

Resumo

Uma rápida análise nas áreas de Reconhecimento de Padrões e Visão Computacional mostra que não existem algoritmos genéricos para resolver qualquer tipo de problema. De acordo com a tarefa a ser desempenhada ou as características de interesse das imagens analisadas, algoritmos extremamente específicos devem ser desenvolvidos. Um levantamento bibliográfico de trabalhos realizados no Brasil mostrou a escassez de estudos voltados para o reconhecimento de moedas nacionais. Esse trabalho de conclusão de curso apresenta um algoritmo para reconhecimento e classificação de moedas de Real através do processamento de imagens digitais. Um algoritmo eficiente pode ser aplicado em programas voltados para acessibilidade e inclusão de deficientes visuais. O algoritmo desenvolvido faz uso da transformada de Hough para reconhecimento das regiões circulares de interesse. Em seguida, o método *Scale-invariant feature transform* (SIFT) retorna as características das regiões, posteriormente normalizadas através do método *Bag-Of-Words* (BOW). Por fim, o método *Support vector machine* (SVM) é usado para treinamento e predição das classes. Em testes realizados, o algoritmo obteve acurácia superior à 75%.

Palavras-chaves: visão computacional, processamento de imagens, reconhecimento de padrões, reconhecimento de moedas, SIFT, SVM, BOW

Abstract

A quick analysis on the Pattern Recognition and Computer Vision fields shows that generic algorithms able to solve any kind of problem are nonexistent. According to the desired task or features of the analyzed images, extremely specific algorithms need to be developed. A survey of Brazilian studies showed the lack of studies on the recognition of national coins. This work aims at proposing an efficient algorithm to recognize and classify Real coins by processing digital images using Hough Transform, Scale-invariant feature transform and Bag-Of-Words (BOW). Lastly, the Support vector machines (SVM) method is used for class training and prediction. An efficient algorithm to solve this issue can be applied to a variety of accessibility software, including those designed for vision-impaired people. During tests, the algorithm displayed an accuracy level above 75%.

Key-words: computer vision, image processing, pattern recognition, coin recognition, SIFT, SVM, BOW

Listas de ilustrações

Figura 1 – Fluxograma que ilustra algoritmo proposto, destacando a entrada de dados, os processos e a saída.	23
Figura 2 – Imagem monocromática de Lena. (GONZALEZ; WOODS, 2008)	25
Figura 3 – Comparação entre imagens resultantes após o acréscimo de ruído descrito por seis diferentes PDFs. (GONZALEZ; WOODS, 2008)	27
Figura 4 – Aplicação da transformada <i>top-hat</i> em uma imagem. (MATHWORKS, 2014)	28
Figura 5 – Comparação entre os resultados da aplicação do método de limiarização de Otsu em uma imagem não-particionada e uma imagem particionada. (GONZALEZ; WOODS, 2008)	29
Figura 6 – Comparação entre os resultados da aplicação de um algoritmo de segmentação de regiões por descontinuidade em uma imagem sólida e uma imagem com textura. (GONZALEZ; WOODS, 2008)	30
Figura 7 – Comparação entre os resultados da aplicação de algoritmos de segmentação de regiões.(GONZALEZ; WOODS, 2008)	31
Figura 8 – Aplicação da Transformada de Hough para círculos em bolas de croqué. (WANG; MARTIN, 2010)	33
Figura 9 – Anversos da segunda família de moedas do Real. (Banco Central do Brasil, 2003)	37
Figura 10 – Reversos da segunda família de moedas do Real. (Banco Central do Brasil, 2003)	37
Figura 11 – Imagem do anverso de uma moeda de 1 real no ambiente padronizado.	40
Figura 12 – Organização de diretórios usada para armazenar as imagens das moedas.	41
Figura 13 – Composição de imagens do anverso de duas moedas de 10 centavos ilustrando diferença nos materiais.	42
Figura 14 – Composição de imagens do anverso de duas moedas de 10 centavos convertidas para escala de cinza.	42
Figura 15 – Anverso de moeda de 1 real em escala de cinza sem filtro.	43
Figura 16 – Anverso de moeda de 1 real em escala de cinza suavizada com filtro de mediana 5.	43
Figura 17 – Anverso de moeda de 1 real, convertida para escala de cinza, suavizada com um filtro de mediana de tamanho 21 e com o círculo de Hough detectado pelo algoritmo em destaque.	44
Figura 18 – Anverso de moeda de 1 real, com <i>keypoints</i> detectados pelo algoritmo SIFT em destaque.	46

Figura 19 – Anverso de moeda de 1 real, com <i>keypoints</i> detectados pelo algoritmo SURF em destaque.	46
Figura 20 – Método <i>Bag-Of-Words</i> ilustrado: Conjunto de imagens avaliadas, histogramas de características extraídas e dicionário formado pelo conjunto de características. (Gil Levi, 2013)	47

Lista de tabelas

Tabela 1 – Média de <i>keypoints</i> por valor de moeda.	45
Tabela 2 – Identificação das classes utilizadas para cada moeda.	49
Tabela 3 – Tabelas comparativas de resultados com diferentes tamanhos de dicionário.	52
Tabela 4 – Tabela de assertividade do algoritmo com um dicionário de 1000 palavras.	55

Lista de abreviaturas e siglas

TCC	Trabalho de Conclusão de Curso
PI	Processamento de Imagens
VC	Visão Computacional
RP	Reconhecimento de Padrões
IA	Inteligência Artificial
PDF	<i>Probability density function</i>
OPF	<i>Optimum Path Forest</i>
GUI	<i>Graphical User Interface</i>
SIFT	<i>Scale-Invariant Feature Transform</i>
SURF	<i>Speeded Up Robust Features</i>
OpenCV	<i>Open Source Computer Vision</i>
BOW	<i>Bag-Of-Words</i>
SVM	<i>Support Vector Machine</i>

Sumário

1	Introdução	21
1.1	Contextualização e Motivação	21
1.2	Objetivo	21
1.2.1	Objetivo Geral	21
1.2.2	Objetivos Específicos	22
1.3	Metodologia	22
1.4	Organização do Texto	23
2	Revisão Bibliográfica	25
2.1	Processamento Digital de Imagens	25
2.1.1	Filtragem	26
2.1.1.1	Filtros de ruído	27
2.1.1.2	Filtros morfológicos	28
2.1.2	Segmentação de regiões	29
2.1.2.1	Detecção de bordas de Canny	30
2.1.2.2	Transformada de Hough para círculos	32
2.1.3	Extração de características	33
2.1.3.1	Descritores de fronteira	34
2.1.3.2	Descritores de regiões	34
2.1.3.3	Descritores de texturas	34
2.1.4	Reconhecimento de objetos	35
2.2	Bibliotecas de Processamento Digital de Imagens	36
2.2.1	OpenCV	36
2.2.2	BIAL	36
2.3	Moedas do Real	36
2.4	Considerações Finais	37
3	Desenvolvimento	39
3.1	Organização de arquivos	40
3.2	Sistema de cores	42
3.3	Filtragem de ruído	43
3.4	Detecção de áreas de interesse	44
3.5	Extração de descritores	45
3.6	Dicionário de características	47
3.7	Treinamento da SVM	48
3.8	Classificação	49

4 Resultados	51
4.1 Dicionário de palavras	51
4.2 Assertividade	55
5 Conclusão	57
Referências	59
Apêndices	61
APÊNDICE A Código Fonte	63

1 Introdução

Esse capítulo tem como objetivo apresentar e contextualizar o trabalho desenvolvido nas seções subsequentes.

1.1 Contextualização e Motivação

A área de Reconhecimento de Padrões tem como objetivo detectar, reconhecer e classificar objetos em categorias ou classes. ([THEODORIDIS; KOUTROUMBAS, 2006](#))

Nessa área, não existem algoritmos genéricos para resolver qualquer tipo de problema. De acordo com a tarefa a ser desempenhada ou as características das imagens analisadas, algoritmos extremamente específicos devem ser desenvolvidos. ([WANGENHEIM, 2007](#))

Este trabalho propõe a elaboração de um algoritmo para realizar o reconhecimento de moedas de Real por meio do processamento de imagens fotográficas.

O reconhecimento de moedas nacionais foi abordado em um trabalho de [Jory \(2011\)](#). Em suas simulações, o algoritmo proposto obteve sucesso em aproximadamente 50% das simulações . Entretanto, seu trabalho se mostrou ineficiente na detecção de moedas falsas, pois o algoritmo desenvolvido se limitava a analisar apenas o diâmetro e a cor das mesmas. Além disso, o algoritmo foi construído usando funções interpretativas do software MATLAB, impactando o desempenho e aumentando o tempo total de execução.

Um algoritmo capaz de identificar e classificar as moedas nacionais sem interferência humana e com alta confiabilidade pode ser utilizado em aplicações com foco em acessibilidade, permitindo a inclusão de deficientes visuais. Além disso, o algoritmo pode ser bastante útil em situações onde seja necessário contabilizar um grande volume de moedas ou, ainda, distinguir moedas verdadeiras de falsas.

1.2 Objetivo

Essa seção irá descrever o objetivo geral e os objetivos específicos desse trabalho.

1.2.1 Objetivo Geral

Desenvolver um algoritmo eficiente capaz de reconhecer e classificar, de maneira sistemática e sem intervenção humana, as moedas da segunda família de Real por meio do processamento digital de imagens fotográficas.

1.2.2 Objetivos Específicos

- Definir um método padronizado para aquisição de imagens;
- Melhorar a qualidade das imagens capturadas por meio de filtros;
- Detectar objeto circular na imagem fotográfica;
- Extrair características do objeto circular detectado;
- Classificar objeto circular detectado com um valor de moeda.

1.3 Metodologia

Inicialmente, foi realizado um estudo bibliográfico sobre técnicas de Processamento de Imagens, Reconhecimento de Padrões e Visão Computacional. Além disso, um levantamento de artigos científicos sobre de reconhecimento de moedas foi efetuado para auxiliar a definir algumas estratégias para reconhecer moedas de Real, objeto de interesse deste trabalho.

Para auxiliar no processo de desenvolvimento e depuração do algoritmo, foi necessária a geração de um conjunto de imagens fotográficas de moedas de Real - em diferentes níveis de qualidade, iluminação e resolução.

Utilizando como base funções definidas em bibliotecas de processamento de imagens, foi desenvolvido um algoritmo capaz de filtrar e melhorar a qualidade de uma imagem de entrada, segmentar as regiões de interesse, extraer características dessas regiões e, por fim, identificar e classificar as regiões segmentadas de acordo com seu valor monetário.

O ponto de partida do algoritmo é a leitura da imagem que será trabalhada. Algumas bibliotecas de Processamento de Imagens, como a OpenCV, trazem métodos nativos para leitura de imagens digitais.

Após a entrada de dados, o algoritmo melhora a qualidade da imagem lida, reduzindo o ruído com filtros medianos.

Em seguida, as regiões de interesse são segmentadas. Como o objetivo do algoritmo é identificar objetos circulares, o algoritmo de *Hough* é usado para definir, com precisão, as regiões circulares da imagem.

Analizando as regiões segmentadas, são extraídas características da região para auxiliar no processo de classificação das moedas.

Com base nas características extraídas, métodos classificadores são executados com o objetivo de identificar qual conjunto de características melhor identifica as regiões segmentadas e, com isso, detectar o valor das moedas.

O fluxo de funcionamento deste processo está ilustrado na Figura 1.

