CENTRO EDUCACIONAL DA FUNDAÇÃO SALVADOR ARENA

FACULDADE DE TECNOLOGIA TERMOMECANICA

BRUNA SELYMES SERGIO

DENIS RAISON DOS SANTOS

**APLICANDO O OPENCV PARA AUXILIO NA CORREÇÃO AUTOMÁTICA DE AVALIAÇÕES**

São Bernardo do Campo

2014

BRUNA SELYMES SERGIO

DENIS RAISON DOS SANTOS

APLICANDO O OPENCV PARA AUXILIO NA CORREÇÃO AUTOMÁTICA DE AVALIAÇÕES

Trabalho de conclusão de curso, realizado sob orientação do Prof. Ms. Eduardo Rosalém Marcelino, apresentado à Faculdade de Tecnologia Termomecanica como requisito para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

São Bernardo do Campo

2014

Bruna Selymes Sergio

Denis Raison dos Santos

Aplicando o opencv para auxilio na correção automática de avaliações

Trabalho de Conclusão de Curso – Faculdade de Tecnologia Termomecanica

Comissão Julgadora

Eduardo Rosalem Marcelino

Karina Duarte

Ricardo S. Jacomini

São Bernardo do Campo

2014

Dedicamos o presente trabalho às nossas famílias pelo apoio, compreensão e incentivo.

AGRADECIMENTOS

Nossossinceros agradecimentos a todos aqueles que de alguma forma doaram um pouco de si para que a conclusão deste trabalho fosse possível:

A Deus, a quem devemos absolutamente todas as nossas conquistas.

Às nossas famílias, em especial a nossos pais: Hilmar Lippi Sergio & Bianca Selymes C. Sérgio e Marcos dos Santos & Elenice de Souza, que com amor e compreensão, nos incentivaram e nos apoiaram até a conclusão de mais esta etapa de nossas vidas.

Ao professor MS. Eduardo Rosalem Marcelino, nosso orientador, pelo tempo dedicado, paciência, incentivo e conselhos que tornaram a conclusão desta monografia possível.

À Fundação Salvador Arena, bem como o corpo docente que nos suportaram com infraestrutura, material e ensino de qualidade para a conclusão desta etapa de nossas vidas acadêmicas.

Aos nossos amigos, pela compreensão a cada momento em que não pudemos estar juntos para nos dedicarmos ao presente trabalho.

“A tarefa não é tanto ver aquilo que ninguém viu, mas pensar o que ninguém ainda pensou sobre aquilo que todo mundo vê.”

Arthur Schopenhauer

RESUMO

Visão computacional é a área da ciência da computação responsável pela visão de máquina, pela forma em que um computador enxerga a partir de imagens captadas por câmeras e sensores. Esta tecnologia vem ganhando grande destaque devido à crescente necessidade de que máquinas consigam reproduzir tarefas que são executadas por seres humanos, principalmente aquelas repetitivas, trabalhosas e perigosas. Este trabalho introduz a área de visão computacional bem como alguns dos algoritmos e ferramentas para o desenvolvimento de aplicações voltadas a esta área. Como forma de aplicar os conceitos estudados, foi desenvolvido um protótipo de um aplicativo de correção automática de avaliações do tipo múltipla escolha utilizando a biblioteca de visão computacional *OpenCV.*

PALAVRAS-CHAVE: Visão Computacional, OpenCV, Corretor automático, Reconhecimento de Padrões

ABSTRACT

Computer vision is the area of the computer science responsible for machine vision, the way in which a computer sees from images captured by cameras and sensors. This technology has been gaining great prominence due to the increasing need to machines to be able to perform tasks that are performed by humans, especially those repetitive, laborious and dangerous. This paper introduces the computer vision area as well as some of the algorithms and tools for developing applications related to this area. In order to apply the concepts studied, we developed a prototype application for automatic correction of multiple-choice assessments using the computer vision OpenCV library.

KEY-WORDS: Computer Vision, OpenCV, Automatic corrector, pattern recognittion

LISTA DE FIGURAS

[Figura 1- Ilustração do sistema visual humano. 23](#_Toc393138690)

[Figura 2 – Analogia olho humano e Câmera fotográfica. 24](#_Toc393138691)

[Figura 3 - Exemplo de reconhecimento de borda. Imagem original e apenas suas bordas. 32](#_Toc393138692)

[Figura 4 - Elementos do processo de análise de imagem. 35](#_Toc393138693)

[Figura 5 – Sistema de reconhecimento de Padrões. 37](#_Toc393138694)

[Figura 6 - Fases do reconhecimento de padrões. 39](#_Toc393138695)

[Figura 7 - Filtragem de ruidos. Esquerda:Imagem com ruido; direita : mesma imagem após filtragem. 41](#_Toc393138696)

[Figura 8 - Extração de Características. Impressão digital e sua imagem direcional. 41](#_Toc393138697)

[Fonte: Revista Ubiquidade, 2011 42](#_Toc393138698)

[Figura 9 - Imagens a serem encontradas. Correspondem a algumas regiões na imagem teste, mas com algumas transformações aplicadas (rotação e escala). 50](#_Toc393138699)

[Figura 10 – Cenário em que as imagens serão encontradas. 51](#_Toc393138700)

[Figura 11- Imagem resultado Resultado. 51](#_Toc393138701)

[Figura 12 – Imagens progressivamente desfocadas, resultado da aplicação do *Gaussian Blur* várias vezes na mesma imagem. 52](#_Toc393138702)

[Figura 13 - A imagem original redimensionada para metade do seu tamanho cada vez que o *Gaussian Blur* é aplicado. 53](#_Toc393138703)

[Figura 14 – Passo LoG (Laplacian de Gaussian). Em representação RGB, imagens com valores próximos de zero são exibidos em preto, esta é a razão das imagens serem escuras. 54](#_Toc393138704)

[Figura 15 - Pixel atual (marcado com um 'X') e os 26 pixels vizinhos na comparação de máximos/mínimos. 55](#_Toc393138705)

[Figura 16 – Etapa de rejeição de pontos sensíveis ao ruído ou que estejam mal localizados. 56](#_Toc393138706)

[Figura 17 - Imagem origem e imagem *template* e resultado da detecção da correspondência entre as duas. 59](#_Toc393138707)

[Figura 18 - Deslocamento na imagem origem. 60](#_Toc393138708)

[Figura 19 - Imagem resultado do deslocamento da imagem buscada sobre a imagem origem. 61](#_Toc393138709)

[Figura 20 - Imagem binária gerada pela função adaptativeThreshold. 62](#_Toc393138710)

[Figura 21- Círculos detectados na imagem utilzando a função houghCircles da OpenCV. 63](#_Toc393138711)

[Figura 22 - Página inicial Questões na Web 66](#_Toc393138712)

[Figura 23 - Versão demo do aplicativo na Google Play 66](#_Toc393138713)

[Figura 24 - Versão paga do aplicativo na Google Play 67](#_Toc393138714)

[Figura 25 – Página inicial GradeCam 68](#_Toc393138715)

[Figura 26 - Criação de gabarito GradeCam 69](#_Toc393138716)

[Figura 27 – Arquitetura geral do corretor automático de gabaritos de múltipla escolha. 71](#_Toc393138717)

[Figura 28 - Modelagem de dados do Sistema 74](#_Toc393138718)

[Figura 29 - Tela inicial da aplicação 76](#_Toc393138719)

[Figura 30 – Tela de criação de novos gabaritos 77](#_Toc393138720)

[Figura 31 – Tela de criação de novos gabaritos 77](#_Toc393138721)

[Figura 32 – Modelo de arquivo PDF do gabarito de respostas 79](#_Toc393138722)

[Figura 33 - Quantidade de paginas PDF 80](#_Toc393138723)

[Figura 34 – Android *Jelly Bean* 4.3. 81](#_Toc393138724)

[Figura 35 - *KitKat* 4.4. 81](#_Toc393138725)

[Figura 36 - Imagem do gabarito a ser corrigido 82](#_Toc393138726)

[Figura 37 - Fluxograma do funcionamento da correção de gabaritos 83](#_Toc393138727)

[Figura 38 - Fluxograma do funcionamento do Web Service 85](#_Toc393138728)

[Figura 39 - Imagem em escala preta e branca 86](#_Toc393138729)

[Figura 40 - *Template* reconhecimento do início e fim do gabarito. 87](#_Toc393138730)

[Figura 41 - Resultado da execução do algoritmo SIFT 87](#_Toc393138731)

[Figura 42 - Resultado do corte na imagem do gabarito 88](#_Toc393138732)

[Figura 43 - Imagem após utilizar o GaussianBlur 89](#_Toc393138733)

[Figura 44 - Imagens após aplicação do adaptativeThreshold 90](#_Toc393138734)

[Figura 45 – Contornos retangulares encontrados na imagem. 91](#_Toc393138735)

[Figura 46- Círculos detectados na imagem, utilizando o algoritmo HoughCircles 91](#_Toc393138736)

[Figura 47 - Gabarito corrigido durante os testes 92](#_Toc393138737)

[Figura 48 – Gabarito corrigido durante os testes 94](#_Toc393138738)

[Figura 49 – Gabarito corrigido durante os testes. 95](#_Toc393138739)

[Figura 50 – Gabarito corrigido sem sucesso durantes os testes 95](#_Toc393138740)

[Figura 51 - Gabarito corrigido durante os testes. 96](#_Toc393138741)

LISTA DE ABREVIATURAS E SIGLAS

3D – Três dimensões

BSD - Berkeley Software Distribution

CSS - Cascading Style Sheets

HTML - Hypertext Markup Language

HTTP - Hypertext Transfer Protocol

IBGE – Instituto Brasileiro de Geografia e estatística

LoG - Laplacian of Gaussian

MER – Modelo Entidade Relacionamento

OCR – Reconhecimento ótico de caracteres

OpenCV – Open source computer vision library

PDF – Portable document format

QR Code – Quick response code

RGB – Red Green Blue

SIFT - Scale Invariant Feature Transform

SO - Sistema Operacional

**SUMÁRIO**

[1. INTRODUÇÃO 18](#_Toc393138573)

[1.1. OBJETIVOS GERAIS 19](#_Toc393138574)

[1.2. OBJETIVOS ESPECÍFICOS 20](#_Toc393138575)

[1.3. JUSTIFICATIVA 21](#_Toc393138576)

[2. VISÃO COMPUTACIONAL 22](#_Toc393138577)

[2.1. INTRODUÇÃO 22](#_Toc393138578)

[2.2. SISTEMA VISUAL HUMANO 22](#_Toc393138579)

[2.3. ESTADO DA ARTE DA VISÃO COMPUTACIONAL 25](#_Toc393138580)

[2.3.1. História 25](#_Toc393138581)

[2.3.2. Definição e objetivos 26](#_Toc393138582)

[2.4. ETAPAS DE UM SISTEMA DE VISÃO COMPUTACIONAL 27](#_Toc393138583)

[2.4.1. Aquisição de imagem 27](#_Toc393138584)

[2.4.2. Pré-processamento 28](#_Toc393138585)

[2.4.3. Extração de características 28](#_Toc393138586)

[2.4.4. Detecção e segmentação 28](#_Toc393138587)

[2.4.5. Processamento de alto nível 29](#_Toc393138588)

[2.5. ÁREAS DE APLICAÇÃO 29](#_Toc393138589)

[2.5.1. Medicina 30](#_Toc393138590)

[2.5.2. Setor industrial 30](#_Toc393138591)

[2.5.3. Reconhecimento ótico de caracteres 31](#_Toc393138592)

[2.5.4. Biometria 31](#_Toc393138593)

[2.6. PROBLEMAS E DIFICULDADES DA VISÃO COMPUTACIONAL 32](#_Toc393138594)

[2.6.1. Detecção de bordas 32](#_Toc393138595)

[2.6.2. Reconhecimento de objetos 32](#_Toc393138596)

[2.6.3. Visão estereoscópica 33](#_Toc393138597)

[2.7. CONCLUSÃO 33](#_Toc393138598)

[3. RECONHECIMENTO DE PADRÕES 34](#_Toc393138599)

[3.1. INTRODUÇÃO 34](#_Toc393138600)

[3.2. CONCEITO DE RECONHECIMENTO DE PADRÕES 34](#_Toc393138601)

[3.2.1. Padrão 38](#_Toc393138602)

[3.2.2. Classe 38](#_Toc393138603)

[3.3. ESTRUTURA DE SISTEMA PARA RECONHECIMENTO DE PADRÕES 39](#_Toc393138604)

[3.3.1. Sensoriamento (medição) 40](#_Toc393138605)

[3.3.2. Pré-processamento e segmentação 40](#_Toc393138606)

[3.3.3. Extração de características 41](#_Toc393138607)

[3.3.4. Classificação 42](#_Toc393138608)

[3.3.5. Pós-processamento 42](#_Toc393138609)

[3.4. ÁREAS DE APLICAÇÃO 43](#_Toc393138610)

[3.5. CONCLUSÃO 44](#_Toc393138611)

[4. OPENCV 45](#_Toc393138612)

[4.1. INTRODUÇÃO 45](#_Toc393138613)

[4.2. INTRODUÇÃO AO OPENCV 45](#_Toc393138614)

[4.3. APLICAÇÕES 47](#_Toc393138615)

[4.4. CONCLUSÃO 47](#_Toc393138616)

[5. ALGORITMOS 48](#_Toc393138617)

[5.1. INTRODUÇÃO 48](#_Toc393138618)

[5.2. SIFT 48](#_Toc393138619)

[5.2.1. Breve demonstração de implementação do sift 50](#_Toc393138620)

[5.2.2. Etapas de implementação do sift: construção de escala 52](#_Toc393138621)

[5.2.3. Etapas de implementação do sift: detecção de extremos 53](#_Toc393138622)

[5.2.4. Etapas de implementação do sift: detecção de pontos-chave 55](#_Toc393138623)

[5.2.5. Etapas de implementação do sift: descritor de pontos-chave 57](#_Toc393138624)

[5.3. TEMPLATE MATCHING 57](#_Toc393138625)

[5.3.1. funcionamento do template matching 58](#_Toc393138626)

[5.4. ADAPTATIVE THRESHOLD 61](#_Toc393138627)

[5.5. FIND CONTOURS 62](#_Toc393138628)

[5.6. BOUNDING RECT 63](#_Toc393138629)

[5.7. HOUGHCIRCLES 63](#_Toc393138630)

[5.8. CONCLUSÃO 64](#_Toc393138631)

[6. FERRAMENTAS EXISTENTES NO MERCADO 65](#_Toc393138632)

[6.1. INTRODUÇÃO 65](#_Toc393138633)

[6.2. QUESTÕES NA WEB 65](#_Toc393138634)

[6.2.1. Pontos fortes e fracos 67](#_Toc393138635)

[6.3. GRADECAM 67](#_Toc393138636)

[6.3.1. Pontos fortes e fracos 69](#_Toc393138637)

[6.4. CONCLUSÃO 69](#_Toc393138638)

[7. ESTUDO DE CASO 70](#_Toc393138639)

[7.1. INTRODUÇÃO 70](#_Toc393138640)

[7.2. ARQUITETURA GERAL DO CORRETOR DE AVALIAÇÕES 70](#_Toc393138641)

[7.3. Limitações do projeto 73](#_Toc393138642)

[7.4. MODELAGEM DE DADOS 73](#_Toc393138643)

[7.5. APLICAÇÃO WEB 75](#_Toc393138644)

[7.6. APLICATIVO PARA SMARTPHONE 80](#_Toc393138645)

[7.7. WEB SERVICE 84](#_Toc393138646)

[7.8. ALGORITIMO PARA CORREÇÃO DO GABARITO 85](#_Toc393138647)

[7.9. APLICAÇÃO 93](#_Toc393138648)

[8. CONSIDERAÇÕES FINAIS e trabalhos futuros 97](#_Toc393138649)

[9. REFERÊNCIAS 99](#_Toc393138650)

# 

# INTRODUÇÃO

A área de visão computacional, mais precisamente de reconhecimento de padrões, tem sido objeto de estudo e desenvolvimento devido ao seu grande número de aplicações e dada sua relevância principalmente para a automatização de tarefas do dia-a-dia e também pela necessidade crescente de que a máquina se torne cada vez mais parecida com o homem.

Visão computacional tem se mostrado como ferramenta fundamental em vários ramos como, por exemplo, na robótica onde sistemas autônomos são atualmente capazes de interpretar e tomar decisões a partir de imagens, na indústria, onde a automação vem retirando cada vez mais o homem de tarefas repetitivas, aumentando a agilidade e melhorando a qualidade da produção.

