UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA/ELETRÔNICA
CURSO SUPERIOR DE ENGENHARIA DE COMPUTAÇÃO

Renan Kruchelski Machado
Ricardo Farinhaki
Thiago Avelino da Silva

SETA: AMBIENTE DE REALIDADE AUMENTADA PARA O ENSINO DE ALGORITMOS DE APRENDIZAGEM DE MÁQUINA

TRABALHO DE CONCLUSÃO DE CURSO

Curitiba
2013

Renan Kruchelski Machado
Ricardo Farinhaki
Thiago Avelino da Silva

SETA: AMBIENTE DE REALIDADE AUMENTADA PARA O ENSINO DE ALGORITMOS DE APRENDIZAGEM DE MÁQUINA

Documento de Projeto da Disciplina de Trabalho de Conclusão do Curso, de Engenharia de Computação, apresentado à UTFPR como requisito parcial para obtenção do título de Engenheiro de Computação.

Orientador: Prof. Gustavo Benvenutti Borba

Curitiba
2013

RESUMO

MACHADO, Renan K.; FARINHAKI, Ricardo; DA SILVA, Thiago A.. SETA: AMBIENTE DE REALIDADE AUMENTADA PARA ENSINO DE ALGORITMOS DE APRENDIZAGEM DE MÁQUINA. Trabalho de Conclusão de Curso – Curso Superior de Engenharia de Computação, Universidade Tecnológica Federal do Paraná. Curitiba, 2013.

A realidade aumentada (RA) consiste na sobreposição de imagens virtuais em imagens de um ambiente real, em tempo real. Ambientes de RA podem ser utilizados como recursos didáticos complementares para o ensino de diferentes conteúdos. Este trabalho apresenta um ambiente de realidade aumentada, denominado Seta, para o ensino de algoritmos de aprendizagem de máquina, especificamente de diferentes versões do algoritmo de clusterização *k-means* e dos algoritmos de regressão linear e polinomial. Dado um conjunto de pontos em um quadro branco. A imagem é capturada através de uma webcam conectada a um computador, que consiste na plataforma do sistema. O algoritmo de processamento de imagens que realiza a segmentação e interpretação das imagens de entrada utiliza as técnicas de limiarização global, morfologia matemática, rotulação e transformada de Hough. O professor pode selecionar quais algoritmos aplicar e então projetar os resutados do algoritmo sobre a imagem real, em tempo real, contendo os pontos desenhados O software foi desenvolvido em Java e utilizadas as bibliotecas de processamento de imagens OpenCV e ImageJ e a biblioteca de aprendizagem de máquina Weka. Os testes realizados demonstraram que o sistema é capaz de operar satisfatoriamente sob as condições de iluminação observadas nas situações reais, e com uma taxa de atualização satisfatória para a aplicação.

Palavras-chave: realidade aumentada, aprendizagem de máquina, informática no ensino.

ABSTRACT

MACHADO, Renan K.; FARINHAKI, Ricardo; DA SILVA, Thiago A.. SETA: AMBIENTE DE REALIDADE AUMENTADA PARA ENSINO DE ALGORITMOS DE APRENDIZAGEM DE MÁQUINA. Trabalho de Conclusão de Curso – Curso Superior de Engenharia de Computação, Universidade Tecnológica Federal do Paraná. Curitiba, 2013.

Augmented reality (AR) is the overlay of virtual images into images of a real environment, in real time. AR environments can be used as supplementary teaching resources for different content . This paper presents an augmented reality environment called SETA, for teaching machine learning algorithms, specifically different versions of k -means clustering algorithm and algorithms of linear and polynomial regression. Given a set of points on a whiteboard , the teacher can select which algorithms to apply and then display the results on the real image in real time, containing the plotted points . The image is captured using a webcam connected to a computer, which is the system platform. The image processing algorithm which performs segmentation and interpretation of the input images uses a global thresholding technique , mathematics morphology, labelling and Hough transform. The software was developed in Java and it was used image processing libraries (ImageJ and OpenCV) and machine learning library Weka . The tests showed that the system is able to operate satisfactorily under the lighting conditions observed in real situations , and with a refresh rate suitable for the application .

Keywords: augmented reality, machine learning, using computer on education.

Sumário

Lista de Siglas

RA - Realidade Aumentada

RV - Realidade Virtual

VR - Virtual Reality

AR - Augmented Reality

VE - Na língua inglesa significa *Virtual Environments* e em português Ambientes Virtuais.

1 INTRODUÇÃO

A Realidade Aumentada (RA) pode ser considerada uma tecnologia que permite imagens virtualmente geradas por computadores serem sobrepostas em um ambiente real e em tempo real. Com o desenvolvimento desta tecnologia, várias áreas profissionais começaram a empregá-la no seu dia-a-dia. Encontram-se exemplos reais de aplicação em serviços militares, em prospecção de terrenos na geologia, em simulação de voos ou de mergulhos, em áreas médicas como ferramenta de auxílio em cirurgias, e ainda em diversas áreas ligadas à educação.

Nos últimos anos, tanto em ambientes corporativos como em escolas, a utilização da RA vem sendo melhor aproveitada. Isto se deve às tecnologias atuais mais desenvolvidas que possibilitam construir softwares com uma ampla gama de funcionalidades para uso em computadores pessoais e smartphones.

1.1 MOTIVAÇÕES E OBJETIVOS

A introdução de novas práticas educacionais tem como foco suplantar as deficiências encontradas nos métodos tradicionais de ensino, como por exemplo, a busca do engajamento de estudantes através de tecnologias adequadas durante o processo de ensino (SILVA et. al, 2008).

Tem-se como objetivo geral deste trabalho o auxílio ao professor no ensino de algoritmos de aprendizagem de máquina utilizando realidade aumentada, de forma que seja possível maior interatividade no processo de exposição dos conteúdos e facilite a visualização resultados dos algoritmos por parte dos alunos.

Como cenário principal de uso, pode-se descrever uma sala de aula com o professor à frente, escrevendo em um pequeno quadro branco ou folha de papel. O professor desenha um gráfico 2D contendo pontos esparsos e pretende mostrar aos alunos como um algoritmo de clusterização k-means, por exemplo, irá classificar estes pontos. Então, o professor executa a ferramenta de auxílio de aprendizagem por realidade aumentada e as seguintes etapas são realizadas: 1) a imagem é adquirida por uma câmera conectada a um computador, posicionada de frente para o quadro; 2) os dados relevantes da imagem (pontos no gráfico) são segmentados e interpretados (coordenadas); 3) as coordenadas dos pontos são submetidas ao algoritmo k-means, que classifica estes pontos em k classes (k especificado pelo

professor); 4) os resultados do k-means são sobrepostos na imagem adquirida no passo 1) e esta imagem pode ser apresentada para os alunos. A figura 1 apresenta a visão geral do projeto de forma ilustrada.

Figura 1 – Visão Geral do projeto.
Fonte: Autoria própria

Os crescentes estudos e trabalhos que vêm sendo desenvolvidos com RA na última década apontam para um ambiente promissor em se tratando do desenvolvimento de novas tecnologias. A área da educação aparece como um dos principais alvos de desenvolvimento de software envolvendo esta tecnologia. Alguns exemplos de aplicativos de RA que já existem na área da educação são:

- Aplicativo para o ensino de astronomia que identifica constelações através da câmera de um smartphone;
- Aplicativo em que é possível colocar gráficos através da câmera do smartphone sobre cenários reais;
- Aplicativo que informa a latitude e longitude de uma posição geográfica de acordo com o destino que se deseja chegar;
- Aplicativo para o ensino de crianças através de recursos que constroem um mundo virtual para a criação de histórias (JUGARU, 2012);

Para atingir o objetivo principal do projeto, é necessário atingir os seguintes objetivos específicos:

- Compreensão dos algoritmos de aprendizagem de máquina;
- Compreensão dos algoritmos de Processamento Digital de Imagens;
- Compreensão de algoritmos e técnicas de RA;
- Especificação e modelagem do sistema de RA;
- Implementação, testes e correção no sistema;

O produto final desenvolvido ao fim desse trabalho constitui-se de um software, nomeado Seta, que fará a captura do que o professor desenha no quadro, o processamento desses dados e sua manipulação pelos algoritmos de aprendizado de máquina e, por fim, a apresentação dos dados já processados com o resultado final aos alunos.

2. FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta um resumo de tecnologias e conhecimentos que embasam o projeto.

2.1 REALIDADE AUMENTADA

Apesar de o termo Realidade Aumentada (*Augmented Reality - AR*) vir sendo correntemente empregado a partir da década de 90, sua origem pode ser identificada algumas décadas antes. A partir de 1957, Morton Helig começou a construir uma máquina chamada Sensorama, proposta como uma máquina cinematográfica de imersão multi-sensorial com diversas funcionalidades simultâneas como a vibração do assento em que o usuário estaria sentado, correntes de ar simulando vento real do ambiente, sons sendo tocados e a projeção de um ambiente 3D ao redor do campo de visão do espectador. Tal experimento, apesar de ser mais adequadamente classificado na sua integridade como um ambiente de Realidade Virtual (*Virtual Reality VR*), apresentava também elementos do que mais tarde passou a ser chamado de RA (SUNG, 2011).

Por volta de 1990, o termo RA foi colocado pelo professor Tom Caudell, que trabalhava nos serviços de computadores da Boeing. Em pesquisas para encontrar

melhores formas de manufatura e engenharia de processos, Caudell começou a utilizar tecnologia de Realidade Virtual e desenvolveu um software que poderia sobrepor as posições onde determinados cabos no processo de manufatura deveriam estar. Em 1992 L. B. Rosenberg criou o que é reconhecido como sendo o primeiro sistema de realidade aumentada para a força aérea dos Estados Unidos, ficando conhecido como *Virtual Fixtures*. Tratava-se de dicas, projetadas em letras muito grandes, para ajudar os usuários em suas tarefas (SUNG, 2011).

Até 1999, a tecnologia de Realidade Aumentada permaneceu restrita mais no âmbito de pesquisas científicas, envolvendo equipamentos caros e software complexos, do que como ferramenta para desenvolvimento de softwares comerciais para consumidores. O que tornou a área mais conhecida foi o lançamento do ARToolKit, por Hirokazu Kato do *Nara Institute of Science and Technology*, para a comunidade de código aberto. O ARToolKit consiste em uma biblioteca para construir diversos tipos de aplicações de RA e pela primeira vez foi possível fazer rastreamento de captura de vídeo do mundo real para combinar com a interação de objetos virtuais e gráficos 3D que podiam ser sobrepostos em qualquer plataforma de sistema operacional (SUNG, 2011).

A RA é uma tecnologia que permite imagens virtualmente geradas por computadores serem sobrepostas em um ambiente real e em tempo real. (ZHOU, DUH, & BILLINGHURST, 2008). A RA se diferencia da Realidade Virtual pois em VR as pessoas estão em um ambiente virtual gerado pelo computador, ao passo que em RA o ambiente é real mas com imagens de informações e dados virtuais do sistema. Em outras palavras, RA preenche a lacuna entre o real e o virtual (KANGDON LEE, 2002).

Segundo Milgram e Kishino (1994) a Realidade Aumentada é considerada uma variação do estudo de ambientes virtuais (*Virtual Environments VE*). MILGRAM e KISHINO (1994) explicam que a idéia de VE é a imersão de um indivíduo em um ambiente totalmente virtual no qual este agente não possui contato com o mundo real. A partir desta premissa, a AR se diferencia por auxiliar o indivíduo a observar o mundo real com objetos virtuais sobrepostos. Com isso, MILGRAM e KISHINO (1994) concluem que a AR suplanta a realidade, pois proporciona uma nova visão, na qual o real e o virtual coexistem. A imagem da Figura 2 foi retirada do artigo de AZUMA (1996) e mostra um exemplo de realidade aumentada. Nesta imagem pode-se observar um ambiente real, uma sala com uma mesa e elementos virtuais

sobrepostos. Os elementos virtuais inseridos são em 3D e se assemelham muito ao mundo real e que nos conduzem a dúvida do que é virtual ou não.

Figura 2: Mesa real com uma luminária e cadeiras virtuais.
Fonte: (AZUMA, 1996)

Para AZUMA (1996), a realidade aumentada apresenta as seguintes características:

- Combina o real com o virtual
- A interação ocorre em tempo real
- Interpreta os elementos em 3D

2.1.1 Problemas e Desafios da Realidade Aumentada

Segundo pesquisas de AZUMA (1996) a realidade aumentada encara alguns problemas que limitam o seu desenvolvimento sendo o principal o alinhamento dos objetos virtuais com o mundo real, que devem ser propriamente alocados para que a ilusão do objeto virtual e real coexistam. Este problema ocorre devido a erros de registro de imagens que são classificados como estáticos e dinâmicos (AZUMA, 1996). As principais fontes de erros estáticos são:

Distorção ótica: Este tipo de distorção existe na maioria de câmeras e sistemas de lentes e isto implica tanto na câmera que filma o ambiente quanto no projetor da imagem. Nas proximidades do centro do campo de visão a imagem é relativamente livre de distorção, porém se afastarmos do centro, a imagem fica mais distorcida.

Em geral estas pertubações da imagem são causadas em função da distância radial do eixo ótico.

Erros no sistema de rastreamento: Erros de saída do sistema no qual os sensores falham por não detectar uma mudança. Estes erros geralmente são não-sistêmicos o que dificulta a sua detecção e correção.

Desalinhamentos mecânicos: São as discrepâncias entre o modelo ou especificação do hardware e as atuais propriedades físicas do sistema real.

Parâmetros de visualização incorretos: Os parâmetros de visualização especificam como converter a imagem de saída da câmera em uma matriz para o gerador da imagem final. Alguns erros podem ser introduzidos nestes parâmetros e geralmente são sistêmicos.

Os erros dinâmicos podem ser causados em tempo de execução, por exemplo, devido ao atraso no processamento da imagem (AZUMA, 1996).

2.1.2 Processamento de imagens

Conversão para escala de cinza (*grayscale*): Se cada *pixel* de cor é definido pela tripla (R, G, B) representando a intensidade da tonalidade de vermelho (*red R),* verde (*green G)* e azul (*blue B)*, pode-se mapear esses valores para um único número que representa um valor em escala de cinza. Pode-se definir a conversão de imagens coloridas para escala de cinza seguindo os passos:

1) Receber os valores *red, green* e *blue* de um dado *pixel.*
2) Utilizar uma fórmula matemática para converter esses três números em um único valor em escala de cinza.
3) Substituir os valores originais *red, green* e *blue* pelo novo valor em escala de cinza (COOK, 2009).

Alguns exemplos de algoritmos que fazem a conversão para escala de cinza são:

1) *Lightness*: método faz a média da cor mais proeminente com a cor menos proeminente entre as três escalas - (max(R, G, B) + min(R, G, B)) / 2.
2) *Average*: método que faz a conversão pela média aritmética dos três valores – (R + G + B)/3.
3) *Luminosity*: é realizada uma média ponderada para adequar o resultado à percepção humana. O ser humano tende a ser mais sensível à cor verde

do que às outras cores, então a escala *green* tem peso maior, seguida da escala *red e da blue* - 0,21 R + 0,71 G + 0,07 B (COOK, 2009).

Binarização: A conversão de uma imagem em níveis de cinza para uma imagem com representação binária (apenas dois níveis) é importante para inúmeros objetivos, como por exemplo:

- identificar objetos e separá-los do fundo da imagem;
- quando analisar a forma da imagem é mais importante que a intensidade dos pixels;
- apresentar a imagem em um dispositivo de saída que tem somente um bit de resolução de intensidade, ou seja, um dispositivo de dois níveis, como uma impressora.

