UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

SAMUEL SANCHES DE FREITAS

DETERMINAÇÃO DO VALOR TOTAL DE MOEDAS EM IMAGENS DIGITAIS

QUIXADÁ
2014

SAMUEL SANCHES DE FREITAS

DETERMINAÇÃO DO VALOR TOTAL DE MOEDAS EM IMAGENS DIGITAIS

Trabalho de Conclusão de Curso submetido à Coordenação do Curso Bacharelado em Sistemas de Informação da Universidade Federal do Ceará como requisito parcial para obtenção do grau de Bacharel.

Área de concentração: computação

Orientador Prof. Críston Pereira de Souza

QUIXADÁ

2014

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca do Campus de Quixadá

F936d Freitas, Samuel Sanches de

Determinação do valor total de moedas em imagens digitais / Samuel Sanches de Freitas. – 2014.

55 f. : il. color., enc. ; 30 cm.

Monografia (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Sistemas de Informação, Quixadá, 2014.

Orientação: Prof. Dr. Críston Pereira de Souza

Área de concentração: Computação

1. Redes Neurais (Computação) 2. Inteligência artificial 3. Computação Evolutiva I. Título.

CDD 006.3

SAMUEL SANCHES DE FREITAS

DETERMINAÇÃO DO VALOR TOTAL DE MOEDAS EM IMAGENS DIGITAIS

Trabalho de Conclusão de Curso submetido à Coordenação do Curso Bacharelado em Sistemas de Informação da Universidade Federal do Ceará como requisito parcial para obtenção do grau de Bacharel.

Área de concentração: computação

Aprovado em: _____ / junho / 2014.

BANCA EXAMINADORA

_____

Prof. Dr. Críston Pereira de Souza (Orientador)
Universidade Federal do Ceará-UFC

_____

Prof. MSc. Ricardo Reis Pereira
Universidade Federal do Ceará-UFC

_____

Prof. MSc. Ticiana Linhares Coelho da Silva
Universidade Federal do Ceará-UFC

Aos meus pais...

AGRADECIMENTOS

Agradeço à comunidade de software aberto pelas incontáveis horas de esforço e dedicação contribuídas ao ensino e à propagação do conhecimento, sem as quais este trabalho simplesmente não seria possível. Àqueles que alimentam a comunidade com suas dúvidas, experiências e contribuições nos clareiam a mente quando os livros não são capazes.

Agradeço o professor Críston Souza pela paciência na orientação e incentivo que tornaram possível a conclusão desta monografia.

Agradeço aos meus pais por todos os sacrifícios feitos ao longo dos anos. Ao meu irmão e tios. A todos estes, o meu "muito obrigado" pelo apoio incondicional, alegria, exemplo e estímulo.

Em nome de Erica Eva, Geovanny Filho, Rodrigo Matihara, Lucas Ismailly, Felipe Freitas, Daniel Araújo, Caio Balthazar, Ítalo Pessoa, Paulo Filipe, Emerson Fernandes e Glauco Aquino agradeço a todos os meus amigos, essenciais nessa etapa da minha vida.

Finalmente, à banca avaliadora, por prontamente aceitar fazer parte desse momento tão especial. Aos companheiros de trabalho, funcionários e desconhecidos que contribuíram sem perceber para que esse sonho se realizasse.

“O que você sabe não tem valor;
o valor está no que você faz
com o que sabe. ”
(*Bruce Lee*)

RESUMO

Um problema comum encontrado por pessoas com problema de visão é a contagem e o reconhecimento de moedas. Para facilitar essa tarefa esse trabalho propõe um algoritmo para fazer isso de forma automática. Para permitir a detecção e a contagem automática de moedas, é importante realizar uma contagem com uma baixa taxa de erro. Uma maneira de realizar esta contagem é aplicar um algoritmo de classificação. Neste trabalho testamos abordagem, e quais seriam as melhores entradas para o classificador. Concluímos que é possível obter uma taxa de acerto acima de 90 % utilizado árvores de decisão em um dos modelos propostos.

Palavras chave: Visão Computacional, Redes Neurais, Árvores de Decisão.

ABSTRACT

A common problem encountered by people with vision problems is the count and recognition of coins. To facilitate this task this paper proposes an algorithm to do this automatically. To enable detection and automatic coin counting, it is important to perform a count with a low error rate. One way to accomplish this count is to apply a classification algorithm. In this paper we test if this is a good approach. Conclude that it is possible to obtain an accuracy rate above 90% used decision trees in one of the proposed models.

Keywords: Computer Vision, Neural Networks, Decision Tree.

LISTA DE TABELAS

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

De acordo com pesquisa divulgada pelo site “Droider” [1], existem aproximadamente 19 milhões de smartphones ativados no Brasil, com uma previsão de crescimento de 44% em seis meses. Esses smartphones geralmente são utilizados para navegar pela web, ler notícias, interagir com redes sociais e etc. Por conta disso os smartphones são uma ferramenta cada dia mais presente no Brasil.

No Brasil, existem mais de 35,7 milhões de pessoas com deficiência visual de acordo com uma pesquisa do IBGE divulgada no site “G1” [2]. Onde 18,8% dos entrevistados tem dificuldade em ver mesmo com óculos. No Brasil já existem diversas tecnologias com a finalidade de facilitar uma inclusão social para essas pessoas, possibilitando uma maior acessibilidade para elas. Por exemplo, produtos com texto em braile e imagens em relevo. Em smartphones, as adaptações para deficientes visuais também ocorrem. Alguns exemplos são o aumento do contraste, inversão de cores e a navegação guiada.

Os avanços das técnicas de visão computacional estão possibilitando o surgimento de novas soluções para melhorar a acessibilidade para pessoas com deficiência visual. Um exemplo é a utilização de técnicas de visão computacional para ajudar o usuário daltônico a perceber as diferenças entre as cores [3]. Com os avanços na área de visão computacional, surgiram várias técnicas possibilitando um programa extrair e interpretar características relevantes em imagens. O objetivo dessa área é simular a visão humana em máquinas [4]. Porém, mesmo com essa evolução, a acessibilidade em relação à contagem de moedas não melhorou muito. Por exemplo, para a contagem de grandes quantidades de moedas, normalmente são utilizadas as seguintes técnicas: (i) utilização de máquinas para esse fim e (ii) a contagem manual. No entanto, essas técnicas geram os problemas discutidos a seguir. A contagem manual pode se tornar muito demorada, dependendo da quantidade de moedas, principalmente quando isso é feito por um deficiente visual. Já a utilização de máquinas específicas para a contagem de moedas normalmente tem um custo elevado. Por exemplo, em uma pesquisa foram encontradas as seguintes máquinas para contagem: “Semacon S-120 Coin Counter”, “Semacon S-40 Coin Counter” e “Cassida C800” com valores entre 229,00U$ e 530,00U$ no site OfficeMax [5]. Essas máquinas além de terem um custo elevado são pouco portáteis.

Portanto, baseado no contexto apresentado e nos problemas anteriores, esse projeto busca implementar um algoritmo eficiente para contagem do valor total de moedas em

imagens digitais usando visão computacional. Este algoritmo deve ser eficiente o bastante para permitir sua execução em dispositivos móveis (tablets e smartphones).

Com essa implementação, duas aplicações deste projeto são (i) permitir que deficientes visuais determinem facilmente o valor de moedas, e (ii) permitir que comerciantes determinem rapidamente o valor total em moedas em um caixa.

Esse documento está divido nas seguintes seções. Na Seção 1.1 são abordados os objetivos do projeto. Na Seção 2, é feita uma revisão bibliográfica sobre alguns conceitos importantes para o entendimento do projeto como visão computacional, aprendizagem de máquina e alguns artigos relacionados ao tema. Na Seção 3, são detalhados os procedimentos metodológicos utilizados para implementar e obter os dados para testes do algoritmo e como eles são avaliados. Na Seção 4, é detalhado o funcionamento do algoritmo de detecção das moedas. Na Seção 5, é detalhado o um modelo de classificação de moedas utilizando pontos de interesses. Na Seção 6, é detalhado um modelo de classificação de moedas utilizando o raio da moeda. Na Seção 8, são detalhados os resultados dos experimentos. Na Seção 9, são feitas as considerações finais deste trabalho.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

- Implementar um algoritmo de contagem do valor total de moedas em imagens digitais.

1.1.2 Objetivos Específicos

- Elaborar um algoritmo para localização de possíveis moedas em imagens digitais, usando visão computacional.
- Elaborar um algoritmo para classificação de moedas de acordo com seu valor monetário.
- Executar o algoritmo em dispositivos móveis.

2 REVISÃO BIBLIOGRÁFICA

O desenvolvimento do software de contagem de moedas em imagens digitais está relacionado com as áreas de visão computacional e aprendizagem de máquina. Durante essa seção serão abordados os principais conceitos relacionados à visão computacional, aprendizagem de máquina e avaliação de classificadores. São apresentados também os principais trabalhos relacionados encontrados na literatura.

2.1 Visão Computacional

Visão computacional é uma subárea da computação gráfica responsável pela criação de técnicas computacionais para simular a visão humana em máquinas [4]. Essa área permite que um computador realize extração de informações relevantes de imagens ignorando as informações não relevantes, para fazer o reconhecimento de objetos, orientações e tamanho [4].

Embora atualmente já existam várias pesquisas na área da visão computacional para aumentar a precisão para a extração de dados de imagens, um dos grandes desafios da visão computacional é fazer a extração de informações de imagens de forma eficiente como é feito por um humano, por conta da quantidade de ruídos que geralmente são encontrados nas imagens [4].

Para realizar o processamento de imagens, é necessário digitalizá-la. A imagem digital consiste em um número finito de pontos e cores, onde cada ponto é chamado pixel. Essa imagem é representada por uma matriz de N linhas e M colunas [6]. Os pixels são divididos em canais para a representação das cores. Por exemplo, em uma imagem monocromática os pixels têm apenas um canal, mas em um a imagem colorida eles terão vários canais. Esse número de canais dependerá da composição do sistema utilizado para a representação das cores, como o RGB, HSV e outros [6].

2.1.1 Histograma

Os histogramas são ferramentas para processamento de imagens com várias aplicações. Entre essas aplicações é possível destacar: melhora na definição da imagem, compressão e segmentação de imagens. Os histogramas são determinados a partir de valores de intensidade dos pixels [7]. O histograma de uma imagem é um conjunto de números

indicando o percentual de pixels naquela imagem que apresentam um determinado nível de uma característica escolhida, como a escala de cinza, brilho, saturação ou outras características. Estes valores são normalmente representados por um gráfico de barras que fornece para cada nível de cinza o percentual de pixels correspondentes na imagem. Através da visualização do histograma de uma imagem obtemos uma indicação de sua qualidade quanto ao nível de contraste e quanto ao seu brilho médio (se a imagem é predominantemente clara ou escura) [4].

O histograma também pode ser definido em imagens coloridas. Mas nesse caso, a imagem é decomposta de alguma forma em componentes RGB e para cada componente é calculado o histograma correspondente [8]. Um exemplo de comparação de histogramas pode ser visto na Figura 1. Na imagem à esquerda é mostrada uma imagem e seu respectivo histograma. Nessa imagem foi feita uma equalização do histograma. A equalização do histograma é uma técnica onde é feita uma redistribuição de uma determinada característica dos valores no histograma de forma que ele fique mais uniforme [8]. Nesse caso foi feita uma equalização do histograma baseada na distribuição de cinza na imagem, fazendo a imagem ficar mais nítida. O resultado dessa imagem pode ser visto na imagem à esquerda.

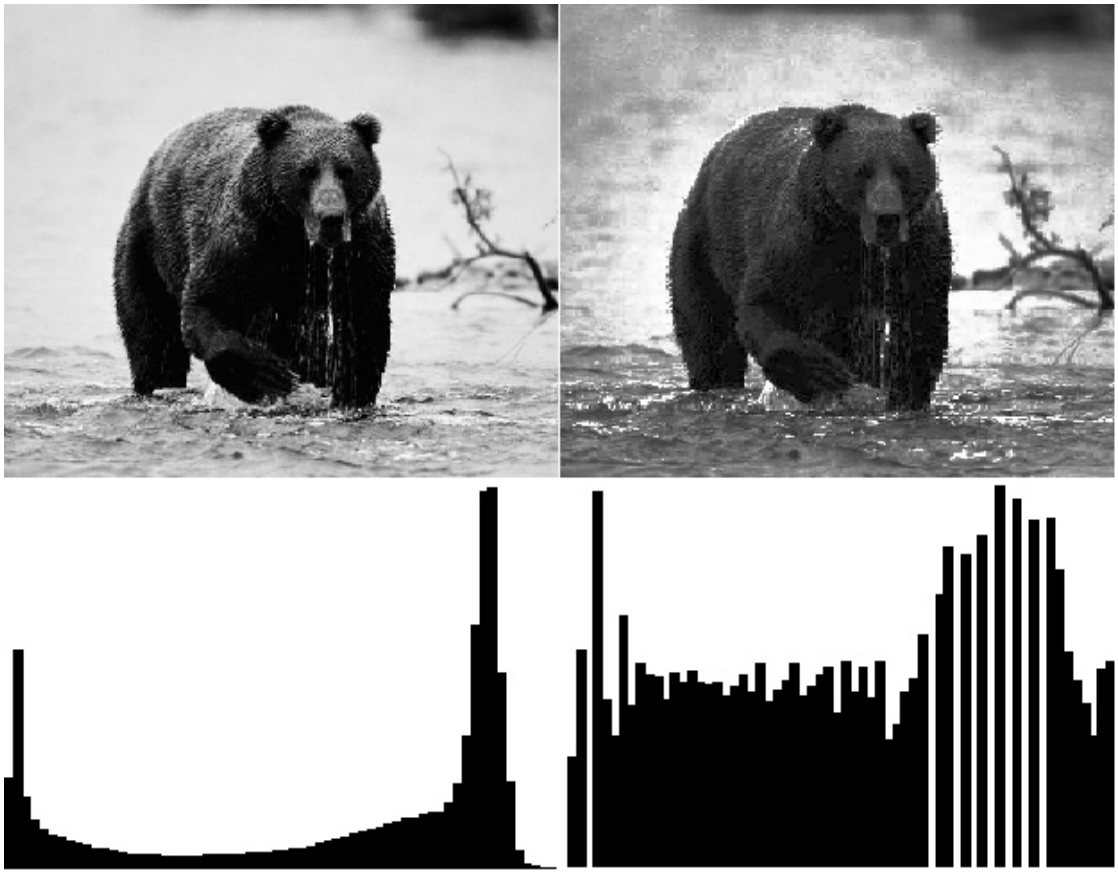

Figura 1 - No topo a imagem base e abaixo dela o histograma calculado. Fonte: [8].