O universo de Visão Computacional cada vez mais ganha espaço e relevância, aumentando assim o número de técnicas e algoritmos para o desenvolvimento e implementação de aplicações voltadas para essa área.

Esta monografia apresenta um protótipo que se utiliza da tecnologia da área de Visão Computacional para apresentar uma proposta de um corretor automático de avaliações do tipo múltipla escolha utilizando para seu desenvolvimento a biblioteca *OpenCV*.

Este trabalho está organizado conforme descrito a seguir.

O primeiro capítulo apresenta brevemente o tema do trabalho, objetivos geral e específicos, a problemática abordada bem como a relevância para o desenvolvimento do mesmo.

No Capítulo 2 o tema de visão computacional é abordado, incluindo exemplos e aplicações.

No Capítulo 3 são abordados conceitos de reconhecimento de padrões, ponto chave para o desenvolvimento deste trabalho.

No Capítulo 4 a biblioteca *OpenCV* é apresentada, com suas diversas aplicações.

No Capítulo 5 são descritos alguns dos algoritmos e métodos existentes na área de Visão Computacional e que foram testados e utilizados no desenvolvimento deste trabalho.

O Capítulo 6 apresenta algumas das ferramentas existentes no mercado que possuem proposta similar à apresentada por este trabalho.

O Capítulo 7 descreve o estudo de caso, com todo o procedimento, limitações do projeto, tecnologias e ferramentas utilizadas para chegar à conclusão deste trabalho de pesquisa.

Finalmente, serão apresentadas as considerações finais e sugestões e as referências bibliográficas utilizadas para o desenvolvimento desta monografia.

## OBJETIVOS GERAIS

O principal objetivo deste trabalho é demonstrar a relevância e benefícios da automatização de correções de avaliações através da apresentação de um protótipo de sistema corretor de avaliações do tipo múltipla escolha, que utiliza para isso técnicas de visão computacional e reconhecimento de padrões.

## OBJETIVOS ESPECÍFICOS

Para atingir o objetivo geral desse trabalho os seguintes pontos foram abordados:

Foi realizada uma pesquisa sobre o funcionamento, onde e como podem ser aplicadas as técnicas de Visão Computacional.

Foram estudados algoritmos para reconhecimento de padrões em imagens e vídeos para a aplicação dos mesmos sobre o gabarito da avaliação a ser corrigido.

Foi realizado um estudo sobre o *framework* de visão computacional OpenCV que disponibiliza em suas bibliotecas os algoritmos que foram utilizados para a realização da correção dos gabaritos de resposta.

Um protótipo de um software foi desenvolvido, para efetuar a correção automática de avaliações no formato múltipla escolha utilizando os conceitos estudados neste trabalho.

## JUSTIFICATIVA

Educadores de todas as áreas, trabalhando em qualquer instituição, têm como parte de sua rotina tarefas como o desenvolvimento, correção e gerenciamento de avaliações de seus alunos. Em muitos casos, professores trabalham em mais de uma instituição de ensino, consequentemente possuem mais de uma disciplina, mais de uma turma e, por consequência, essas atividades multiplicam o esforço e tempo que são consumidos por eles.

Por conta disso, as tarefas de criar as avaliações e as corrigir, consideradas básicas para qualquer professor, se tornam exaustivas e também suscetíveis a falhas e erros. Além disso, o consumo de tempo dedicado à realização dessas tarefas é alto e poucas são as ferramentas de auxílio aos educadores.

No Brasil, o número de pessoas que possuem *smarphones,* vem crescendo significantemente. Esta maior facilidade de acesso impulsionou o desenvolvimento aplicações que facilitam e/ou automatizam tarefas diárias. Seguindo essa tendência é que foi proposto o desenvolvimento de um sistema computacional direcionado a automatização das tarefas de criação e correção de gabaritos de avaliações de múltipla escolha, através de câmeras de *smartphones,* para o melhor aproveitamento de tempo, além de diminuir a probabilidade de ocorrerem erros durante correções das avaliações.

Do ponto de vista prático, espera-se que o estudo contribua no sentido de ampliar os conhecimentos e possibilidades de sistemas de auxilio voltados à automação de correção de avaliações no formato de múltipla escolha.

# VISÃO COMPUTACIONAL

## INTRODUÇÃO

Visão computacional é a área da ciência computacional responsável pela visão de máquina (MILANO; HONORATO, 2010). É responsável pelo estudo da extração de informações de objetos de uma imagem e foca-se explicitamente na obtenção e manipulação de dados de uma imagem e no uso destes para diferentes propósitos (RIOS, 20??). Em resumo, a visão computacional é definida como a técnica de “ensinar computadores a enxergarem como humanos” (GREENEMEIER, 2008).

## SISTEMA VISUAL HUMANO

O sistema visual humano é uma área análoga ao campo de visão computacional, devido ao fato de que enquanto a primeira estuda processos psicológicos envolvidos na formação e percepção de imagens por humanos, a outra estuda os processos e algoritmos usados por máquinas para enxergar (RIOS, 20??).

O sistema visual humano pode ser dividido em três principais locais de processamento: Olhos, que agem como entrada para o sistema visual, caminhos visuais, responsáveis por transportar e organizar a informação visual e o córtex visual, que é a parte do cérebro que gera a percepção da imagem (Figura 1) (LUEBKE et al., 2003).

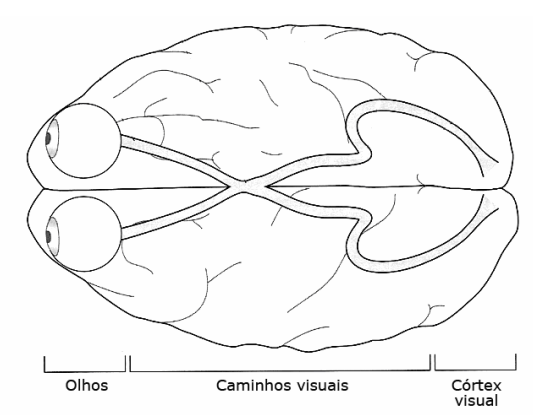


Figura 1- Ilustração do sistema visual humano.

**Fonte: Adaptado de Luebke et al, 2003, p. 240**

O olho humano é semelhante a câmeras fotográficas (Figura 2) e filmadoras de vídeo. Assim como esses dois equipamentos, o olho possui três componentes essenciais: orifício que controla a entrada da luz (íris, que abre e fecha a pupila), uma lente para melhor focar a luz (cristalino, que foca objetos próximos ou distantes mudando sua curvatura) e um elemento capaz de fazer o registro dessa imagem (cérebro, que interpreta e registra sensações visuais) (Instituto de Física da USP, 1998).

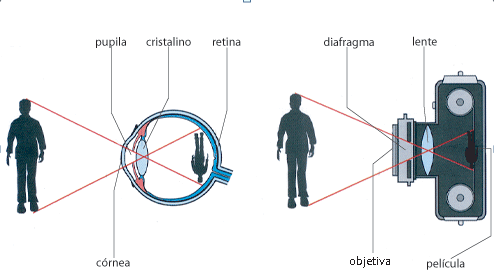


Figura 2 – Analogia olho humano e Câmera fotográfica.

**Fonte: Adaptado de Centro oftalmológico do nordeste, 2013.**

Segundo Milano e Honorato (2010), é através do córtex visual no cérebro que o olho humano consegue interpretar objetos em imagens de forma tão rápida, sendo essa uma das partes mais complexas no sistema de processamento cerebral. Alguns cientistas concentram seus estudos na tentativa de entender o funcionamento dessa parte do cérebro, para então trazer tais ideias para a visão computacional.

## ESTADO DA ARTE DA VISÃO COMPUTACIONAL

### HISTÓRIA

As primeiras aplicações nas categorias de visão computacional e processamento de imagens ocorreram na década de 1920 (MARQUES FILHO; VIEIRA NETO, 1999), mas avanços notáveis nas áreas relacionadas ao processamento de imagens tiveram início na década de 1960, com o início dos programas espaciais Norte Americano e Russo que geraram grandes quantidades de dados de imagem e demandaram estudos para a melhoria de visualização e utilização dos mesmos, principalmente a partir de imagens da lua que eram transmitidas por uma sonda e processadas por computadores para a correção de distorções dessas imagens. Essas técnicas foram base para métodos aprimorados de realce e restauração de imagens de outros programas espaciais posteriores (MORRIS, 2003).

Da década de 1960 aos dias atuais, a área de processamento de imagens vem apresentando crescimento expressivo e suas aplicações abrangem vários ramos da atividade humana (MARQUES FILHO; VIEIRA NETO, 1999).

De acordo com Morris (2003), a diminuição do custo de *hardware* e o grande avanço tecnológico é que possibilitaram que computadores cada vez mais poderosos chegassem ao mercado. Computadores esses suficientemente poderosos para serem facilmente utilizados no processamento de imagens ou aplicações de visão computacional, fazendo com que seu uso, anteriormente restrito ao governo, universidades e laboratórios indústrias, passasse ao uso prático na vida cotidiana.

### DEFINIÇÃO E OBJETIVOS

A visão computacional, de acordo com Ballard e Brown (1982), é a ciência relacionada com a automatização e integração de uma grande variedade de processos e representações usadas para a percepção visual. A visão computacional inclui técnicas como, por exemplo, o processamento de imagens (transformação, codificação e transmissão de imagens) e a classificação de padrões estatísticos.

Sistemas que processam imagens tentam simular o comportamento da visão biológica. Computadores trabalham no processamento dessas imagens assim como o cérebro faz com as imagens que são captadas pelos olhos. Sendo assim os sistemas de visão computacional precisam de dispositivos (câmeras) que transforme as imagens reais em imagens que um computador possa processar (imagem digital) (MANZI, 2007).

O objetivo da visão computacional é a tomada de decisões úteis sobre objetos físicos reais e cenas baseadas em imagens sensoriais (SHAPIRO; STOCKMAN, 2000). Ela busca gerar descrições inteligentes e úteis de cenas e sequência visuais, e seus objetos, através da realização de operações sobre os sinais recebidos de câmeras (DAUGMAN, 2013).

Outro objetivo da visão computacional é buscar um modelo de representação que se aproxime da visão biológica (MANZI, 2007).

## ETAPAS DE UM SISTEMA DE VISÃO COMPUTACIONAL

Não existe um modelo padrão para a implementação das aplicações de visão computacional, entretanto, as principais operações a serem efetuadas sobre uma imagem são: aquisição de imagem, pré-processamento, extração de características, detecção e segmentação e processamento de alto nível (MILANO; HONORATO, 2010). Essas etapas serão detalhadas a seguir.

### AQUISIÇÃO DE IMAGEM

A fase de aquisição de imagem é o primeiro passo para um sistema de visão computacional. É como o próprio nome já adianta: o processo de aquisição de imagem, que ocorre a partir de sensores de câmeras, onde os pixels de cada imagem obtida indicam as coordenadas de luz e propriedades físicas. (MILANO; HONORATO, 2010).

### PRÉ-PROCESSAMENTO

A segunda etapa é a fase de pré-processamento, que tem como principal função melhorar a qualidade da imagem para as etapas seguintes (MARQUES FILHO; VIEIRA NETO, 1999). Nesta etapa, são aplicados métodos a fim de facilitar a identificação de um objeto, por esse motivo, este é um processo realizado antes de se obter informações de uma imagem (MILANO; HONORATO, 2010).

### EXTRAÇÃO DE CARACTERÍSTICAS

De acordo com Milano e Honorato (2010), a fase de extração de características é a etapa responsável pela extração de características matemáticas da imagem através de descritores que permitem caracterizar cada dígito com precisão. Faz-se importante ressaltar que a entrada desta etapa é uma imagem, porém a saída é um conjunto de dados que correspondem a essa imagem (MARQUES FILHO; VIEIRA NETO, 1999).

### DETECÇÃO E SEGMENTAÇÃO

É na etapa de detecção e segmentação que as regiões da imagem, que são relevantes para a aplicação, são destacadas baseadas em suas características (MILANO; HONORATO, 2010). A tarefa básica da segmentação é justamente a divisão da imagem em unidades significativas. Esta etapa, apesar de simples, é uma das mais complexas de ser implementada (MARQUES FILHO; VIEIRA NETO, 1999).

### PROCESSAMENTO DE ALTO NÍVEL

A última etapa refere-se ao processamento de alto nível, onde algoritmos são executados para a validação da satisfação dos dados, estimativa de parâmetros sobre a imagem e a classificação das características dos objetos obtidos. Essa etapa consiste em atribuir significado aos objetos que foram reconhecidos (MARQUES FILHO; VIEIRA NETO, 1999) (MILANO; HONORATO, 2010).

## ÁREAS DE APLICAÇÃO

As aplicações de visão computacional têm apresentado um crescimento muito expressivo e em quase todos os ramos da atividade humana, por ser um campo bastante amplo e por se referir ao mecanismo de percepção mais fundamental existente (MARQUES FILHO; VIEIRA NETO, 1999) (MILANO; HONORATO, 2010).

Tendo em vista que a visão computacional tem um campo enorme de abrangência, a seguir, serão mencionadas algumas das principais áreas de aplicação desta tecnologia.

### MEDICINA

A medicina é uma das áreas onde o emprego de sistemas de visão computacional é mais relevante. Seu uso pode ser exemplificado com a realização de diagnostico a partir do recebimento de uma imagem de um paciente, como um raio X ou uma tomografia. Outro exemplo de aplicação é o estudo do movimento do corpo humano por visão computacional, para ajudar pacientes com deficiências motoras (RIOS, 20??).

### SETOR INDUSTRIAL

A visão de máquina é empregada geralmente nos setores industriais, mais especificamente a área de controle de qualidade. A utilização de sistemas de visão computacional no controle de qualidade em indústrias permite que o processo de inspeção, anteriormente manual, torne-se mais rápido e consequentemente mais barato, com menor probabilidade de erros. É um dos campos mais antigos onde a visão de máquina computacional é empregada. (SOUZA; CAPOVILLA; ELEOTÉRIO, 2010) (RIOS, 20??).

### RECONHECIMENTO ÓTICO DE CARACTERES

O reconhecimento de caracteres em uma imagem (OCR) é o processo de ler texto impresso, escrito ou datilografado e transformá-lo em texto digital (RIOS, 20??). Sua aplicação vai desde interpretação de texto impresso ou escrito à mão ao reconhecimento de assinaturas em cheques e registro de placas de carro (MORRIS, 2003).

### BIOMETRIA

A biometria é outra área que faz uso de sistemas de visão computacional. Biometria é a ciência de identificação de pessoas por suas características físicas. E isso quer dizer, no contexto de visão computacional, o reconhecimento de imagens dessas características, por exemplo, o reconhecimento de impressões digitais, reconhecimento de íris, reconhecimento facial, entre outros (MORRIS, 2003).

## PROBLEMAS E DIFICULDADES DA VISÃO COMPUTACIONAL

### DETECÇÃO DE BORDAS

A detecção de bordas (figura 3) é um dos problemas mais importantes encontrados no reconhecimento de objetos em imagens. Basicamente, é a detecção de regiões na imagem onde ocorre uma mudança de brilho da imagem que representam, geralmente, uma mudança das características do que está sendo visto. Esses problemas podem afetar a forma como os dados extraídos são analisados, frequentemente levando a interpretações falsas sobre o conteúdo. (RIOS, 20??).

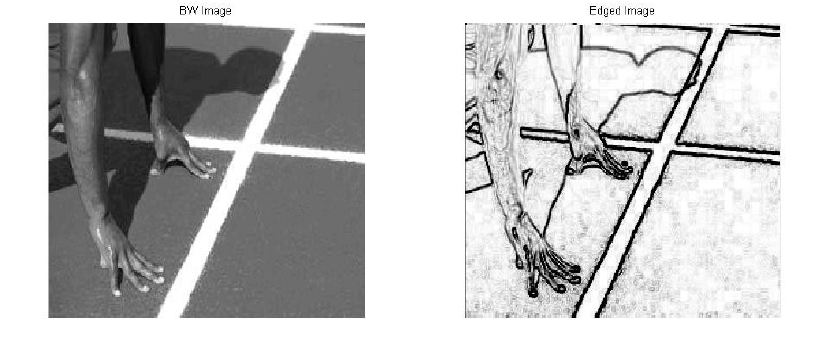


Figura 3 - Exemplo de reconhecimento de borda. Imagem original e apenas suas bordas.