A figura 3 mostra um histograma de uma determinada imagem contendo pixels de níveis de cinza claros e escuros, em uma distribuição denominada bimodal. (CARNEIRO, 2013):

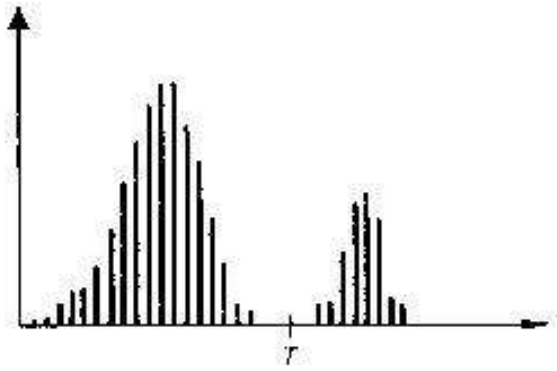


Figura 3: histograma de imagem mais clara sob um fundo escuro

Fonte: (CARNEIRO, 2013)

A transformação chamada de binarização retorna uma imagem contendo apenas dois níveis de cinza, e pode ser descrita através da aplicação da função: s = T(r). A figura 4 ilustra esse procedimento (CARNEIRO, 2013).

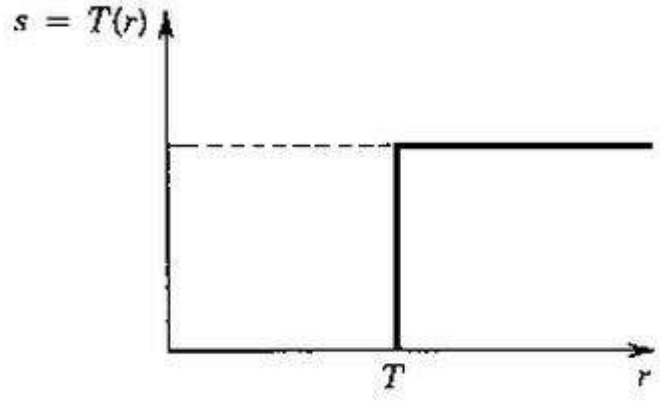


Figura 4: binarização através da função s = T(r)
Fonte: (CARNEIRO, 2013)

A função T(r) compara o sinal de entrada com um valor de *threshold* (T), escolhido como referência para a separação dos níveis de cinza. O sinal de saída, apresentado é obtido pela relação: s = 1 para r > T; s = 0 para r < T.

Transformada de Hough: A Transformada de Hough (*Hough Trasform - HT)* foi desenvolvida por Paul Hough, em 1962, e foi patenteada pela IBM logo em seguida. Foi elaborada com a proposta para detectar as características analiticamente representáveis em imagens binarizadas, assim como figuras básicas como linhas, círculos e elipses.

Na última década se transformou em uma ferramenta de uso comum na visão artificial para o reconhecimento destas características.

A Transformada de Hough consiste em um método padrão para detecção de formas que facilmente parametrizadas (linhas, círculos, elipses, etc.) em imagens computacionais. Normalmente, a transformada é aplicada na imagem após um pré-processamento nesta imagem, como por exemplo a detecção de bordas de uma figura (JAMUNDÁ, 2000).

O conceito central da Transformada de Hough é criar um mapeamento entre o espaço de imagem e o espaço de parâmetros. Cada borda da imagem é transformada através desse mapeamento para indicar as células no espaço de parâmetros, determinadas pelas primitivas definidas através do ponto analisado. Essas células são incrementadas e indicarão, por meio da máxima local do acumulador, quais os parâmetros correspondentes à forma especificada (JAMUNDÁ, 2000).

A transformada de Hough permite mapear um pixel da imagem em uma curva no espaço de parâmetros, organizado em forma de um acumulador n dimensional, em que n representa o número de parâmetros.

Há várias parametrizações possíveis para o espaço de linhas. Como representação paramétrica de uma linha, por exemplo, Hough usou a equação demonstrada na figura 5.

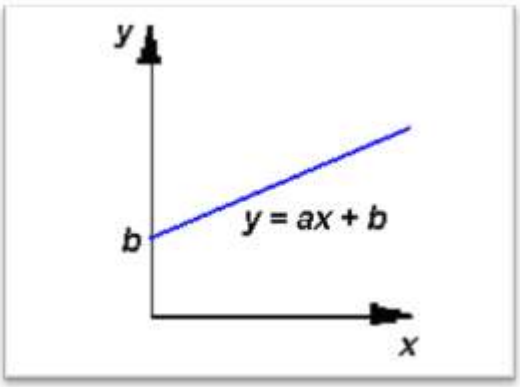


Figura 5: Parametrização linear

Fonte: (JAMUNDÁ, 2000)

O algoritmo de Hough exige um acumulador de dimensão igual ao número de parâmetros desconhecidos na equação da família de curvas que estão sendo buscadas. Assim, encontrar segmentos de linhas por meio da equação requer achar dois parâmetros para cada segmento: a e b da equação 1.

$$y = ax + b \qquad (1)$$

Utilizando uma matriz acumuladora A, o algoritmo de Hough analisa cada *pixel* e calcula os parâmetros da curva especificada para encontrar a equação da curva que passa pelo *pixel*. Depois de calcular os parâmetros de um determinado *pixel*, eles são ‘quantizados’ para um valor correspondente *a* e *b*, e o acumulador A(a , b) é incrementado. Quando todos os *pixels* tiverem sido processados, procura-se no acumulador A os maiores valores. Eles indicam os parâmetros das prováveis linhas na imagem.

Morfologia Matemática: Morfologia Matemática (*Mathematical Morphology - MM)* é uma técnica para análise e processamento de estruturas geométricas, baseada na teoria dos conjuntos, em estudo de topologias e em matemática discreta. A *MM* consiste em uma ferramenta bastante utilizada para o processamento de imagens e de outros dados discretos. Duas das principais operações estudadas em MM são as

operações de dilatação e de erosão (BURGER; BURGE, 2006). O Elemento Estruturante na transformação morfológica se fundamenta na comparação de uma determinada imagem com outra imagem menor, chamada de elemento estruturante, e cuja geometria é conhecida. Em Morfologia Binária (*Binary Morphology)*, o elemento estruturante (*structuring element)* consiste em uma matriz com valores 0 ou 1, apenas (BURGER; BURGE, 2006):

$$\mathrm{H(i,j)} \in \{0,1\} \quad (2)$$

Dilatação é uma das operações fundamentais da Morfologia Matemática é a operação de dilatação. A operação de dilatação pode ser vista como a dilatação do elemento estruturante H sendo replicado em cada pixel de primeiro plano (*foreground pixel* - *pixel* cujo valor é 1) da equação 3:

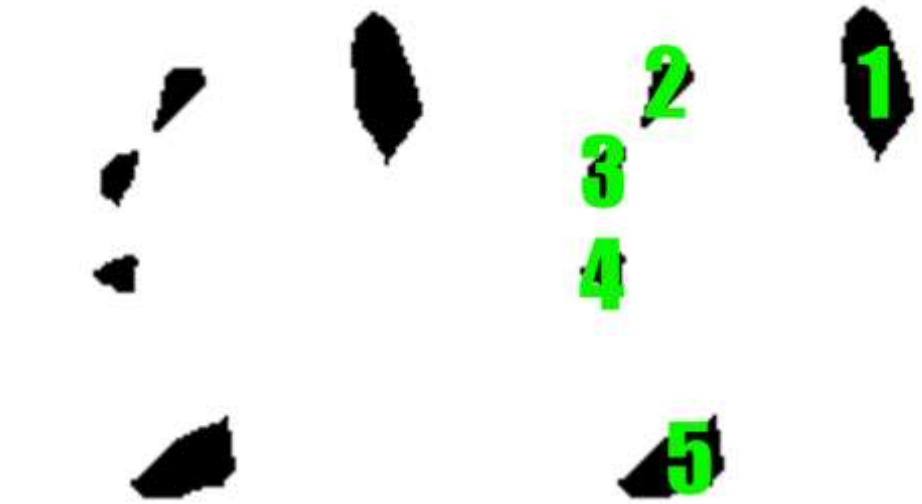
$$\mathbf{I} \oplus \mathbf{H} = \cap_{\mathbf{p}\epsilon\mathbf{I}} \quad \mathbf{H_p} = \cap_{\mathbf{q}\epsilon\mathbf{H}} \quad \mathbf{I_q} \quad (3)$$

com Hp, Iq denotando os conjuntos H, I deslocados por p e q, respectivamente (BURGER; BURGE, 2006).