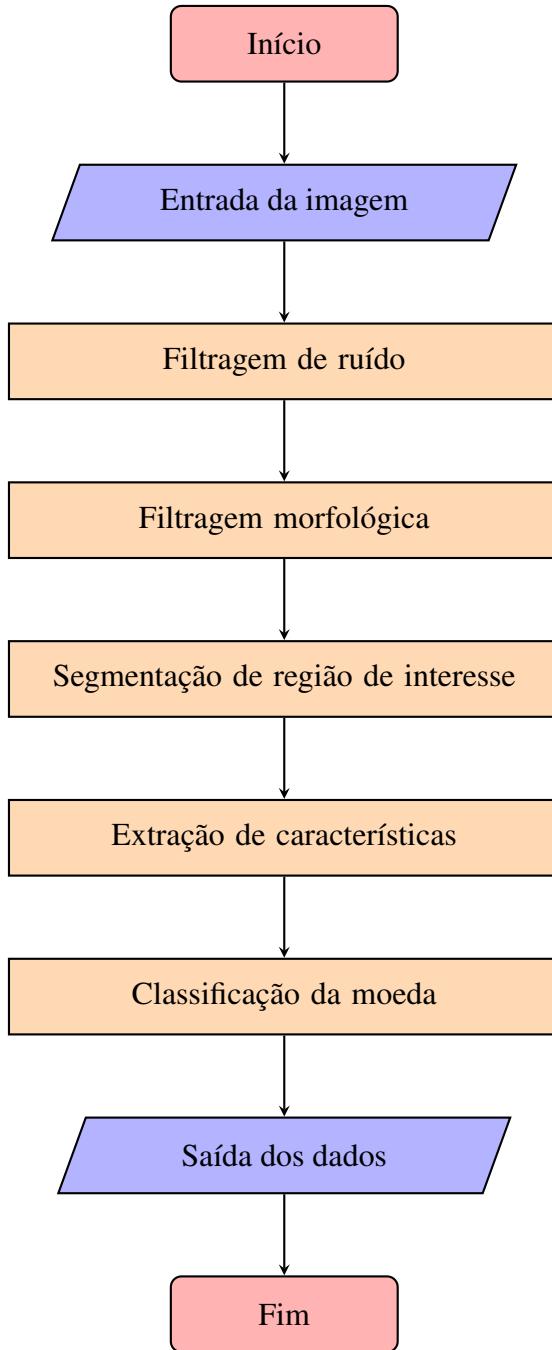


Figura 1 – Fluxograma que ilustra algoritmo proposto, destacando a entrada de dados, os processos e a saída.

Após o processo de desenvolvimento, o algoritmo foi colocado a prova e dados de assertividade foram gerados a partir da realização de uma série de testes com imagens em diferentes níveis de qualidade.

1.4 Organização do Texto

Neste capítulo introdutório, o objetivo do trabalho foi apresentado e contextualizado, acompanhado das motivações e justificativas de seu desenvolvimento. Além disso, também foi

apresentada uma metodologia descrevendo como esse objetivo foi atingido.

No capítulo Revisão Bibliográfica, é apresentada a fundamentação teórica necessária para o compreendimento dos assuntos abordados neste trabalho, incluindo tópicos de Processamento Digital de Imagens, Reconhecimento de Padrões, Visão Computacional e também alguns dados relevantes sobre as moedas nacionais, objeto de interesse do trabalho.

Em seguida, em Desenvolvimento, é descrita a metodologia, dificuldades e soluções encontradas durante o desenvolvimento da solução para o problema apresentado.

No capítulo de Resultados, são apresentados os resultados obtidos nos testes do algoritmo proposto.

Por fim, em Conclusão, o projeto e seus resultados finais são sumarizados.

2 Revisão Bibliográfica

Este capítulo tem como objetivo apresentar conceitos e métodos da áreas de Processamento Digital de Imagens, Reconhecimento de Padrões e Visão Computacional, fundamentais para o desenvolvimento deste trabalho. Além disso, são apresentadas algumas bibliotecas que implementam os métodos descritos. Por fim, é apresentada uma breve descrição da história e também das características físicas e morfológicas das moedas de Real.

Salvo nas subseções onde explicitamente citado, o texto contido nesta seção usa como base o conteúdo do livro “Processamento Digital de Imagens” de [Gonzalez e Woods \(2008\)](#).

2.1 Processamento Digital de Imagens

Uma imagem digital pode ser definida como uma função $f(x, y)$ finita e discreta, tal que o par x e y denota as coordenadas de um ponto (também chamado de *pixel*) em um plano e a função f retorna a intensidade (ou nível de cinza) desse ponto.

Por exemplo, a Figura 2 traz a imagem digital monocromática de Lena¹. Nesta imagem, cada *pixel* tem seu valor representado por 8 bits, sendo que o valor 00000000 (0) denota a cor preta, o valor 11111111 (255) denota a cor branca e os valores intermediários, tons de cinza.



Figura 2 – Imagem monocromática de Lena. ([GONZALEZ; WOODS, 2008](#))

¹ Lena Söderberg, fotografada por Dwight Hooker para a edição de novembro de 1972 da revista *Playboy*. Esse recorte de sua aparição na revista foi usado pela primeira vez em um artigo científico em 1973 e, desde então, passou a ser considerado padrão para testes na área de Processamento de Imagens. ([POL, 1997](#))

Computacionalmente, imagens digitais podem ser representadas por meio de estruturas de matrizes bidimensionais de M linhas e N colunas, com a intensidade de cada pixel sendo definida pela função:

$$f(x, y) = \begin{vmatrix} f(0, 0) & f(0, 1) & \dots & f(0, N - 1) \\ f(1, 0) & f(1, 1) & \dots & f(1, N - 1) \\ \dots & \dots & & \dots \\ f(M - 1, 0) & f(M - 1, 1) & \dots & f(M - 1, N - 1) \end{vmatrix}$$

A área de Processamento Digital de Imagens compreende o processamento de imagens por meio processos que, de maneira geral, apresentam imagens digitais como entrada e saída. Paralelamente, existe o campo da Visão Computacional, que tem como objetivo emular aspectos da visão humana por meio do uso de computadores. Situada entre esses dois campos, está a área de Análise de Imagens, que visa compreender e mapear significados a objetos extraídos de imagens processadas.

Os algoritmos utilizados no Processamento Digital de Imagens podem ser agrupados em três níveis, com base em suas aplicações:

- *Nível baixo*: Envolve operações primitivas e relacionadas ao pré-processamento de imagens como filtros de redução de ruído ou ajuste de contraste.
- *Nível médio*: Relaciona os métodos usados para segmentar regiões e identificar atributos e características de regiões de interesse em uma imagem.
- *Nível alto*: Engloba os algoritmos usados na interpretação de objetos reconhecidos.

2.1.1 Filtragem

As técnicas de filtragem de imagens digitais são utilizadas para melhorar a qualidade das imagens, ajustando o contraste entre os elementos, removendo ruídos ou acentuando características de maior interesse.

Os algoritmos de filtragem são classificados em duas categorias:

- *Filtragem espacial*: Filtros que atuam no domínio espacial, ou seja, no plano da imagem, realizando manipulação direta dos pixels que a compõe.
- *Filtragem no domínio da frequência*: Filtros que atuam no domínio da transformada de Fourier de uma imagem digital e que, portanto, fazem o uso de operações adicionais de transformação para o domínio da transformada e transformação inversa para retornar ao domínio espacial.

De maneira geral, filtragens espaciais são menos complexas e requerem menos recursos computacionais para serem realizadas. No entanto, algumas operações que trabalham com amostragem de imagens, como *aliasing*, apresentam melhor desempenho no domínio de frequência.

Nesse trabalho, serão utilizados filtros de ruído e filtros morfológicos.

2.1.1.1 Filtros de ruído

Durante o processo de aquisição e transmissão de imagens digitais, é comum perceber a presença de ruído, ou seja, variações aleatórias indesejadas nas informações de brilho ou cor nas mesmas. Esse fenômeno ocorre devido à uma série de fatores que variam desde a qualidade dos equipamentos envolvidos na aquisição das imagens até iluminação e temperatura do ambiente onde as imagens são adquiridas.

Por se tratarem de fenômenos aleatórios, modelos de ruídos são descritos utilizando funções de densidade de probabilidade (ou PDF, do inglês *probability density function*), como exemplificado na Figura 3.

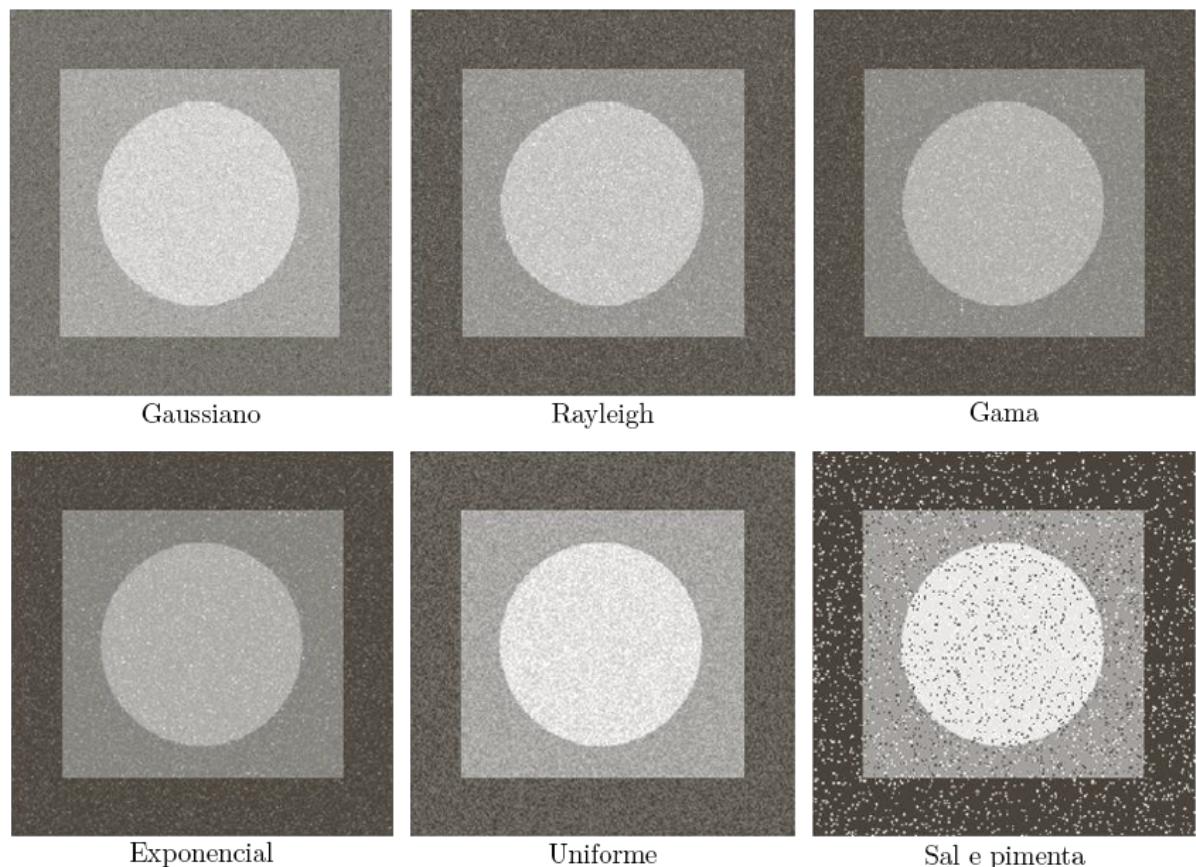


Figura 3 – Comparaçāo entre imagens resultantes apōs o acrēscimo de ruído descrito por seis diferentes PDFs. ([GONZALEZ; WOODS, 2008](#))

Os filtros de ruído podem atuar tanto no domínio espacial quanto nos de frequência. Normalmente, quando a imagem apresenta apenas ruído aleatório, filtros espaciais, como os de média e mediana, são preferidos. Entretanto, quando há necessidade de analisar e determinar a forma do ruído periódico, são aplicados filtros no domínio de frequência que atuam rejeitando ou permitindo determinadas faixas de frequência.