**Fonte: Rios, 20??**

### RECONHECIMENTO DE OBJETOS

O reconhecimento de objetos é a tarefa de reconhecer, numa cena, um objeto predefinido na base de conhecimento ou aprendido. Geralmente, os métodos empregados usam *templates* para gerar o conjunto de bordas do objeto requerido e compara suas bordas com as bordas da imagem (RIOS, 20??).

### VISÃO ESTEREOSCÓPICA

A visão estereoscópica computacional é a extração de dados 3D de uma imagem digital. É um processo que faz analogia à estereopsia nos animais, que são capazes de perceber profundidade combinando imagens que chegam aos dois olhos. Ao realizar a análise de diversas imagens diferentes do mesmo objeto em diferentes ângulos, é possível analisar as mudanças nas perspectivas de uma imagem para a outra, obter representação tridimensional do objeto, dando sensação de profundidade a ele (RIOS, 20??).

## CONCLUSÃO

Neste capítulo foi feita uma breve introdução à tecnologia de visão computacional. Para tanto, um paralelo entre visão computacional e biológica foi traçado, bem como um resumo do estado da arte de visão computacional e suas áreas de aplicação e etapas. Algumas dificuldades e problemas também foram descritos.

# RECONHECIMENTO DE PADRÕES

## INTRODUÇÃO

Reconhecimento de padrões é a técnica que trata da descrição e classificação de objetos. Desde o princípio da computação, a tarefa de implementar algoritmos que emulem a capacidade humana de reconhecer e classificar objetos, se apresenta como uma das mais intrigantes e desafiadoras.

O problema de reconhecimento de padrões surge com muita frequência em diversas áreas e consiste basicamente em determinar se dado objeto (padrão) ocorre em alguma parte de outro objeto (ANDRADE, 1993).

Este capítulo traz uma abordagem ao tema de reconhecimento de padrões, tal como alguns dos algoritmos usados para o reconhecimento de padrões.

## CONCEITO DE RECONHECIMENTO DE PADRÕES

O processo de análise de uma imagem consiste na técnica de aplicar sobre uma imagem transformações sucessivas visando extrair mais facilmente a informação contida nela. As técnicas de análise ou processamento de imagem podem ser divididas em três partes principais (figura 4): processamento de baixo nível, processamento de nível intermediário e processamento de alto nível, onde o reconhecimento de padrões está incluso nesta última (SOUZA, 2011).

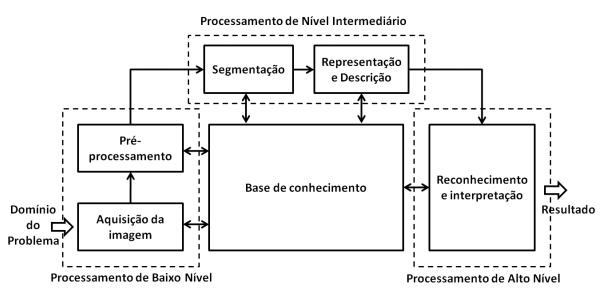


Figura 4 - Elementos do processo de análise de imagem.

**Fonte: Gonzalez; Woods, 2000**

De acordo com Devroye, Györfi e Lugosi (1996), reconhecimento de padrões refere-se à tarefa de adivinhar ou prever a natureza desconhecida de uma observação, como preto ou branco, zero ou um, doente ou saudável, verdadeiro ou falso. Uma observação é um conjunto de medidas numéricas, como uma imagem, por exemplo, que é uma sequência de pixels. O termo reconhecimento de padrões refere-se à tarefa de classificar um objeto baseando-se em suas medidas (TOHKA, 2013).

Esta é a parte onde os sistemas de visão possuem uma intersecção com a área de inteligência artificial (MARENGONI; STRINGHINI, 2009). Visão de máquina é uma área na qual o reconhecimento de padrões é de extrema importância. Um sistema de visão de máquina captura imagens via câmera e as analisa para produzir descrições do que está sendo visualizado (THEODORIDIS; KOUTROUMBAS, 2009).

Para Castro e Prado (2002), o reconhecimento de padrões é uma habilidade extremamente desenvolvida nos animais e seres humanos. O ser humano é capaz de reconhecer rostos, vozes, caligrafias e até emoções de pessoas conhecidas. Sendo assim, pode-se concluir que padrões são os meios pelos quais o mundo é interpretado e a partir destas interpretações é que se tomam atitudes e decisões. Nota-se também que a facilidade no reconhecimento destes padrões está relacionada aos estímulos em que o indivíduo foi exposto anteriormente, ou seja, está relacionado a aprendizado e evolui com a experiência.

Um sistema que mede certo objeto e em seguida o classifica é chamado de sistema de reconhecimento de padrões. Um exemplo de um sistema de reconhecimento de padrões pode ser um filtro de spam que reconhece automaticamente e-mails maliciosos e os guarda em pastas apartadas da caixa de entrada do usuário (TOHKA, 2013). Outro exemplo seria o reconhecimento de rostos em fotos.

Segundo Osório e Bittencurt (2000), a partir de uma imagem original pode ser realizado um processamento que permite analisar esta imagem e identificar seus componentes. Assim, é possível extrair uma descrição de alto nível dos seus componentes, agrupar os componentes que são similares, ou então classificar seus componentes em grupos predefinidos (classes de objetos). Portanto, pode-se afirmar que um sistema de reconhecimento de padrões é basicamente composto por um sensor, que obtém observações a serem classificadas, o mecanismo de extração das características e por fim, o esquema de classificação destes dados, cujos resultados serão categorizados de acordo com seus padrões. De acordo com Tohka (2013), a tarefa do sistema de reconhecimento de padrões é classificar um objeto em uma classe correta com base nas medições sobre este objeto. A figura 5 ilustra esse processo de funcionamento de sistemas de reconhecimento de padrões.

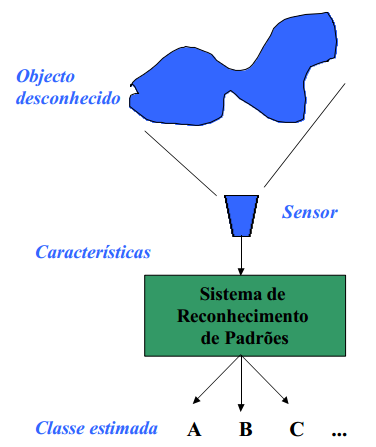


Figura 5 – Sistema de reconhecimento de Padrões.

**Fonte: Fred, 2001**

Segundo Marengoni e Stringhini (2009), o reconhecimento de objetos é umas das principais funções da área de visão computacional e está relacionado diretamente com o reconhecimento de padrões. Um objeto é geralmente definido por mais de um padrão (textura, forma, cor, dimensão etc.) e o reconhecimento de cada um destes padrões pode facilitar o reconhecimento do objeto como um todo.

Algumas áreas que utilizam o reconhecimento de padrões são: identificação através de impressões digitais e análise da íris, diagnóstica médicos, visão computacional, alguns diagnósticos de câncer, reconhecimento de voz, processamento de imagens, análise de caracteres manuscritos, análise de eletrocardiogramas, dentre outras (CASTRO; PRADO, 2002).

### PADRÃO

Padrão pode ser definido como as propriedades que possibilitam o agrupamento de objetos semelhantes dentro de uma determinada classe ou categoria, mediante a interpretação de dados de entrada, que permitem a extração das características desses objetos (CASTRO; PRADO, 2002). Em outras palavras, padrões envolvem uma série de características que se repetem e podem ser identificadas, por exemplo, sinal de voz, rosto humano, imagens etc.

### CLASSE

Classe de um padrão pode ser definida como um conjunto de atributos comuns aos objetos de estudo, ou seja, o agrupamento das características que diferentes objetos possuem em comum (CASTRO; PRADO, 2002).

Sendo assim, reconhecimento de padrões pode ser definido como um procedimento que busca a identificação de estruturas em dados de entrada comparados a estruturas conhecidas e sua classificação dentro de categorias (CASTRO; PRADO, 2002).

## ESTRUTURA DE SISTEMA PARA RECONHECIMENTO DE PADRÕES

De acordo com Castro e Prado (2002), um sistema para reconhecimento de padrões envolve três grandes etapas: representação de dados de entrada e sua mensuração, extração das características e a identificação e classificação do objeto de estudo.

A figura 6 ilustra com mais detalhes as fases do reconhecimento de padrões.

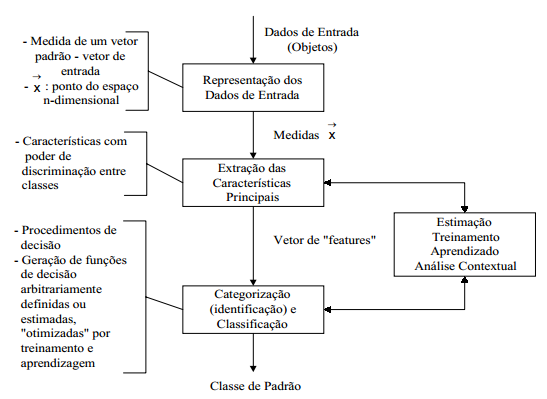


Figura 6 - Fases do reconhecimento de padrões.

**Fonte: Castro; Prado, 2002**

Dentro destas três grandes etapas do reconhecimento de padrão, Tohka (2013) destaca que existem cinco estágios que a maioria dos sistemas para reconhecimento de padrões possuem: Sensoriamento (medição), pré-processamento e segmentação, extração de características, classificação e pós-processamento. Estes estágios serão brevemente descritos, apesar de serem específicos da aplicação, ou seja, variam de acordo com a mesma.

### SENSORIAMENTO (MEDIÇÃO)

Para Castro e Prado (2002) é o estágio que descreve os padrões característicos do objeto, para a posterior classificação em uma classe. Tohka (2013) afirma que muitas vezes uma medição inclui informações de diversos objetos a serem classificados, por exemplo, para reconhecer um endereço escrito em um envelope, será necessário classificar alguns caracteres para que seja possível reconhecer o endereço todo. Os dados, neste caso, provavelmente serão uma imagem de um envelope incluindo as sub-imagens de todos os caracteres a serem classificados mais o plano de fundo que não tem nada a ver com a tarefa de reconhecimento de padrões.

### PRÉ-PROCESSAMENTO E SEGMENTAÇÃO

Segundo Tohka (2013), o estágio de pré-processamento refere-se à filtragem dos dados crus para a eliminação de ruídos e outras operações realizadas sobre os dados brutos para melhorar a sua qualidade (figura 7). Já segmentação, para Consi e Monteiro (200?), consiste num processo cujo objetivo é separar em regiões distintas que possuem o mesmo conteúdo no contexto de uma aplicação. Usando o mesmo exemplo do tópico anterior, a imagem do endereço inteiro precisa ser dividida em outras imagens que representem caracteres individualmente. O resultado da segmentação pode ser representado como um vetor que é chamado de *“pattern vector”*.

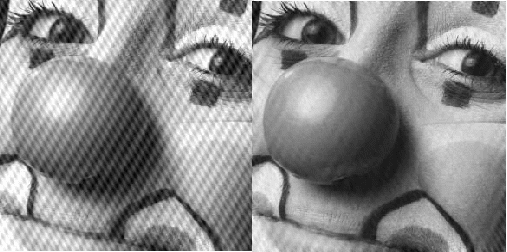


Figura 7 - Filtragem de ruidos. Esquerda:Imagem com ruido; direita : mesma imagem após filtragem.

**Fonte: Souza; Capovilla; Eleotério, 2010.**

### EXTRAÇÃO DE CARACTERÍSTICAS

A escolha das características é de suma importância para o bom desempenho do classificador, portanto essa escolha é feita a partir do que se pretende classificar, sendo assim, exige um conhecimento sobre o problema em estudo (CASTRO; PRADO, 2002). Os objetivos básicos são: a redução da dimensionalidade do vetor característico, sem que isso implique em perda de informação que possa ser relevante para a classificação, objetivando a redução do esforço computacional e a seleção das características significativas para a tarefa de classificação. A figura 8 ilustra a etapa de extração de características aplicada à identificação de impressões digitais.



Figura 8 - Extração de Características. Impressão digital e sua imagem direcional.

**Fonte: Revista Ubiquidade, 2011**

### CLASSIFICAÇÃO

Envolve a determinação de procedimentos para a identificação e classificação do objeto em uma classe de objetos, diferente do estágio anterior, independe da natureza do problema. O extrator de características tem como função determinar e extrair as características mais significativas que contribuam para a descrição do objeto, dentre as infinitas características que possam descrevê-lo, além disso, o extrator de características varia com o sistema a ser analisado. (CASTRO; PRADO, 2002). Uma vez extraídas as características é necessária a classificação do objeto. Esta classificação indica a designação do objeto a uma determinada classe, dentre as várias que se apresentam. É nesta etapa que o classificador aprende a distinguir dentre as classes analisadas, a qual delas o objeto pertence.

### PÓS-PROCESSAMENTO

Segundo Tohka (2013), a tarefa final do sistema de reconhecimento de padrões é decidir sobre uma ação com base no resultado da classificação. De volta ao exemplo de reconhecimento de endereço, tem-se o resultado da classificação para múltiplos caracteres e a tarefa é decidir, ou descobrir, o endereço que esses caracteres formam. Portanto, é possível utilizar o contexto, neste caso, o uso de outros resultados da classificação para corrigir um possível erro de classificação. Supondo que o resultado da classificação tenha sido *‘Hollywoud Boulevard’*, em seguida, o endereço é provavelmente ‘*Hollywood Boulevard’*. Ou seja, o estágio de pós-processamento é o de decisão do que se fará agora com a informação já trabalhada e classificada.

## ÁREAS DE APLICAÇÃO

As áreas de aplicação de reconhecimento de padrões são as mais diversas, devido ao fato de ser uma tecnologia em ascensão e da necessidade em diversas áreas de fazer com que pessoas e máquinas se comuniquem de forma cada vez mais parecida, para usar poder computacional em tarefas do dia a dia ou que sejam naturais para o homem.

De acordo com Orrú (200?), alguns dos exemplos de áreas de aplicação das técnicas de reconhecimento de padrões mais notórios são:

* Medicina: análise de imagens para diagnóstico médico, classificação de doenças;
* Polícia e investigação: detecção criminal a partir da fala, escrita manual, impressões digitais, fotografias;
* Comunicação homem-máquina: reconhecimento automático de fala, reconhecimento da escrita, compreensão de fala, processamento da linguagem natural;
* Defesa: reconhecimento automático de alvos, orientação e controle;
* Estudo e estimativa de recursos naturais: agricultura, extrativismo, geologia, ambiente;
* Indústria: teste e montagem de produtos, controle e inspeção de qualidade;
* Bioinformática: análise de sequências do genoma
* Mineração de dados (*data mining*): a busca por padrões significativos obtidos de grandes bases de dados e *data warehouses*.
* Classificação de documentos da Internet;
* Busca e classificação em base de dados multimídia;
* Reconhecimento biométrico, incluindo faces, íris ou impressões digitais.

## CONCLUSÃO

Neste capítulo foi feita uma breve introdução à tarefa de reconhecimento de padrões. Para tanto, foi apresentada uma visão geral do assunto, suas áreas de aplicação, bem como as competentes etapas e estágios de sistemas para reconhecimento de padrões.

# OPENCV

## INTRODUÇÃO

Visão computacional é uma área da computação em rápido crescimento, devido ao barateamento e aperfeiçoamento de câmeras, melhor poder de processamento e também devido ao amadurecimento de algoritmos de visão (BRADSKI; KAEHLER, 2008). O OpenCV (*open source computer vision library*) teve grande relevância no crescimento de técnicas de visão computacional por permitir que cada vez mais pessoas possam realizar trabalhos em visão.

O OpenCV ajuda na implementação de projetos e iniciação de pesquisas provendo infraestrutura de aprendizagem de máquina e visão computacional que anteriormente estavam apenas disponíveis em alguns modernos laboratórios de pesquisa (BRADSKI; KAEHLER, 2008).

Este capítulo traz uma visão geral da biblioteca OpenCV, tal como aplicações e algoritmos.

## INTRODUÇÃO AO OPENCV

O OpenCV é uma biblioteca de visão computacional e processamento de imagens *open source* escrita em C e C++ que roda em LINUX, Windows e MAC OS, iOS e Android, e interface com Python, Ruby, Matlab, entre outras linguagens (BRADSKI; KAEHLER, 2008).

OpenCV foi criada para manter uma infraestrutura para aplicações em visão computacional e acelerar o uso de percepção de máquina. Por ser um produto distribuído sob a licença BSD[[1]](#footnote-1) e, portanto, ser grátis tanto para uso acadêmico quanto comercial, é mais fácil de utilizar e modificar o código (OpenCV, 200?a).