2.1.2 Segmentação

A segmentação é o processo de dividir uma imagem em regiões ou objetos distintos. Esse processo geralmente é feito baseado nas características do objeto ou região que será segmentado, como contraste, brilho e cor. O nível de detalhamento da segmentação depende do objeto que será segmentado, resolução da imagem e do processo que será executado [8]. As técnicas de segmentação utilizadas nesse trabalho serão: (i) segmentação por detecção de borda e (ii) segmentação por corte [8].

A segmentação por bordas é baseada na detecção de bordas na imagem analisada. Essas bordas são caracterizadas por uma mudança na intensidade dos pixels. Para isso são utilizados detectores de borda para encontrar esse tipo de variação de pixels, e quando esses pixels estão próximos eles podem ser conectados formando uma borda ou contorno e com esse contorno é possível definir um objeto [4]. Um dos principais detectores de borda é o algoritmo Canny [9]. Esse algoritmo satisfaz três critérios de qualidade: (i) baixa taxa de erros, (ii) boa precisão em relação à distância entre a borda real e a borda detectada e (iii) apenas uma borda detectada por região [9]. Na Figura 2, é possível ver um exemplo de saída do algoritmo Canny [9].

O segundo tipo de segmentação usada é a segmentação por corte. Esse tipo de segmentação é simples de ser implementada e rápida em termos computacionais [4]. Nela geralmente são utilizadas propriedades intuitivas para criar a imagem segmentada. A segmentação por corte divide a imagem em regiões baseadas em propriedades escolhidas. Um exemplo desse tipo de segmentação é baseado no histograma da imagem, verificando quantas regiões existem (picos e vales) [8]. E com essa informação segmentar a imagem. Um exemplo pode ser visto na Figura 3. Utilizando essa técnica é possível definir várias regiões e com isso é possível ter uma imagem com diferentes propriedades e com os ruídos minimizados [8]. A Figura 3 exemplifica a segmentação por corte. Nesse exemplo a imagem processada será a imagem à esquerda, onde sistema de cores possui apenas um canal e os valores desse canal podem variar de 0 a 255. Então baseado em uma análise do histograma dessa imagem exibido ao centro, foi escolhido um valor de limiar para executar a segmentação. Baseado no valor escolhido foi feita uma limiarização da imagem, onde todos os pixels que possuem o valor menor que o limiar receberá o valor 0 e todos os pixels com valor acima receberá o valor 255, resultando na imagem binarizada à esquerda.

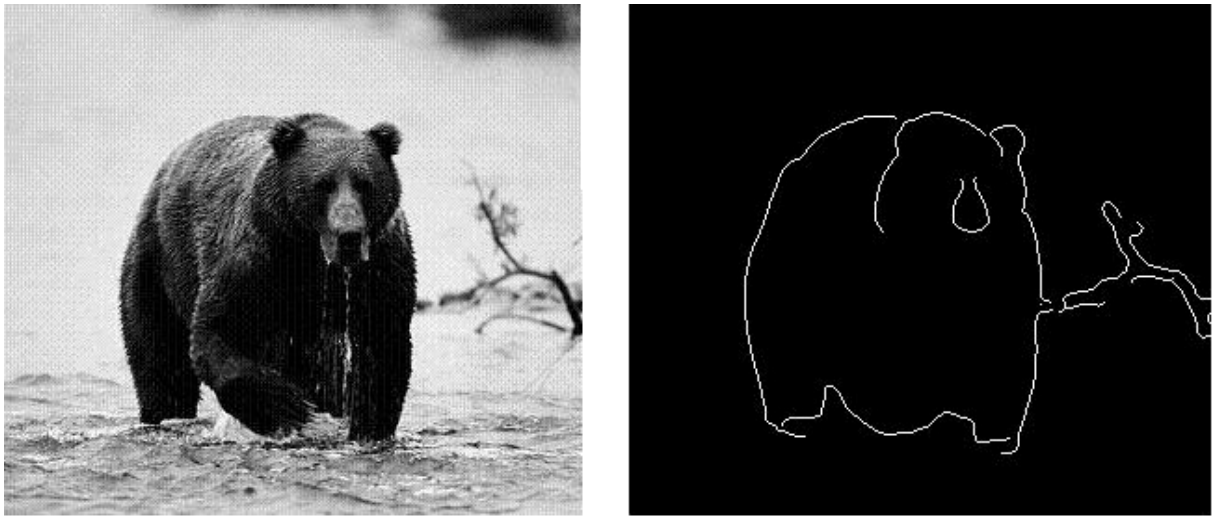

Figura 2 - Aplicação do Canny. Fonte: [8].

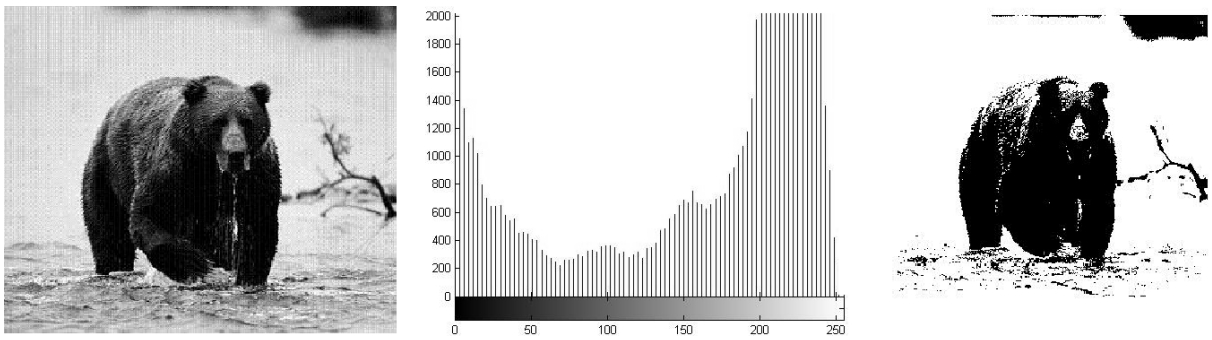

Figura 3 - Segmentação de imagem utilizando a técnica de corte. À esquerda a imagem em nível de cinza, ao centro o histograma com o ponto de corte e à direita a imagem binarizada. Fonte: [8].

2.1.3 Pontos de interesse em imagens.

Pontos de interesse, em visão computacional, são muito utilizados em problemas onde é necessário o reconhecimento de imagens, rastreamento e reconstrução 3D [7]. Esses pontos são baseados na ideia de analisar pontos-chave na imagem, em vez de analisar toda a imagem.

Para que as análises de imagens utilizando pontos de interesse sejam precisas, é necessário que sejam detectados vários pontos com características distintas e estáveis, como linhas, círculos e cores [7]. Por exemplo, ao analisar o alfabeto, a letra **A** pode ser descrita como duas linhas inclinadas na vertical e uma linha na horizontal cruzando essas duas linhas. Enquanto a letra **O** pode ser descrita como uma elipse. Para o reconhecimento desses pontos, podem ser utilizados vários algoritmos como o SIFT, que será explicado na próxima seção.

2.1.4 Reconhecimento de pontos de interesse usando o SIFT.

O SIFT (Scale-Invariant Feature Transform) [10], é um algoritmo para obtenção de pontos de interesse em imagens. O seu funcionamento é dividido em fases, onde as imagens são analisadas e adaptadas individualmente. O algoritmo começa por identificar os locais candidatos a pontos-chave, como máximos e mínimos locais no histograma de uma região da imagem. Depois é feita uma interpolação dos pontos com baixo e alto contraste. Em objetos com bordas circulares, como no caso das moedas, primeiro é localizada melhor circunferência na borda, nessa circunferência o ponto de interesse será o centro. Os outros pontos são rejeitados para melhorar a precisão. Os pontos que sobrevivem à filtragem são atribuídos uma orientação, baseada nas direções dominantes dos gradientes espaciais. Após a atribuição de orientação, cada ponto chave poderá ser calculado em relação a outro ponto, escala e orientação. Finalmente o cálculo dos descritores é feito para cada ponto dividindo o espaço entorno do ponto chave em uma grade, depois é calculado o histograma de cada quadrado da grade, concatenando os histogramas em um vetor. Cada elemento desse vetor é considerado um ponto de interesse [10].

Na Figura 4, é demonstrado uma aplicação do algoritmo SIFT [10] para identificar os pontos de interesse em duas imagens de uma placa, e depois compara com os pontos de interesse identificados nas imagens.

Figura 4 - Exemplo de identificação de pontos de interesse em imagens diferentes. Fonte: [11].

2.1.5 OpenCV

O OpenCV é uma biblioteca de funções C++ desenvolvida e mantida pela Intel, e possui mais de 500 funções [12]. Foi construída com o objetivo de tornar a visão computacional mais acessível à programadores em áreas como interação humano-computador em tempo real e robótica. A biblioteca, seu código fonte e os executáveis estão otimizados para processadores Intel. Um programa utilizando OpenCV, ao ser executado invoca uma DLL (Dynamic Linked Library) que detecta o tipo de processador e carrega uma DLL otimizada para o processador. Junto com o pacote OpenCV é fornecido também a documentação e um conjunto de exemplos.

O OpenCV está dividido em cinco grupos de funções: Processamento de imagens; Análise estrutural; Análise de movimento e rastreamento de objetos; Reconhecimento de padrões e Calibração de câmera e reconstrução 3D [12].

2.2 Aprendizagem de máquina.

A aprendizagem de máquina é uma área onde o objetivo é o desenvolvimento de técnicas computacionais para a construção de sistemas capazes de adquirir conhecimento de forma automática [13].

O tipo de aprendizagem a ser utilizado neste projeto será a classificação utilizando a aprendizagem supervisionada, onde é dado um conjunto de exemplos para treinamento, um algoritmo gera como saída um classificador. Esse classificador pode ser usado para classificar exemplos ainda não classificados, de forma que a cada novo exemplo o classificador possa retornar uma classe com uma maior precisão [13]. Um exemplo de classificação é um conjunto de objetos com classes pré-definidas, onde cada objeto possui um atributo representando a classe e outros atributos que serão avaliados para fazer a classificação. Nesse conjunto de objetos o objetivo do classificador é predizer o atributo classe dos objetos, com base nos outros atributos.

Os classificadores que serão utilizados nesse projeto são: redes neurais artificiais (RNAs) e árvores de decisão. Esses classificadores serão explicandos nas próximas seções.

2.2.1 Redes Neurais Artificiais.

As Redes Neurais Artificiais (RNA) são modelos matemáticos baseados em estruturas neurais biológicas e que tem a capacidade classificação adquirida por meio de aprendizado [13]. Uma RNA é equivalente a um grafo orientado, onde os nós são os neurônios e as arestas são as conexões. Em uma RNA as informações são coletadas pelos neurônios, onde ocorre um processamento para definir pesos para informação até o próximo neurônio.

Cada conexão tem um peso, que será usado para generalizar o padrão aprendido pela RNA. Em cada processo de aprendizagem, os valores dos pesos são ajustados para que o valor retornado pela RNA se aproxime do valor esperado.

Para fazer o ajuste nos neurônios da rede neural é utilizado um algoritmo chamado 'backpropagation', onde cálculo do ajuste de pesos inicia na última camada de neurônios e termina nas camadas mais próximas da camada inicial.

Na Figura 5 é mostrado um exemplo de uma RNA, onde a entrada E é envidada para os neurônios da primeira camada, e de acordo como o processamento feito pelo neurônio a informação processada é enviada para as outras conexões. Após todo o processamento é retornada a saída S pela RNA.