2.1.1.2 Filtros morfológicos

No processo de captura de imagens, fatores como o posicionamento das fontes de iluminação e da câmera podem gerar alterações morfológicas nas imagens digitais resultantes, como inhomogeneidade na intensidade da coloração dos objetos que compõe a cena ou variações no sombreamento dos mesmos.

Essas deformações observadas nas imagens capturadas podem gerar resultados indesejados nos procedimentos de segmentação e limiarização de imagens, assuntos que são abordados em detalhe na próxima seção.

Com o objetivo de reduzir tais deformidades e melhorar a qualidade das imagens, são utilizados filtros morfológicos.

As transformadas *top-hat* e *bottom-hat* são aplicadas para corrigir variações de sombreamento em uma região, decorrentes de iluminação não-uniforme. A transformada top-hat é aplicada quando existe a intenção de acentuar objetos claros em um fundo escuro, como no exemplo mostrado na Figura 4. Já a transformada *bottom-hat* é utilizada para destacar objetos mais escuros em um fundo claro.

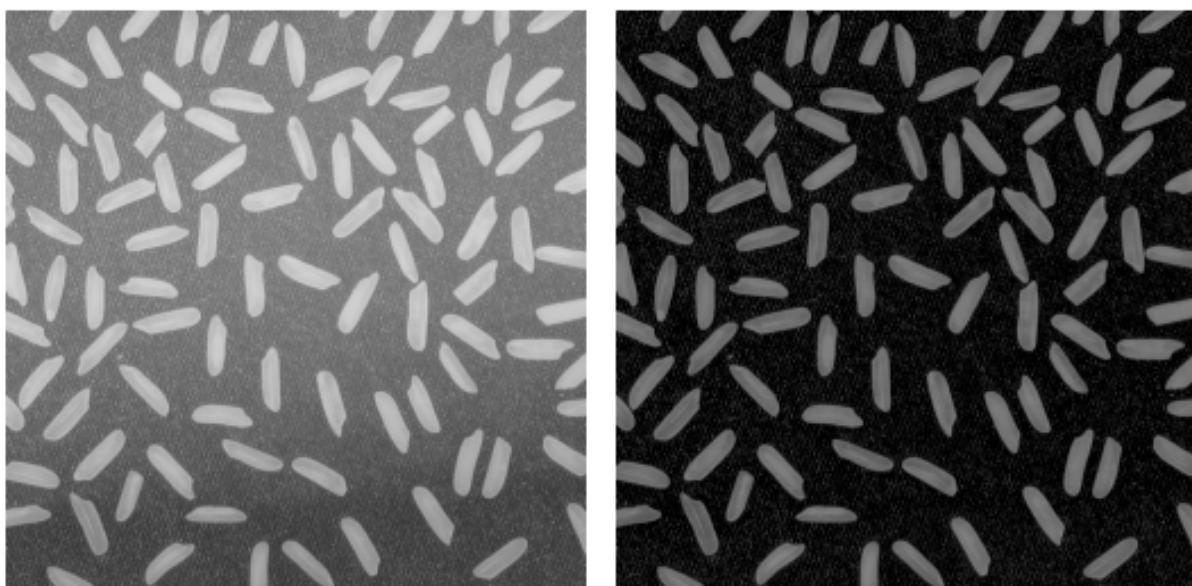


Figura 4 – Aplicação da transformada *top-hat* em uma imagem. ([MATHWORKS, 2014](#))

Outro fenômeno decorrente de iluminação inadequada é a inomogeneidade, observado como uma variação gradiente na intensidade das cores de um objeto ou na forma de reflexos de luz em sua superfície. Uma metodologia relativamente simples utilizada para processar imagens onde esse fenômeno ocorre é o particionamento da imagem principal em sub-imagens onde seja possível observar uma iluminação mais uniforme, como é observado na Figura 5.

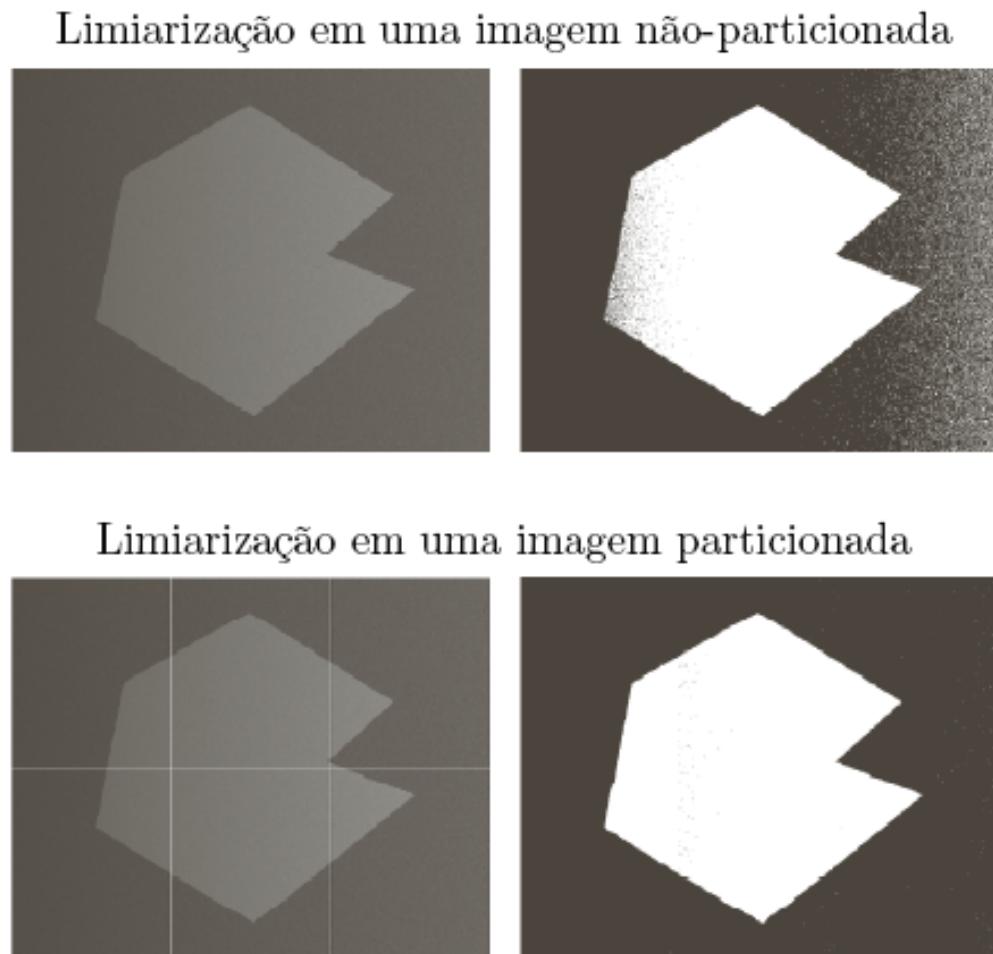


Figura 5 – Comparação entre os resultados da aplicação do método de limiarização de Otsu em uma imagem não-particionada e uma imagem partitionada. ([GONZALEZ; WOODS, 2008](#))

2.1.2 Segmentação de regiões

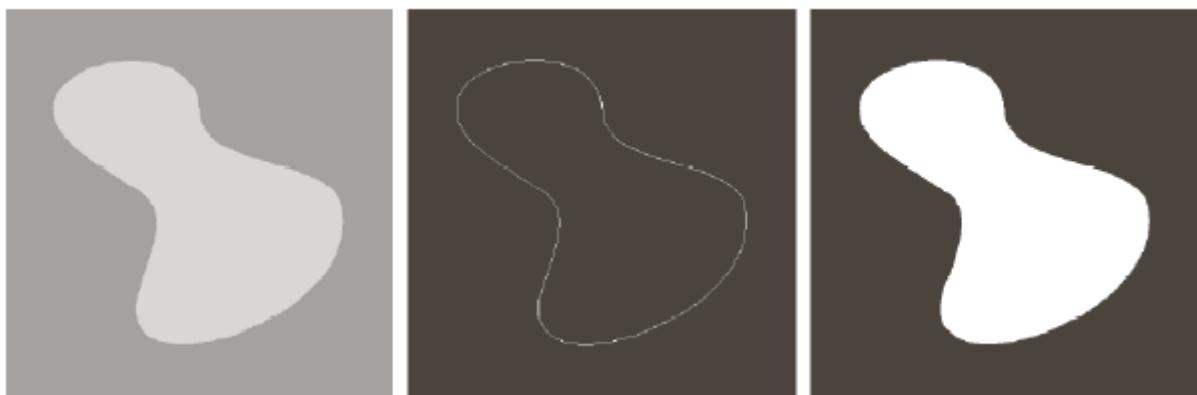
Na seção anterior, foram descritos métodos de pré-processamento de imagens digitais onde tanto a entrada quanto a saída são imagens. Os processos de segmentação de imagens recebem imagens como entrada e retornam atributos dessas imagens como saída.

O conteúdo e nível de detalhamento desses atributos varia de acordo com o objetivo desejado. Normalmente, os atributos extraídos nessa etapa são usados como entrada em algoritmos cognitivos, que tentam dar sentido às imagens.

De maneira geral, os algoritmos de segmentação de imagens são divididos em duas categorias, de acordo com a abordagem:

- *Abordagem de descontinuidade*: buscam mudanças bruscas nos valores de intensidade dos pixels da imagem, como algoritmos de detecção de bordas. Exemplo na Figura 6.
- *Abordagem de similaridade*: buscam regiões semelhantes em uma imagem, de acordo com critérios pré-definidos, como algoritmos de limiarização e de crescimento de região.

Segmentação de uma região com intensidade constante



Segmentação de região com textura



Figura 6 – Comparação entre os resultados da aplicação de um algoritmo de segmentação de regiões por descontinuidade em uma imagem sólida e uma imagem com textura. ([GONZALEZ; WOODS, 2008](#))

Nas sub-seções que seguem serão apresentados dois algoritmos de segmentação de imagens usados nesse trabalho.

2.1.2.1 Detecção de bordas de Canny

Entre os algoritmos de detecção de bordas, o algoritmo de Canny se destaca por ser um método que, apesar de sua relativa complexidade, é considerado ideal para detecção de bordas.

Em seu trabalho, Canny propôs uma abordagem ótima de detecção de bordas, que tinha como objetivo principal atender a três critérios:

- *Baixa taxa de erro.* As bordas da figura devem ser encontradas em sua totalidade e o algoritmo não deve retornar nenhum ponto para não-bordas.
- *Pontos de borda bem localizados.* A distância entre os pontos de borda detectados e os pontos de borda reais devem ser mínimas.
- *Resposta única para cada ponto de borda.* Cada ponto de borda encontrado deve corresponder a um, e a somente um, ponto de borda verdadeiro.

Quando comparado à outros métodos de detecção de borda, o algoritmo de Canny apresenta maior complexidade e, portanto, sua aplicação pode ser limitada quando o tempo de processamento é um requisito. Por outro lado, a qualidade da borda obtida é, geralmente, superior à obtida por outros algoritmos de detecção, como ilustrado na Figura 7.