A operação de erosão pode ser definida como sendo praticamente a operação inversa da operação de dilatação. Uma posição p no resultado $\mathrm{I} \ominus \mathrm{H}$ se, e somente se, o elemento estruturante H - quando colocado na posição p - está completamente contido nos *pixels* de primeiro plano da imagem original. Ou seja, se $\mathrm{H_p}$ é um subconjunto de I. Assim, analogamente a operação de dilatação, pode-se definir erosão por meio da equação (BURGER; BURGE, 2006):

$$\mathrm{I} \ominus \mathrm{H} = \{\mathrm{p} \in \mathrm{Z}^2 | \mathrm{H_p} \subseteq \mathrm{I}\} \quad (4)$$

O algoritmo conhecido como *Region Labeling* tem dois passos fundamentais: (1) rotular (*label*) preliminarmente as regiões da imagem e (2) resolver casos onde determinadas regiões são rotuladas mais de uma vez (BURGER; BURGE, 2006). A figura 6 ilustra o procedimento de *labelling* e de segmentação.



(A) (B)

Figura 6: Imagem original (A) e imagem após *labelling* (B).

Fonte: Autoria própria

A tarefa principal do conjunto dos procedimentos de segmentação e *labelling* em uma dada imagem é pré-requisito para a maioria dos sistemas de reconhecimento e de classificação.

2.1.3 Projetos existentes de Realidade Aumentada

Nesta seção são apresentados alguns exemplo de projetos envolvendo diferentes conceitos de Realidade Aumentada.

O ARQuake é uma versão do popular jogo Quake que utiliza a realidade aumentada para a interação do usuário com o jogo. Segundo THOMAS et. al. (2000), o objetivo do projeto é desenvolver um aplicação com a perspectiva em primeira pessoa com os seguintes atributos:

1) A aplicação está situada no mundo real;

2) O ponto de vista mostrado pela aplicação mostra ao usuário é determinado pela posição e orientação de sua cabeça;

3) Informações relevantes são mostradas através de um display semi-transparente;

4) O usuário é livre para andar através do espaço mostrado no display;

5) A aplicação será operacional tanto em ambientes externos quanto internos;

6) A interface do usuário com o ambiente se dá apenas com um botão.

Na Figura 7, pode-se observar um tela do jogo ARQuake.

Figura 7: ARQuake.

Fonte: http://wearables.unisa.edu.au/projects/arquake/

ARiSE é um acrônimo em inglês para Realidade Aumentada em Ambientes Escolares (*Augmented Reality in School Environments*) e é uma plataforma que tem o objetivo de utilizar apresentações 3D para o melhor entendimento de conteúdo através da maior motivação por parte do aluno. Para isso, a plataforma integra a realidade aumentada no ambiente do dia-a-dia do aluno e professor, apresentando conteúdo audio-visual e multimídia de acordo com as necessidades identificadas por estudiosos na área do ensino. A figura 8 presenta um exemplo de ARiSE.

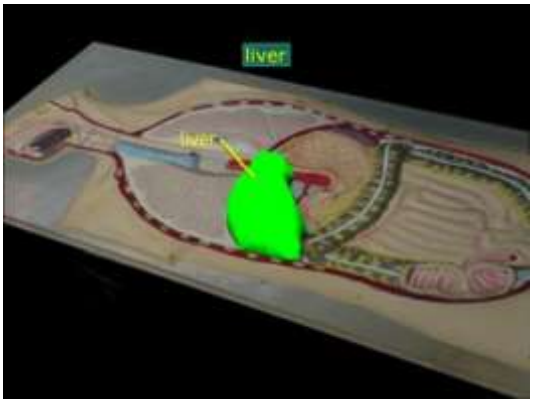


Figura 8: Aplicação do ARiSE no ensino sobre o corpo humano.

Fonte: http://www.arise-project.org/uploads/pics/scenario1-1_03.jpg

2.2 ALGORITMOS DE APRENDIZAGEM DE MÁQUINA

A área de estudo da aprendizagem de máquina busca construir sistemas computacionais que melhoram automaticamente seu desempenho de acordo com a experiência. Um de seus focos é o reconhecimento de padrões complexos em um conjunto de dados e, assim, tomar decisões inteligentes para uma determinada ação (MITCHELL, 2006). Esta família de algoritmos é útil para problemas sobre os quais o conhecimento humano não dispõe de uma quantidade relevante de dados, passando para a máquina a tarefa de adquirir novos conhecimentos e remodelá-los. Dessa forma, essa classe de algoritmos auxilia o computador a executar diversas ações sem que seja previamente programado com algoritmos específicos para essas ações (ZHANG, 2000). Os algoritmos de aprendizagem de máquina são classificados em taxonomias de acordo com o resultado desejado. As principais taxonomias são: aprendizagem supervisionada (*supervised learning*), aprendizagem não-supervisionada (*unsupervised learning*), aprendizagem semi-supervisionada (*semi-supervised learning*), aprendizagem por reforço (*reinforcement learning*), transdução (*transduction*) e *learning to learn.* Optou-se, neste trabalho, por concentrar os estudos na primeira e segunda classe de algorimos.

2.2.1 Aprendizagem supervisionada e não-supervisionada

Na aprendizagem supervisionada, o objetivo geralmente é fazer o sistema aprender um sistema de classificação criado pelo programador. Reconhecimento de dígitos utilizando-se redes neurais é um exemplo de aplicação de aprendizado por classificação.

Em Inteligência Artificial, a aprendizagem supervisionada é a técnica mais comum para treinamentos de redes neurais e de árvores de decisão. Assim a técnica depende de classificações pré-determinadas que, em cada caso específico tem um propósito. Nas redes neurais, por exemplo, a classificação é utilizada para determinar o erro da rede e com isso tentar minimizá-lo.

O objetivo na aprendizagem não-supervisionada aparenta ser mais complexo, pois busca-se que o computador faça alguma coisa sem que o informemos

previamente de como fazer. Existem algumas principais abordagens no âmbito do aprendizado não-supervisionado. A primeira consiste em um sistema de recompensas, no qual o objetivo não é gerar nenhuma classificação, mas sim produzir decisões que maximizem as recompensas. Essas recompensas indicam o caminho correto, o sucesso da ação. Essa forma de aprendizado pode ser interessante porque não assume nenhum exemplo de classificação pré-descoberto.

Um dos métodos mais comuns de aprendizagem não-supervisionada é o da clusterização(*clustering*). Nesse modelo de algoritmo o objetivo não é maximizar a função de utilidade (*utility function*), mas em vez disso simplesmente encontrar similaridades nos dados de treinamento. A ideia é que as descobertas dos *clusters* corresponderão a uma classificação intuitiva. Um exemplo de aplicação nesse segmento é a classe de algoritmos que filtram informações sociais, como no site *Amazon.com*, em que são recomendados livros aos usuários com base no princípio de encontrar grupos semelhantes de pessoas e então atribuir novos usuários ao grupo.

2.2.2 Algoritmos de Clusterização

Algoritmos de clusterização dividem os dados em classes úteis ou significativas designadas *clusters*, nas quais a similaridade entre dados de uma mesma classe é maximizada e a similaridade entre dados de classes diferentes é minimizada. Estes *clusters* descobertos podem ser usados para indicar as características da distribuição dos dados subjacentes e com isso servir como base para várias técnicas de análise e *Data Mining (*mineração de dados). Algumas aplicações genéricas da *clusterização* incluem, por exemplo, a caracterização de diferentes grupos de clientes baseado nos padrões de compra, classificação de documentos na internet, agrupamento de genes e de proteínas que possuem funcionalidade semelhante (FONSECA; BELTRAME, 2011).

K-means Clustering: O *K-Means Clustering,* ou simplesmente *K-Means,* é um tipo de algoritmo que pode ser enquadrado na categoria de algoritmos de aprendizagem não-supervisionada. É um dos algoritmos de *Data Mining* que serão empregados no projeto. A ideia principal do algoritmo é classificar as informações de acordo com os próprios dados. Essa classificação é baseada em comparações e análises entre os

valores numéricos desses dados. Por isso, será gerada sempre uma classificação automática sem a necessidade de monitoramento humano, isto é, sem uma pré-classificação existente. O algoritmo *K-Means* é numérico, não supervisionado, não-determinístico e iterativo (AYODELE 2012).