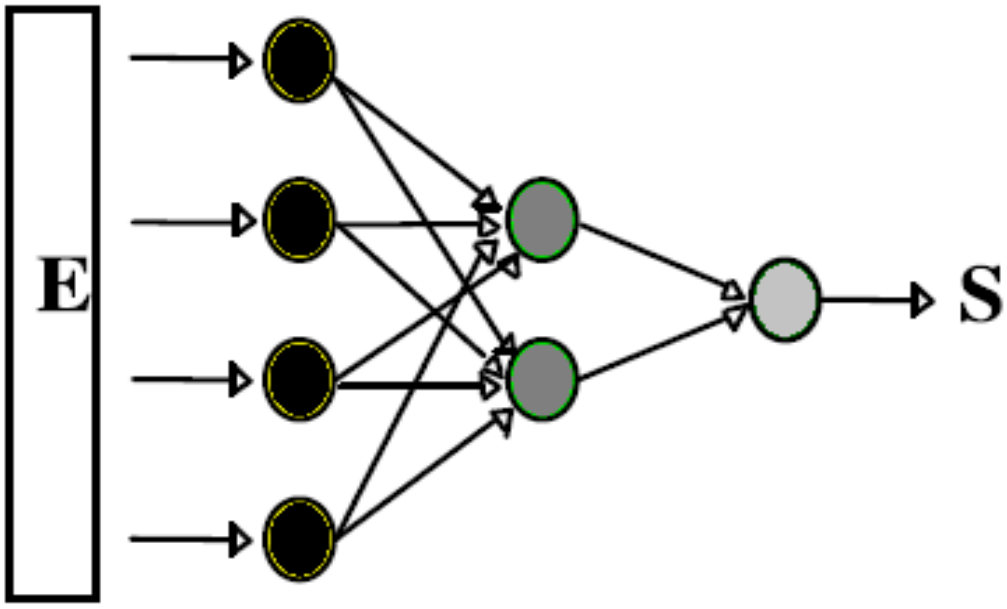

Figura 5 – Exemplo de uma rede neural. Fonte: O Autor.

2.2.2 Árvore de decisão.

Uma árvore de decisão é uma representação de um classificador utilizada por sistemas de aprendizado de máquina para dar ao agente classificador a capacidade de aprender e tomar decisões [13]. Uma árvore de decisão utiliza a estratégia divisão e conquista, ou seja, um problema complexo é decomposto em subproblemas mais simples e recursivamente a mesma estratégia é aplicada aos subproblemas [14].

Uma árvore de decisão é organizada em nós, ramos e folhas, onde os nós representam os atributos. Os ramos criados a partir desses nós representam os possíveis valores desse atributo. E as folhas que representam as diferentes classes de um conjunto de exemplos de treinamento. Na árvore de decisão cada caminho representa uma regra de classificação.

O aprendizado das árvores de decisão ocorre na medida em que um algoritmo observa suas interações com os exemplos, estruturando a árvore de forma que: (i) cada nó não-folha seja rotulado com o nome de um dos atributos; (ii) os ramos saindo de um nó interno sejam rotulados com valores do atributo naquele nó; (iii) cada folha é rotulada com uma classe, a qual é a classe prevista para exemplos que pertençam àquele nó folha [13].

A Figura 6 apresenta um exemplo de árvore de decisão com dados para classificar se uma pessoa pode ou não comprar um computador. Nesse caso as possíveis classes são “sim” e “não”. Os atributos são: “Idade”, “Estudante” e “Credito”. O atributo “Crédito”, por exemplo, possui as seguintes possibilidades: “alto” ou “baixo”. Nessa árvore para obter a classificação, basta ir aplicando os valores que se quer saber a classificação nas ramificações correspondentes.

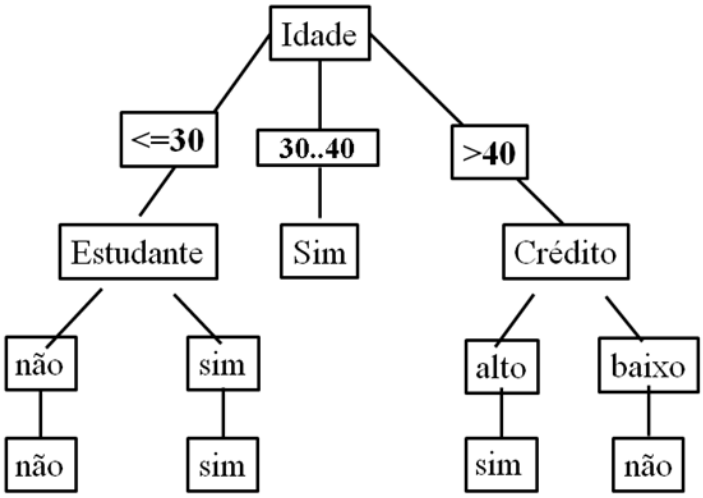

Figura 6 - Árvore de decisão para comprar computador. Fonte: [15].

2.2.3 Avaliação de algoritmos de classificação.

Após o treino de um classificador é utilizado um conjunto de testes para avaliar a sua eficiência. Porém o problema de avaliar a eficiência está na interpretação dos resultados. Um dos métodos mais comuns para reduzir esse problema é a utilização da análise ROC (Receiver Operating Characteristc) e usando métricas como f-score [16].

A análise ROC é feita sobre um gráfico ROC, que é um gráfico bidimensional onde os eixos X e Y representam os falsos positivos (TFP) e verdadeiros positivos (TVP) [17]. Cada exemplo classificado será representado por um ponto no gráfico ROC e para cada classificador será feito um gráfico ROC. Com esses gráficos será possível comparar os classificadores e analisar qual se adapta melhor ao problema. Além do gráfico ROC, podem ser utilizadas outras métricas para melhorar a análise dos classificadores, como a precisão, que é a porcentagem de objetos classificados corretamente, e a cobertura, que é a porcentagem da classe de interesse [18]. Essas medidas são calculadas usando a matriz de confusão mostrada na Figura 7.

		Existe uma instância do objeto na imagem	
		Positivo (SIM)	Negativo (Não)
O algoritmo detectou uma instância do objeto na imagem	Positivo (SIM)	Verdadeiro Positivo (VP) Detecção correta	Falso Positivo (FP) Erro na detecção
	Negativo (Não)	Falso Negativo (FN) Erro na detecção	Verdadeiro Negativo (VN)

Figura 7 – Matriz de confusão. Fonte: [19].

Na análise ROC são comuns os seguintes termos: Verdadeiro Positivo (VP), Falso Positivo (FP), Falso Negativo (FN) e Verdadeiro Negativo (VN). Esses termos nesse projeto serão definidos como:

- Verdadeiro Positivo (VP): ocorre quando o objeto classificado pelo classificador recebe a classificação correta.
- Falso Positivo (FP): ocorre quando o objeto é classificado como “moeda” pelo classificador, mas não é uma moeda, ou quando o classificador classificar um valor incorreto para a moeda.

- Falso Negativo (FN): ocorre quando o objeto é classificado como “não moeda”, sendo uma moeda.
- Verdadeiro Negativo: ocorre quando o objeto é classificado como “não moeda”, não sendo uma moeda.

Para os cálculos das métricas serão utilizadas as seguintes fórmulas:

$$\text{Taxa de verdadeiros positivos } (TVP) = VP/(VP + FN)$$

$$\text{Taxa de falsos positivos } (TFP) = FP/(FP + VN)$$

$$\text{Precisão} = VP/(VP + FP)$$

$$\text{Acurácia} = (VP + VN)/(VP + FP + VN + FN)$$

$$\text{Cobertura} = VP/(VP + FN)$$

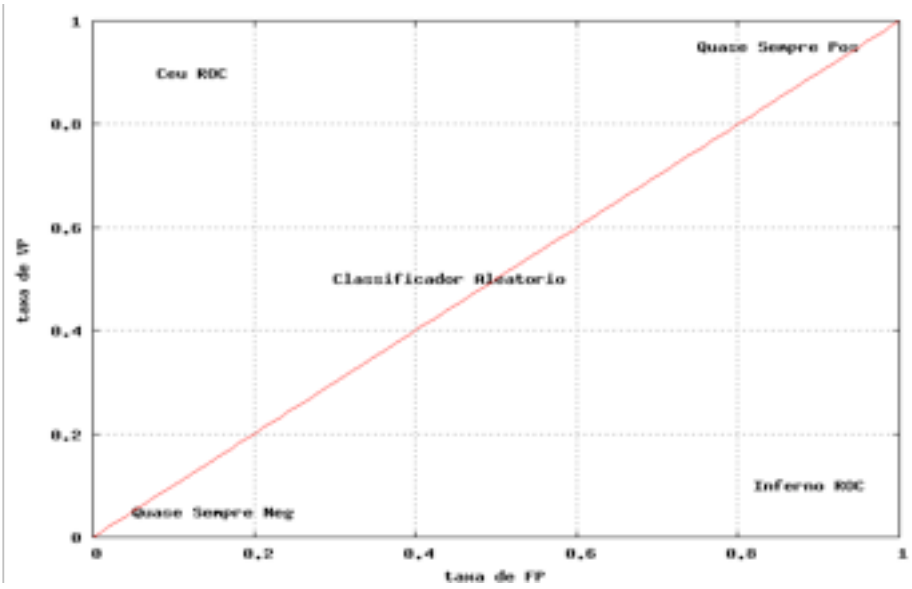

Figura 8 - Exemplo de um gráfico ROC. Fonte: [18].

A Figura 8 é um exemplo de gráfico ROC, nela é possível identificar cinco regiões importantes: Céu ROC; Inferno ROC; Quase Sempre Neg e Quase Sempre Pos, e área de classificadores aleatórios.

O “Céu ROC” ocorre quando o classificador sempre retorna o resultado esperado. Nesse quadrante ficam os pontos dos classificadores onde a classificação quase sempre é perfeita.

O “Inferno ROC” acontece quando o classificador sempre erra o resultado esperado. Nesse quadrante ficam os pontos que representam os classificadores que nunca acertam a classificação.

O quadrante “Quase Sempre Neg”, é a região que representa os classificadores que quase sempre classificam os exemplos como negativos. Assim o número de exemplos classificados errados é baixo, assim como o número de exemplos classificados corretamente.

O quadrante “Quase Sempre Pos”, é a região que representa os classificadores que quase sempre classificam os exemplos como positivos. Assim quase todos os exemplos positivos são classificados corretamente e quase todos os negativos são classificados incorretamente.

Outra maneira de avaliar algoritmos de classificação que será usada é a métrica f-score. Ela é uma medida que considera tanto a precisão e a cobertura do teste para calcular a pontuação. F-score é uma métrica que retorna a média harmônica entre a precisão e a cobertura baseada na fórmula a baixo:

$$\text{F-Score} = 2x\ (\ precisão\ X\ cobertura\ )/(precisão + cobertura)$$

2.3 Trabalhos Relacionados.

No trabalho de Zambanini e Kampel [20], os autores implementam um algoritmo classificador eficiente computacionalmente para reconhecimento de moedas antigas. Já que para esse caso, o uso de classificadores para esse reconhecimento seria muito lento por conta da grande quantidade de moedas a serem comparadas. Outro desafio para esse tipo de classificador é a rotação das moedas, já que o classificador utilizado nesse trabalho tem uma tolerância baixa para essa rotação. A estratégia para a resolução desse problema foi o treinamento de um classificador usando como entrada pontos de interesse produzidos por uma adaptação do algoritmo SIFT [20]. Um exemplo de saída desse algoritmo pode ser visto na Figura 9. Assim, a quantidade de dados analisados pelo classificador é reduzida, e como a distância dos pontos de interesse em relação ao centro será parecida no padrão aprendido pelo classificador e o detectado na imagem, o problema da rotação da imagem será parcialmente resolvido.

Já no artigo [21], o problema abordado é a classificação de imagens circulares utilizando texturas. O problema dessa classificação são os casos onde as imagens classificadas estão rotacionadas, o que causa uma acurácia menor por conta da mudança na posição das texturas. Um exemplo dessa mudança pode ser observado na Figura 10. Para resolver esse problema, os autores criaram um algoritmo que reduz as diferenças que ocorreriam na comparação das imagens rotacionadas. Para remover essas diferenças os autores fazem uma ordenação dos pixels. Para ordenar os pixels, primeiro a imagem é binarizada utilizando um limiar que é escolhido de acordo com a moda, media e mediana de pixels no histograma. Depois a imagem é dividida em nove quadrantes, e os quadrantes são ordenados de forma decrescente de acordo com a quantidade de pixels que cada quadrante possui.

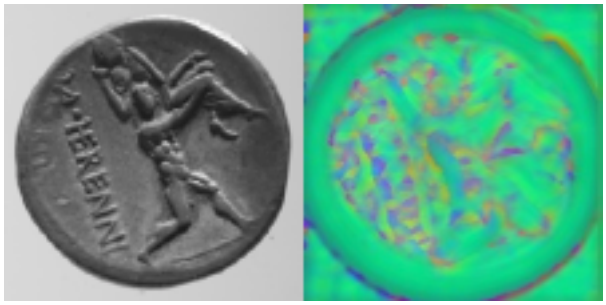

Figura 9 – À esquerda a imagem original da moeda e a direita a imagem após aplicar o algoritmo para detecção dos pontos de interesse. Fonte: [20].

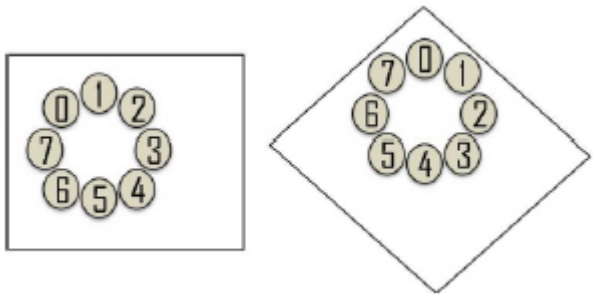

Figura 10 – À esquerda a representação de uma imagem dividida em quadrantes e a direita a imagem rotacionadas. Fonte: [21].

Outro artigo relacionado é o [22], onde os autores implementam um algoritmo para classificação de moedas utilizando como entrada, valores detectados pelo algoritmo SIFT[10]. Esse algoritmo é executado em duas fases. Na primeira fase o algoritmo converte a imagem para uma escala de cinza, e utilizando a diferença entre os valores máximo e mínimo da intensidade de cinza ele binariza a imagem. Após isso, baseado na borda circular recorta a moeda do restante da imagem. Após esse corte é aplicado o algoritmo SIFT [10] para

detecção dos pontos de interesse da imagem. Com esses pontos é criando um histograma de tamanho 8, utilizando as posições dos pontos na imagem, desconsiderando os pontos com baixo contraste e que estejam nas bordas. O histograma calculado é enviado para o classificador.