Segundo Bradski e Kaehler (2008) a OpenCV foi projetada visando eficiência computacional com forte foco em aplicações em tempo real. Um de seus objetivos é prover um uso simples a infraestrutura de visão computacional que ajude as pessoas a construírem de forma rápida aplicações sofisticadas de visão.

De acordo com o site oficial da biblioteca OpenCV (200?a), essa biblioteca possui mais de 2500 algoritmos otimizados, que incluem um conjunto abrangente de algoritmos de visão computacional e aprendizado de máquina. Esses Algoritmos podem ser usados para detectar e reconhecer faces, identificar objetos, classificar ações em vídeos, encontrar similaridades entre imagens num banco de dados, seguir movimentos de olhos, entre muitas outras aplicações.

Ainda segundo o site oficial da OpenCV(200?a), o *download* dessa biblioteca já supera os sete milhões e a comunidade de usuários possui mais de 47 mil membros até a data desta pesquisa. A biblioteca é utilizada por empresas, grupos de pesquisa e órgãos governamentais. A licença da biblioteca permite a criação de produtos comerciais a partir do uso parcial ou total da biblioteca, ou seja, não há a obrigação de que o produto final que seja *open source* só porque a biblioteca usada para sua construção é.

## APLICAÇÕES

A biblioteca OpenCV contém mais de 500 funções que abrangem muitas áreas em visão, incluindo inspeção de produtos em fábricas, imagens médicas, segurança, interface com usuário, calibração de câmeras e visão robótica (BRADSKI; KAEHLER, 2008).

Algumas das aplicações mais relevantes do OpenCV já registradas, de acordo com Bradski e Kaehler (2008) são: a junção de imagens à mapas e satélites, alinhamento de imagens digitalizadas, redução de ruídos em imagens médicas, análise de objetos, sistemas de segurança e detecção de intrusão, sistemas automáticos de monitoramento, sistemas de inspeção em fábricas, aplicações militares. Também já foi aplicada em reconhecimento de sons e músicas, onde técnicas de reconhecimento de visão são aplicadas a imagens de espectrogramas de sons.

## CONCLUSÃO

Este capítulo apresentou uma breve introdução sobre a biblioteca *Open source Computer Vision Library* (OpenCV). Foi apresentado brevemente as áreas de aplicação mais relevantes e um breve histórico da tecnologia.

# ALGORITMOS

## INTRODUÇÃO

Este capítulo fará uma abordagem dos métodos avaliados durante a confecção deste trabalho. O estado da arte destes métodos foi pesquisado a fim de determinar quais deles seriam relevantes e/ou aplicáveis para a aplicação prática proposta neste trabalho.

## SIFT

SIFT – *Scale invariant feature transforms* é um método apresentado por Lowe, D.G., (2004) para a extração de características de imagens que podem ser utilizadas para efetuar a correspondência entre os diferentes pontos de um objeto ou cena. Como o próprio nome do método sugere, uma das propriedades mais significantes do método é: características são invariantes às mudanças de escala e rotação, e fornecem correspondência robusta através de uma gama considerável de distorção mudança de ponto de vista em 3D, além de ruído e mudança na iluminação (RIGUAL, 2012).

De acordo com o site oficial da OpenCV (201?b), é um algoritmo que extrai “pontos-chave” e computa seus descritores. Este algoritmo é usado principalmente para a detecção de objetos em imagens através de características destas imagens. O uso do SIFT está relacionado ao ponto de correspondência entre os diferentes pontos de vista de uma cena 3D, por esse motivo o SIFT tem se mostrado muito útil para a correspondência de imagem e reconhecimento de objetos em condições do mundo real (LOWE; 19??). Segundo Rigual (2012), o descritor SIFT ainda parece o descritor mais atraente para usos práticos, e, portanto, também o mais utilizado hoje em dia.

Para entender o SIFT, é importante distinguir entre características (*features*), pontos-chave (*keypoints*)e descritor (*descriptor*):

* Características: Parte característica, evidente ou proeminente de uma imagem.
* Pontos-chave: Armazenam informações para localizar as características da imagem. Possuem coordenadas X e Y (posição da característica dentro da imagem), escala e orientação.
* Descritor: é um vetor para cada ponto-chave. É altamente distinto e parcialmente invariante à iluminação e ponto de vista em 3D. Contém 128 elementos que com valor de 0-255.

Existem quatro etapas principais envolvidas no algoritmo SIFT (LOWE, 2004). Primeiramente, este método encontra as características das imagens para comparar. Uma vez que as características são determinadas, é calculado um descritor para cada característica, chamado descritor SIFT. Dois descritores devem ser semelhantes, se e somente se as suas características correspondentes forem semelhantes também. Em seguida todas as características de ambas as imagens são comparadas com as características da outra imagem. A correspondência é feita através da comparação dos descritores. Finalmente, para saber se um objeto é encontrado em ambas as imagens (objetos devem ter um grande número de características), correspondências positivas são testadas usando métodos de montagem robustos (RIGUAL, 2012).

### BREVE DEMONSTRAÇÃO DE IMPLEMENTAÇÃO DO SIFT

Sinha (2010) a título de ilustração usou SIFT para encontrar as imagens da na . As imagens se relacionam perfeitamente, mesmo que algumas transformações de escala e rotação tenham sido aplicadas, como pode ser visto na , onde os retângulos grandes marcam as imagens correspondentes. Os quadrados menores marcam as características individuais encontradas nessas regiões.

.

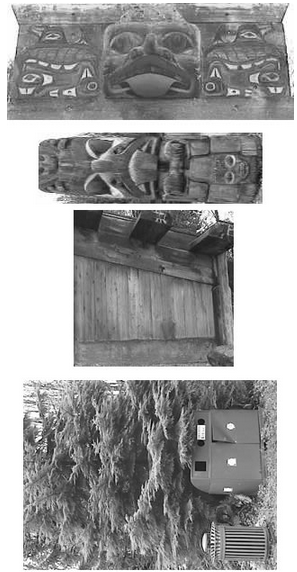


Figura 9 - Imagens a serem encontradas. Correspondem a algumas regiões na imagem teste, mas com algumas transformações aplicadas (rotação e escala).

**Fonte: Rigual, 2012.**



Figura 10 – Cenário em que as imagens serão encontradas.

**Fonte: Rigual, 2012.**

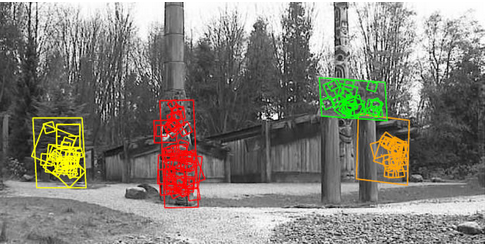


Figura 11- Imagem resultado Resultado.

**Fonte: Rigual, 2012.**

### ETAPAS DE IMPLEMENTAÇÃO DO SIFT: CONSTRUÇÃO DE ESCALA

Segundo Rigual (2012), esta é a preparação inicial. Onde são criadas representações internas da imagem original para garantir a invariância de escala. Isto é feito pela geração de um "espaço de escala". A obtenção de imagens menos detalhadas permite o método olhar para objetos de forma mais geral ignorando as pequenas características. *Gaussian blur* alcança esse efeito e não acrescenta novos dados falsos. O *GaussianBlur()* desfoca uma imagem usando um filtro Gaussiano. Um exemplo pode ser visto na . A imagem original é redimensionada para metade do tamanho de cada vez antes de aplicar o *Gaussian Blur*. Imagens do mesmo tamanho formam uma oitava. A ilustra quatro oitavas com cinco imagens cada (a primeira oitava teve de ser cortada por motvos de espaço). As imagens individuais são formadas devido ao aumento da "escala" (quantidade de desfoque). Lowe (2004), o autor do SIFT, recomenda trabalhar com quatro oitavas e cinco níveis de *blur* (desfoque).



Figura 12 – Imagens progressivamente desfocadas, resultado da aplicação do *Gaussian Blur* várias vezes na mesma imagem.

**Fonte: Rigual, 2012.**

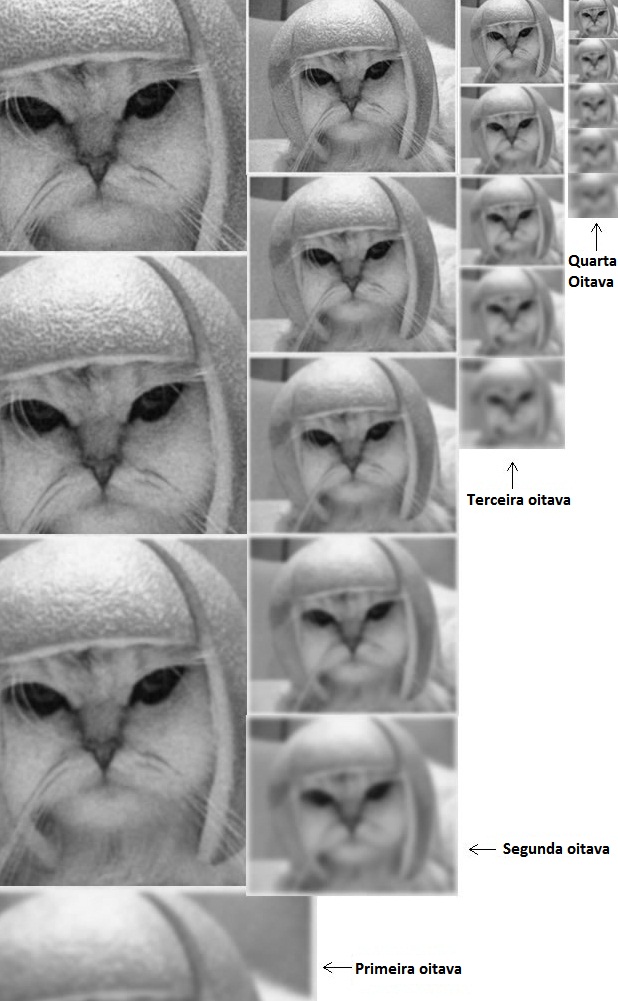


Figura 13 - A imagem original redimensionada para metade do seu tamanho cada vez que o *Gaussian Blur* é aplicado.

**Fonte: Adaptado de Sinha, 2010.**

### ETAPAS DE IMPLEMENTAÇÃO DO SIFT: DETECÇÃO DE EXTREMOS

O Laplacian de Gauss (LoG) é um filtro de detecção de bordas e cantos nas imagens (OPENCV, 200?b), é usado para encontrar pontos-chave em uma imagem (SINHA, 2010). Duas imagens consecutivas em uma oitava são escolhidas e uma é subtraída a partir da outra. Em seguida, o mesmo processo é feito para o par seguinte e o processo repete-se para todas as oitavas (RIGUAL; 2012). Um exemplo pode ser visto na .

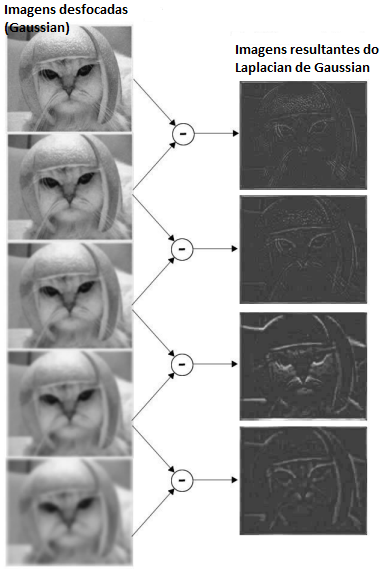


Figura 14 – Passo LoG (Laplacian de Gaussian). Em representação RGB, imagens com valores próximos de zero são exibidos em preto, esta é a razão das imagens serem escuras.

**Fonte: Rigual, 2012.**

### ETAPAS DE IMPLEMENTAÇÃO DO SIFT: DETECÇÃO DE PONTOS-CHAVE

Os pontos-chave são os pontos onde o Laplacian de gaussian obtém os valores máximos e mínimos dos locais. A comparação não é apenas realizada contra os oito pixels vizinhos de uma mesma imagem, mas sim contra os 26 pixels vizinhos a partir das imagens do nível de desfoque *(blur)* anterior e posterior, exemplificado na .

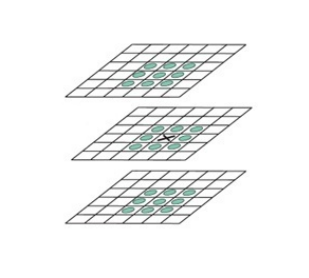


Figura 15 - Pixel atual (marcado com um 'X') e os 26 pixels vizinhos na comparação de máximos/mínimos.

**Fonte: Lowe, 2004.**

Uma vez que um potencial ponto-chave foi encontrado, comparando pixel a pixel com seus vizinhos, segundo Lowe (2004), o próximo passo é a realização de um ajuste detalhado para os dados quanto à localização, escala e proporção das curvaturas principais, para obter a localização mais exata de extremos. Se a intensidade nestes extremos é inferior ao valor de um *threshold* definido, o local é rejeitado. Este limiar é chamado *contrastThreshold* na *OpenCV*. Esta informação permite a rejeição de pontos que tenham baixo contraste (sensíveis ao ruído) ou que estejam mal localizados ao longo de uma borda.

A ilustra essa etapa: a parte (a) da imagem demonstra a imagem original que possui 233x189 pixels. A Parte (b) exibe as primeiras 832 localizações de pontos-chave em máximas e mínimas da função Laplacian de Gauss. Pontos-chave são exibidos como vetores indicando escala, orientação e localização. Parte (c), Após a aplicação de um limiar de contraste mínimo, 729 pontos-chave permanecem. (d) Os 536 pontos-chave que permanecem após um *threshold* adicional nas principais curvaturas.

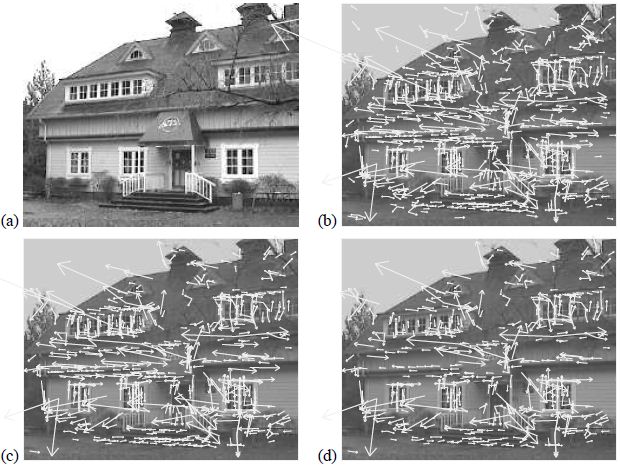


Figura 16 – Etapa de rejeição de pontos sensíveis ao ruído ou que estejam mal localizados.

**Fonte: Lowe, 2004.**

### ETAPAS DE IMPLEMENTAÇÃO DO SIFT: DESCRITOR DE PONTOS-CHAVE

A etapa final se refere à criação de um descritor para cada ponto chave da imagem. Descritores devem ser fáceis de calcular e uma boa representação do ponto chave: dois descritores devem ser semelhantes se e somente se os pontos-chave são semelhantes também. Estes são transformados numa representação que permite níveis significativos de distorção da forma local e as alterações na iluminação (LOWE, 2004).

## TEMPLATE MATCHING

*Template Matching* é um método para pesquisar e encontrar a localização de pequenas partes de uma imagem que corresponde à imagem de um *template.* Segundo Marius, Pennathur e Rose (200?), a ideia do *Template Matching* é realizar cruzamentos entre a imagem recebida e um modelo (*template*)que é representativo da imagem. *O* OpenCV disponibiliza a função cv2.matchTemplate() para esta finalidade (OPENCV, 201?c) *Template Matching* possui diversas aplicações como por exemplo, detecção de face e processamento de imagens médicas.

De acordo com Jain, Duin e Mao (2000), o padrão a ser reconhecido é comparado com o *template* armazenado, levando em consideração todas as posições possíveis e as mudanças de escala. A medida de similaridade pode ser otimizada baseada em conjuntos de treinamento. Muitas vezes, o próprio modelo é aprendido a partir de treinamentos. Portanto, conclui-se que o funcionamento do *template matching* se baseia no prévio treinamento do algoritmo para aprendizado das partes buscadas na imagem.