Como exemplo, pode-se imaginar uma tabela com linhas e colunas com os dados a serem classificados. Cada coluna é chamada de dimensão, e cada linha contém informações das dimensões (chamadas de ocorrências ou pontos). O algoritmo então indicará um *cluster* (classe) e determinará as linhas que pertencem a essa classe. O usuário fornece a quantidade k de classes a serem utilizadas (o k, do nome *K-Means*, tem origem neste número de classes).

Finalmente, para criar as classes e classificar as ocorrências, o algoritmo pode comparar cada valor de cada linha utilizando a distância euclidiana como base. Então calcula-se o quanto uma ocorrência está longe de outra. Em seguida é definido um centróide para cada classe estipulada, que vai sendo refinado iterativamente assim que novas ocorrências vão sendo analisadas. São gerados portanto k centróides e as ocorrências da tabela serão dispostas conforme suas respectivas distâncias em relação ao centróide.

Dessa forma, pode-se construir o algoritmo *K-Means* seguindo os seguintes passos:

- O conjunto é particionado em *k* classes sendo representadas por *k* centróides. No início esses centróides são gerados aleatoriamente ou então escolhe-se os *k* primeiros valores da tabela;
- Calcula-se a distância entre cada ponto do conjunto e cada centróide. Se há N pontos e *k* centróides, serão calculadas N x *k* distâncias;
- Os pontos são então classificados de acordo com sua distância dos centróides de cada classe. O ponto vai pertencer à classe representada pelo centróide que está mais perto do ponto. O algoritmo termina se nenhum ponto mudar de classe na interação, ou seja, se ao final de duas iterações consecutivas todos os pontos permanecerem na mesma classe;
- Os valores das coordenadas dos centróides de cada classe são calculados novamente. Para cada classe que possui mais de um ponto, o valor novo dos centróides é calculado por meio da média das coordenadas de cada ponto que pertence a esta classe;

- O algoritmo volta para o passo 2 da iteração repetindo o ‘refinamento’ do cálculo das coordenadas dos centróides (WITTEN; FRANK, 2005).

O método *Farthest First* é uma modificação do *K-Means.* Este algoritmo coloca os centróides de cada classe, no momento do cálculo dos novos centróides, no ponto mais afastado desde centróide de forma que esteja dentro da mesma classe. Procedendo-se desta maneira a velocidade de *clusterização* é aumentada na maioria dos casos, já que são realizadas menos modificações e realinhamentos. (WEKA, 2013).

No método Hierárquico o processo de identificação das classes (*clusters*) é ordenado recursivamente, utilizando tanto objetos quanto grupos já identificados previamente como entrada para o processamento. Assim, cria-se uma hierarquia de grupos de objetos, no formato de uma árvore. Os métodos hierárquicos possuem algumas características particulares:

- São divididos em dois tipos: aglomerativos e divisivos. Nos métodos aglomerativos, uma vez que dois elementos são unidos, eles permanecem unidos até o final do procedimento. Nos métodos divisivos acontece o contrário, isto é, uma vez que dois elementos são separados, eles jamais voltarão a fazer parte do mesmo agrupamento;
- Como se trata de um método hierárquico, é possível saber de onde um determinado elemento veio anteriormente, ou seja, é mantido um histórico dos passos anteriores de cada elemento;
- Possui uma desvantagem bastante significativa, pois são impraticáveis para grandes bases de dados devido ao seu alto custo computacional. (KAUFMAN; ROUSSEEUW, 2005).

2.2.3 Algoritmo de Regressão Linear e de Regressão Polinomial

O algoritmo de regressão linear é um método para se estimar o valor condicional esperado de uma variável *y*, dados os valores de outras variáveis *x*. A variável de resposta é uma função linear de certos parâmetros, por isso o nome de regressão linear. A figura 9 mostra um exemplo de regressão linear.

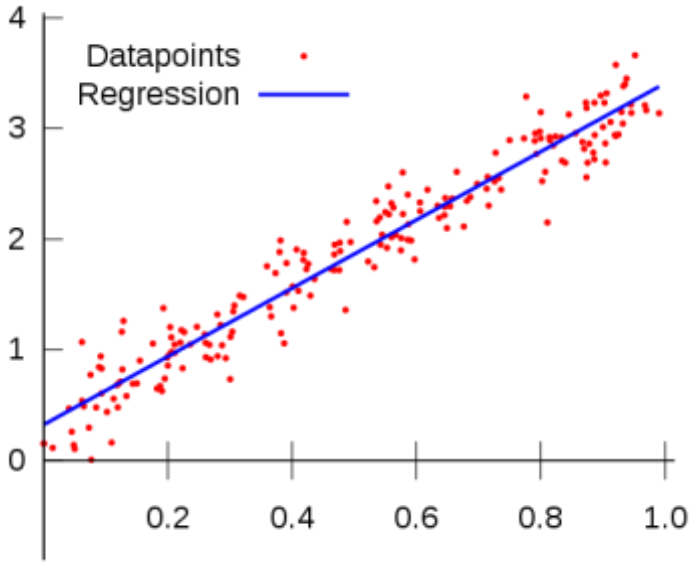


Figura 9: Regressão Linear

Fonte: (WIKIPEDIA, 2013)

Por outro lado, a variável resposta pode ser um polinômio de grau maior do que 1, caso em que o método é chamado de regressão polinomial, como ilustra a figura 10.

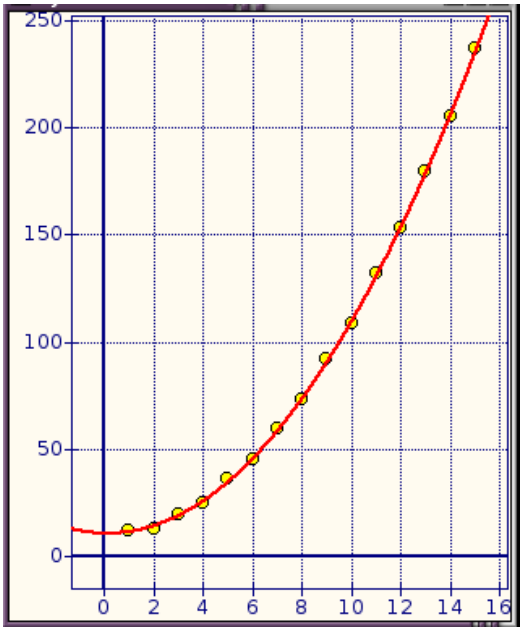


Figura 10: Regressão Polinomial

Fonte: (WIKIPEDIA, 2013)

Dessa forma a regressão polinomial é uma generalização da regressão linear. Pode-se obter o resultado da regressão linear a partir de um ajuste da equação 5 na qual α_n representam os coeficientes da equação e x é a variável :

$$y = \alpha_0 + \alpha_1 x + \epsilon \qquad (5)$$

Já para a regressão polinomial utiliza-se a equação 6 completa a qual α_n representam os coeficientes da equação e x é a variável:

$$y = \alpha_0 + \alpha_1 x + \alpha_2 x^2 + \ldots + \alpha_m x^m + \epsilon \qquad (6)$$

Para obter os parâmetros dessa função, deve-se resolver um sistema de $m + 1$ equações lineares. (SEBER; LEE, 2003).

3. ESPECIFICAÇÕES DO PROJETO

Este capítulo apresenta os requisitos funcionais, e não funcionais e a modelagem UML do projeto.

3.1 REQUISITOS FUNCIONAIS

Requisitos funcionais são aqueles determinados pelas funcionalidades exigidas do sistema, ou seja, as ações que se espera que o sistema execute. Foram levantados os seguintes requisitos funcionais:

- O sistema deve extrair informações de um quadro branco com linhas e pontos em cor preta;
- O sistema deve reconhecer o desenho de um gráfico bidimensional desenhado em um quadro;
- O sistema deve ter uma IHM amigável e intuitiva;
- O sistema deve ser capaz de calcular os seguintes algoritmos de clusterização: *K-Means*, Hierárquico e *Farthest First*;
- O sistema deve projetar na imagem reproduzida pela câmera o resultado do cálculo do algoritmo de regressão linear;
- O sistema deve projetar na imagem reproduzida pela câmera o resultado do cálculo do algoritmo de regressão polinomial;
- O sistema deve habilitar o usuário a gravar a imagem gerada;
- O sistema deverá ser capaz de exibir vídeo na resolução de 640x480 *pixels*.