2.4 Comparação com trabalhos relacionados

Diversos autores utilizam abordagens diferentes para o reconhecimento de moedas. No entanto, este trabalho se destaca, pelo reconhecimento de múltiplas moedas na mesma imagem e a capacidade para ser portável para dispositivos móveis.

Autores como Zambanni e Kampel (2011), Saiphullah et al (2011) e Huber-Mörk et al (2011), utilizam abordagem baseadas em pontos de interesse utilizando o algoritmo SIFT [10]. No trabalho aqui desenvolvido, em uma das abordagens propostas também é utilizado o algoritmo SIFT [10], e para resolver problemas causados pela rotação das moedas também é utilizada uma abordagem semelhante à utilizada no trabalho de Saiphullah et al (2011), dividindo a imagem em regiões e analisando-as separadamente. No entanto, como no trabalho atual são utilizadas moedas com muitas semelhanças, a abordagem utilizando pontos de interesse se torna ineficiente como demonstrado na Seção 8.3.

Para a extração de moedas, é utilizada uma abordagem semelhante à de Huber-Mörk et al (2011), binarizando a imagem e tentando localizar as moedas baseando-se em seu formato circular.

3 PROCEDIMENTOS METODOLÓGICOS.

Para o desenvolvimento do algoritmo de contagem de moedas em imagens digitais, o projeto foi dividido em quatro fases: (i) coleta e rotulação de imagens para o treinamento e avaliação dos classificadores; (ii) identificação de atributos gráficos dos objetos na imagem (estas características devem facilitar a diferenciação de moedas em relação a outros objetos na imagem); (iii) implementação de um algoritmo para a classificação do valor monetário de moedas, e; a (iv) preparação os algoritmos para execução em dispositivo móvel.

Na primeira fase, são coletadas as imagens, para o treinamento e avaliação dos classificadores. Estas imagens são coletadas através de um dispositivo móvel (celular ou

tablet). Um exemplo de imagem coletada pode ser observado na Figura 11. Além disso, estas imagens são rotuladas. Para o classificador, as imagens são rotuladas de acordo com o valor monetário de cada moeda.

Figura 11 – Exemplo de amostra imagem coletada. Fonte: O Autor.

Na segunda fase ocorre a implementação do algoritmo em C++, utilizando biblioteca OpenCV [12], para detecção de possíveis moedas em imagens. Ele será construído baseado na análise das imagens para determinar quais características gráficas podem ser utilizadas como diferencial entre as moedas e o restante da imagem. A imagem processada será suavizada para reduzir os ruídos como excesso de luminosidade e granularidade. Após o processamento, as partes da imagem detectadas como possível moeda serão recortadas da imagem e processadas separadamente. Com isso serão removidas partes desnecessárias da imagem para evitar um processamento desnecessário.

Combinado as possíveis moedas recortadas da imagem, é identificado também um marcador para servir de base para calcular o tamanho das moedas. Esse marcador é identificado com base em uma cor pré-definida. Utilizando o tamanho real e o tamanho em pixels do marcador é feito um cálculo para converter o tamanho em pixels de uma moeda no tamanho real aproximado dessa moeda.

Na terceira fase é implementado um algoritmo para a classificação do valor de cada moeda detectada. Esse classificador é treinado usando imagens de moedas rotuladas (na

primeira fase) com o valor das moedas. O classificador receberá como entrada o valor real aproximado do raio da moeda, a saturação e a cor predominante e utiliza esses valores para classificar a moeda. Para avaliar a precisão do classificador é utilizada a matriz de confusão.

Na quarta fase é feita uma prova de conceito, embarcando o código em um dispositivo móvel para avaliar o tempo de execução dos algoritmos nestes dispositivos. Para portar é criada uma API em C++ para manipulação dos algoritmos de classificação. Esse código é compilado para um dispositivo móvel com o sistema operacional IOS.

4 RECORTE DE MOEDAS.

Para extrair as possíveis moedas da imagem é feita uma segmentação por corte, binarizando a imagem como é mostrado na **Figura 12**. Após essa segmentação são aplicadas duas operações morfológicas na imagem. A primeira é uma erosão e logo após uma dilatação dos objetos na imagem. A erosão é feita para melhor definir os objetos na imagem com pode ser observado na **Figura 13**. Após a erosão é feita uma dilatação para remover pequenos ruídos causados ou deixados pela erosão da imagem. O resultado pode ser visto na **Figura 14**. Agora com os objetos bem definidos na imagem é utilizado um algoritmo do OpenCV chamado HoughCircles para localizar os possíveis círculos na imagem. Como esse algoritmo é muito flexível e não é muito preciso foram necessárias as etapas anteriores para reduzir o aparecimento de falsos círculos nas imagens e otimizar o desempenho. Para cada círculo encontrado é gerada uma imagem para ser utilizada como máscara para extração da possível moeda representada pelo círculo. Um exemplo dessa máscara pode ser visto na **Figura 15**, onde foi gerada uma imagem com todas as máscaras para extração das imagens. Usando essa máscara é possível fazer a extração da possível moeda da imagem original, como na **Figura 16**. Após esse recorte a imagem é processada para a determinação do valor da moeda. A **Figura 17**, demonstra as etapas deste processo utilizando um diagrama de atividades.