*Template Matching* exige bastante poder computacional, mas o avanço das tecnologias e disponibilidade de processadores cada vez mais rápidos tem feito esta abordagem mais viável. Embora seja um algoritmo eficaz para o reconhecimento de padrões, o *Template Matching* apresenta algumas desvantagens. Por exemplo, este algoritmo falha quando os padrões estão distorcidos devido ao processamento da imagem, mudança de ponto de vista ou grandes variações de padrões (JAIN; DUIN; MAO, 2000).

### FUNCIONAMENTO DO TEMPLATE MATCHING

De acordo com a documentação oficial da biblioteca *OpenCV* (200?c), inicialmente dois componentes primários são necessários: a imagem na qual espera-se encontrar uma correspondência para a imagem do modelo, chamada de imagem origem (*source image)* e a imagem modelo (*template image),* queé o pedaço da imagem que será comparada buscando sua correspondência na imagem fonte.

A título de ilustração, a demonstra o resultado da detecção do pedaço de imagem a ser detectado na imagem de entrada, onde o objetivo é detectar a área de maior correspondência.



Figura 17 - Imagem origem e imagem *template* e resultado da detecção da correspondência entre as duas.

**Fonte: OpenCV, 200?.**

Para identificar a área correspondente, faz-se necessário comparar a imagem do modelo contra a imagem de origem convolando-a, como ilustrado pela figura 18. Convolar quer dizer que é necessário mover o pedaço da imagem um pixel de cada vez (da esquerda para a direita, de cima para baixo). Em cada local a métrica é calculada para que represente quão “boa” ou “má” é a correspondência naquele local (ou quão similar o pedaço da imagem procurada é, de determinada área da imagem de origem) (OPENCV; 200?c).



Figura 18 - Deslocamento na imagem origem.

**Fonte: OpenCV, 200?.**

Para cada localização da imagem *template* sobre a imagem de destino, uma métrica é armazenada em uma matriz resultado. Cada localização (x, y) na matriz resultado contém a métrica da correspondência. A é o resultado do deslocamento usando uma métrica especifica, onde os locais mais brilhantes representam as maiores correspondências. O local marcado por um círculo vermelho é provavelmente o local com o maior valor de correspondência, de modo que esta localização é considerada a localização correspondente (OPENCV; 200?c).

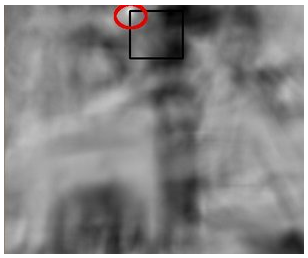


Figura 19 - Imagem resultado do deslocamento da imagem buscada sobre a imagem origem.

**Fonte: OpenCV, 200?.**

## ADAPTATIVE THRESHOLD

*Adaptative Threshold* (*adaptativeThreshold()*) é uma função do *OpenCV* que aplica uma limiar *(threshold)* adaptativa à um *array.* Essa função transforma uma imagem em tons de cinza em uma imagem binária, como apresentado pela (OPENCV, 200?d).

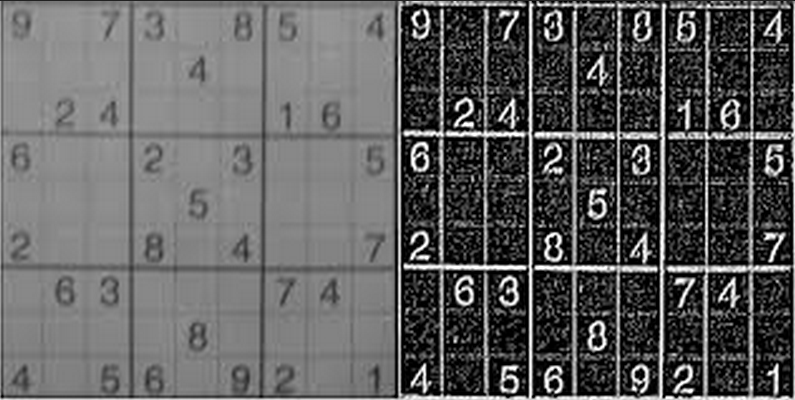


Figura 20 - Imagem binária gerada pela função adaptativeThreshold.

**Fonte: StackOverflow, 2012.**

O *threshold* utilizado para cada pixel é dependente das características da subárea da imagem que contém esse pixel.

## FIND CONTOURS

O conceito de contorno pode ser explicado como a curva que junta os pontos contínuos, que possuem a mesma cor ou intensidade. Os contornos são ferramentas úteis para a análise de forma e detecção e reconhecimento de objetos. *Find Contours (findContours())* é uma função do *OpenCV* que encontra contornos em imagens binárias. Na *OpenCV*, encontrar contornos é o mesmo que encontrar objetos brancos em um fundo preto, ou seja, o contorno a ser encontrado deve, obrigatoriamente, ser branco, enquanto o fundo preto. Além disso, a função não leva em consideração bordas de imagens de 1 pixel (OPENCV, 200?e).

## BOUNDING RECT

Bounding Rect (*boundingRect())* é uma função da *OpenCV,* que calcula um retângulo vertical delimitador para um contorno específico, que cobre o objeto por completo (OPENCV, 200?f). A relevância de criar uma caixa delimitadora de um conjunto é que a partir dela, é possível extrair textos ou pedaços da imagem com maior facilidade.

## HOUGHCIRCLES

*Houghcircles* é um algoritmo da OpenCV, que encontra círculos em imagens em tons de cinza, como ilustrado pela (OpenCV, 200?g).



Figura 21- Círculos detectados na imagem utilzando a função houghCircles da OpenCV.

Fonte: OpenCV, 2012.

## CONCLUSÃO

Este capítulo apresentou os principais algoritmos estudados durante o desenvolvimento do protótipo deste trabalho. Alguns deles foram testados e sua aplicação no desenvolvimento do protótipo foi satisfatória, por outro lado, alguns não foram utilizados durante o desenvolvimento.

# FERRAMENTAS EXISTENTES NO MERCADO

## INTRODUÇÃO

Este capítulo tem por objetivo apresentar alguns dos produtos já existentes no mercado que possuem proposta semelhante à apresentada pelo modelo de protótipo relativo ao presente trabalho. Neste capítulo serão exaltados pontos fortes das ferramentas e alguns dos pontos fracos apresentados por cada uma delas.

## QUESTÕES NA WEB

Questões na web é um sistema que permite a criação e reutilização de questões feitas por outros usuários para montar avaliações. Além disso, o sistema possui a opção da utilização de um aplicativo para Android, para a correção de provas de múltipla-escolha (QUESTÕES NA WEB, 20??).

A Figura 22 ilustra a página inicial da ferramenta Questões na Web. Esta tela possui as opções de criação de avaliações, criação e publicação de gabaritos e a opção que redireciona o usuário ao *download* do aplicativo para a correção automática dos gabaritos. Ainda nessa página também pode ser encontrado alguns anúncios publicitários relacionados à área da educação.



Figura - Página inicial Questões na Web

Como apresentado na Figura 23 e Figura 24, o aplicativo de correção de gabaritos possui uma versão gratuita limitada a correção de apenas cinco questões e também uma versão *full* que é paga.



Figura - Versão demo do aplicativo na Google Play

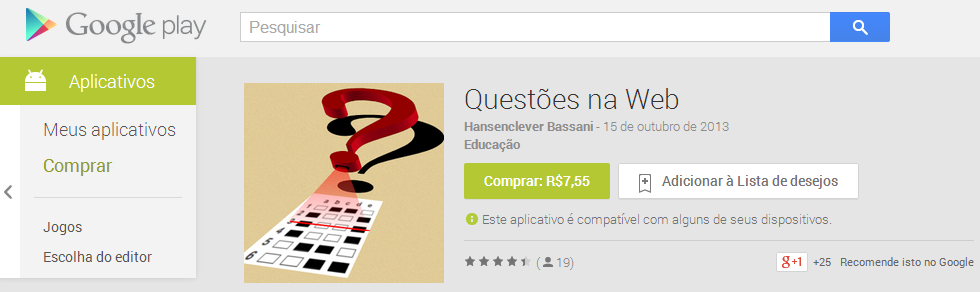


Figura - Versão paga do aplicativo na Google Play

### PONTOS FORTES E FRACOS

O Sistema Questões na web possui como um de seus pontos fortes, mais relevantes a possibilidade de acessar no sistema através de uma conta do Facebook ou do Google. O sistema permite desde a criação de questões, até a correção de gabaritos através de um aplicativo para Android. Por outro lado, um dos pontos que podem ser considerados negativos é que as questões que forem geradas no sistema, mesmo que selecionada a opção de questões privadas, obrigatoriamente, após seis meses, tornam-se públicas, fazendo com que um aluno possa acessar o sistema e pegar respostas de semestres anteriores.

## GRADECAM

Gradecam é um *software* *online* que permite educadores criar e corrigir gabaritos do tipo múltipla-escolha utilizando uma *webcam* (GRADECAM, 20??)*.*

A Figura 25 apresenta a tela de início da ferramenta GradeCam. Na página inicial da ferramenta há um vídeo introdutório da ferramenta para familiarizar o leitor com a ferramenta.

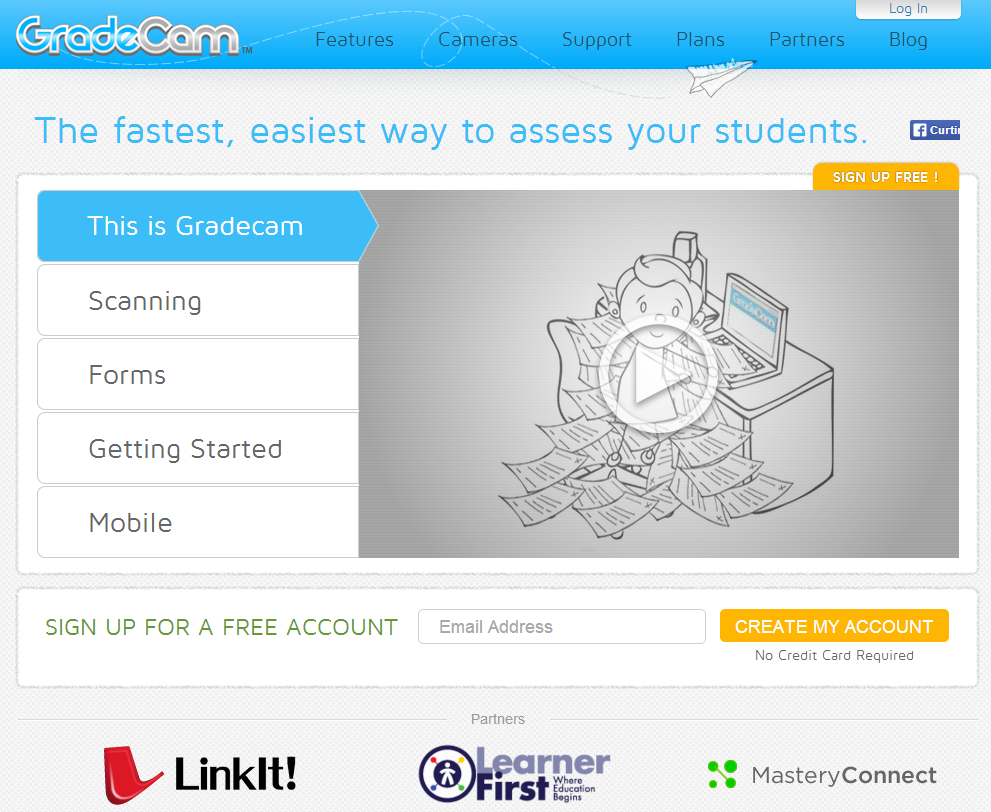


Figura – Página inicial GradeCam

A Figura 26 apresenta um gabarito sendo desenvolvido através da GradeCam, onde é possível determinar o numero de questões, o peso para cada uma delas, além de selecionar qual a resposta correta para cada uma das questões.

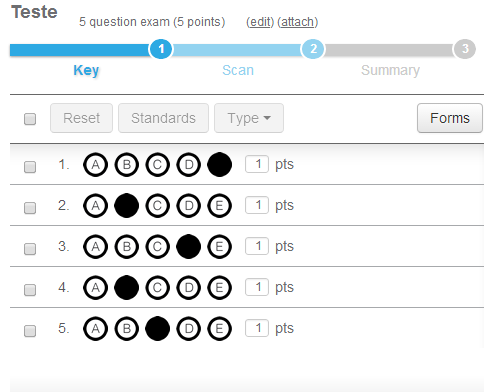


Figura - Criação de gabarito GradeCam

### PONTOS FORTES E FRACOS

Um dos pontos fortes que mais se destacam no Gradecam é a facilidade na criação de gabaritos e alta performance na correção dos mesmos. Um dos pontos fracos é ele não possuir uma versão gratuita, apenas *trial* por 15 dias e não possui a opção de criação de questões, se limitando apenas a criação e correção do gabarito, além de não possuir a opção do idioma português.

## CONCLUSÃO

Este capítulo apresentou uma visão geral sobre algumas das ferramentas existentes no mercado que possuem proposta semelhante à do protótipo desenvolvido para este trabalho. Foram levantados os pontos fortes e fracos de cada ferramenta.

# ESTUDO DE CASO

## INTRODUÇÃO

Neste capítulo será apresentada toda a metodologia, o desenvolvimento e os resultados do artefato criado para correção de gabaritos, utilizando os conhecimentos adquiridos sobre visão computacional durante o desenvolvimento deste trabalho.

## ARQUITETURA GERAL DO CORRETOR DE AVALIAÇÕES

Segundo dados do Instituto Brasileiro de Geografia e Estatística (IBGE), o número total de brasileiros que possuem *smartphones* teve um crescimento de107,2% de 2005 a 2011. Esse aumento fez-se possível devido aos constantes avanços tecnológicos que fazem com que tecnologias, anteriormente caras, cada vez mais baixem seu custo, tornando-se acessível a uma parcela cada vez maior da população. Esta maior facilidade de acesso às tecnologias, principalmente *smartphones,* impulsionou o desenvolvimento aplicações que facilitam e/ou automatizam tarefas diárias. Seguindo essa tendência é que foi proposto o desenvolvimento de um sistema computacional direcionado a automatização das tarefas de criação e correção de gabaritos de avaliações de múltipla escolha, através de câmeras de *smartphones.*

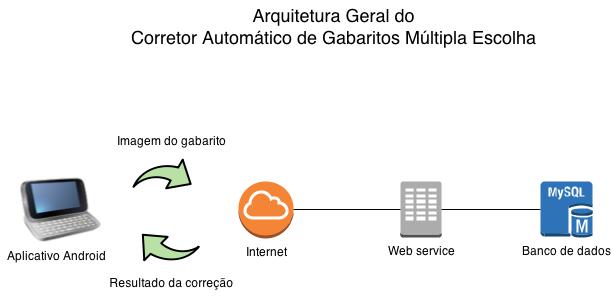


Figura 27 – Arquitetura geral do corretor automático de gabaritos de múltipla escolha.

A apresenta a arquitetura geral da solução do sistema de correção automática de gabaritos desenvolvido durante este trabalho. Como demonstrado na arquitetura, para o sistema funcionar corretamente, o *smartphone* necessita obrigatoriamente possuir conexão com a internet.

A proposta da folha do gabarito a ser corrigido é composta obrigatoriamente por quatro elementos principais, são eles, as marcações do início e fim do gabarito, um *QR Code*[[2]](#footnote-2)que contém os dados do gabarito a ser corrigido como, o identificador único do gabarito e o identificador único do aluno a qual pertence o gabarito, o número das alternativas e por fim os círculos das alternativas que serão corrigidas.

Para efetuar a correção do gabarito o sistema passa por três principais etapas, são elas:

* Aplicativo para *smarphones Android* – Nesta etapa o usuário focaliza com a câmera do *smartphone* o gabarito a ser corrigido. Após estar focalizado corretamente (com a largura completa da câmera do *smartphone*) a tela deverá ser tocada e então a imagem do gabarito a ser corrigido é enviada para o servidor *web*.
* Aplicação *Web* / *Web Service* – Nesta etapa o *web service* reconhece que existe uma imagem dentro da requisição recebida via *internet*, então a imagem da requisição é removida e enviada para processamento, utilizando os algoritmos de visão computacional para realizar a correção do gabarito. Caso a imagem recebida não esteja correta, uma mensagem é retornada para que o usuário envie a imagem novamente. Caso a imagem seja aceita e corrigida pelo sistema, o resultado da correção é gravado na base de dados criada e então o resultado da avaliação é retornado para o *smartphone* do usuário.
* Manipulação dos resultados no banco de dados – Quando o corretor recebe os resultados do gabarito, obtido através dos algoritmos de visão computacional, os resultados obtidos são gravados em um banco de dados para que os resultados possam ser avaliados.