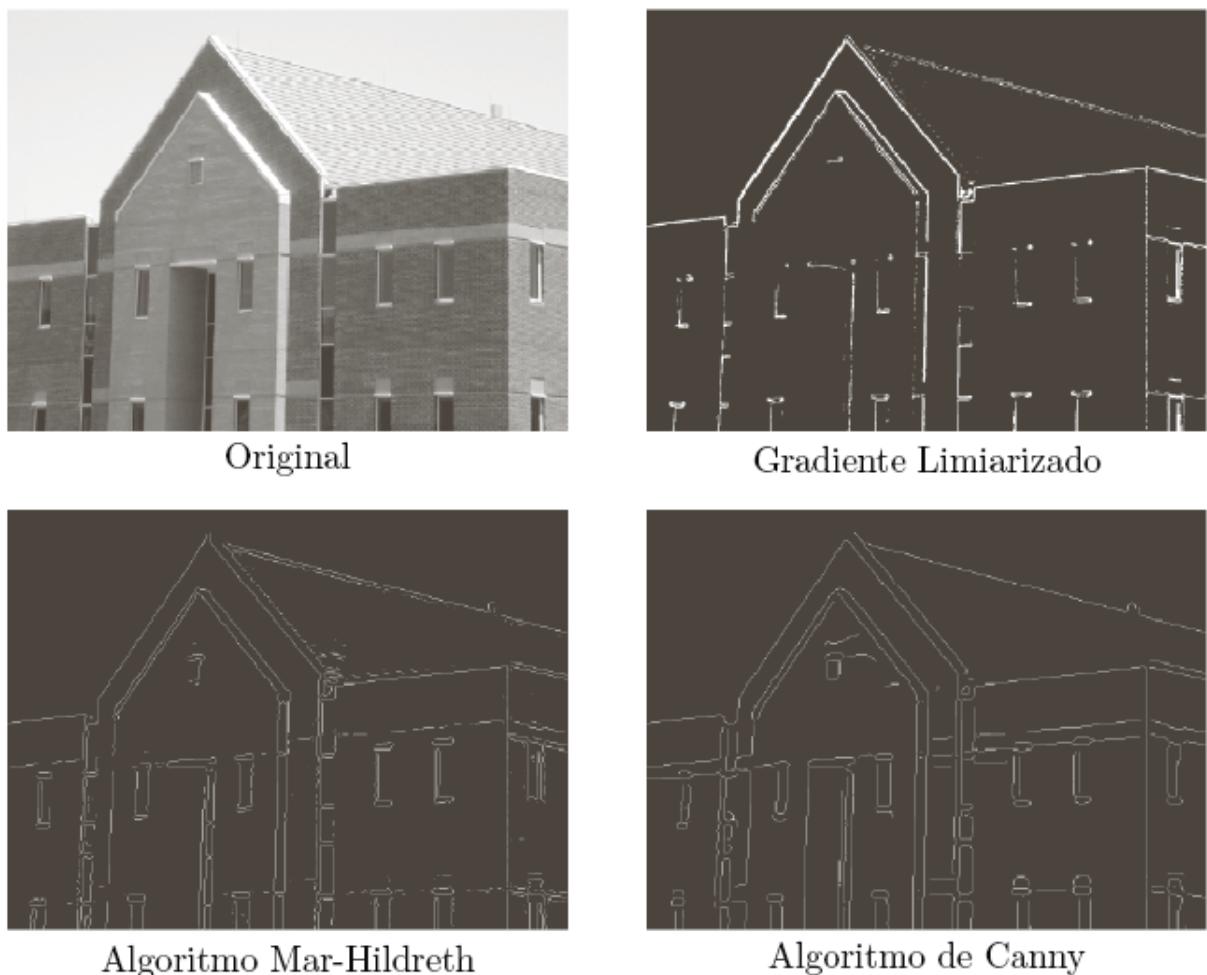


Figura 7 – Comparaçāo entre os resultados da aplicação de algoritmos de segmentaçāo de regiões.(GONZALEZ; WOODS, 2008)

O algoritmo proposto por Canny é composto por quatro passos:

- Suavização da imagem de entrada por meio de um filtro gaussiano de ruído, reduzindo o efeito indesejado onde pontos ruídosos são detectados como borda.
- Cálculo da magnitude dos gradientes de intensidade da imagem em quatro sentidos - vertical, horizontal, à 45º e à 135º - para determinar o sentido da borda.
- Aplicação da supressão não máxima nos mapas de magnitude obtidos na etapa anterior para definir qual sentido retornou a maior magnitude em cada pixel.
- Detecção de bordas por meio de dupla limiarização tomando como parâmetro os gradientes de intensidade mais elevada, ou seja, com mais chance de serem bordas e seu sentido, para tentar conectar as possíveis bordas detectadas.

A saída do algoritmo é uma imagem binária com as mesmas dimensões da imagem de entrada. Nesta imagem, cada pixel é identificado como pixel de borda ou pixel de não-borda. ([CANNY, 1986](#))

2.1.2.2 Transformada de Hough para círculos

De maneira geral, é esperado que algoritmos de detecção de bordas retornem exclusivamente pixels que compõe a borda de um objeto. Entretanto, ao tratar imagens que apresentam ruído e iluminação não uniforme, é comum a presença de descontinuidades nos pixels que delimitam essas fronteiras.

Na área de processamento de imagens, existem três abordagens principais para tratar esse tipo de fenômeno.

- *Processamento local*: Envolve a análise da vizinhança de cada pixel declarado como ponto de borda para determinar os próximos pontos.
- *Processamento regional*: Trabalham conectando pontos pré-determinados das fronteiras aproximadas de uma região. Para isso, requer o conhecimento prévio da localização das fronteiras das regiões de interesse.
- *Transformação global (Transformada de Hough)*: Detecta formas que podem ser descritas como equações parametrizadas como retas, círculos e elipses. Essa abordagem requer como entrada um conjunto de possíveis pontos de borda.

Como o objeto de interesse desse trabalho são moedas circulares, será apresentada uma abordagem da Transformada de Hough para círculos, sugerida por [Davies \(1986\)](#). Em seu trabalho, Davies organizou o problema em duas fases: primeiro é encontrado o centro (x,y) do círculo e, em seguida, é definido o raio r .

O centro de um círculo pode ser determinado pelo ponto de encontro entre as retas normais à tangente de cada ponto da borda. Devido a possíveis inconsistências nos pontos de borda recebidos como entrada, não é possível garantir que todas as normais se cruzem em um único ponto e, portanto, toma-se o ponto de encontro máximo das retas como centro do círculo.

O raio de um círculo é dado pela distância entre seu centro e suas extremidades. Novamente, devido à inconsistências nos pontos de borda recebidos, são calculados os valores de distância entre cada o centro e cada ponto de borda e, em seguida, é computado o raio através de aproximação matemática.

A Figura 8 mostra um exemplo da Transformada de Hough para círculos em uma imagem monocromática contendo bolas plásticas de croqué com diferentes intensidades de cinza.

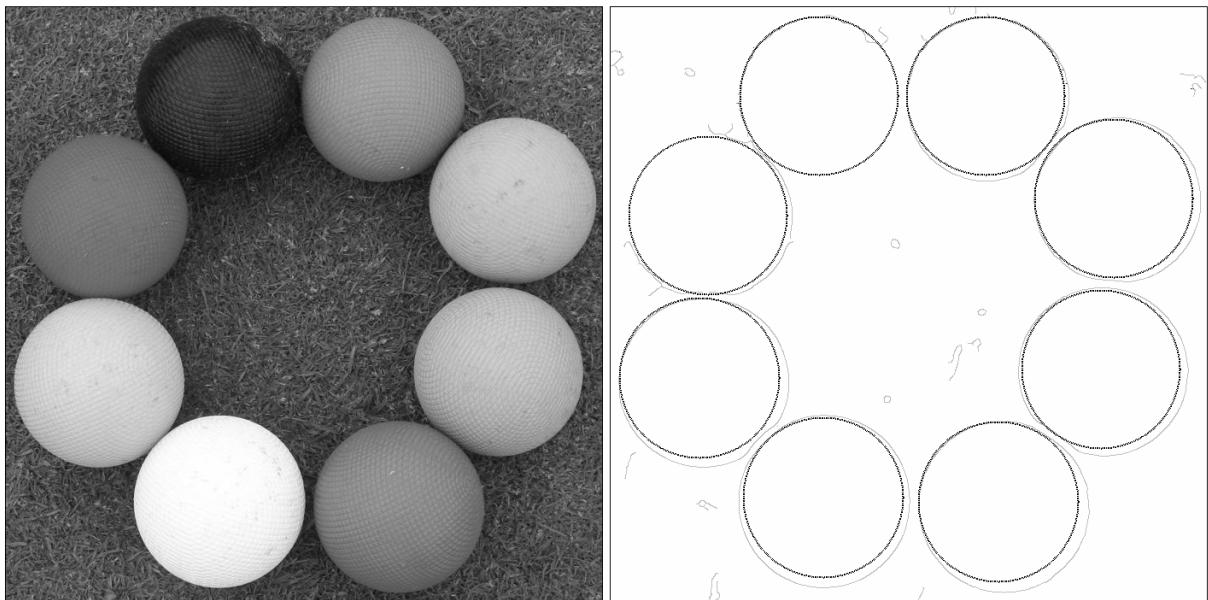


Figura 8 – Aplicação da Transformada de Hough para círculos em bolas de croqué. ([WANG; MARTIN, 2010](#))

2.1.3 Extração de características

Apesar de serem capazes de definir e delimitar regiões de interesse em imagens digitais, os métodos apresentados na seção anterior não trazem mais nenhuma informação útil sobre essas regiões.

Analizando as regiões segmentadas, é possível extrair uma série de características úteis para descrever as mesmas. Por exemplo, uma análise dos pixels que compõe a fronteira de uma região pode trazer informações relevantes sobre sua forma. Já uma varredura dos pixels internos de uma região irá retornar propriedades relacionadas a textura e a cores.

Nas sub-seções que seguem, serão apresentados descritores de características úteis para representar algumas propriedades relevantes de moedas.

2.1.3.1 Descritores de fronteira

Os pixels que formam a fronteira de uma região fornecem informações sobre suas características morfológicas.

Uma simples contagem do número de pixels que compõe a fronteira retorna uma aproximação do seu comprimento. O diâmetro de uma fronteira pode ser determinado pela distância máxima entre dois pontos extremos da fronteira. Uma análise na taxa de mudança de inclinação define a curvatura de um segmento da fronteira e permite que tais segmentos sejam classificados como concavo ou convexo.

2.1.3.2 Descritores de regiões

Uma análise dos pixels internos de uma região pode revelar informações relevantes para descrever como essa região é composta.

Uma contagem dos pixels de um segmento define sua área aproximada. Já o perímetro pode ser calculado analisando o comprimento de sua fronteira. Usando os valores contidos nesses descritores, é possível descrever o coeficiente de compacidade de um segmento através da expressão $\frac{\text{perímetro}^2}{\text{área}}$.

Para descrever objetos circulares, como moedas, o descritor de circularidade pode ser útil. Esse descritor, dado pela expressão $\frac{4\pi \text{área}}{\text{perímetro}^2}$, possui o valor 1 para objetos perfeitamente circulares e $\frac{\pi}{4}$, para objetos quadrados.

Descritores topológicos são usados para descrever sub-regiões dentro de regiões de interesse como, por exemplo, o número de furos em uma região ou a quantidade de elementos distintos conexos na mesma.

Ao trabalhar com imagens coloridas, também é possível extrair descritores regionais relacionados a cor, como cor predominante e cor média de uma região.

2.1.3.3 Descritores de texturas

De maneira geral, é possível qualificar texturas em imagens digitais em relação a suavidade, rugosidade ou regularidade das mesmas. Computacionalmente, existem três abordagens principais para analisar e descrever a textura de uma região.

As abordagens estatísticas analisam padrões estatísticos para definir uma textura como suave, rugosa ou granulada. Já as abordagens estruturais são usadas para identificar texturas compostas por padrões de formas primitivas, como retas ou círculos. Por fim, abordagens espectrais são aplicadas para identificar padrões periódicos globais de textura, facilmente distinguíveis analisando concentrações de energia no espectro de Fourier de uma região.