Figura 12 – Imagem binarizada contendo as moedas. Fonte: O Autor

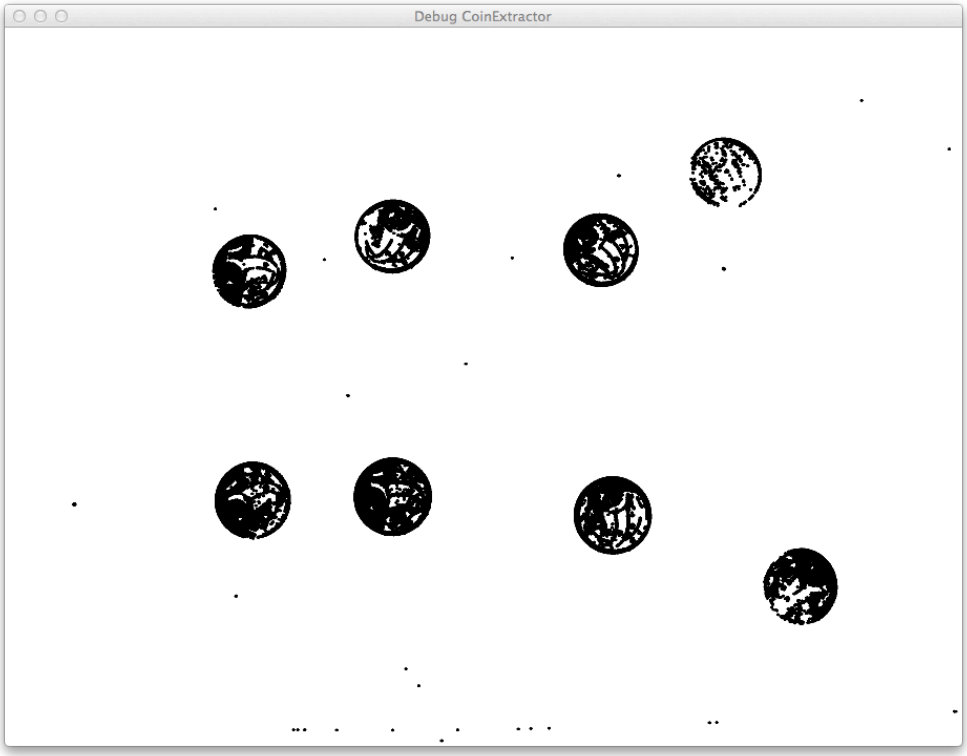

Figura 13 – Resultado da operação morfológica de erosão. Fonte: O Autor

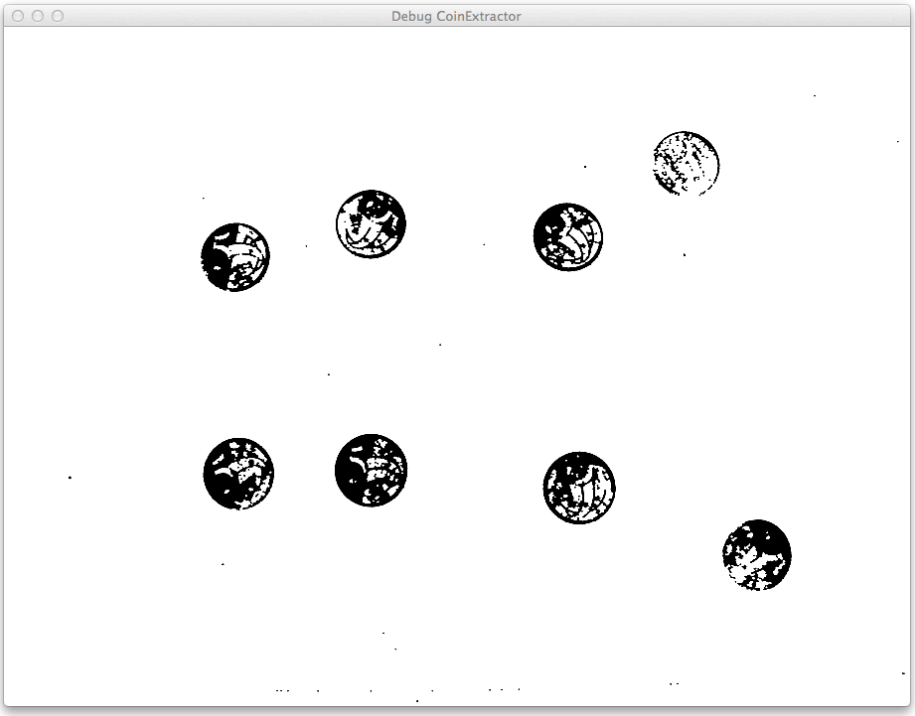

Figura 14 – Resultado da operação morfológica de dilatação. Fonte: O Autor

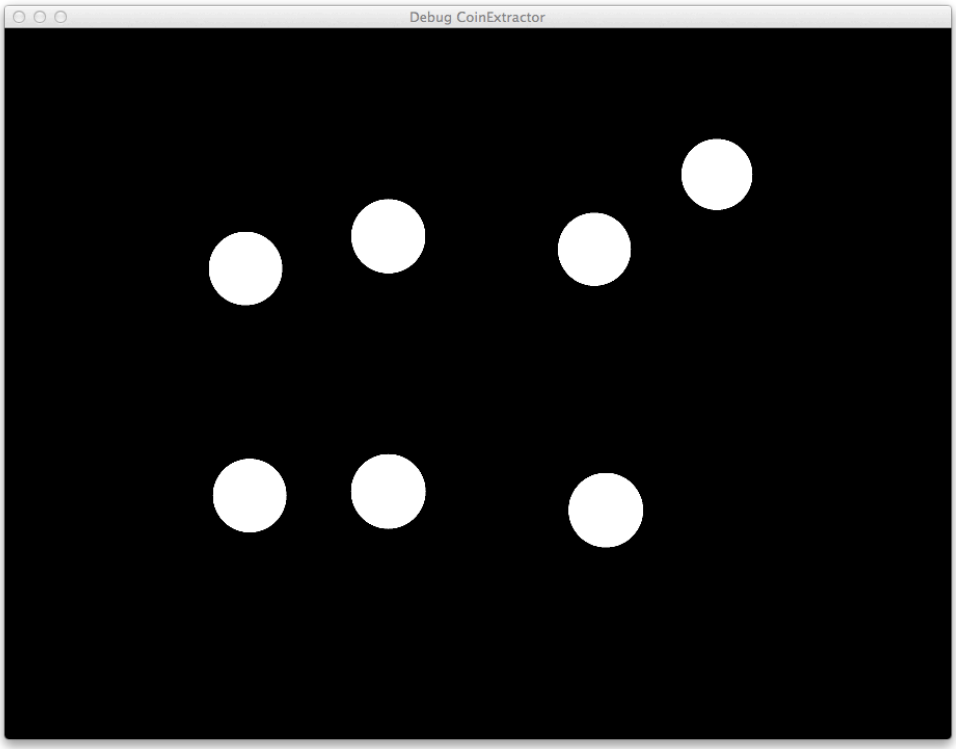

Figura 15 – Exemplo de máscara para extração de moedas da imagem. Fonte: O Autor

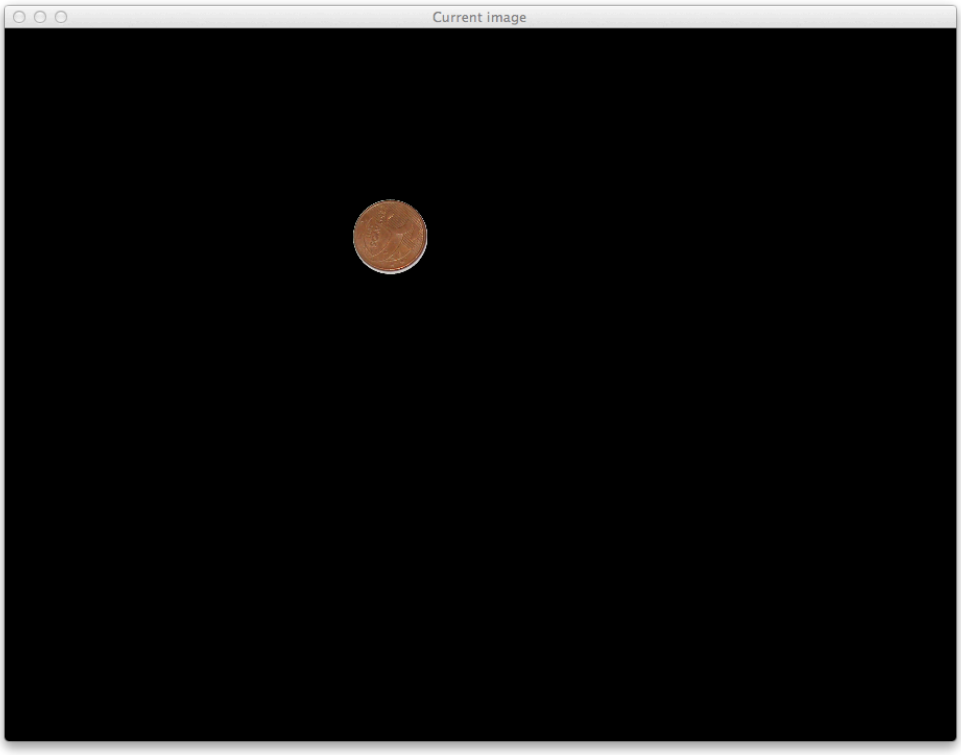

Figura 16 – Moeda recortada utilizando uma máscara. Fonte: O Autor

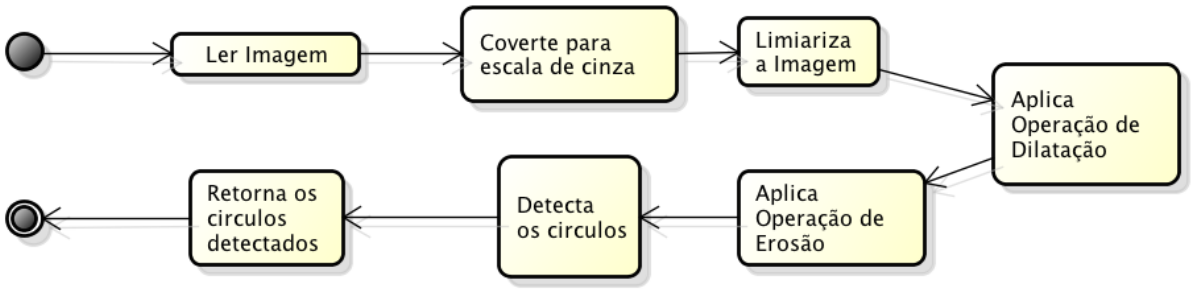

Figura 17 – Diagrama de Atividade do Recorte de moedas. Fonte: O Autor.

5 RECONHECIMENTO DO VALOR DE MOEDAS UTILIZANDO PONTOS DE INTERESSE – MODELO A.

Um abordagem utilizada para a classificação das moedas é baseada em pontos de interesse. Nessa abordagem, para cada recorte da imagem que pode ser uma moeda, é executado o algoritmo SIFT[10] para identificar os pontos de interesse como pode ser observado na **Figura 18**.

Com os pontos de interesse detectados são calculadas as distâncias de cada ponto de interesse ao centro da moeda. Como a distância do observador até a moeda pode variar,

podemos ter uma variação na distância dos pontos de interesse ao centro. Para reduzir essa variação, os valores das distâncias são divididos pelo raio da moeda. A fórmula para esse cálculo é apresentada abaixo, onde P é a distância do ponto ao centro no eixo x (em pixels), C a distância do ponto ao centro no eixo y (em pixels) e R é o raio da moeda (em pixels).

$$\text{Fórmula para cálculo das distâncias (FCD)} = \left(\frac{\sqrt{P^2 + C^2}}{R} \right)$$

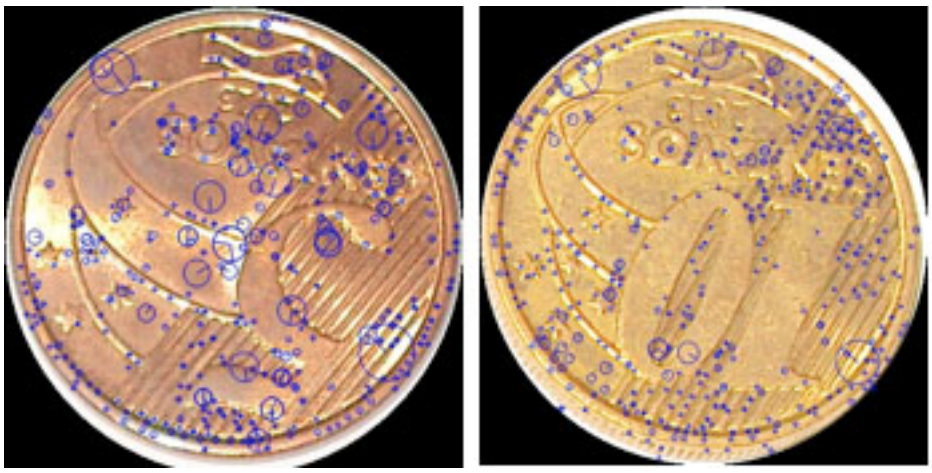

Figura 18 – Moedas com os pontos de interesse detectados. Fonte: O Autor

Com os valores das distâncias calculados, é criado um histograma com vinte intervalos largura 0,05 conforme pode ser visto na **Figura 19**. Essa quantidade de intervalos do histograma foi determinada empiricamente. Cada intervalo desse histograma representa uma região da moeda como é mostrado na **Figura 20**, e o valor dessa região é o número de pontos de interesse existentes nessa região. Como as variações das distâncias estão organizadas em um histograma, essa abordagem funciona mesmo quando a moeda é rotacionada pois a rotação das moedas mantém os pontos na mesma região.

Outro problema que pode ocorrer é uma grande variação do número de pontos de interesse detectados na moeda devido à variação da distância da câmera até a moeda, visto que essa variação pode variar a resolução das moedas recortadas. Para reduzir essa interferência no histograma é feita uma normalização para que cada faixa do histograma passe a ter a porcentagem de pontos em cada faixa. Assim como o aumento ou a redução de pontos de interesse é proporcional em toda a imagem, espera-se que a proporção de pontos em cada região seja mantida. Na **Error! Reference source not found.**, é ilustrado a sequência de passos para a execução do modelo utilizando um diagrama de atividades.

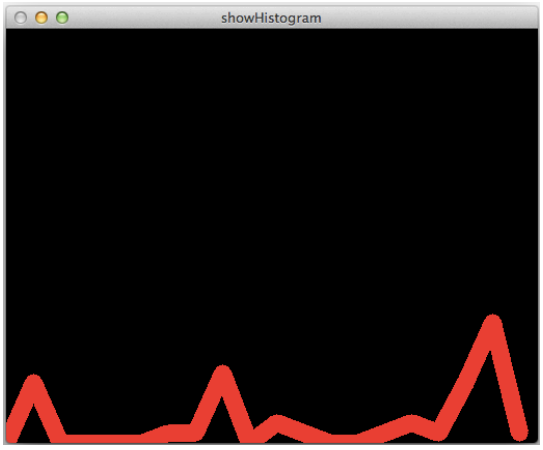

Figura 19 – Desenho de um histograma criado usando as distâncias calculadas. Fonte: O Autor

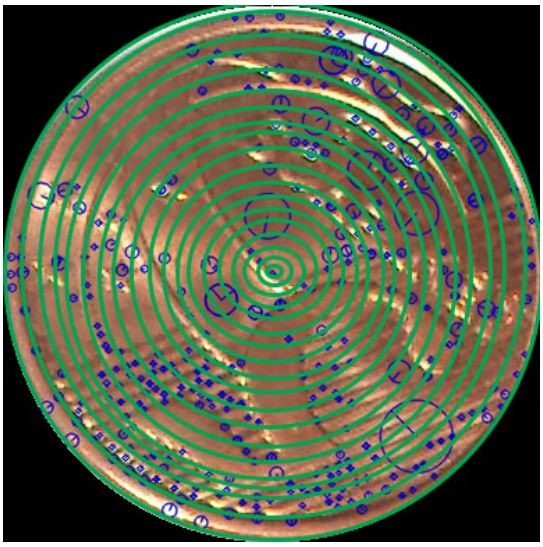

Figura 20 – Exemplo da divisão de regiões na moeda. Fonte: O Autor

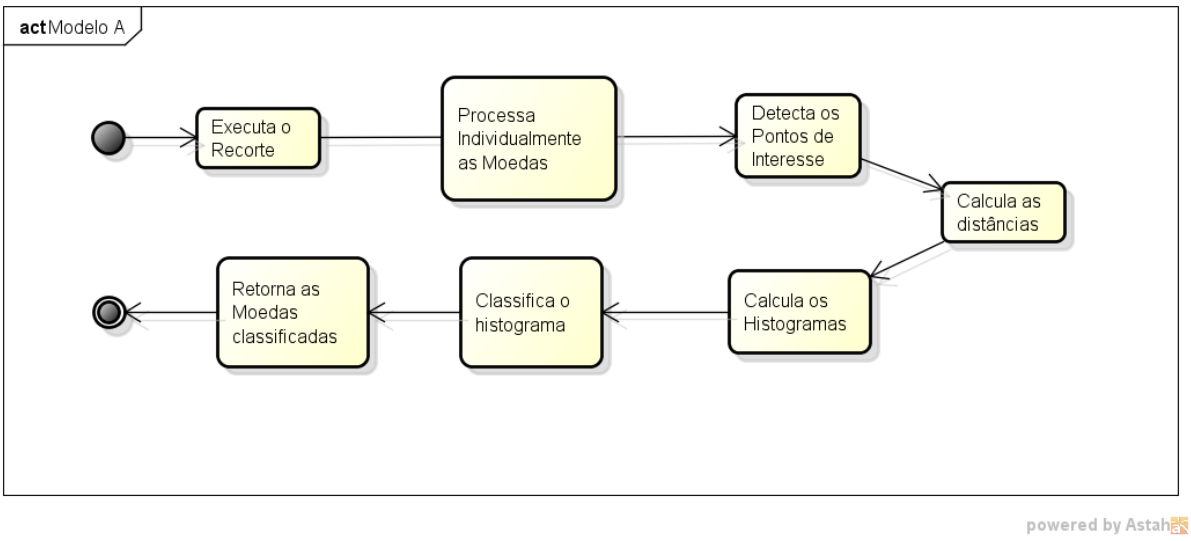

Figura 21 – Diagrama de Atividade com o processamento do Modelo A. Fonte: O Autor.

6 RECONHECENDO O VALOR DAS MOEDAS UTILIZANDO O RAIO DA MOEDA – MODELO B.

Na abordagem utilizando o raio da moeda, foi necessário utilizar um marcador para calcular o tamanho real aproximado das moedas. Para facilitar a identificação, o marcador escolhido foi uma moeda de 5 centavos pintada de vermelho. Para fazer a extração do marcador a imagem foi convertida para o formato de representação RGB, onde cada pixel possui três canais com os valores de intensidade das cores vermelho, verde e azul. Após a conversão foi feita uma segmentação por corte, baseado na intensidade de cores do canal vermelho eliminando todos os pixels com uma baixa intensidade. Em seguida a imagem é convertida para uma representação binarizada da imagem em apenas um canal, gerando assim uma mascará. O exemplo desse processamento é demonstrado na **Figura 22**.

Com a máscara detectada é feita uma correção de perspectiva para reduzir a diferença de tamanho dos objetos causada pela inclinação da câmera em relação ao plano onde estão as moedas. Para fazer essa correção é necessário identificar quatro pontos-chave e calcular as novas posições para esses pontos. Com esses pontos é possível calcular uma matriz chamada Homografia com os valores necessários para modificar a imagem atual e efetuar a correção de perspectiva. A **Figura 23** ilustra o retângulo que envolve o marcador, permitindo assim selecionar seus quatro cantos como pontos-chave. Com esses pontos são calculados os novos pontos utilizando a média da soma da altura e largura do triângulo. Com esses valores, utilizando como referência o ponto superior esquerdo do retângulo são calculados os novos pontos, transformando o retângulo em um quadrado como mostrado na **Figura 23** a direita. Com esses pontos o OpenCV consegue realizar a transformação de perspectiva. A **Figura 24** demonstra as diferenças na imagem antes e depois da correção de perspectiva. Na **Error! Reference source not found.**, é ilustrado a sequência de passos para a execução do modelo utilizando um diagrama de sequência.

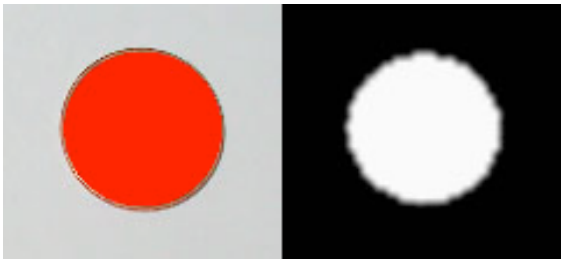

Figura 22 – A esquerda a imagem do marcador e a direita a máscara criada com a segmentação do marcador. Fonte: O Autor.

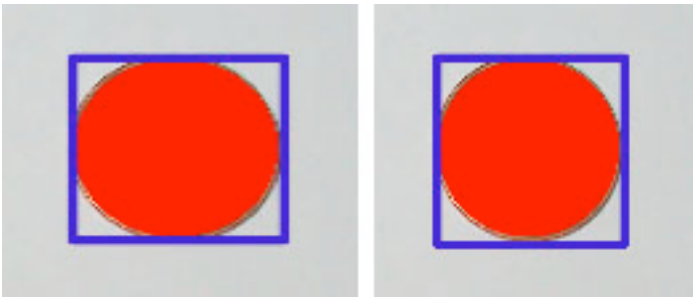

Figura 23 – A esquerda a imagem do marcador antes da correção de perspectiva e a esquerda a imagem do marcador após a correção de perspectiva. Fonte: O Autor.