Para o trabalho em questão foi criado um modelo de protótipo para correção de gabaritos, ele será composto por pela maior parte dos fatores citados acima, exceto o *QR Code* para reconhecimento do aluno e do gabarito, essa funcionalidade não foi desenvolvida por motivos de falhas e mal funcionamento no momento que foram efetuados os testes.

## Limitações do projeto

O protótipo possui algumas limitações para ter seu funcionamento correto, suas limitações são:

* o *smartphone* utilizado deverá ter acesso a internet para que seja efetuada a correção, pois o algoritmo de correção fica somente no servidor sendo o *smartphone* utilizado apenas para enviar a imagem do gabarito.
* A quantidade de questões a ser corrigida pelo protótipo varia de um a nove questões, não sendo possível a utilização de mais de nove questões.
* A imagem a ser enviada pelo *smartphone* para a correção deve ser ajustada com a largura total da tela do aparelho e deverá estar bem focalizada pela câmera, não é possível a correção de gabaritos que estejam com a imagem desfocada, distantes ou tremidas.

## MODELAGEM DE DADOS

A proposta do sistema não é unicamente a correção de gabaritos, nele existem funcionalidades que permitem aos seus usuários a criação de um perfil de próprio, cadastrar turmas e os alunos para as turmas e por fim criar folhas de respostas com base na turma para qual a avaliação está sendo criada. A demonstra como foi feita a modelagem dos dados do corretor de avaliações.

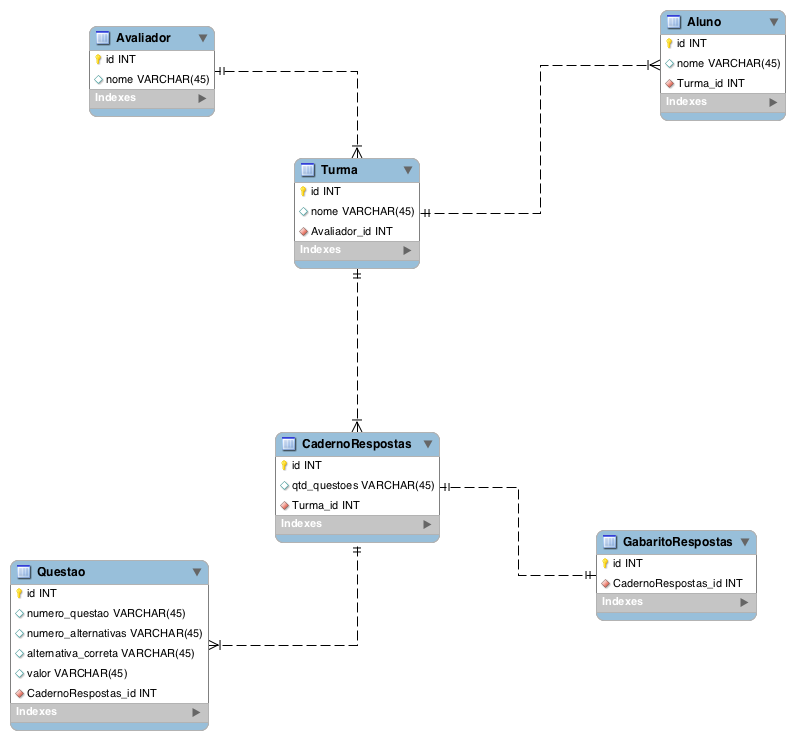


Figura 28 - Modelagem de dados do Sistema

De acordo com o Modelo de Entidade e Relacionamento (MER) apresentado pela , existem seis modelos de entidade que serão utilizados no sistema. São eles:

* Avaliador – Entidade que possui o perfil do avaliador
* Turma – Entidade da turma de alunos
* Aluno – Entidade que possui os dados do aluno
* CadernoRespostas – Entidade que possui as questões do gabarito a ser criado
* GabaritoRespostas – Entidade que guarda as questões corretas do caderno de resposta
* Questao – Entidade que guarda os dados de cada questão do caderno de resposta, como o número da questão, número de alternativas, alternativa correta e o valor da questão.

## APLICAÇÃO WEB

Como citado na arquitetura geral do corretor de avaliação, o protótipo possui uma aplicação servidor que fica hospedado em um servidor na *web*, essa aplicação foi desenvolvida utilizando o *framework* Django. Esse *framework* foi escolhido, pois, de acordo com a documentação disponibilizada no site oficial do Django Projects, ele permite a construção de sistemas *web* de alto desempenho de forma elegante e rápida.

O sistema *web* possui duas telas principais. Conforme ilustrado pela , a tela inicial apresenta uma *dashboard,* cuja função é exibir todos os gabaritos criados pelo usuário que está *logado* no sistema, permitindo ao usuário visualizar, excluir e concluir os gabaritos criados. Essa tela também possui o botão que redireciona rapidamente o usuário para a criação de novos gabaritos.

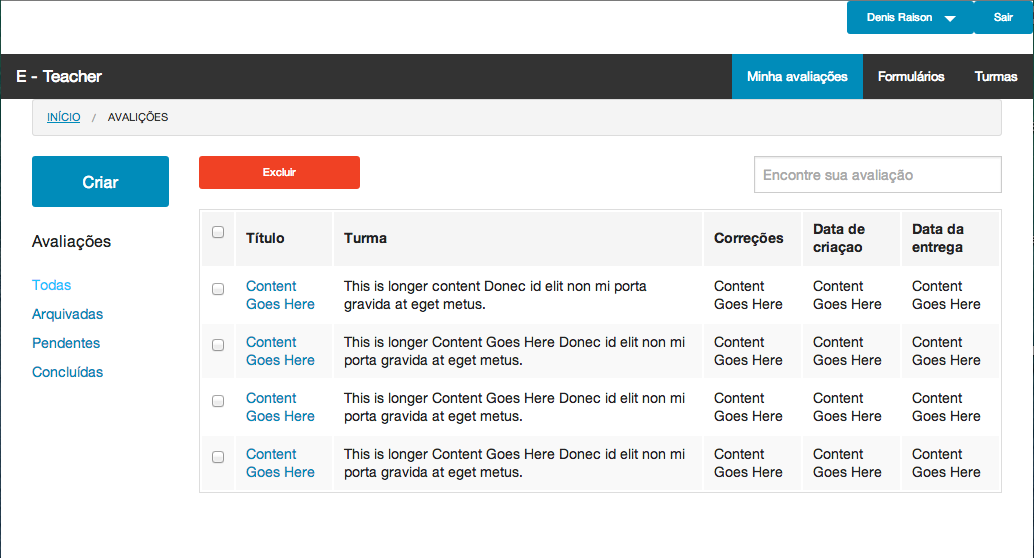


Figura 29 - Tela inicial da aplicação

As e apresentam a tela do formulário de criação de novos gabaritos de resposta pelo usuário. Esse formulário possui os campos para criação de novos gabaritos, são eles:

* Título do gabarito – Nomeia o gabarito para facilitar a visualização dele na *dashboard;*
* Descrição – Campo não obrigatório que permite que o usuário adicione uma descrição ao gabarito criado, para facilitar a busca por esse gabarito;
* Turma – Campo de seleção em que todas as turmas cadastradas pelo usuário que estiver *logado* são mostradas, esse campo determinará o número de folhas de respostas que serão criadas pelo sistema;
* Quantidade de questões – Campo em que o usuário define o número de questões que irão compor o gabarito;

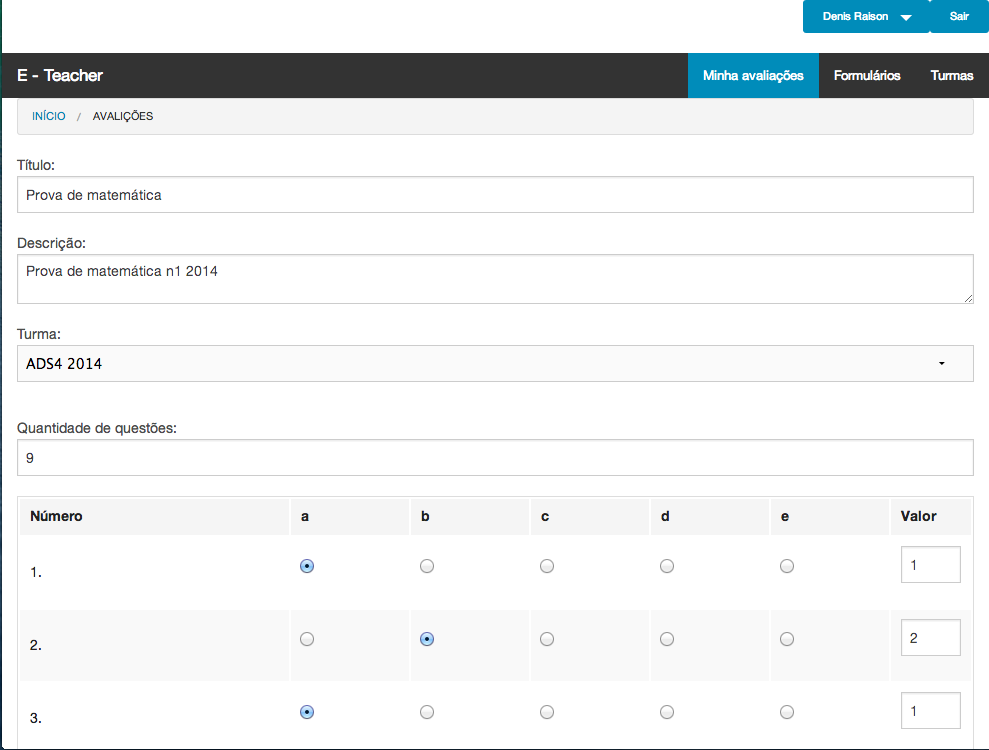


Figura 30 – Tela de criação de novos gabaritos

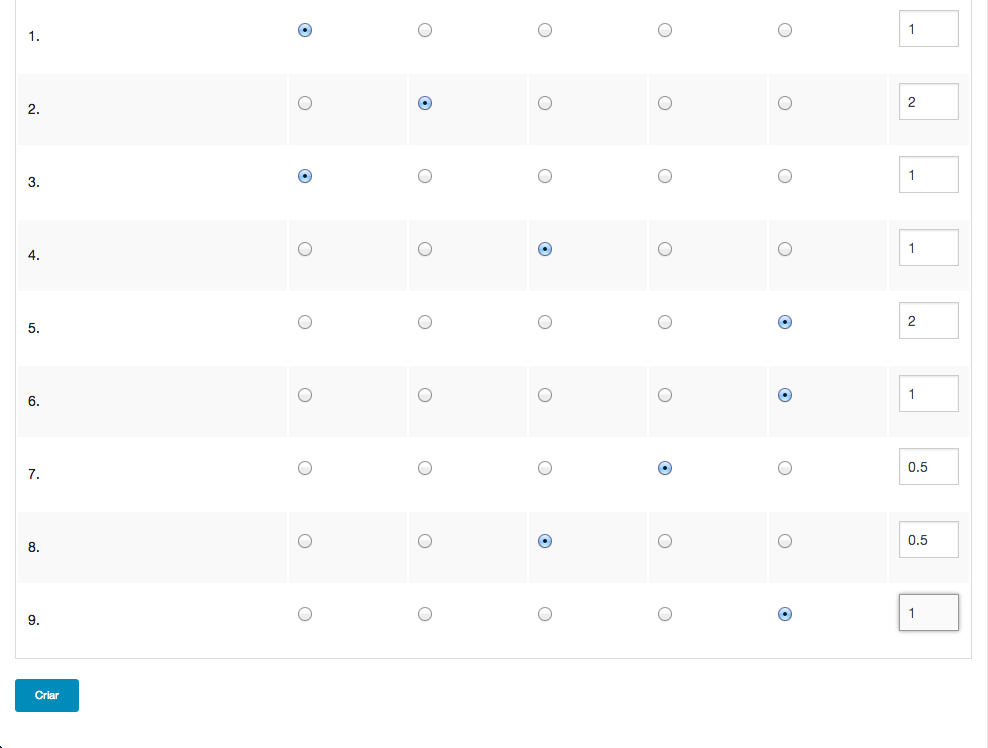


Figura 31 – Tela de criação de novos gabaritos

A tabela abaixo do formulário de cadastro exibe todas as questões definidas pelo usuário no campo “Quantidade de questões”, porém, por limitações técnicas, no protótipo desenvolvido para esta monografia houve uma limitação de nove questões para serem corrigidas em cada gabarito. Na tabela existem as colunas:

* Número - que mostra o número da questão, “a”, “b”, “c”, “d”, “e”, que são as alternativas das questões
* Valor - que é definido o valor da questão para aquela avaliação, para que seja possível dar uma nota para o aluno ao final da correção.

Para cada questão é preciso, obrigatoriamente, selecionar uma alternativa correta e também o valor da questão. Após ter preenchido corretamente todos os dados do formulário, deve-se selecionar o botão “Criar”.

Após selecionar o botão “Criar”, o sistema processa os dados recebidos do formulário preenchido anteriormente e gera um arquivo PDF. Para a criação do arquivo PDF foi utilizada a biblioteca *xhtml2pdf*. De acordo com o site oficial da *xhtml2pdf*, essa biblioteca é escrita em Python e por esse motivo, independente de qualquer plataforma, ela converte documentos HTML e CSS em um arquivo no formato PDF e tem suporte para o *framework* Django utilizado na aplicação.

Após a criação do arquivo PDF o usuário é redirecionado para uma tela que apresenta o arquivo PDF criado e os gabaritos de resposta de cada aluno criados para a avaliação, como apresentado pela .

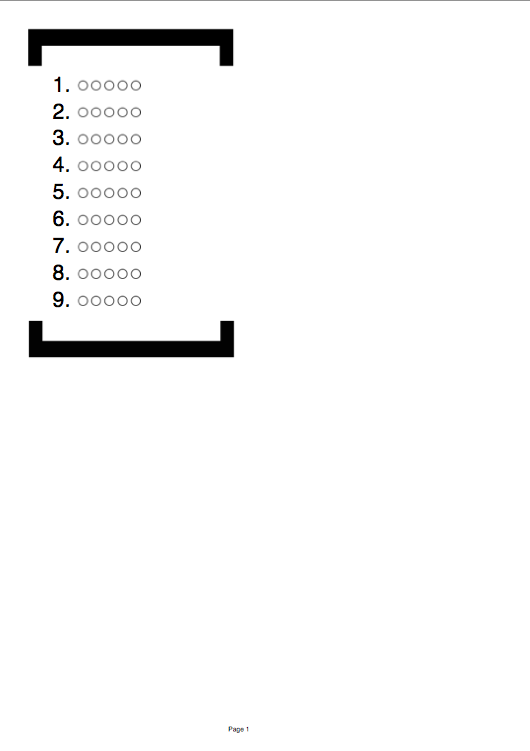


Figura 32 – Modelo de arquivo PDF do gabarito de respostas

A , demonstra que no arquivo PDF gerado foram criadas dez páginas. Cada uma dessas páginas corresponde a um aluno previamente cadastrado na turma para qual foi gerado o gabarito, cada página possui um gabarito e cada folha de gabarito possui o número de questões definidas no formulário de edição.

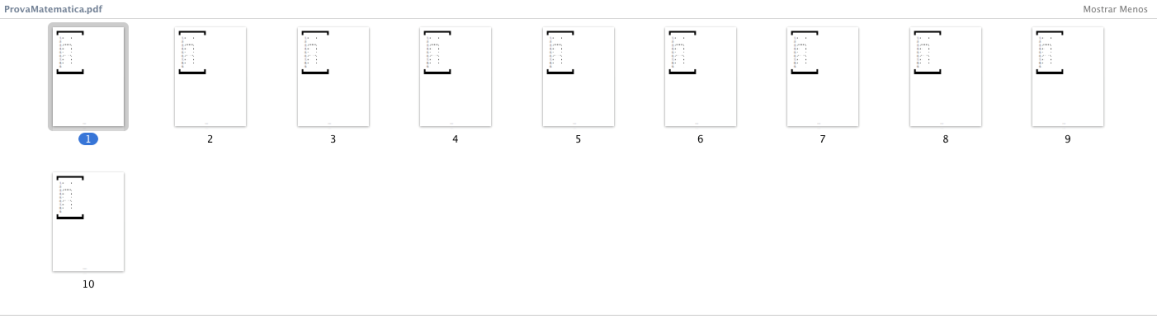


Figura 33 - Quantidade de paginas PDF

## APLICATIVO PARA SMARTPHONE

Como citado na arquitetura geral do corretor de avaliações, um aplicativo para *smartphone* também foi desenvolvido,para isso foi escolhido o Sistema Operacional Android, devido ao fato de que, é o sistema operacional móvel mais popular do mundo atualmente e também porque existe uma versão para Android da biblioteca *Opencv* utilizada pela a aplicação (Android, 201?).