2.1.4 Reconhecimento de objetos

Na seção anterior, foram apresentados importantes métodos para descrever propriedades de regiões segmentadas em uma imagem digital. Nesta seção, são apresentados fundamentos e métodos para reconhecimento de objetos, ou seja, classificação de regiões segmentadas de uma imagem em padrões conhecidos.

Os métodos de reconhecimento de objetos pertencem à área de Visão Computacional, que abrange conhecimento das áreas de Processamento Digital de Imagens e de Inteligência Artificial.

O objetivo principal dos métodos de reconhecimento de objetos é associar, por meio de uma função de decisão, o padrão (conjunto de descritores ou características) de um objeto de interesse à uma classe de padrões, geralmente predefinida.

Com base nesse método, foram desenvolvidas diversas abordagens de reconhecimento de objetos, detalhadas a seguir:

- *Casamento (matching)*: Um padrão inicialmente desconhecido, representado por um vetor de características, é associado à uma classe predefinida, também representada por um vetor de características, utilizando como critério de comparação a distância (euclidiana, por exemplo) entre esse dois vetores.
- *Classificadores bayesianos*: Segue o mesmo princípio da abordagem anterior, com a inclusão de um comportamento probabilístico para tratamento de erros. Cada classificação incorreta é denominada como perda e é atribuída à um classificador que busca minimizar a perda média total, tratando a probabilidade de acontecimento de novas classificações incorretas.
- *Redes neurais*: Método usado para solucionar problemas de Inteligência Artificial. Consiste no uso de modelos computacionais inspirados no sistema nervoso humano, capazes de reconhecer padrões e realizar aprendizado de máquina utilizando sua experiência, adaptando sua função de decisão após sucessivas classificações corretas.
- *Optimum Path Forest*: Proposto por Papa et al. (2012), o OPF é um classificador com aprendizado fundamentado em Teoria dos Grafos. Em geral, apresenta eficácia similar aos métodos descritos anteriormente, porém, tem eficiência superior no treinamento, melhorando o desempenho do processo final.
- *Support Vector Machine*: Máquina de vetores de suporte, as SVM são um conjunto de métodos do aprendizado supervisionado que analisam os dados e reconhecem padrões, usado para classificação e análise de regressão.

2.2 Bibliotecas de Processamento Digital de Imagens

Nessa seção, serão apresentadas duas bibliotecas livres e de código aberto com foco nas áreas de Processamento Digital de Imagens e Visão Computacional.

2.2.1 OpenCV

A OpenCV (*Open Source Computer Vision Library*) é uma biblioteca multiplataforma, livre e de código aberto, originalmente desenvolvida pela Intel e agora mantida pelas organizações Willow Garage e Itseez, voltada ao desenvolvimento de aplicações de Visão Computacional. Escrita em C/C++, com foco em eficiência computacional e aplicações de tempo real, a biblioteca é otimizada para trabalhar com processamento concorrente.

Além do módulo principal de Visão Computacional, a biblioteca também conta com módulos de Estruturas de Dados, Álgebra Linear, Aprendizado de Máquinas e Interface Gráfica do Usuário (*GUI*). ([BRADSKY; PISAREVSKY; BOUGUET, 2006](#))

2.2.2 BIAL

A BIAL (*Biomedical Image Analysis Library*) é uma biblioteca livre, de código aberto e portável para multiplas plataformas, inicialmente desenvolvida como projeto de extensão de um grupo de alunos e professores do Instituto de Ciência e Tecnologia da Universidade Federal de São Paulo (Unifesp).

Essa biblioteca se destaca das demais por oferecer suporte nativo à operações de Processamento de Imagens e Análise de Imagens comuns na área da saúde e biomedicina. ([CAP-PABIANCO, Fábio A. M., 2014](#))

2.3 Moedas do Real

Após sucessivas trocas de unidades monetárias, o Brasil adotou o Real como moeda padrão em julho de 1994, durante o mandato do presidente Itamar Franco e sob o comando de Fernando Henrique Cardoso, então ministro da Fazenda.

A primeira família de moedas do Real, lançada em 1994, é composta por seis moedas nos valores de 1, 5, 10, 25, 50 centavos e 1 real. As moedas eram cunhadas em aço inoxidável e seu diâmetro variava entre 20mm e 24mm, de acordo com o valor. No anverso, apresentavam o valor da moeda entre ramos de louro estilizados (com exceção da moeda de 25 centavos, que continha ondas). No reverso, as moedas traziam a Efígie da República.

A segunda família, objeto de interesse desse trabalho, foi lançada em 1998 e apresenta seis moedas nos mesmos valores da primeira família. No entanto, as moedas são cunhadas em materiais diversos como aço, cobre e bronze e contam com uma variação maior de diâmetro,

entre 17mm e 27mm. O anverso é padronizado e traz o valor da moeda acompanhado de uma alegoria que faz alusão à Bandeira Nacional. Já o reverso traz imagens de figuras históricas como Pedro Álvares Cabral, Tiradentes e D. Pedro I. A morfologia das moedas está ilustrada nas Figuras 9 e 10. ([Banco Central do Brasil, 2003](#))



Figura 9 – Anversos da segunda família de moedas do Real. ([Banco Central do Brasil, 2003](#))



Figura 10 – Reversos da segunda família de moedas do Real. ([Banco Central do Brasil, 2003](#))

2.4 Considerações Finais

Neste capítulo, foram apresentados tópicos e conceitos básicos das áreas de Processamento Digital de Imagens, Reconhecimento de Padrões e Visão Computacional, além de informações sobre bibliotecas relacionadas à essas áreas.

3 Desenvolvimento

O objetivo deste capítulo é descrever a metodologia, dificuldades e soluções encontradas durante o desenvolvimento da solução para o problema apresentado.

Nesta seção, é apresentada uma visão geral do algoritmo desenvolvido, com o intuito de facilitar a compreensão do mesmo. Maiores informações e detalhes de implementação serão explicados nas seções que seguem.

Numa visão macro, podemos dividir o algoritmo em oito passos bem definidos com objetivos específicos. Cada passo será explicado com mais detalhes nas seções que seguem.

1. **Organização de arquivos:** Interface com o sistema de arquivos.
2. **Sistema de cores:** Conversão das imagens coloridas para imagens em escala-de-cinza.
3. **Filtragem de ruído:** Aplicação de filtro de mediana para suavizar ruídos.
4. **Detecção de áreas de interesse:** Transformada de Hough aplicada para encontrar regiões circulares de interesse.
5. **Extração de descritores:** Extração de vetores capazes de descrever as regiões de interesse utilizando a transformada SIFT (*Scale-Invariant Feature Transform*).
6. **Dicionário de características (*Bag-of-Words*):** Leitura de imagens do conjunto de treinamento, extração de vetores de características (ou “palavras”) das regiões de interesse e criação de um dicionário de uso comum contendo essas “palavras”.
7. **Treinamento da SVM:** Com base no dicionário criado no passo anterior, um novo extrator de características é criado e aplicado sobre as imagens do conjunto de treinamento, extraíndo “palavras normalizadas”. Em seguida, essas novas “palavras” e suas respectivas classificações são indexadas em um SVM para treinar o classificador.
8. **Classificação:** Com base no classificador criado e treinado no passo anterior, uma nova imagem não pertencente ao conjunto de testes é carregada e tentativamente classificada em uma das categorias de moedas existentes.

Por exigir grande esforço computacional, os passos de criação de dicionário e treinamento da SVM são executados apenas uma vez. Os dados resultantes da criação da SVM são armazenados em um arquivo que é automaticamente carregado em execuções posteriores, reduzindo o tempo de classificação das moedas.

3.1 Organização de arquivos

O primeiro passo do desenvolvimento consistiu na definição de um protocolo de aquisição de imagens digitais dos objetos de interesse, moedas de Real. Durante a execução de testes preliminares, ficou clara a necessidade de definir um protocolo padronizado de aquisição das imagens devido à forte semelhança entre as texturas que ilustram os objetos.

Como o objetivo do algoritmo é identificar e classificar as moedas de Real através da análise de suas características internas, como cor e textura e não externas, como tamanho, foi necessário definir um ambiente padrão para aquisição de imagens, com câmera em distância fixa e fundo branco, para que variantes como resolução e iluminação pudessem ser controladas. Vale ressaltar que, mesmo dentro de um ambiente controlado, duas fotografias da mesma moeda podem apresentar diferenças notáveis uma das outras.



Figura 11 – Imagem do anverso de uma moeda de 1 real no ambiente padronizado.

Para cada valor de moeda, foram adquiridas 80 imagens, com 40 ilustrando o anverso e, as outras 40, o reverso da moeda representada. Além disso, para cada valor, foram utilizadas de 3 a 5 exemplares de cada moeda, com variações de desgaste, causado pelo tempo, e limpeza.

Em seguida, as imagens foram organizadas numa estrutura de diretórios visando facilitar sua leitura no algoritmo. As pastas nomeadas *0*, *1*, *2*, *3*, *4* e *5*, correspondem aos valores R\$0,01, R\$0,05, R\$0,10, R\$0,25, R\$0,50 e R\$1,00, respectivamente. Dentro de cada pasta, duas pastas, nomeadas *0* e *1*, armazenam as imagens dos anversos e reversos, respectivamente, de cada moeda fotografada. Por fim, a nomenclatura das imagens nas pastas segue ordem crescente, de *0.png* até *n.png*.

A figura 12 ilustra a organização adotada.

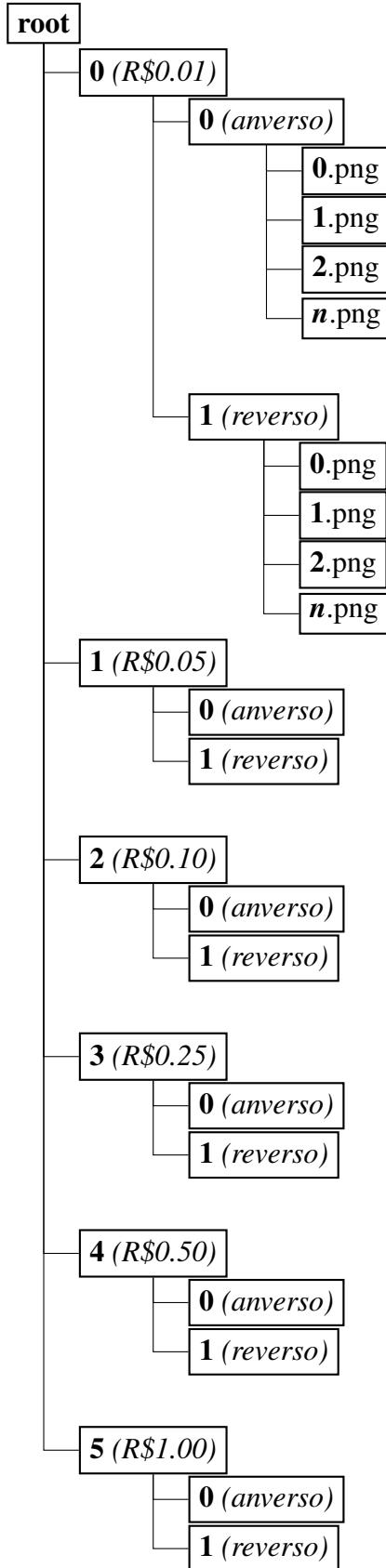


Figura 12 – Organização de diretórios usada para armazenar as imagens das moedas.

3.2 Sistema de cores

Durante o estudo da fundamentação teórica, foi levantada a possibilidade de explorar o histograma de cores das moedas com o objetivo de extrair informações úteis na tarefa de classificação das mesmas.