Figura 24 – À esquerda a imagem antes da correção de perspectiva, e à direita a imagem após a correção de perspectiva. Fonte: O Autor.

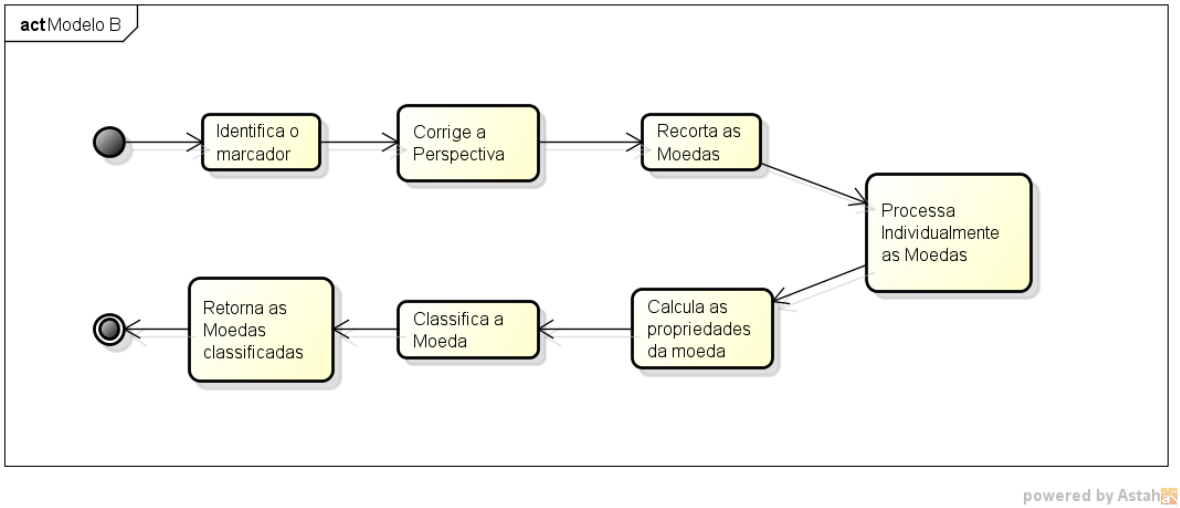

Figura 25 – Diagrama de atividades com o processamento do Modelo B. Fonte: O Autor.

Após a identificação do marcador e a correção de perspectiva da imagem finalizada, é feita a segmentação das moedas da imagem, onde para cada moeda identificada é

calculado o raio da moeda com base o raio do marcador. Por exemplo, como o marcador tem o tamanho de 21mm (o tamanho de uma moeda de 5 centavos da segunda família de moedas), se esse marcador possuir o raio de 210 pixels, uma moeda com raio de 220 pixels terá o raio real aproximado de 22mm. Esse raio real é usado para diferenciar as moedas entre si. Como em alguns casos as moedas possuem tamanhos parecidos também foram identificados outros atributos para aumentar a quantidade de acertos do classificador. Os outros atributos utilizados são a cor média e a saturação media da moeda. Isso faz sentido pois ocorre uma variação de cores entre as moedas, como pode ser observado na **Figura 26**, fazendo com que a cor média seja o valor da cor predominante da moeda. A saturação da imagem é utilizada para as moedas que possuem cores opacas como as moedas de 50 centavos e 1 real, já que a cor prata predominante nessas moedas possui uma saturação baixa, fazendo com que o atributo que indica a cor da imagem seja incapaz de determinar a cor da moeda.

Figura 26 – Exemplo da diferença de cores entre as moedas. Fonte: O Autor.

8 RESULTADOS EXPERIMENTAIS.

Nessa seção são mostrados os resultados experimentais para dois classificadores, o primeiro baseado nos pontos de interesse e o segundo utilizando o raio e as cores das moedas, ambos testados utilizando os algoritmos de rede neural e árvore de decisão conforme descritos na Sessão 2.2.

8.1 Base de dados.

Para a realização dos experimentos foram coletadas aproximadamente 400 instâncias de moedas, onde cada instância é um recorte de uma moeda em uma imagem. As

imagens foram obtidas utilizando três dispositivos 1 iPhone 3GS, 1 iPad 2 e 1 iPhone 5. Em todos os dispositivos foram capturadas imagens com 3 Megapixels, 5 Megapixels e 8 Megapixels conforme mostrado na Tabela 1. Essa variação ocorreu para permitir o teste do algoritmo em resoluções mais baixas. Nessas imagens as quantidades de moedas variam entre 5 e 20 moedas.

As imagens obtidas foram separadas em dois grupos, imagens com marcador e imagens sem marcador, para que fosse possível testar as duas abordagens. Cada grupo foi dividido em dois subgrupos, imagens para treinamento e imagens para teste para que o classificador não fosse testado com as mesmas instâncias que foram utilizadas para fazer o treinamento, para assim evitar a utilização de moedas de uma mesma imagem nos conjuntos de treino e teste, que poderia favorecer erroneamente o modelo.

Resolução	Valor	Quantidade
3 MP	5	10
	10	36
	25	9
	50	9
	100	10
5 MP	5	55
	10	40
	25	40
	50	10
	100	26
8 MP	5	11
	10	48
	25	39
	50	30
	100	28
Total		401

Tabela 1 – Tabela com a distribuição de moedas pelo valor e resolução. Fonte: O Autor.

8.2 Resultado do Recorte.

O algoritmo de recorte obteve uma boa taxa de acerto. No entanto, para que o algoritmo pudesse suportar algumas variações do fundo, optou-se por um recorte mais conservador. Com isso o algoritmo tem uma boa taxa de acerto nas moedas detectadas, porém algumas moedas são ignoradas.

Os cenários mais comuns onde as moedas são ignoradas no recorte ocorrem quando a saturação da imagem é baixa, quando as moedas estão afastadas do centro da imagem, e quando ocorre algum reflexo do flash nas moedas. Na **Figura 30**, é apresentado uma imagem que sofreu forte influência do flash da câmera.

O algoritmo ignora as moedas com saturação baixa, pois ao converter a imagem para escala de cinza, os pixels com baixa saturação ficam com valores muito altos (aproximando do branco) e ao fazer a operação de binarização da imagem o valor do limiar precisaria ser muito alto para não eliminar esses pixels. Porém se o valor do limiar é muito alto, o algoritmo reduz a tolerância na mudança de fundo. Na **Figura 27** é demonstrado o processamento de uma imagem com a saturação baixa. À esquerda a imagem original e à direita a imagem binarizada.

Outro problema na detecção das moedas ocorre devido a distorções causadas pela lente da câmera. Um exemplo dessa distorção é mostrado na **Figura 28** e **Figura 29**, quando uma moeda fica na área onde ocorre essa distorção a imagem da moeda passa a ter um formato de uma elipse. Por conta desse formato o algoritmo de detecção de círculos do OpenCV passa a ignorar essas moedas.

Para tentar resolver esses problemas foi feita uma implementação alternativa de um método para o reconhecimento de moedas. Nessa implementação são desenhados círculos com o menor tamanho para envolver esses objetos e depois são desprezados os objetos com o raio 10% menor e raio 30% maior que o raio do marcador. Com essa implementação o algoritmo passou a reconhecer mais moedas, mesmo quando elas caem em algum dos problemas anteriores. Porém essa implementação passou a reconhecer uma quantidade maior de falsas moedas e em alguns casos ocorreram a detecção de moedas dentro de moedas.

O Algoritmo de recorte teve uma taxa de acerto de aproximadamente 72%. A Tabela 2, mostra a proporção de acertos no recorte em função da resolução. É possível perceber que quanto menor a resolução da imagem menor é a taxa de acerto. Isso ocorre devido a baixa quantidade de pixels e devido a uma maior granularidade nas imagens com a resolução mais baixa fazendo o algoritmo de detecção ignorar alguns círculos.

	Total	Encontradas	Acerto
3 MP	120	30	25%
5 MP	191	152	79,58%
8 MP	247	225	91,09%
Total	**558**	**407**	**72,94%**

Tabela 2 – Porcentagem de acerto do recorte em função da resolução. Fonte: O Autor.

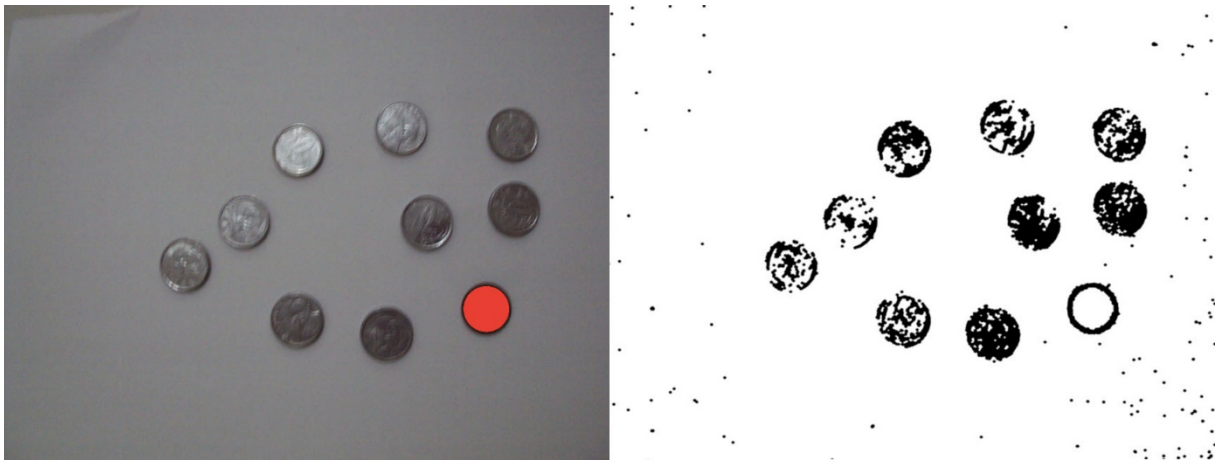

Figura 27 – Exemplo de processamento de uma imagem com baixa saturação. Fonte: O Autor.

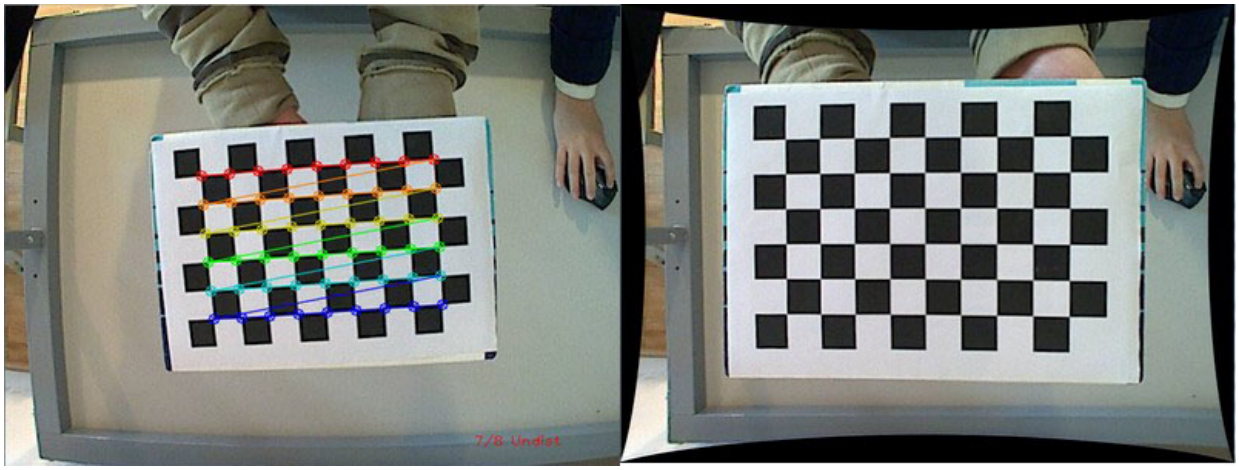

Figura 28 - Exemplo de processamento de uma imagem com distorção. Fonte: [23].

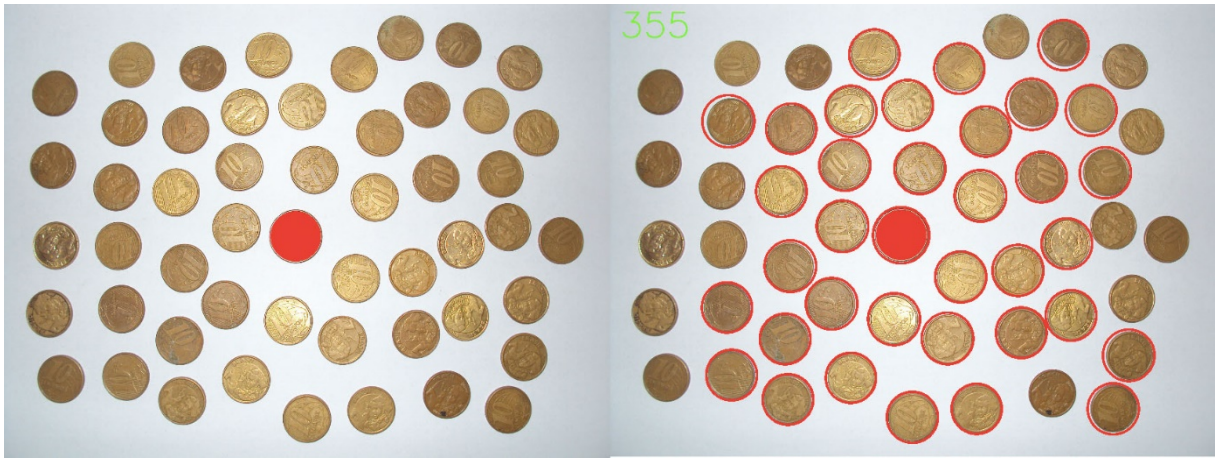

Figura 29 - Exemplo de processamento de uma imagem com moedas nas áreas distorcidas. Fonte: O Autor.