O aplicativo foi desenvolvido para as versões *Ice Cream Sandwich* 4.0.3, *Jelly Bean* 4.3 () e *KitKat* 4.4 () do SO Android, portanto podem haver erros ou má execução do aplicativo caso instalado em versões diferentes. Para que seja possível utilizar o aplicativo, no momento do *download,* deverá ser autorizado pelo usuário que o aplicativo acesse a câmera do dispositivo e o acesso ao cartão de memória ou armazenamento interno do dispositivo.



Figura 34 – Android *Jelly Bean* 4.3.

**Fonte: Android, 201?.**



Figura 35 - *KitKat* 4.4.

**Fonte: Android, 201?.**

Ao iniciar o aplicativo no *smartphone,* automaticamente é executada a câmera do dispositivo. Nesse momento deve-se enquadrar o gabarito com a sua marcação dentro da área total da tela do *smartphone,* como pode ser visto na e tocar na tela do dispositivo.

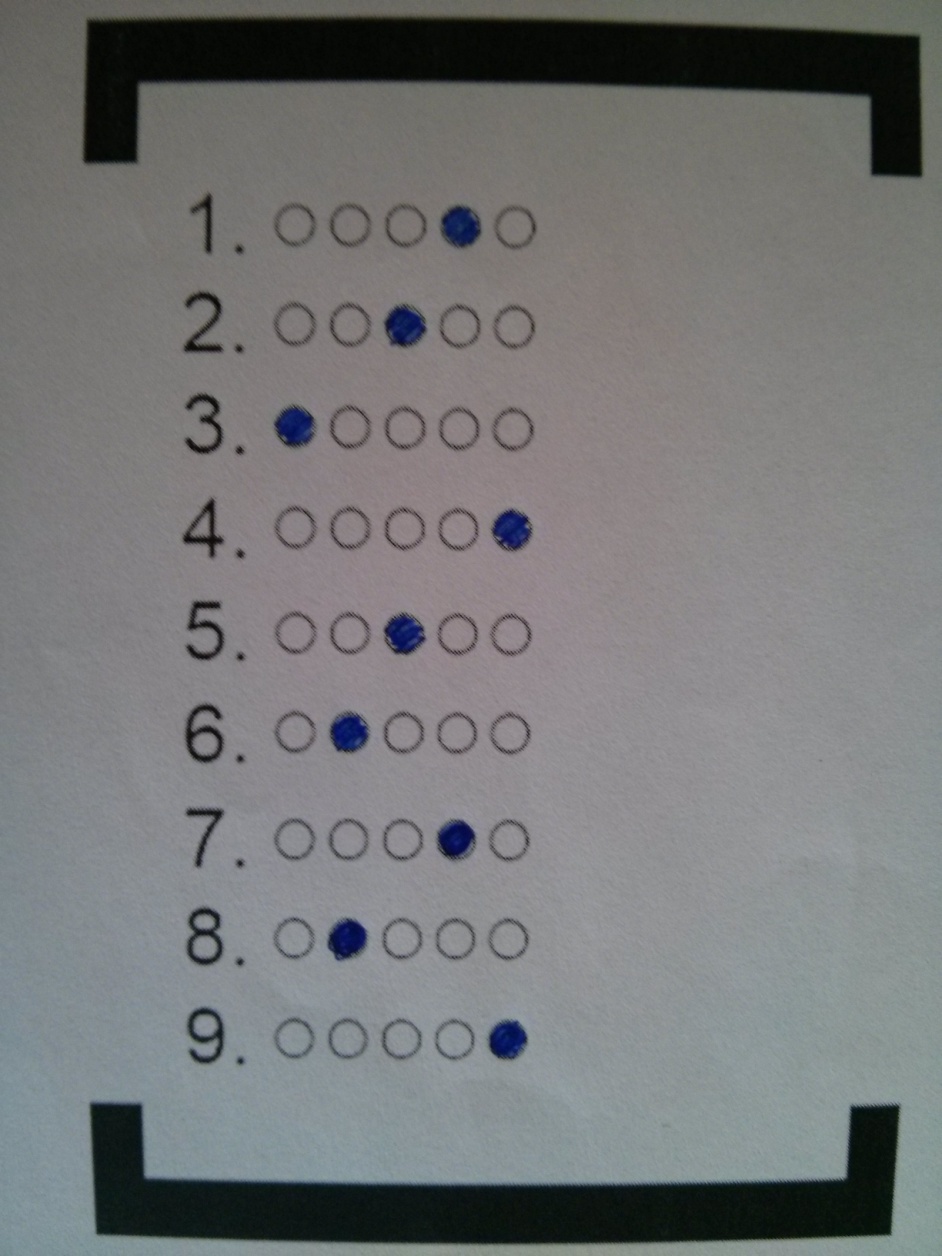


Figura 36 - Imagem do gabarito a ser corrigido

Após efetuar o toque na tela do dispositivo, o aplicativo captura a imagem enquadrada na câmera. Utilizando a biblioteca do java.net *HttpURLConnection* é criada uma conexão com o servidor *web,* citado no capítulo anterior, então a imagem é encapsulada dentro de um conteúdo *HTTP*. A imagem é então enviada através da internet para o servidor *web* e tem como resultado dessa transação uma resposta do servidor, que pode ser uma mensagem de erro, fazendo com que o usuário tenha que reenviar o gabarito para a correção ou retornando o resultado da correção obtido através dos algoritmos de visão computacional. A descreve o fluxo de funcionamento do aplicativo de correção de gabaritos.

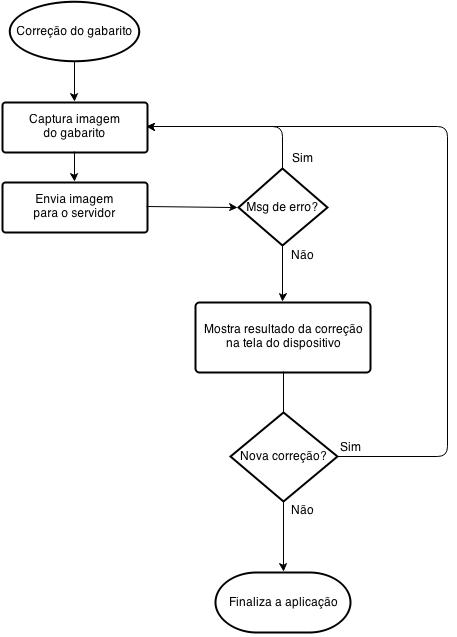


Figura 37 - Fluxograma do funcionamento da correção de gabaritos

## WEB SERVICE

Conforme citado na arquitetura geral do corretor de gabaritos essa etapa é responsável pela transmissão e recepção de dados do *smartphone*, ela é o *middleware*[[3]](#footnote-3) de todo o sistema, ou seja, essa é a etapa responsável pela integração de ambas as partes do sistema, tais como a Aplicação *Web* e a Aplicação Android.

O *web service* utilizado no sistema foi escrito na linguagem Python, e é um *web service* que roda em cima do Django na sua versão 1.6. Elepossui um serviço que fica aguardado a chamada do aplicativo Android. Como dito no capítulo anterior o aplicativo Android se conecta com um endereço do *web service* e envia a imagem encapsulada para o servidor e nesta etapa é onde o sistema corrige o gabarito, grava o resultado e o retorna para o aplicativo Android.

De acordo com a , é no *web service* que ocorre a etapa mais importante do sistema, a correção efetiva dos gabaritos de provas, pois a imagem enviada pelo *smartphone* chega ao *web service* e é enviada por ele para o algoritmo que efetua a correção do gabarito, quando as respostas são corrigidas e salvas do banco de dados, elas são enviadas novamente ao *web service* que as envia de volta para o *smartphone*.

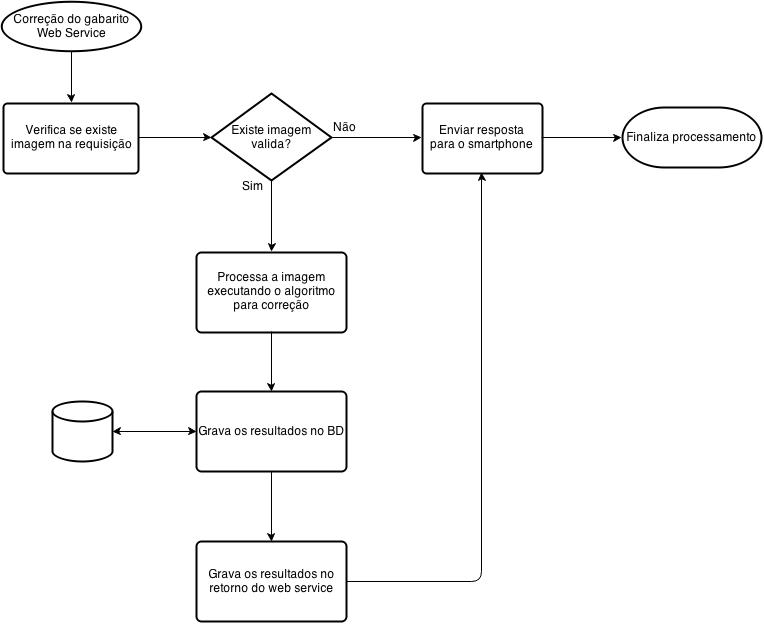


Figura 38 - Fluxograma do funcionamento do Web Service

## ALGORITIMO PARA CORREÇÃO DO GABARITO

O algoritmo utilizado para a correção do gabarito foi desenvolvido na linguagem de programação Python e utiliza métodos e funções do *framework* Opencv,apresentado no capítulo 4.

Para que se possa dar inicio a correção do gabarito, é preciso de duas principais imagens, a imagem do gabarito a ser corrigido e uma imagem de *template,* que é a imagem que será rastreada na imagem de entrada, neste caso é a imagem do gabarito a ser corrigido.

A imagem recebida pelo aplicativo Android primeiramente deve ser transformada em preto e branco, como pode ser visto na . Esse processo é feito utilizando a função cvtColor do OpenCV, que converte imagens em diferentes escalas de um espaço de cor. As imagens em RGB foram convertidas para imagens em preto e branco, neste trabalho.

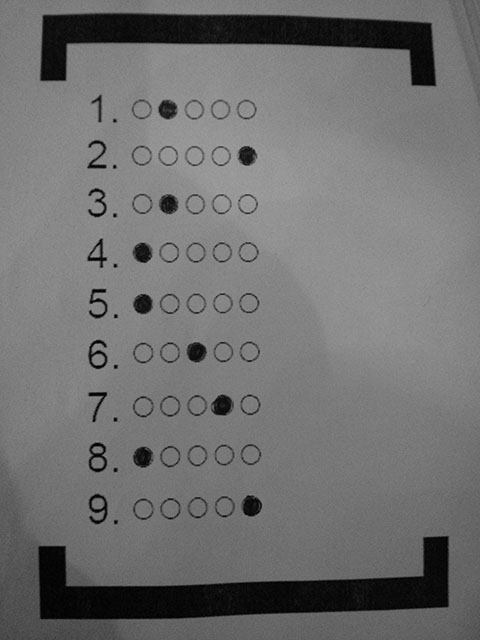


Figura 39 - Imagem em escala preta e branca

A apresenta a imagem que será carregada como *template* para que seja feito o reconhecimento do inicio e do final do gabarito de respostas que está sendo corrigido.

template_head

Figura 40 - *Template* reconhecimento do início e fim do gabarito.

A partir dessa imagem, o sistema pode reconhecer se de fato a imagem enviada é um gabarito. Para executar esta tarefa utilizou-se o algoritmo para reconhecimento de padrões chamado SIFT, descrito no capítulo . O algoritmo SIFT reconhece pontos-chave (através das características da imagem) na *template* e encontra os mesmos na imagem. A , ilustra os padrões reconhecidos tanto no início do gabarito quanto no final do gabarito.

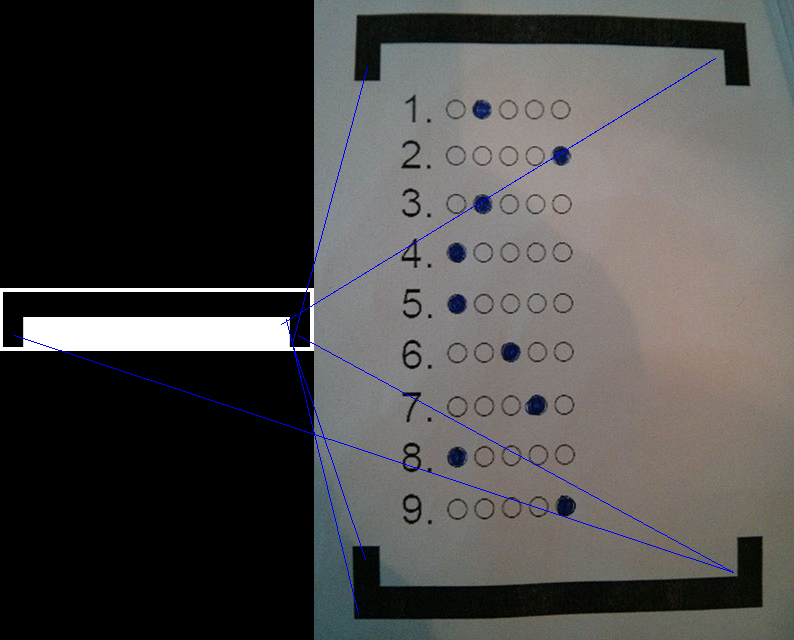


Figura 41 - Resultado da execução do algoritmo SIFT

Após o reconhecimento da área interessante para o processo, o algoritmo realiza um corte na imagem nos pontos encontrados, ficando apenas com o que é realmente necessário para que possa ser realizada a correção do gabarito, como apresentado na .

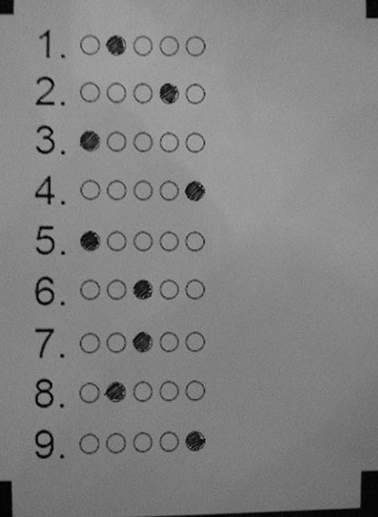


Figura 42 - Resultado do corte na imagem do gabarito

Após a imagem cortada, inicia-se a etapa para o reconhecimento das questões e alternativas, para essa etapa a imagem passa por dois tratamentos de imagem:

* Blur – A imagem passa pela função GaussianBlur do Opencv, conforme comentado no capítulo 5.2.2. Nessa etapa, a imagem sofre alterações quanto a sua nitidez ();

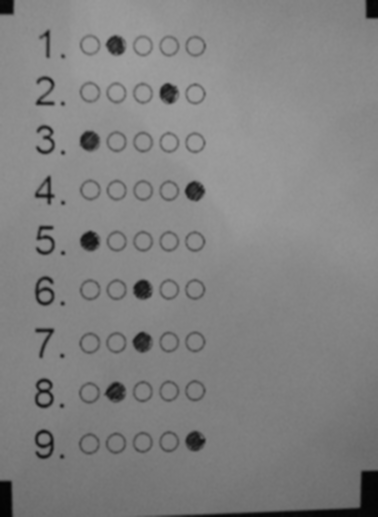


Figura 43 - Imagem após utilizar o GaussianBlur

* adaptiveThreshold - A imagem passa pela função adaptiveThreshold do Opencv, descrita no capítulo , esse algoritmo transforma uma imagem em escala de preto e branco em uma imagem binária, conforme ilustrado pela , após a execução do algoritmo só existe na imagens pontos branco e pretos;

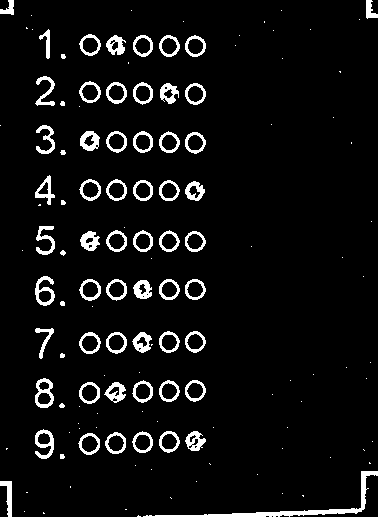


Figura 44 - Imagens após aplicação do adaptativeThreshold

A manipulação na imagem é feita para que o método *findCountour* do Opencv*,* citado no capítulo , seja executado. Esse algoritmo encontra todos os contornos e sequências de contornos dentro da imagem. Após serem encontrados os contornos é utilizado o algoritmo *countorArea* para trabalhar apenas com contornos que sejam retângulos ou quadrados completos retirando assim todos os números das alternativas das imagens, como pode ser visto na .