No entanto, durante experimentos preliminares realizados com o histograma de cores das moedas, foi possível observar que, devido a diferenças nos materiais de confecção das moedas, moedas do mesmo valor apresentam colorações bastante distintas, como podemos observar na figura 13.



Figura 13 – Composição de imagens do anverso de duas moedas de 10 centavos ilustrando diferença nos materiais.

Esta falta de padronização no espectro de cores das moedas poderia, potencialmente, dificultar no processo de classificação. Além disso, trabalhar com imagens coloridas aumenta o custo computacional de todas as operações envolvidas. Por essas razões, foi tomada a decisão de, antes de qualquer operação com as imagens, realizar uma conversão para escala de cinza.

Para executar a conversão, é utilizada a função `cvtColor()` do OpenCV.



Figura 14 – Composição de imagens do anverso de duas moedas de 10 centavos convertidas para escala de cinza.

3.3 Filtragem de ruído

Conforme explicado anteriormente, mesmo em um ambiente controlado, as imagens obtidas irão apresentar diferenças sutis, causadas por ruído e variações na iluminação.

Com o objetivo de suavizar essas variações, porém, sem perder informações consideradas importantes para o reconhecimento das moedas, foi aplicado um filtro mediana de tamanho 5 pixels, com a função `medianBlur()` do OpenCV.



Figura 15 – Anverso de moeda de 1 real em escala de cinza sem filtro.



Figura 16 – Anverso de moeda de 1 real em escala de cinza suavizada com filtro de mediana 5.

3.4 Detecção de áreas de interesse

Para padronizar a metodologia de extração de características e garantir que apenas informações do interior das moedas seja considerado, é necessário definir um método de extração de regiões de interesse.

Como os objetos de interesse são circulares, a função `HoughCircles()` do OpenCV, configurada com os parâmetros adequados, é capaz de encontrar todos os objetos circulares em cada imagem e retornar seus pontos centrais e raios.

Uma boa configuração de parâmetros é fundamental para o funcionamento desta função. Após testes empíricos realizados no conjunto de imagens de treinamento variando estes parâmetros, foi possível chegar em uma combinação capaz de encontrar o maior círculo presente em uma imagem, ou seja, as fronteiras de cada moeda.

Para evitar que pontos da extrema borda das moedas, idênticos na maioria das situações, sejam incluídos nas regiões de interesse, os círculos tem seu raio diminuído em 10%. Com ponto central e raio definidos, é possível plotar um círculo em uma matriz binária destacando a região de interesse.

A implementação do método de detecção de círculos Hough faz uso, internamente, do método de detecção de bordas de Canny. Os testes iniciais revelaram que algumas moedas (como a de 1 real) possuíam texturas internas que eram identificadas como borda e confundiam o algoritmo de Hough. Para evitar esse comportamento indesejado, antes da etapa de detecção, aplica-se um filtro de mediana com tamanho 21 para borrar as bordas internas das moedas.



Figura 17 – Anverso de moeda de 1 real, convertida para escala de cinza, suavizada com um filtro de mediana de tamanho 21 e com o círculo de Hough detectado pelo algoritmo em destaque.

3.5 Extração de descritores

Com as regiões de interesse devidamente detectadas, o próximo passo é extrair um conjunto de características (“palavras”) capazes de descrever talas regiões.

Nativamente, a biblioteca OpenCV traz uma série de funções capazes de extrair características de regiões. Como o modelo proposto busca identificar e classificar moedas independente de variações de dimensão ou rotação, os algoritmos SIFT (*Scale-Invariant Feature Transform*) e SURF (*Speeded Up Robust Features*) se destacaram como possibilidades viáveis dentre os métodos disponíveis dentro do OpenCV ([OpenCV 2.4.11.0 documentation, 2015](#)).

O funcionamento de ambos os algoritmos segue, em linhas gerais, a mesma metodologia. Inicialmente, o algoritmo seleciona “pontos de interesse” (*keypoints*) da imagem a partir de características salientes de seu espaço linear obtidas realizando uma convolução da imagem original com uma série de imagens filtradas em diferentes escalas. Em seguida, as características locais são organizadas em um histograma de características. Por realizar cálculos mais rapidamente, o SURF é recomendado para aplicações em tempo-real ([OYALLON; RABIN, 2013](#)).

Contudo, em testes empíricos realizados aplicando os dois algoritmos no conjunto de imagens de moedas, não foi possível observar nenhum ganho na velocidade de execução do SURF em relação ao SIFT. Além disso, as execuções do SIFT mostraram “pontos de interesse” visivelmente mais interessantes do que as do SURF. Por estes motivos, optou-se por utilizar o algoritmo SIFT no decorrer deste trabalho. As figuras [18](#) e [19](#) ilustram os resultados.

Por serem extremamente sensíveis a variações na moeda, o número de *keypoints* obtidos por imagem pode variar consideravelmente, como é possível observar na tabela [1](#).

Tabela 1 – Média de *keypoints* por valor de moeda.

		Valor da Moeda	Média de Keypoints
R\$0,01	anverso	260	
	reverso	258	
R\$0,05	anverso	630	
	reverso	291	
R\$0,10	anverso	608	
	reverso	592	
R\$0,25	anverso	890	
	reverso	978	
R\$0,50	anverso	870	
	reverso	713	
R\$1,00	anverso	1321	
	reverso	848	



Figura 18 – Anverso de moeda de 1 real, com *keypoints* detectados pelo algoritmo SIFT em destaque.

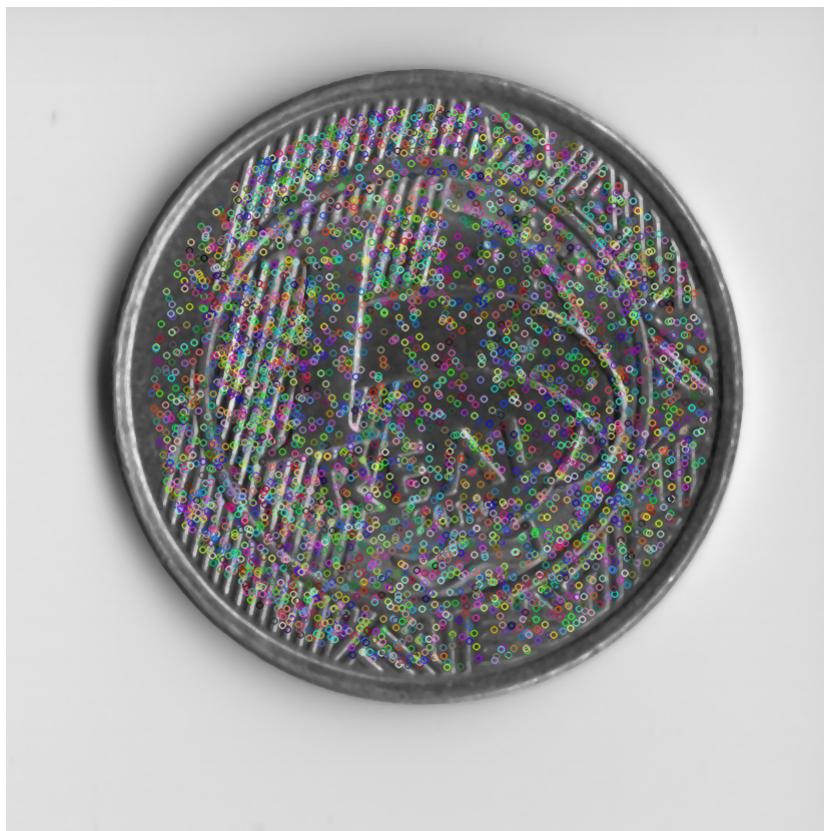


Figura 19 – Anverso de moeda de 1 real, com *keypoints* detectados pelo algoritmo SURF em destaque.

3.6 Dicionário de características

Na seção anterior, foi apresentado um eficiente método de extração de características (“pontos de interesse”) dos objetos que devem ser reconhecidos e classificados.

No entanto, como foi possível observar na tabela 1, o método poderá retornar conjuntos de dimensões e valores distintos para descrever moedas do mesmo valor. Antes que seja possível realizar a comparação de vetores de características das moedas, é necessário colocá-los em caráter de equivalência.

Na área de reconhecimento textual, o modelo *Bag-Of-Words* é usado com frequência para categorização de documentos. Em [Csurka et al. \(2004\)](#), foi proposta uma variação desse modelo, capaz de trabalhar com imagens (“palavras visuais”).

Esse modelo preve a quantização de uma série de palavras visuais encontradas em um conjunto de documentos com o objetivo de destacar as palavras que aparecem com maior frequência e criar um dicionário de palavras. O tempo de execução desse passo é relativamente lento e varia de acordo com o tamanho do dicionário.

Aliada ao extrator de características *SIFT*, esta variação do método *Bag-Of-Words* se destaca como alternativa válida para problema da comparação de descritores de tamanho variado. Além disso, por não ser dependente da geometria dos objetos de interesse, este método se mostra robusto o suficiente para normalizar características de qualquer região pré-definida.

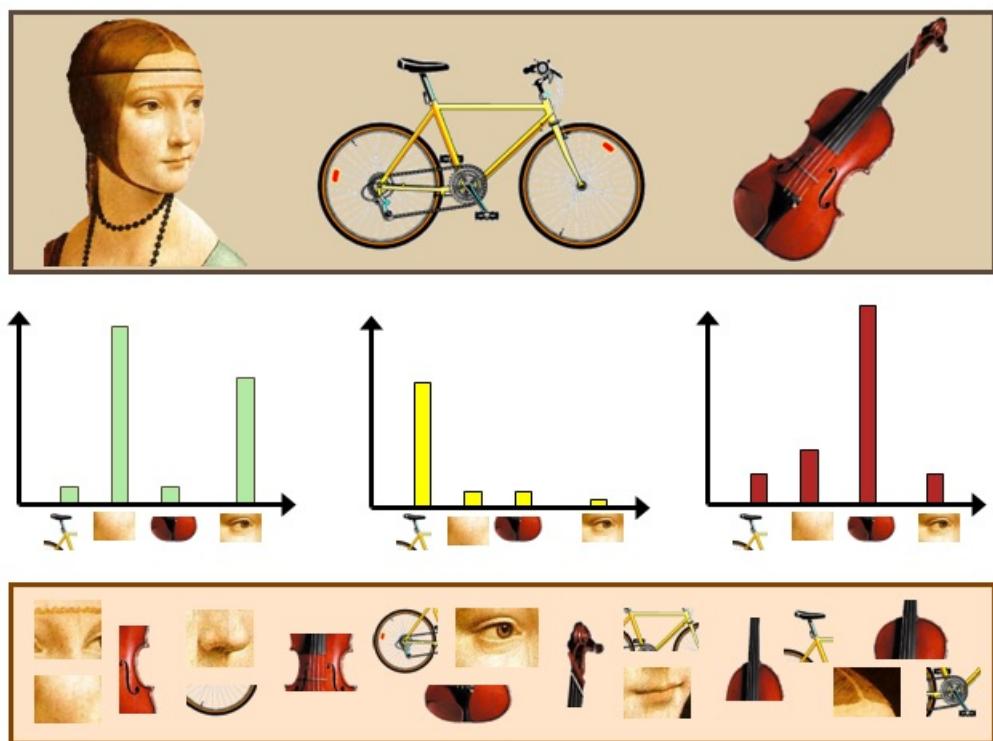


Figura 20 – Método *Bag-Of-Words* ilustrado: Conjunto de imagens avaliadas, histogramas de características extraídas e dicionário formado pelo conjunto de características. ([Gil Levi, 2013](#))

Com o dicionário de características construído, é necessário reavaliar as imagens tomando como referência estas características globais estabelecidas. Neste passo, o conjunto de documentos é avaliado novamente porém, dessa vez, contabilizando em um histograma apenas o aparecimento das palavras-chave do dicionário.