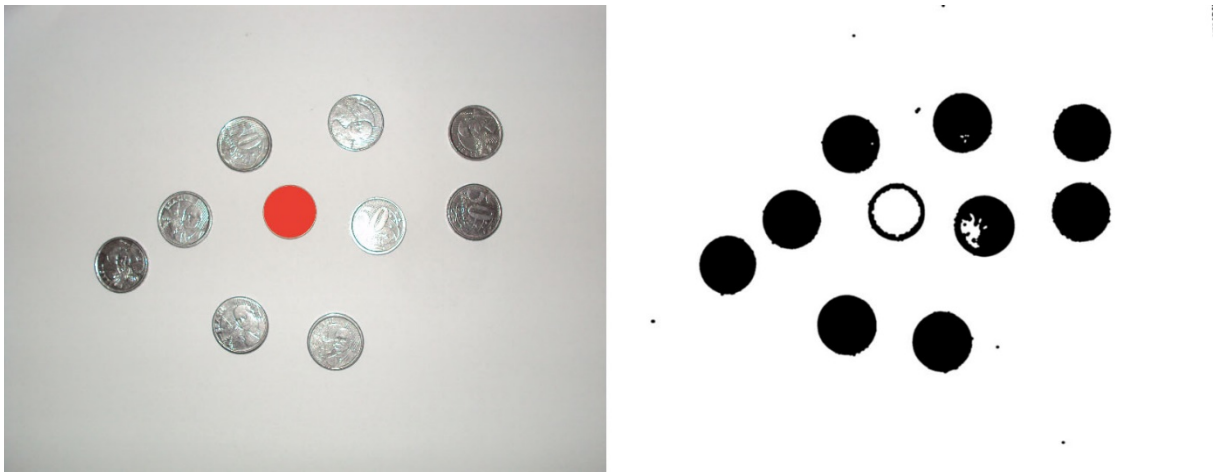

Figura 30 - Exemplo de processamento de uma imagem com o reflexo do flash. Fonte: O Autor.

8.3 Resultados para o algoritmo baseado em pontos de interesse – Modelo A.

O algoritmo baseado em pontos de interesse obteve uma baixa taxa de acerto nos modelos utilizando árvore de decisão e na rede neural (Multilayer Perceptron). Isso ocorreu pois os algoritmos de detecção de pontos de interesse detectam pontos com a mesma proporção em todos os tipos de moedas, como pode ser visto na **Figura 31**. Isto reduz a diferenciação entre as moedas, dificultando a tarefa de classificação baseada em pontos de interesse. Ou seja, isso fez com que os histogramas tivessem apenas pequenas variações dificultando a classificação das imagens. Outro problema encontrando nessa abordagem foi a quantidade necessária de recursos computacionais necessários para a sua execução, onde em

alguns casos o algoritmo para detecção de pontos de interesse chega a utilizar mais de 1 Gigabyte de memória e consome muito tempo para ser executado.

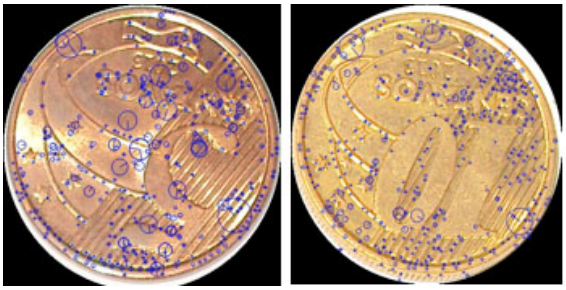

Figura 31 - Exemplo da divisão de regiões na moeda. Fonte: O Autor.

8.3.1 Classificação utilizando uma árvore de decisão.

Os resultados da classificação utilizando a árvore de decisão ficou muito próximo de classificadores aleatórios. Em um classificador "aleatório", a probabilidade de classificar uma classe é igual à proporção de instâncias desta classe. Classificadores aleatórios possuem a taxa de acerto próxima a 50% e sua matriz de confusão possui os valores distribuídos de forma uniforme em vez de possuir uma maior concentração de pontos na diagonal principal.

INSTÂNCIAS CLASSIFICADAS CORRETAMENTE	128	34%
INSTÂNCIAS CLASSIFICADAS INCORRETAMENTE	238	66%
TOTAL DE INSTÂNCIAS	366	100%

Tabela 3 – Estatísticas de classificação utilizado uma árvore de decisão. Fonte: O Autor.

5	**10**	**25**	**50**	**100**	**←Classificado como:**
25	15	20	8	8	**5**
31	36	17	14	10	**10**
17	27	25	5	6	**25**
4	14	6	11	8	**50**
4	6	12	6	31	**100**

Tabela 4 – Matrix de confusão da classificação utilizando uma árvore de decisão. Fonte: O Autor.

A Tabela 3, mostra a porcentagem de acerto do classificador. Como a taxa de acerto desse classificador foi apenas de 34%, por conta disso ele pode ser considerado como tendo qualidade próxima a do classificador aleatório como dito anteriormente, já que nesse casso a probabilidade de uma classe ser sorteada de maneira aleatória é de 20%. Na Tabela 4 é mostrada a matriz de confusão sua distribuição de erros e acertos de classificação. Nela pode ser observado que a distribuição das classes foi feita proporcionalmente de acordo com a quantidade de classes existentes.

8.3.2 Classificação utilizando uma rede neural.

Os resultados de classificação utilizando redes neurais também foram muito ruins. Embora a taxa de acertos tenha melhorado em 7% o classificador se manteve próximo de uma classificação aleatória, como pode ser observado na Tabela 5. Esse aumento da taxa de acerto pode ser explicado comparando a Tabela 6 que exibe a matriz de confusão de redes neurais com a Tabela 4. Nessa comparação pode ser notado que a taxa de acertos da moeda de 10 centavos aumentou de 36 para 53, e na moeda de 25 centavos os acertos aumentaram de 25 para 30, enquanto para as outras moedas temos apenas uma pequena variação.

INSTÂNCIAS CLASSIFICADAS CORRETAMENTE	149	41%
INSTÂNCIAS CLASSIFICADAS INCORRETAMENTE	217	59%
TOTAL DE INSTÂNCIAS	366	100%

Tabela 5 – Estatísticas de classificação utilizado Rede Neural. Fonte: O Autor.

5	**10**	**25**	**50**	**100**	**←Classificado como:**
25	25	13	9	4	**5**
25	53	17	10	3	**10**
16	19	30	3	12	**25**
7	15	7	12	2	**50**
6	4	16	4	29	**100**

Tabela 6 – Matrix de confusão da classificação utilizando Rede Neural. Fonte: O Autor.

8.4 Resultados para o algoritmo baseado no raio da moeda – Modelo B.

O modelo utilizando o raio da moeda apresentou resultados bem melhores. Como nessa abordagem foi utilizada uma característica relativamente fácil de ser extraída utilizando visão computacional, a utilização de recursos computacionais foi menor, com o tempo de execução médio por imagem de 3 segundos e no máximo 200 Megabytes de memória. Utilizando apenas o raio da moeda foi obtida a taxa de acerto de 80%. Porém foram adicionados outros atributos para melhorar a taxa de acerto. Os atributos adicionados foram a média de cor da imagem e a média de saturação da imagem. Com esses novos atributos a taxa de acerto aumentou para 90%. Outro atributo extraído das moedas com o objetivo de melhorar a classificação das moedas de 1 real foi a diferença da saturação entre as partes interna (66 % do raio na parte mais interna) e externa (33% do raio na parte mais externa) da moeda, porém na geração dos algoritmos de classificação ele foi ignorado.

8.4.1 Classificação utilizando uma árvore de decisão.

Na classificação utilizando árvores de decisão essa abordagem obteve uma boa taxa de acerto comparada à abordagem utilizando pontos de interesse. Na abordagem atual (Modelo B) foi obtida a taxa de 90,1% de acerto, como é mostrada na Tabela 7, contra 34% do Modelo A. Na Tabela 8, é possível ver na matriz de confusão que a diagonal principal contém uma maior concentração de pontos, diferente da matriz de confusão da Tabela 4. Analisando a tabela 8 é possível notar que a maior quantidade de erros ocorre nas moedas de 5 e 10 centavos, isso ocorreu devido ao fato dos valores do raio e da cor serem parecidos.

INSTÂNCIAS CLASSIFICADAS CORRETAMENTE	328	90.1%
INSTÂNCIAS CLASSIFICADAS INCORRETAMENTE	36	9.9%
TOTAL DE INSTÂNCIAS	364	100%

Tabela 7 – Estatísticas de classificação utilizado árvore de decisão. Fonte: O Autor.

5	10	25	50	100	←Classificado como:
81	7	4	0	2	**5**
7	94	0	1	0	**10**
7	0	63	0	5	**25**
0	0	0	38	1	**50**
0	0	2	0	52	**100**

Tabela 8 – Matrix de confusão da classificação utilizando árvore de decisão. Fonte: O Autor.

8.4.2 Classificação utilizando uma rede neural.

Na classificação utilizando redes neurais a taxa de acerto foi quase a mesma da classificação utilizando árvore de decisão. Como pode ser observado na Tabela 9, esse classificador obteve uma taxa de acerto de 90,7%, com uma pequena diferença de 0,6% se comparado ao classificador anterior. Essa diferença representa apenas duas instâncias na base de dados. Observando a Tabela 10, é possível notar que essa diferença foi causada pelo aumento de acertos nas moedas de 10, 25, 50 centavos. Porém ocorreu um aumento na quantidade de erros nas moedas de 5 centavos e de 1 real.

INSTÂNCIAS CLASSIFICADAS CORRETAMENTE	330	90.7%
INSTÂNCIAS CLASSIFICADAS INCORRETAMENTE	34	9.3%
TOTAL DE INSTÂNCIAS	364	100%

Tabela 9 – Estatísticas de classificação utilizado uma rede neural. Fonte: O Autor.

5	10	25	50	100	←Classificado como:
83	5	3	3	0	**5**
4	98	0	0	0	**10**
6	0	62	0	7	**25**
1	1	0	36	1	**50**
0	0	2	1	51	**100**

Tabela 10 – Matrix de confusão da classificação utilizando uma rede neural. Fonte: O Autor.

8.5 Execução em dispositivo Móvel.

Como um dos objetivos do projeto é executar o algoritmo em um dispositivo móvel, foram feitos dois aplicativos utilizando um *port* oficial para dispositivos IOS, um utilizando o Modelo A baseado em pontos de interesse e o Modelo B baseado no raio da moeda.

Na execução utilizando do Modelo A, não foi possível utilizar o algoritmo de detecção de pontos de interesse do OpenCV, pois que ele não foi implementado no *port* oficial até a versão 2.2, versão utilizada pelo aplicativo.

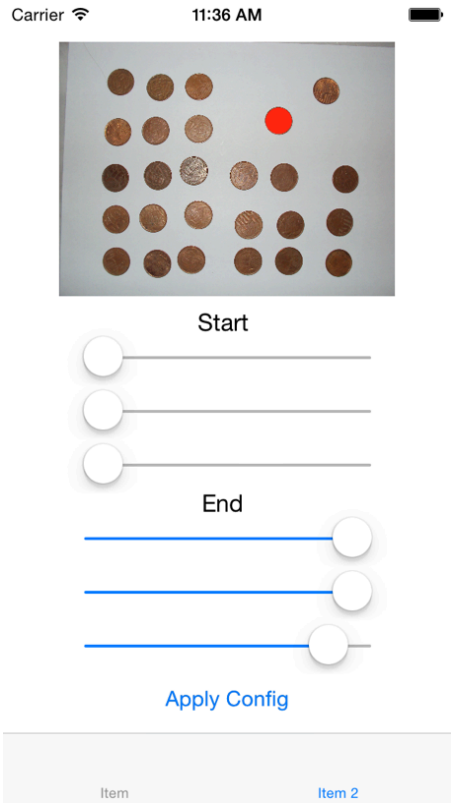

Figura 32 – Imagem com os controles para configurar a segmentação do marcador. Fonte: O Autor.

Na execução utilizando do Modelo B, foi utilizada uma classe para converter o formato de imagens utilizado pelo IOS para formato reconhecido pelo OpenCV. Como ao fazer essa conversão o formato de representação dos canais é alterada e o IOS não possui uma representação padrão, foi necessário criar uma tela para poder configurar manualmente os valores para executar a segmentação do marcador. Essa tela é exibida na **Figura 32**, essa tela possui seis controles, três para os valores iniciais e outros três para os valores máximos dos canais e um botão para aplicar essa configuração. Com essa configuração o algoritmo funcionou normalmente. A **Figura 33**, mostra a tela da aplicação após a contagem das

moedas com um alerta indicando o valor contido na imagem. Porém como uma das motivações desse algoritmo é a utilização por deficientes visuais, esse objetivo não foi atendido.