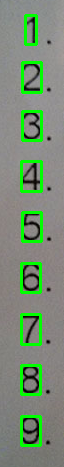


Figura 45 – Contornos retangulares encontrados na imagem.

Após serem reconhecidos todos os números na imagem é feita uma repetição número a número, efetuando cortes na direção horizontal, retirando apenas as alternativas de cada questão para serem trabalhadas. Tendo todas as alternativas divididas por questões, um laço de repetição foi executado pelas alternativas de cada questão e dentro de cada laço é utilizado a função *HoughCircles,* como citado no capítulo , a função reconhece dentro da imagem todos os desenhos com formatos de círculo, como pode ser visto na



Figura 46- Círculos detectados na imagem, utilizando o algoritmo HoughCircles

A etapa final do algoritmo é a execução de um laço de repetição em todas as alternativas da questão. Nesta etapa é feita uma verificação se a alternativa está preenchida, caso esteja, essa alternativa é considerada a alternativa correta da questão que está sendo corrigida. Essa verificação é feita checando se a área do círculo detectado pelo algoritmo *HoughCircle,* citado anteriormente, está com a sua cor predominante menor ou igual a setenta, na escala de cinza, pois quanto o número mais se aproxima de zero e se distancia de duzentos e cinquenta e cinco, foi considerado que o círculo em questão está preenchido.

A ilustra o gabarito completo corrigido, essa imagem representa visualmente a maneira que o algoritmo efetua a correção do gabarito.



Figura 47 - Gabarito corrigido durante os testes

## APLICAÇÃO

Foi analisado o funcionamento do protótipo através da realização de um teste com um questionário de oito questões sobre assuntos diversos da área de Tecnologia da Informação e Comunicação para uma turma de uma faculdade de tecnologia de São Bernardo do Campo – SP, na descrição das avaliações estava prescrito para preencher completamente a alternativa que foi considerada correta utilizando apenas caneta da cor azul .

O teste foi realizado com dez pessoas para que fosse analisada a porcentagem de erros e acertos tendo em vista que cada pessoa marca o gabarito de maneiras diferentes. Neste trabalho, através das figuras 43 à 46, foram ilustrados quatro dos dez gabaritos testados.

Ao final do teste foi demonstrado que o sistema possui uma probabilidade de acerto de 98,75%, pois dos dez gabaritos corrigidos, o protótipo corrigiu corretamente nove deles errando a correção de apenas uma alternativa do questionário, que pode ser visto na , porém nesta alternativa do gabarito aluno não preencheu o círculo completamente deixando falhas na pintura da alternativa fazendo com que o protótipo não reconhecesse que a alternativa estava correta.

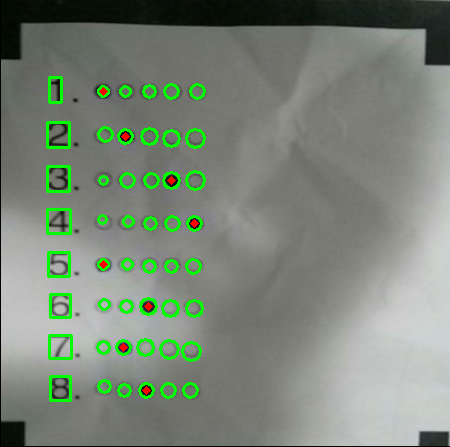


Figura 48 – Gabarito corrigido durante os testes



Figura 49 – Gabarito corrigido durante os testes.



Figura 50 – Gabarito corrigido sem sucesso durantes os testes



Figura 51 - Gabarito corrigido durante os testes.

Ao final dos testes foi demonstrado que apesar de existir uma pequena margem de erro, o sistema permite a correção rápida de avaliações do que de maneira manual.

# CONSIDERAÇÕES FINAIS e trabalhos futuros

Este trabalho teve como principal objetivo o desenvolvimento de um de protótipo de um sistema de auxilio à tarefa de correção de avaliações do tipo múltipla escolha. Para isso, um estudo sobre a área de Visão Computacional e suas tecnologias foi realizado.

A partir de pesquisas, foi identificado que as ferramentas de auxílio a educadores ainda são muito escassas, principalmente a nível nacional. Devido à busca crescente por sistemas rápidos, fáceis e simples é que os aplicativos de dispositivos móveis vêm ganhando relevância e destaque. Atualmente existe uma infinidade de aplicativos de diferentes funções, que aos poucos substituem tarefas rotineiras na vida das pessoas.

Após a realização da pesquisa bibliográfica sobre as principais tecnologias da área de Visão Computacional, foi iniciado o desenvolvimento do protótipo do sistema corretor de avaliações de múltipla escolha, que usa essas tecnologias para seu funcionamento.

Foi analisado o funcionamento do protótipo através da realização de um teste com um questionário de oito questões sobre assuntos diversos da área de Tecnologia da Informação e Comunicação para uma turma de uma faculdade de tecnologia de São Bernardo do Campo. Ao final do teste o protótipo criado corrigiu 98,75% das alternativas corretamente, demonstrando que apesar de existir uma margem de erro, o sistema permite a correção rápida e consequentemente mais eficiente de avaliações do que de maneira manual.

Como trabalhos futuros, sugerimos o aperfeiçoamento do corretor de avaliações, melhorando a forma com que processará as imagens recebidas fazendo com que se reconheça melhor os gabaritos independente do tamanho e forma que o gabarito é enviado ao servidor e também implementando o mecanismo para o reconhecimento de *QR Code* para salvar corretamente os dados de cada aluno e criar uma ata de avaliações dos alunos fazendo assim que o sistemas seja mais completo.

# REFERÊNCIAS

ALTAMISOFT. *User Manual:* adaptive threshold. Disponível em: <http://altamisoft.com/products/altami\_studio/user\_manual\_AS\_3\_1\_0\_en/filters\_en/gray\_transformations\_en/adaptive\_threshold\_en/> Acesso em 14 maio 2014.

ANDRADE, M.V.A.A. *Métodos eficientes para reconhecimento de padrões em texto.* Campinas, 1993.

ANDROID. *Conheça o Android.* Disponível em: http://www.android.com/meet-android/. Acesso em: 12 maio 2014.

BALLARD, D.H.; BROWN, C.M.Computer vision. Nova Jersey: Prentice-Hall, 19??.

BORTOLI, J. Sistemas operacionais de código aberto. Disponível em: <http://www.joeldebortoli.com/2013/02/sistemas-operacionais-de-codigo-aberto.html> Acesso em 29 mar 2014.

BRADSKI, G.; KAEHLER, A. *Learning OpenCV.*Sebastopol: O’Reilly Media, 2008. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.396.1203&rep=rep1&type=pdf> Acesso em: 29 mar 2014.

CASTRO, A.A.M; PRADO, P.P.L. *Algoritmos para reconhecimento de padrões.* Taubaté, 2002.

CENTRO OFTALMOLOGICO DO NORDESTE, *Você sabe como o homem criou a máquina fotográfica?,* 2013. Disponível em: <http://coneoftalmo.com.br/noticia/voce+sabe+como+o+homem+criou+a+maquina+fotografica-85 > Acesso em: 18 fev 2014.

CONCI, A.; MONTEIRO, L.H.*Reconhecimento de placas de veículos por imagens.* Disponível em: <http://www2.ic.uff.br/~aconci/CONENPLACAS.pdf> Acesso em: 05 mar 2014.

DAUGMAN, J.G. *Computer Vision:* computer science tripos, part II. Disponível em: <http://www.cl.cam.ac.uk/teaching/1314/CompVision/CompVisNotes2014.pdf> Acesso em: 21 fev 2014.

DEVROYE, L.; GYORFI, L.; LUGOSI, G. *A Probabilistic Theory of Pattern Recognition*.Disponível em: <http://www.szit.bme.hu/~gyorfi/pbook.pdf> Acesso em: 05 mar 2014.

DJANGO. Django documentation. Disponível em: <https://docs.djangoproject.com/en/1.6/ > Acesso em: 02 maio 2014.

GRADECAM. *About.* Disponível em: <http://gradecam.com/> Acesso em: 21 maio 2014.

GREENEMEIER, L. *Visionary research*: teaching computers to see like a human. Disponível em: <http://www.scientificamerican.com/article/visionary-research/>. Acesso em 19 fev 2014.

INSTITUTO DE FÍSICA DA USP. *Leituras de física:* óptica para ver, fazer e pensar. São Paulo, 1998. Disponível em: < http://www.if.usp.br/gref/optica/optica1.pdf>. Acesso em 19 fev 2014.

JAIN, A.K.; DUIN, R.P.W.; MAO, J. *Statistical pattern recognition:* a review. Disponível em: <http://dspace.dle.more.net/bitstream/handle/123456789/11/pattern%20recognition.pdf?sequence=1> Acesso em: 30 abr 2014.

LOWE, D.G.*Object recognition from local scale-invariant features.* Disponível em: < http://www.cs.ubc.ca/~lowe/papers/iccv99.pdf>. Acesso em 18 abr 2014.

LOWE, D.G. *Distinctive image features from scale-invariant keypoints.* Disponível em: <http://www.cs.berkeley.edu/~malik/cs294/lowe-ijcv04.pdf>. Acesso em 18 abr 2014.

LUEBKE, D. et al.*Level of detail for 3D graphics*. São Francisco: Morgan Kaufmann Publishers, 2003.

MANZI, F.A. *Aplicação de visão computacional para extração de características em imagens do olho humano.* São Carlos: 2007.

MARENGONI, M.; STRINGHINI, D. *Tutorial:* introdução à visão computacional usando OpenCV. Disponível em: <http://seer.ufrgs.br/index.php/rita/article/view/rita\_v16\_n1\_p125/7289> Acesso em 02 mar 2014.

MARIUS, D.; PENNATHUR, S.; ROSE, K. *Face detection using color thresolding and eigenimage template matching.* Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.84.1609&rep=rep1&type=pdf> Acesso em 30 abr 2014.

MARQUES FILHO, O.; VIEIRA NETO, H. *Processamento digital de imagens.*Rio de Janeiro: Brasport, 1999. Disponível em: <http://pessoal.utfpr.edu.br/hvieir/download/pdi99.pdf >. Acesso em: 18 fev 2014.

MILANO, D.; HONORATO, L.B. *Visão computacional.* Disponível em: <http://www.ft.unicamp.br/liag/wp/monografias/monografias/2010\_IA\_FT\_UNICAMP\_visaoComputacional.pdf> Acesso em: 18 fev 2014.

MORRIS, T. *Computer* *graphics and image processing:* part 2: image processing. Disponível em: <http://studentnet.cs.manchester.ac.uk/assessment/feedback/ug/2012/COMP27112.pdf> Acesso em 19 fev 2014.

ORRÚ, T. *Redes* *neurais artificiais para reconhecimento e classificação de padrões.* Campinas, 200?.

OSÓRIO, F; BITTENCOURT, J.R. *Sistemas inteligentes baseados em redes neurais artificiais aplicados ao processamento de imagens.* Disponível em: <http://osorio.wait4.org/oldsite/wia-unisc/wia2000-mini.pdf> Acesso em 22 fev 2014.

OPENCVa. *About.* Disponível em: <http://opencv.org/about.html> Acesso em: 30 mar 2014.

OPENCVe*. Contours:* getting started. Disponível em: <http://docs.opencv.org/trunk/doc/py\_tutorials/py\_imgproc/py\_contours/py\_contours\_begin/py\_contours\_begin.html> Acesso em: 12 maio 2014.

OPENCVg. *Feature detection:* houghCircles. Disponível em: <http://docs.opencv.org/modules/imgproc/doc/feature\_detection.html#houghcircles> Acesso em: 18 maio 2014.

OPENCVc*. Image filtering:* Laplacian. Disponível em: <http://docs.opencv.org/modules/imgproc/doc/filtering.html?highlight=laplacian#laplacian> Acesso em 18 abr 2014.

OPENCVb. *Introduction to SIFT (scale invariant feature transform).*Disponível em: < http://docs.opencv.org/trunk/doc/py\_tutorials/py\_feature2d/py\_sift\_intro/py\_sift\_intro.html> Acesso em: 19 abr 2014.

OPENCVd. *Miscellaneous image transformations:*adaptative threshold.Disponível em: <http://docs.opencv.org/modules/imgproc/doc/miscellaneous\_transformations.html> Acesso em: 14 maio 2014.

OPENCVf. *Structural analysis and shape descriptors*: bounding rect. Disponível em: <http://docs.opencv.org/modules/imgproc/doc/structural\_analysis\_and\_shape\_descriptors.html#boundingrect> Acesso em: 18 maio 2014.

QUESTÕES NA WEB. *Sobre.* Disponível em: <http://questoesnaweb.com.br/Info.html> Acesso em: 21 maio 2014.

RIGUAL, F. *Object recognition applied to móbile robotics.* Disponível em: <http://upcommons.upc.edu/pfc/bitstream/2099.1/16105/1/84801.pdf> Acesso em: 19 abr 2014.

RIOS, L.R. *Visão computacional.* Disponível em: <http://homes.dcc.ufba.br/~luizromario/Apresenta%C3%A7%C3%A3o%20de%20IA/Artigo%20(final).pdf> Acesso em:18 fev 2014.

SHAPIRO, L; STOCKMAN, G. *Computer vision.* Disponível em: <http://www.cse.msu.edu/~stockman/Book/contents.pdf> Acesso em: 21 fev 2014.

SINHA, U. *SIFT: Scale invariant feature transform.*Disponível em: <http://www.aishack.in/2010/05/sift-scale-invariant-feature-transform/>” Acesso em: 04 maio 2014.

SOUZA, C.F.; CAPOVILLA, F.; ELEOTÉRIO, T.C*.Visão computacional*. Disponível em: <http://www.ebah.com.br/content/ABAAABeUIAB/visao-computacional> Acesso em 21 fev 2014.

SOUZA, L.R. *Algoritmos para reconhecimento e acompanhamento de padrões em vídeo.* Disponível em: <www.univasf.edu.br/~ccomp/monografias/monografia\_1.pdf‎ > Acesso em: 06 mar 2014.

STACKOVERFLOW*. Adaptative thresold of blurry image.* Disponível em: <http://stackoverflow.com/questions/13391073/adaptive-threshold-of-blurry-image> Acesso em: 02 maio 2014.

THEODORIDIS, S.; KOUTROUMBAS, K. *Pattern Recognition*. San Diego: Elsevier, 2009.

TOHKA, J. *Introduction to Pattern Recognition.* Disponível em: <http://www.cs.tut.fi/kurssit/SGN-2506/> Acesso em 03 mar 2014.

XHTML2PDF. *HTML/CSS to PDF converter written in Python.* Disponível em: <http://www.xhtml2pdf.com/> Acesso em 02 maio 2014.

1. Licença BSD é uma licença de código aberto inicialmente utilizada nos sistemas operacionais do tipo Berkeley Software Distribution (sistema derivado do Unix) (Bortoli, 200?). [↑](#footnote-ref-1)
2. QR code é um [código de barras](http://pt.wikipedia.org/wiki/C%C3%B3digo_de_barras) [bidimensional](http://pt.wikipedia.org/wiki/Bidimensional) que pode ser facilmente digitalizado usando a maioria dos [telefones celulares](http://pt.wikipedia.org/wiki/Telefone_celular) equipados com câmera. [↑](#footnote-ref-2)
3. Middleware é um termo geral, normalmente utilizado para um de código de software que atua como um aglutinador, ou mediador, entre dois programas existentes e independentes. Sua função é trazer independência das aplicações com o sistema de transmissão (TELECO, 20??). [↑](#footnote-ref-3)