O resultado deste método é um vetor descritor de tamanho fixo, com as mesmas dimensões definidas na criação do dicionário, composto pela frequência do aparecimento de cada palavra-chave nos documentos.

3.7 Treinamento da SVM

Após a criação do dicionário, tomando como partida estes vetores de características normalizados, realiza-se a etapa de aprendizado supervisionado. Para esta tarefa, foi usado o método de Máquina de Vetores de Suporte (SVM, do inglês: *support vector machine*), que já possui implementação no OpenCV.

No OpenCV, para cada amostra, é necessário fornecer seu vetor e um rótulo (ou *label*) capaz de identificar tal amostra. Após o registro de todas as amostras, é executada a função `svm.train()` que realiza o treinamento de uma SVM genérica, ou seja, utilizando os parâmetros padrão, com suporte a múltiplas classes.

O resultado desse método é uma SVM treinada capaz de, dada uma nova amostra, prever a classificação mais aproximada a qual esta amostra deve pertencer.

A tabela 2 traz a organização de classes utilizada neste trabalho. Para facilitar a interface com os arquivos de treinamento, a nomenclatura segue a estrutura estabelecida na tabela 12.

Tabela 2 – Identificação das classes utilizadas para cada moeda.

Valor da Moeda	Classe	
R\$0,01	anverso	0
	reverso	10
R\$0,05	anverso	1
	reverso	11
R\$0,10	anverso	2
	reverso	12
R\$0,25	anverso	3
	reverso	13
R\$0,50	anverso	4
	reverso	14
R\$1,00	anverso	5
	reverso	15

3.8 Classificação

A etapa de classificação consiste em, dada uma moeda com valor não conhecido previamente, realizar uma predição do seu possível valor e retorná-lo ao usuário.

Na seção anterior, foram explicadas as etapas de construção do dicionário de características e treinamento da SVM usadas neste trabalho. Esses processos, apesar de serem consideravelmente mais lentos que o processo de classificação, precisam ser executados uma única vez para um determinado conjunto de moedas de teste. Após sua conclusão, o dicionário e a SVM podem ser salvas em um arquivos de texto e lidas pelo programa durante o início da próxima execução.

Após a leitura do dicionário e da SVM, o programa lê a moeda não conhecida, identifica a região de interesse, extrai suas características (tomando como base o dicionário criado) e, usando a SVM, prediz e retorna o valor provável da mesma.

Para auxiliar no processo de validação do algoritmo criado, as imagens das moedas utilizadas no teste foram estruturadas da mesma forma que as imagens das moedas utilizadas do treinamento. Dessa forma, é possível validar se as predições do SVM estão certas ou erradas e, em seguida, unificar e comparar os resultados em uma tabela de acurácia.

4 Resultados

Neste capítulo, são apresentados os resultados obtidos nos testes do algoritmo proposto.

4.1 Dicionário de palavras

Nesta seção, estão agrupados testes do algoritmo com diferentes tamanhos de dicionários de palavras. Além de expor a acurácia do método desenvolvido, também é possível observar a relação entre a variação do tamanho do dicionário de características e seu impacto no tempo de criação do dicionário e treinamento da SVM e, por fim, nas porcentagens de acerto.

Vale ressaltar que, independente do tamanho do dicionário utilizado, o tempo médio do treinamento da SVM foi de aproximadamente 3 minutos. Esse tempo pode variar consideravelmente quando alterado o número de amostras utilizadas no treinamento.

Além disso, o tempo de reconhecimento da moeda foi de aproximadamente 1 segundo, mesmo levando em conta o tempo de leitura da imagem, processamento e predição.

Para elaboração desses testes, foram criados seis cenários de teste variando o tamanho do dicionário de 100 a 10000 características. No total, foram utilizadas 80 imagens de cada valor de moeda - 40 do anverso de 40 do reverso. Desse conjunto, metade foi aplicada no treinamento do SVM e, a outra metade, nos testes de classificação.

Em todas as execuções, apesar de apresentar pequenas variações na corretude do raio encontrado, o algoritmo de Hough foi capaz de detectar a região circular principal (ou seja, a moeda) de cada imagem.

Além disso, pode-se observar que a predição de valores referentes ao reverso das moedas foi superior a predição de anversos. É possível que isso decorra do fato de que o anverso das moedas seja composto por texturas comuns como linhas horizontais e um conjunto de apenas algarismos (0, 1, 2 e 5) enquanto o reverso apresenta formas mais distintas entre si.

O algoritmo proposto por [Jory \(2011\)](#), exige múltiplas moedas na fotografia para classificá-las por meio da comparação de características como cor, raio e diâmetro e leva aproximadamente 4 minutos para reconhecer cada uma delas.

O algoritmo proposto neste trabalho é capaz de trabalhar com as moedas de maneira independente, extraíndo as características de cada uma e comparando com a SVM criada. Além disso, após os custosos processos de criação do dicionário e treinamento da SVM, a classificação de cada moeda se dá em tempo inferior a 1 segundo.

Na tabela 3, estão agrupados os resultados.

Tabela 3 – Tabelas comparativas de resultados com diferentes tamanhos de dicionário.

Cenário 1		
Tamanho do dicionário		100
Tempo de criação do dicionário		9min
Valor da Moeda		Acurácia
R\$0,01	anverso	45%
	reverso	75%
R\$0,05	anverso	35%
	reverso	60%
R\$0,10	anverso	40%
	reverso	95%
R\$0,25	anverso	25%
	reverso	65%
R\$0,50	anverso	15%
	reverso	75%
R\$1,00	anverso	95%
	reverso	100%
Média de acurácia		60.41%

Cenário 2		
Tamanho do dicionário		500
Tempo de criação do dicionário		20min
Valor da Moeda		Acurácia
R\$0,01	anverso	65%
	reverso	65%
R\$0,05	anverso	50%
	reverso	60%
R\$0,10	anverso	70%
	reverso	100%
R\$0,25	anverso	35%
	reverso	100%
R\$0,50	anverso	60%
	reverso	95%
R\$1,00	anverso	100%
	reverso	100%
Média de acurácia		75%

Cenário 3		
Tamanho do dicionário		1000
Tempo de criação do dicionário		29min
Valor da Moeda		Acurácia
R\$0,01	anverso	60%
	reverso	65%
R\$0,05	anverso	65%
	reverso	55%
R\$0,10	anverso	70%
	reverso	100%
R\$0,25	anverso	35%
	reverso	100%
R\$0,50	anverso	70%
	reverso	100%
R\$1,00	anverso	100%
	reverso	100%
Média de acurácia		76.66%

Cenário 4		
Tamanho do dicionário		2500
Tempo de criação do dicionário		1h 14min
Valor da Moeda		Acurácia
R\$0,01	anverso	60%
	reverso	50%
R\$0,05	anverso	70%
	reverso	40%
R\$0,10	anverso	15%
	reverso	100%
R\$0,25	anverso	45%
	reverso	100%
R\$0,50	anverso	75%
	reverso	95%
R\$1,00	anverso	100%
	reverso	100%
Média de acurácia		70.83%

Cenário 5		
Tamanho do dicionário		5000
Tempo de criação do dicionário		1h 57min
Valor da Moeda		Acurácia
R\$0,01	anverso	40%
	reverso	45%
R\$0,05	anverso	70%
	reverso	45%
R\$0,10	anverso	10%
	reverso	100%
R\$0,25	anverso	20%
	reverso	100%
R\$0,50	anverso	80%
	reverso	100%
R\$1,00	anverso	100%
	reverso	100%
Média de acurácia		67.5%

Cenário 6		
Tamanho do dicionário		10000
Tempo de criação do dicionário		~2h 33min
Valor da Moeda		Acurácia
R\$0,01	anverso	60%
	reverso	45%
R\$0,05	anverso	60%
	reverso	40%
R\$0,10	anverso	10%
	reverso	90%
R\$0,25	anverso	35%
	reverso	100%
R\$0,50	anverso	90%
	reverso	80%
R\$1,00	anverso	100%
	reverso	100%
Média de acurácia		67.5%

4.2 Assertividade

Como foi possível observar na seção anterior, dicionários maiores não significam necessariamente taxas de acurácia superiores. Nos testes realizados, a melhor taxa de acurácia foi de 76.6% e ocorreu no teste utilizando um dicionário de 1000 características.

Para detalhar os resultados do algoritmo, a tabela 4 traz um sumário de todas as classificações - corretas e incorretas - obtidos nos testes.

Nesta bateria de testes, foi aplicada uma técnica de validação cruzada, útil para estimar a precisão de algoritmos de predição. Nessa técnica, o conjunto de amostras é particionado em subconjuntos mutualmente exclusivos. Então, uma partição é utilizada para o treinamento do algoritmo e, as outras, na validação da predição. Este processo segue até que todas as partições individuais tenham sido utilizadas no treinamento. (KOHAVI, 1995)

Tabela 4 – Tabela de assertividade do algoritmo com um dicionário de 1000 palavras.

5 Conclusão

Nas áreas de Processamento de Imagens e Visão Computacional, não é possível aplicar um único algoritmo genérico para solucionar qualquer problema. Com um objetivo em mente, é desenvolvido um procedimento específico para atingir os melhores resultados possíveis.

Durante a revisão bibliográfica e análise dos métodos de Processamento de Imagens existentes, notou-se que é possível realizar um encadeamento de diversos processos para converter as imagens em sua forma bruta em dados de interesse.

No levantamento bibliográfico deste trabalho, ficou evidente a lacuna acadêmica existente no ramo de detecção e identificação de moedas de Real, contando com apenas um trabalho publicado. Um algoritmo eficiente e de alta-confiabilidade pode trazer grandes contribuições para a área e, possivelmente, inspirar novos trabalhos relacionados.

Nos testes realizados, foi possível constatar uma grande acurácia (superior a 75%) do algoritmo em um tempo de processamento mínimo. Como a grande maioria dos algoritmos de processamento de imagens, uma boa parametrização dos métodos utilizados pode trazer resultados ainda superiores. Um trabalho futuro pode abordar a otimização destes parâmetros.

Como trabalho futuro, podem ser explorados o reconhecimento de mais que 1 moeda, o tratamento de fundos diferenciados e, até mesmo, o desenvolvimento uma aplicação para celulares.