3.2 REQUISITOS NÃO-FUNCIONAIS

Os requisitos não funcionais tem o propósito de descrever a arquitetura proposta para o Seta Realidade Aumentada, assim como abordar as diferentes linguagens e bibliotecas que ajudam a integrar o programa.

- O sistema deve reconhecer o gráfico bidimensional em menos de 10 segundos após o desenhado;
- O sistema deve executar os algoritmos em menos de 10 segundos;
- O sistema deve possuir suporte para câmera e área de desenho;
- O sistema deve integrar com a biblioteca Weka;
- O sistema deve possuir interface com o *driver* da câmera;
- A câmera deve ter uma resolução mínima de 1 *Mega Pixel*;
- A Interface gráfica do sistema necessita que haja um Java Runtime Environment (JRE) instalado no sistema operacional;
- A ferramenta de exibição deve ser executada em um sistema capaz de prover um *display* com resolução igual ou maior que a dos vídeos a serem exibidos;
- O sistema deve ser desenvolvido utilizando linguagem de programação Java;
- O sistema deve ser uma aplicação *desktop*;
- O sistema deve possibilitar a instalação em ambiente Windows versões XP, Vista e 7.

3.3 MODELAGEM UML

Antes de um sistema ser desenvolvido é necessário projetar o sistema utilizando práticas de engenharia de *software*. Neste caso a opção escolhida foi a linguagem UML. A linguagem UML auxilia na especificação, visualização e documentação de sistemas de *software*, incluindo a estrutura e *design* de modo que os requerimentos sejam competidos (OMG, 2013).

3.3.1 Diagrama de Casos de Uso

A figura 11 apresenta o diagrama de caso de uso do projeto. O texto a seguir descreve detalhadamente cada caso de uso.

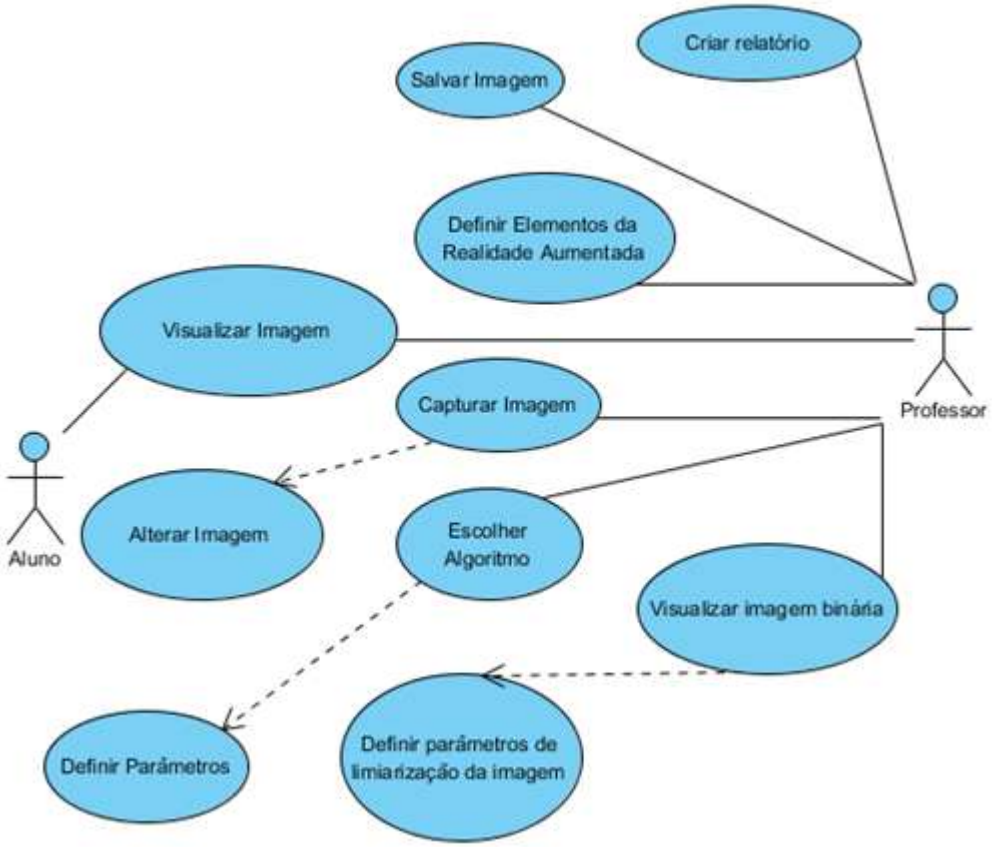


Figura 11: Diagrama de casos de uso do projeto SETA

Fonte: Autoria Própria

Diagrama de caso de uso: Criar relatório

Ator Principal: Professor

Descrição: O professor registra imagens e anotações em um relatório em formato eletrônico pdf.

Pré-Condições:

Professor habilitado no sistema.

Pós-Condições:

Arquivo pdf gerado com as anotações e imagens extraídas pelo professor.

Fluxo Básico:

1. Selecionar o frame da imagem desejado.
2. Se necessário adicionar comentário para a imagem.
3. Salva as informações.

Fluxo Alternativo:

- O usuário pode a qualquer momento cancelar a imagem selecionada, deletar uma imagem salva e/ou sair do sistema.
- Se não houver imagem a ser capturada, será notificado o evento ao usuário.

Regras de Negócio:

O usuário deve estar utilizando o sistema. A câmera deve estar ligada e configurada no computador, e suas especificações devem preencher os pré-requisitos estipulados.

Diagrama de caso de uso: Salvar Imagem

Ator Principal: Professor

Descrição: Salvar imagem projetada no painel principal da aplicação em formato PNG.

Pré-Condições:

Câmera ligada e conexão com o sistema.

Pós-Condições:

Imagem salva em local desejado no formato pdf.

Fluxo Básico:

1. Professor acessa o sistema.
2. Seleciona o frame desejado.
3. Salva a imagem no local desejado.

Regras de Negócio:

- O computador que hospeda a aplicação deve disponibilizar espaço em disco para que seja possível salvar a imagem.

Diagrama de caso de uso: Definir elementos da realidade aumentada

Ator Principal: Professor

Descrição: Este caso de uso possibilita a escolha dos elementos virtuais que serão inseridos na imagem.

Pré-Condições:

Câmera ligada e conexão com o sistema.

Pós-Condições:

Imagem definida a partir dos parâmetros definidos pelo professor.

Fluxo Básico:

1. Professor ativa e desativa os elementos que deseja.

2. A imagem é defina pelos parâmetros escolhidos.

Regras de Negócio:

- A câmera deve capturar a imagem do desenho contida no gráfico XY desenhado com caneta preta em um fundo branco.

Diagrama de caso de uso: Visualizar Imagem

Ator Principal: Professor

Ator Secundário: Aluno

Descrição: Visualização da imagem gerada pela câmera e elementos virtuais.

Pré-Condições:

Câmera ligada e conexão com o sistema.

Pós-Condições:

Imagem exibida no painel principal da aplicação.

Fluxo Básico:

1. Professor conecta a câmera ao computador.
2. Liga o sistema.
3. Professor desenha gráfico XY.
4. Visualização através do monitor do computador ou a partir de um projetor.

Regras de Negócio:

- Câmera conectada ao computador que tem o sistema instalado.
- Elemento de projeção da imagem (monitor ou projetor)

Diagrama de caso de uso: Capturar Imagem

Ator Principal: Professor

Descrição: Captura da imagem necessária para o processamento e inserção dos elementos virtuais.

Pré-Condições:

Câmera ligada e conexão com o sistema.

Pós-Condições:

Imagem padrão capturada.

Fluxo Básico:

1. Professor conecta a câmera ao computador.
2. Liga o sistema.
3. Professor desenha gráfico bidimensional.

Regras de Negócio:

- A iluminação do ambiente deve ser adequada, para que se possa distinguir a diferença entre o quadro e desenhos.
- O professor deve desenhar linhas e pontos com pouca imperfeição.