No Modelo B, o algoritmo obteve uma boa performance com a execução no IOS. A utilização de memória no IOS foi menor que a utilização de memória durante a execução no computador, essa redução ocorre devido às otimizações no gerenciamento de memória feitas pelo compilador do IOS. O tempo de execução por imagem também foi baixo, variando entre 2 e 5 segundos. Os picos de processamento e uso de memória ocorrem apenas durante a detecção de círculos na imagem.

Figura 33 – Imagem do aplicativo após a contagem das moedas identificadas. Fonte: O Autor.

9 CONSIDERAÇÕES FINAIS.

O objetivo geral desse trabalho foi desenvolver um algoritmo eficiente para contagem de moedas em imagens digitais. Nesse sentido, foi realizada uma revisão de técnicas de visão computacional e aprendizagem de máquina para apoiar a viabilidade deste trabalho.

Para implementação do algoritmo, foi feito um algoritmo para a extração de moedas, e outro para a classificação das moedas.

O algoritmo de recorte de moedas, obteve uma taxa de acerto aceitável por conta de ter sido utilizada uma abordagem mais conservadora, onde o algoritmo tem uma alta taxa de acerto porém uma baixa cobertura. Para trabalhos futuros seria proposto a criação de um algoritmo melhor para o reconhecimento de moedas para aumentar a cobertura do algoritmo mantendo a taxa de acertos.

Para a classificação foram utilizados dois modelos: Modelo A, utilizando pontos de interesse e o Modelo B utilizando o tamanho da moeda. O Modelo A, se mostrou ineficiente, com uma baixa taxa de acerto e inviável para a implementação em um dispositivo móvel usando os recursos nativos do OpenCV. O Modelo B, se mostrou mais confiável e com uma melhor performance comparado ao Modelo A. Porém por conta deste modelo ter sido fortemente relacionado a representação de canais da imagem, ao implementa-lo na plataforma IOS, a detecção automática do limiar para extração do marcador se mostrou ineficiente. Com isso foi necessário criar uma ferramenta manual para configuração desse limiar. Portanto para um trabalho futuro é sugerido utilizar uma nova abordagem para o reconhecimento do marcador.

O objetivo principal do trabalho de criar um algoritmo para contagem de moedas capaz de ser executado em uma plataforma mobile foi alcançado, porém a motivação de criar um aplicativo para auxiliar deficientes visuais foi perdida. Isso por conta que é necessário configurar manualmente os valores para a detecção do marcador no Modelo B.

REFERÊNCIAS.

1. DROIDER. **Já são mais de 19 milhões de smartphones no Brasil, segundo pesquisa**. 2011. Disponível em: <http://www.droider.com.br/smartphone/mercado-smartphone/numero-mais-19-milhoes-de-smartphones-no-brasil-pesquisa.html>. Acesso em: 17 jul. 2013.

2. G1. **23,9% dos brasileiros declaram ter alguma deficiência, diz IBGE**. 2012. Disponível em: <http://g1.globo.com/brasil/noticia/2012/04/239-dos-brasileiros-declaram-ter-alguma-deficiencia-diz-ibge.html>. Acesso em: 17 jul. 2013.

3. RAMOS, André Luís Belmiro Moreira. **Uma Abordagem Metodológica para a Avaliação Multidimensional da Acessibilidade de Interfaces com o Usuário para Aplicações Web**. 2011.

4. SZELISKI, Richard. **Computer vision: algorithms and applications**. Springer, 2010.

5. OFFICEMAX. **COIN COUNTERS & SORTERS**. 2013. Disponível em: <http://www.officemax.com/office-supplies/cash-handling/counters-sorters>. Acesso em: 12 jul. 2013.

6. DA SILVA BARROS, Antonio Carlos et al. **Implementação de segmentação da pele atráves matlab/simulink ultilizando o espaço de cores IRgBy**. 2011.

7. LAGANIÈRE, Robert. **OpenCV 2 Computer Vision Application Programming Cookbook: Over 50 Recipes to Master this Library of Programming Functions for Real-time Computer Vision**. Packt Publishing Ltd, 2011.

8. MARENGONI, Maurício; STRINGHINI, Stringhini. **Tutorial: Introdução à visão computacional usando opencv**. Revista de Informática Teórica e Aplicada, v. 16, n. 1, p. 125-160, 2010.

9. OPENCV. **CANNY Edge Detector. OpenCV**. 2013. Disponível em: <http://docs.opencv.org/trunk/doc/tutorials/imgproc/imgtrans/canny_detector/canny_detector.html>. Acesso em: 11 jul. 2013.

10. HESS, Rob. **An open-source SIFTLibrary**. In: Proceedings of the international conference on Multimedia. ACM, 2010. p. 1493-1496.

11. OLIVEIRA, A; MARROQUIM, R. **Processamento de Imagens**. 2012. Disponível em: <http://www.lcg.ufrj.br/Cursos/cos756/13-sift-hough.pdf>. Acesso em: 17 jul. 2013.

12. WILSON, G. **Programmer´s Tool Chest – The OpenCV Library**. Dr. Dobb's. Disponível em: <http://www.drdobbs.com/open-source/the-opencv-library/184404319?pgno=1>. Acesso em: 23 jun. 2013.

13. REZENDE, Solange Oliveira. **Sistemas inteligentes: fundamentos e aplicações**. Editora Manole Ltda, 2003.

14. WITTEN, Ian H.; FRANK, Eibe. **Data Mining: Practical machine learning tools and techniques**. Morgan Kaufmann, 2011.

15. UFPE. **Indução de Árvores e Regras Proposicionais de Decisão**. 2007. Disponível em: <www.cin.ufpe.br/~compint/aulas-IAS/kdd-011/ID3attributiveRules.ppt>. Acesso em: 17 jul. 2013.

16. SOKOLOVA, Marina; JAPKOWICZ, Nathalie; SZPAKOWICZ, Stan. **Beyond accuracy, F-score and ROC: a family of discriminant measures for performance evaluation**. In: AI 2006: Advances in Artificial Intelligence. Springer Berlin Heidelberg, 2006. p. 1015-1021.

17. FAWCETT, Tom. **An introduction to ROC analysis**. Pattern recognition letters, v. 27, n. 8, p. 861-874, 2006.

18. MATSUBARA, EDSON TAKASHI; MONARD, MARIA CAROLINA. **Relações entre Ranking, Análise ROC e Calibração em Aprendizado de Máquina**, 2008.

19. OLIVEIRA, G. **Detecção de Ausência de Cinto de Segurança em Imagens de Transito**, 2012.

20. ZAMBANINI, Sebastian; KAMPEL, Martin. **Automatic coin classification by image matching**. In: Proceedings of the 12th International conference on Virtual Reality, Archaeology and Cultural Heritage. Eurographics Association, 2011. p. 65-72.

21. SAIPULLAH, Khairul Muzzammil; KIM, Deok-Hwan; LEE, Seok-Lyong. **Rotation invariant texture feature extraction based on Sorted Neighborhood Differences**. In: Multimedia and Expo (ICME), 2011 IEEE International Conference on. IEEE, 2011. p. 1-6.

22. HUBER-MÖRK, Reinhold et al. **Identification of ancient coins based on fusion of shape and local features**. Machine vision and applications, v. 22, n. 6, p. 983-994, 2011.

23. OPENCV. **Camera calibration With OpenCV**. 2013. Disponível em: <http://docs.opencv.org/doc/tutorials/calib3d/camera_calibration/camera_calibration.html>. Acesso em: 06 mai. 2014.

APÊNDICES

APÊNDICE A – Diagramas de Sequência

Nesse apêndice será apresentado os diagramas elaborados durante o desenvolvimento.

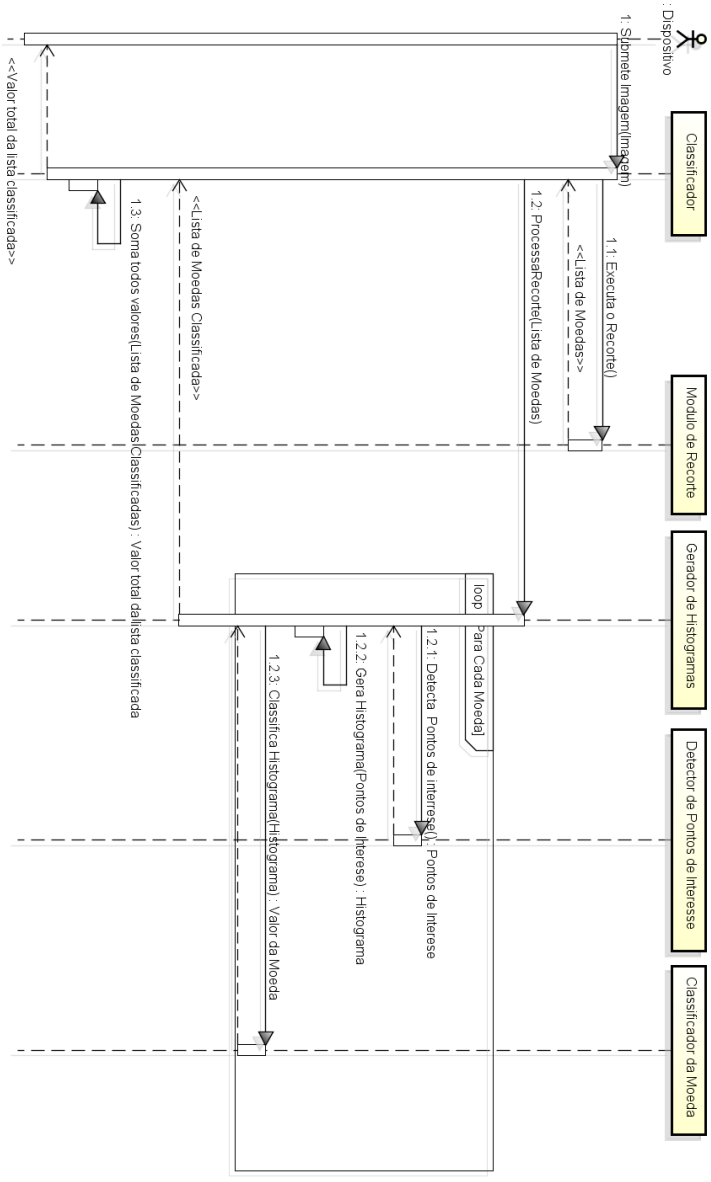

Figura 34 - Diagrama de Sequência com o processamento do Modelo A. Fonte: O Autor.

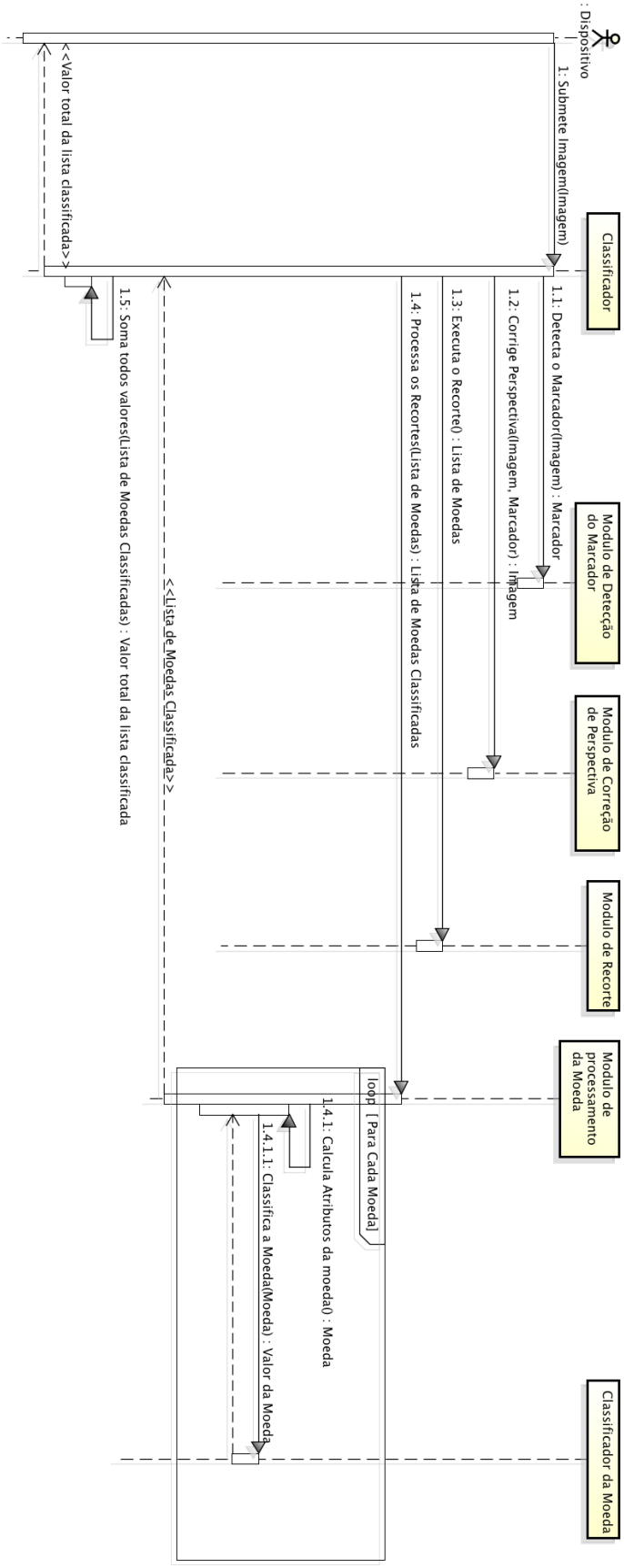

Figura 35 - Diagrama de sequência com o processamento do Modelo B. Fonte: O Autor.