Referências

- Banco Central do Brasil. *Moedas*. 2003. <https://www.bcb.gov.br/?MOEDA>. Online; acessado em: 9 de Abril de 2014. Citado 2 vezes nas páginas 13 e 37.
- BRADSKY, G. R.; PISAREVSKY, V.; BOUGUET, J. *Learning OpenCV: Computer Vision with the OpenCV Library*. [S.l.]: Springer, 2006. Citado na página 36.
- CANNY, F. J. A Computational Approach to Edge Detection. v. 8, n. 6, p. 679–698, 1986. Citado na página 32.
- CAPPABIANCO, Fábio A. M. *Currículo do Sistema de Currículos Lattes*. 2014. <http://lattes.cnpq.br/7438076121387151>. Online; acessado em: 27 de Junho de 2014. Citado na página 36.
- CSURKA, G. et al. Visual categorization with bags of keypoints. In: *In Workshop on Statistical Learning in Computer Vision, ECCV*. [S.l.: s.n.], 2004. p. 1–22. Citado na página 47.
- DAVIES, E. Image space transforms for detecting straight edges in industrial images. *Pattern Recognition Letters*, v. 4, n. 3, p. 185 – 192, 1986. ISSN 0167-8655. Citado na página 32.
- Gil Levi. *Bag of Words Models for visual categorization*. 2013. <https://gilscvblog.wordpress.com/2013/08/23/bag-of-words-models-for-visual-categorization/>. Online; acessado em: 7 de Maio de 2015. Citado 2 vezes nas páginas 14 e 47.
- GONZALEZ, R. C.; WOODS, R. E. *Digital Image Processing*. 3rd. ed. Upper Saddle River, N.J.: Prentice Hall, 2008. – p. ISBN 9780131687288 013168728X 9780135052679 013505267X. Citado 6 vezes nas páginas 13, 25, 27, 29, 30 e 31.
- JORY, N. M. *Reconhecimento de moedas via processamento de imagens*. São José, Santa Catarina, Brasil: [s.n.], 2011. Monografia; Graduação em Sistemas de Telecomunicações. Citado 2 vezes nas páginas 21 e 51.
- KOHAVI, R. A study of cross-validation and bootstrap for accuracy estimation and model selection. In: . [S.l.]: Morgan Kaufmann, 1995. p. 1137–1143. Citado na página 55.
- MATHWORKS. *Top-hat filtering*. 2014. <http://www.mathworks.com/help/images/ref/imtophat.html>. Online; acessado em: 29 de Junho de 2014. Citado 2 vezes nas páginas 13 e 28.
- OpenCV 2.4.11.0 documentation. *Feature Detection and Description — OpenCV 2.4.11.0 documentation*. 2015. http://docs.opencv.org/modules/features2d/doc/feature_detection_and_description.html. Online; acessado em: 3 de Maio de 2015. Citado na página 45.
- OYALLON, E.; RABIN, J. An analysis and implementation of the surf method, and its comparison to sift. 2013. Citado na página 45.
- PAPA, J. P. et al. Efficient supervised optimum-path forest classification for large datasets. *Pattern Recognition*, v. 45, n. 1, p. 512–520, 2012. Citado na página 35.

POL, L.-M. *Lenna 97: A Complete Story of Lenna*. 1997. <http://www.ee.cityu.edu.hk/~lmpo/lenna/Lenna97.html>. Online; acessado em: 29 de Junho de 2014. Citado na página 25.

THEODORIDIS, S.; KOUTROUMBAS, K. *Pattern Recognition*. 3rd. ed. [S.l.]: Elsevier, 2006. ISBN 0123695317. Citado na página 21.

WANG, T. T.; MARTIN, W. *Hough Transform: Circles*. 2010. <http://www.cs.cmu.edu/~wmartin/vision/circles.html>. Online; acessado em: 29 de Abril de 2014. Citado 2 vezes nas páginas 13 e 33.

WANGENHEIM, A. von. *Análise de Sinais e Imagens para Reconhecimento de Padrões*. 2007. <http://www.inf.ufsc.br/~patrec/imagens.html>. Online; acessado em: 7 de Abril de 2014. Citado na página 21.

Apêndices

APÊNDICE A – Código Fonte

```

#include <opencv/cv.h>
#include <opencv/highgui.h>
#include <opencv2/nonfree/features2d.hpp>
#include <opencv2/ml/ml.hpp>

#include <iostream>
#include <ctime>

using namespace cv;
using namespace std;

#define BUILD_DICTIONARY true
#define BUILD_SVM true
#define DICTIONARY_SAMPLES 20
#define DICTIONARY_SIZE 2500

//Funcao auxiliar responsavel por retornar o tempo atual
string getNow(){
    time_t rawtime;
    struct tm * timeinfo;
    char buffer[80];

    time (&rawtime);
    timeinfo = localtime(&rawtime);

    strftime(buffer,80,"%d-%m-%Y %I:%M:%S",timeinfo);
    string timeString(buffer);

    return timeString;
}

//Dado um valor de classe c, retorna classificacao textual da moeda
string getCoinValue(int c){
    switch (c) {
        case 0: return "R$0.01 (anverso)";

```

```
    case 1: return "R$0.05 (anverso)";  
    case 2: return "R$0.10 (anverso)";  
    case 3: return "R$0.25 (anverso)";  
    case 4: return "R$0.50 (anverso)";  
    case 5: return "R$1.00 (anverso)";  
    case 10: return "R$0.01 (reverso)";  
    case 11: return "R$0.05 (reverso)";  
    case 12: return "R$0.10 (reverso)";  
    case 13: return "R$0.25 (reverso)";  
    case 14: return "R$0.50 (reverso)";  
    case 15: return "R$1.00 (reverso)";  
    default: return "Indefinido";  
}  
}  
  
//Leitura de imagens  
Mat readImage(string path){  
    Mat image;  
    image = imread(path);  
    if(!image.data){  
        exit(-1);  
    }  
    return image;  
}  
  
//Dada uma imagem de entrada, retorna sua versão em escala-de-cinza  
Mat colorToGray(Mat input){  
    Mat output;  
    cvtColor(input, output, CV_BGR2GRAY);  
    return output;  
}  
  
//Dada uma imagem de entrada e um tamanho de máscara d, retorna uma  
//versão suavizada da imagem  
Mat smoothFilter(Mat input, int d){  
    Mat output;  
    medianBlur(input, output, d);  
    return output;  
}
```

```
//Dada uma imagem de entrada, retorna mascara contendo o maior
    circulo detectado atraves da funcao HoughCircles
Mat extractHoughCircleMask(Mat input) {
    vector<Vec3f> circles;
    Mat inputFilterGray = smoothFilter(colorToGray(input), 21);

    HoughCircles(inputFilterGray, circles, CV_HOUGH_GRADIENT, 2,
                 input.rows, 100, 100, 0, input.rows);
    if(circles.size() == 0){
        exit(-1);
    }

    Mat mask(input.size(), CV_8UC1);
    mask.setTo(0);

    Point center(cvRound(circles[0][0]), cvRound(circles[0][1]));
    int radius = cvRound(circles[0][2]);
    radius = radius-(radius*10/100);
    circle(mask, center, radius, 1 ,-1, 8, 0);

    return mask;
}

//Dada uma imagem e uma mascara, extrai descritores (utilizando o
    metodo Sift)
Mat extractDescriptorsUsingSift(Mat input, Mat mask){
    Mat inputFilterGray = smoothFilter(colorToGray(input), 5);

    SiftFeatureDetector detector(0);
    vector<KeyPoint> keypoints;
    detector.detect(inputFilterGray, keypoints, mask);

    Mat descriptors;
    SiftDescriptorExtractor descriptorExtractor;
    descriptorExtractor.compute(inputFilterGray, keypoints,
                               descriptors);

    return descriptors;
}
```

```
//Dada uma imagem, uma mascara e um extrator de descritores BOW,
    extrai descritores
Mat extractDescriptorsUsingBOW(Mat input, Mat mask,
    BOWImgDescriptorExtractor descriptorExtractor) {
    Mat inputFilterGray = smoothFilter(colorToGray(input), 5);

    SiftFeatureDetector detector(0);
    vector<KeyPoint> keypoints;
    detector.detect(inputFilterGray, keypoints, mask);

    Mat descriptors;
    descriptorExtractor.compute(inputFilterGray, keypoints,
        descriptors);

    return descriptors;
}

//Funcao responsavel por criar um dicionario de caracteristicas e
    salvar em path
void buildDictionary(string path, int dictionarySize){
    printf("%s Iniciando criacao do dicionario amostrando %d imagens
        de cada moeda.\n", getNow().data(), DICTIONARY_SAMPLES);

    Mat descritoresMat;
    for(int c=0;c<6;c++) {
        for(int l=0; l<2; l++) {
            for(int i=0;i<DICTIONARY_SAMPLES;i++) {
                char fileName[50];
                sprintf(fileName, "%s%d%s%d%s%d%s", "train/", c, "/", l,
                    "/", i, ".png");
                Mat input = readImage(fileName);
                Mat mask = extractHoughCircleMask(input);
                Mat descriptors = extractDescriptorsUsingSift(input, mask);
                descritoresMat.push_back(descriptors);
            }
        }
    }

    printf("%s Construindo dicionario com %d palavras em %s... \n",
        getNow().data(), dictionarySize, path.data());
    BOWKMeansTrainer bowTrainer(dictionarySize);
```

```

    Mat dictionary=bowTrainer.cluster(descritoresMat);

    FileStorage fs(path, FileStorage::WRITE);
    fs << "coinDictionary" << dictionary;
    fs.release();

    printf("%s Dicionario construido com sucesso!\n", getNow().data());
}

//Funcao responsavel por treinar uma SVM com base no extrator de
//caracteristicas gerado pelo dicionario e salvar em path
void trainSVM(string path, BOWImgDescriptorExtractor
descriptorExtractor) {
printf("%s Iniciando dados da SVM amostrando %d imagens de cada
moeda.\n", getNow().data(), DICTIONARY_SAMPLES);

Mat trainingData;
Mat labels;

for(int c=0;c<6;c++) {
    for(int l=0; l<2; l++) {
        for(int i=0;i<DICTIONARY_SAMPLES;i++) {
            char fileName[50];
            sprintf(fileName, "%s%d%s%d%s%d%s", "train/", c, "/", l,
                    "/", i, ".png");
            Mat input = readImage(fileName);
            Mat mask = extractHoughCircleMask(input);
            Mat bowDescriptors = extractDescriptorsUsingBOW(input,
                mask, descriptorExtractor);

            trainingData.push_back(bowDescriptors);
            int label = c + l*10;
            labels.push_back(label);
        }
    }
}

printf("%s Treinando a SVM e salvando em %s.\n", getNow().data(),
path.data());

CvSVMParams params;

```

```
params.svm_type = CvSVM::C_SVC;
params.kernel_type = CvSVM::LINEAR;
params.term_crit = cvTermCriteria(CV_TERMCRIT_ITER, 100, 1e-6);

CvSVM svm;
svm.train(trainingData, labels, Mat(), Mat(), params);
svm.save(path.data());

printf("%s SVM construida com sucesso!\n",getNow().data());
}

//Funcao que realiza a leitura de um dicionario de caracteristicas
//em path
Mat readDictionary(string path){
    Mat dictionary;
    FileStorage fs(path, FileStorage::READ);
    fs["coinDictionary"] >> dictionary;
    fs.release();
    return dictionary;
}

int main(int argc, char *argv[]){
    //Fase 1. Criacao do dicionario de caracteristicas
    if(BUILD_DICTIONARY){
        buildDictionary("coinDictionary.yml", DICTIONARY_SIZE);
    }
    Mat dictionary = readDictionary("coinDictionary.yml");

    //Fase 2. Treinamento da SVM amostrando os descritores extraidos
    //pelo extrator baseado no dicionario
    BOWImgDescriptorExtractor descriptorExtractor(new
        SiftDescriptorExtractor(), new FlannBasedMatcher());
    descriptorExtractor.setVocabulary(dictionary);
    if(BUILD_SVM){
        trainSVM("coinSVM.yml", descriptorExtractor);
    }
    CvSVM svm;
    svm.load("coinSVM.yml");

    //Fase 3. Predicao de valores de moedas
```

```
//exemplo de entrada
int cTest=5; //moeda de 1 real
int lTest=1; //lado reverso
int id=35; //identificador unico da imagem no banco de imagens

printf("%s Testando moeda: %s\n",getNow().data(),
       getCoinValue(cTest).data());

char fileName[50];
sprintf(fileName, "%s%d%s%d%s%d%s", "train/", cTest, "/", lTest,
        "/", id, ".png");

Mat inputTest = readImage(fileName);
Mat maskTest = extractHoughCircleMask(inputTest);
Mat bowDescriptorsTest = extractDescriptorsUsingBOW(inputTest,
                                                    maskTest, descriptorExtractor);

float response = svm.predict(bowDescriptorsTest);
printf("%s Resposta obtida: %s - ", getNow().data(),
       getCoinValue((int)response).data());

if((int)response == cTest+lTest*10) {
    return 1; //sucesso
}

return 0; //falha
}
```