Diagrama de uso: Alterar imagem

Ator Principal: Professor

Ator de suporte: Aluno

Descrição: O professor pode realizar alterações no gráfico capturado inserindo novos pontos ou retirando pontos já existentes.

Pré-Condições:

Professor habilitado no sistema e imagem já capturada.

Pós-Condições:

Imagem modificada.

Fluxo Básico:

1. Selecionar a imagem com a qual deseja trabalhar.
2. Definir se deseja acrescentar ou remover pontos.
3. Colocar os pontos a serem acrescentados ou os pontos a serem removidos.

Regras de Negócio:

- As imagens devem ter sido corretamente capturadas.

Diagrama de caso de uso: Escolher Algoritmos

Ator Principal: Professor

Descrição: O sistema deverá proporcionar a escolha de algoritmos que poderão ser aplicados nas imagens recebidas através da interface da câmera.

Pré-Condições:

Professor habilitado no sistema e imagem já capturada.

Pós-Condições:

Imagem com elementos virtuais que representam os resultados do processamento dos algoritmos.

Fluxo Básico:

1. Selecionar a imagem com a qual deseja-se trabalhar.
2. Escolher o algoritmo disponibilizado pelo sistema.
3. Visualizar a imagem com os resultados representados em elementos virtuais.

Regras de Negócio:

- As imagens deverão ter sido corretamente capturadas.
- A imagem extraída deverá conter um gráfico XY desenhado e pontos esparsos que representam dados de exemplo.

Diagrama de caso de uso: Definir Parâmetros

Ator Principal: Professor

Descrição: O sistema deverá proporcionar a escolha de parâmetros relativos aos algoritmos disponibilizados pelo sistema.

Pré-Condições:

Professor habilitado no sistema, imagem capturada e algoritmo escolhido.

Pós-Condições:

Imagem com elementos virtuais que representam os resultados do processamento dos algoritmos e seus parâmetros.

Fluxo Básico:

1. Selecionar a imagem com a qual deseja-se trabalhar.
2. Escolher o algoritmo disponibilizado pelo sistema.
3. Definir os parâmetros do algoritmo.
4. Visualizar a imagem com os resultados representados em elementos virtuais.

Regras de Negócio:

- As imagens deverão ter sido corretamente capturadas.
- A imagem extraída deverá conter um gráfico XY desenhado e pontos esparsos que representam dados de exemplo.

Diagrama de caso de uso: Visualizar Imagem Binária

Ator Principal: Professor

Descrição: Visualização da imagem binária gerada pela câmera.

Pré-Condições:

Câmera ligada e conexão com o sistema.

Pós-Condições:

Imagem exibida no painel principal da aplicação.

Fluxo Básico:

1. Professor conecta a câmera ao computador.
2. Liga o sistema.

3. Professor desenha gráfico XY.

4. Professor ativa a opção de imagem binária

4. Visualização através do monitor do computador ou a partir de um projetor.

Regras de Negócio:

- Câmera conectada ao computador que tem o sistema instalado.
- Elemento de projeção da imagem (monitor ou projetor)

Diagrama de caso de uso: Definir parâmetros de limiarização de imagem

Ator Principal: Professor

Descrição: Definição de limite de limiarização binária da imagem gerada pela câmera.

Pré-Condições:

Câmera ligada e conexão com o sistema.

Pós-Condições:

Imagem exibida no painel principal da aplicação.

Fluxo Básico:

1. Professor conecta a câmera ao computador.
2. Liga o sistema.
3. Professor desenha gráfico XY.
4. Professor ativa a opção de imagem binária.
5. Definição de um *treshold* na qual o pixel pode ser considerado “0” ou “1”
4. Visualização através do monitor do computador ou a partir de um projetor.

Regras de Negócio:

- Câmera conectada ao computador que tem o sistema instalado.
- Elemento de projeção da imagem (monitor ou projetor)

3.3.2 Diagrama de Classes

É possível encontrar o detalhamento do diagrama de classes no ANEXO A.

4. DESENVOLVIMENTO

4.1 GESTÃO DE PROJETO

Neste projeto foi utilizado a ferramenta diagrama de Gantt (cronograma detalhado) na qual possibilita a visualização da estrutura de tarefas que devem ser realizadas para a finalização do projeto bem como o tempo necessário para o desenvolvimento. No ANEXO B é possível verificar o diagrama de Gantt do projeto por completo.

O planejamento deste projeto é descrito no quadro 01.

Data de início	**Atividade**	**Duração**	**Integrantes**
01/01/2013	Estudo dos algoritmos de segmentação de imagem e realidade aumentada.	2 meses	Renan, Ricardo e Thiago
01/03/2013	Estudo de algoritmos de clusterização.	1 mês	Renan
01/03/2013	Especificação da ferramenta a ser desenvolvida e desenvolvimento do projeto.	2 meses	Ricardo e Thiago
01/04/2013	Implementação dos algoritmos de clusterização.	1 mês	Renan
01/05/2013	Testes e Desenvolvimento do documento final do trabalho.	1 mês	Renan, Ricardo e Thiago

Quadro 01 - Atividades planejadas

Conforme planejamento das atividade se estimou 8 horas de trabalho semanais de cada participantes que resultaram em 580 horas de trabalho. Na tabela 02 é possível ver os custos do projeto.

Descrição	**Custo (R$)**
580h de trabalho de engenharia (R$38,00/h)	22.040,00

Câmera web cam	100,00
Suporte	100,00
Quadro branco e caneta piloto	50,00
Total:	22.290,00

Quadro 02 - Custos do projeto

4.1.1 DADOS DO PROJETO

Na figura 12 é apresentado os dados reais do desenvolvimento deste projeto.

Estudo e Pesquisa	**01/06/13**	**18/07/13**	**35**
Estudo sobre processamento de imagens	01/06/13	19/06/13	14
Testes com bibliotecas existentes	20/06/13	09/07/13	14
Prototipagem Matlab	10/07/13	17/07/13	6
Definição Metodologia	18/07/13	18/07/13	1
Desenvolvimento	**20/07/13**	**08/11/13**	**81**
Desenvolvimento interface camera	20/07/13	29/07/13	7
Processamento da Imagem	30/07/13	27/09/13	44
Binarização da Imagem	30/07/13	12/08/13	10
Erosão e Dilatação	13/08/13	16/08/13	4
Labeling	19/08/13	03/09/13	12
Centróides	04/09/13	13/09/13	8
Algoritmo de Hough	16/09/13	27/09/13	10
Desenho Interface da Aplicacao	19/08/13	10/10/13	39
Algoritmo Kmeans	02/09/13	01/11/13	45
Regressão Linear	02/10/13	08/11/13	28
Integração WEKA	17/10/13	08/11/13	17
Testes, Refinamento e Fechamento	01/09/13	25/11/13	62
Relatório	01/09/13	25/11/13	62
Testes e Ajustes da Aplicação	08/11/13	25/11/13	12

Figura 12 - Dados referente ao desenvolvimento das tarefas.

Fonte: Autoria Própria

Com o objetivo de confrontar os dados reais e dados planejados, verifica-se uma grande diferença entre as datas de inicio e final do projeto. Isto se deve ao fato do calendário acadêmico estar defasado devido à greve. Também se nota que o tempo gasto em desenvolvimento foi maior que o planejado e isso se deve ao fato do refinamento de escopo no fim do projeto.

4.2 PLATAFORMA DE HARDWARE

Foi utilizado como hardware um computador PC comum e uma câmera do tipo *webcam* para realizar a aquisição das imagens do quadro branco.

4.3 RECURSOS DE SOFTWARE

A partir da análise das bibliotecas de processamento de imagem existentes e necessárias para este projeto, é possível concluir que a linguagem Java pode ser utilizada. É descrito as vantagens da utilização desta linguagem neste projeto:

- Utiliza o paradigma de orientação à objeto;
- Facilidades de Internacionalização - Suporta nativamente caracteres Unicode;
- Simplicidade na especificação, tanto da linguagem como do "ambiente" de execução (JVM);
- É distribuída com um vasto conjunto de bibliotecas (ou APIs);
- Possui facilidades para criação de programas distribuídos e multitarefa (múltiplas linhas de execução num mesmo programa);
- Desalocação de memória automática por processo de coletor de lixo;
- Carga Dinâmica de Código - Programas em Java são formados por uma coleção de classes armazenadas independentemente e que podem ser carregadas no momento de utilização.

4.4 BIBLIOTECAS UTILIZADAS

A biblioteca ImageJ foi projetada para o ambiente Java e contém algoritmos de análise de imagem. A API desta biblioteca pode ser encontrada no link: http://rsbweb.nih.gov/ij/developer/api/index.html

A biblioteca OPENCV foi desenvolvida pela Intel para desenvolvimento de softwares da área de visão computacional. Essa biblioteca possui mais de 500 funções e está dividida em seis grupos:

1) Processamento de imagens;
2) Análise estrutural,
3) Rastreamento de objetos;

4) Reconhecimento de padrões;

5) Calibração de câmera e reconstrução em 3D.

A API desta biblioteca e um detalhamento do conteúdo desta biblioteca pode ser encontrada no link: http://opencv.willowgarage.com/wiki/

A biblioteca Weka é uma coleção de algoritmos de aprendizagem de máquina para tarefas de mineração de dados. O algoritmo pode ser aplicado diretamente em um grupo de dados assim definido em um código Java. Weka possui ferramentas para pré-processamento de dados, classificação, regressão, clusterização, regras de associação, e visualização. O Weka é também utilizado no desenvolvimento de novas estratégias de aprendizagem de máquina (WEKA, 2013).

4.5. METODOLOGIA DE DESENVOLVIMENTO DO PROJETO

O planejamento, arquitetura e execução seguem a metodologia de desenvolvimento de projetos de engenharia de software chamada Rational Unified Process (RUP*).* O Processo Unificado da Rational foi desenvolvido para apoiar o desenvolvimento de sistemas orientado a objetos, o qual fornece uma forma sistemática iterativa para se obter vantagens no *design* do sistema e flexibilidade para mudanças de requerimentos no decorrer do projeto (RUP, 2013).

O RUP arquiteta o desenvolvimento de software em um conjunto de fases, nas quais são tratadas as questões de planejamento, levantamento de requisitos, análise, desenvolvimento, teste e instalação do software. Cada fase tem uma função fundamental para que o objetivo seja cumprido, distribuídos entre vários profissionais que compõem a equipe (RUP, 2013).

A metodologia é representada pelo gráfico da figura 13, na qual se observa a atividade de cada fase.

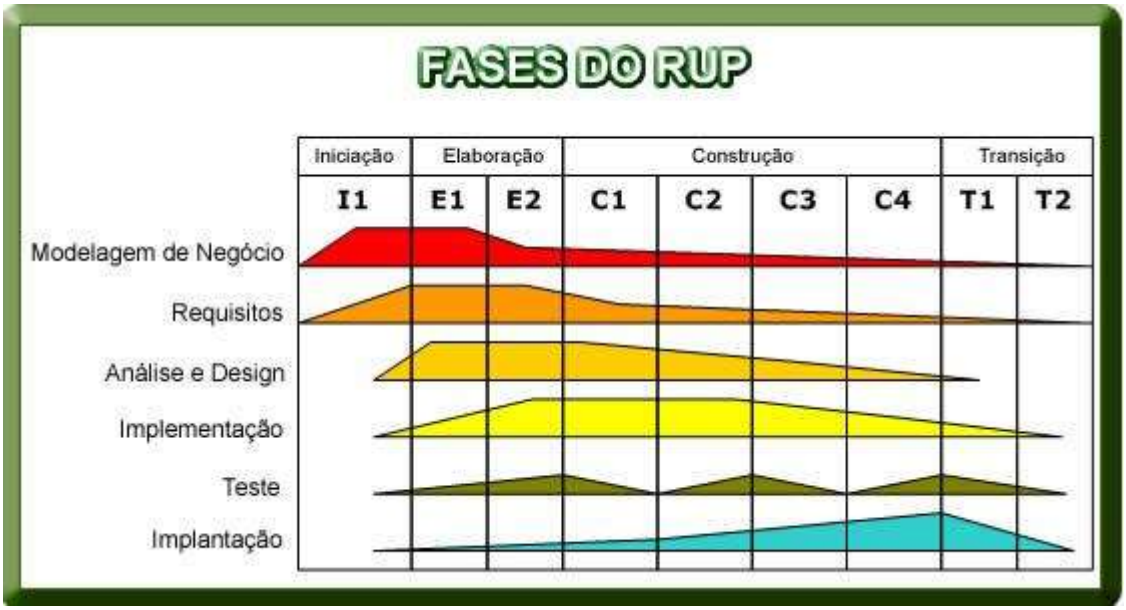


Figura 13: Fases do RUP e intensidade.

Fonte: Info Escola - http://www.infoescola.com/engenharia-de-software/rup/

4.6 ARQUITETURA DO SISTEMA

Seguindo os preceitos da engenharia de software, se faz necessário que o código siga uma arquitetura definida. A arquitetura escolhida para este desenvolvimento é assim denominada “Princípio da Responsabilidade Única”. O princípio da responsabilidade única, no paradigma de orientação a objeto, define que cada classe deve somente possuir uma única responsabilidade (MARTIN, 2002). Desta maneira a arquitetura objetiva uma aplicação que possua uma modularidade, portabilidade e que evita acoplamento de funções e responsabilidades que comprometem um longo ciclo de vida do sistema (MARTIN, 2002). A estrutura apresentada na figura 14 viabiliza uma organização que prioriza a separação dos componentes de serviço e de ferramentas.

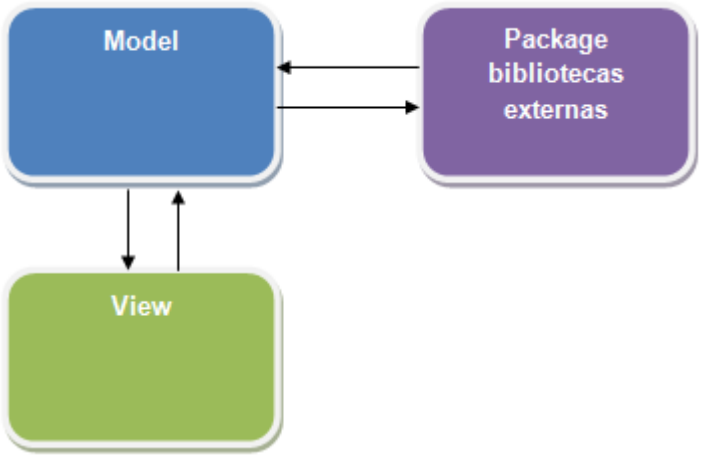


Figura 14: Arquitetura do sistema.

Fonte: Autoria Própria

Cada bloco representa um conjunto de funções que tem como objetivo modularizar e definir responsabilidades. A seguir é possível encontrar a descrição:

- **View:** Conjunto de classes que tem por escopo manipular os elementos da interface do usuário;
- **Model:** Conjunto de entidades do sistema que descrevem o comportamento lógico da aplicação, bem como possui o papel de interface das bibliotecas utilizadas neste projeto. Neste bloco há uma importante tarefa de definir as regras de funcionamento do sistema e desempenho;
- **Package bibliotecas externas:** É necessário realizar uma separação das bibliotecas externas utilizadas no projeto e o código fonte produzido. Através desta arquitetura se viabiliza a portabilidade de conexão em uma estrutura que facilita a conectividade de novos modelos, algoritmos e funcionalidades.

4.7 CONTROLE DE VERSÕES

Pelo fato da equipe contar com três integrantes desenvolvedores, há necessidade de utilizar uma ferramenta para o controle de versões do projeto. Primeiramente, para realizar testes de algumas funcionalidades foi preciso modificar significamente o código que, em certos casos, não gerou resultados satisfatórios, sendo necessário retornar à um ponto estável e funcional. Além deste motivo, o

sistema de controle de versões escolhido, o Git, possui ferramentas de resolução de conflitos eficientes que auxiliam o desenvolvimento simultâneo.
O Git é um software livre e gratuito distribuído sob a licença GNU *General Public License* versão 2 (CONSERVANCY, 2013).

Aliado a esta ferramenta, foi utilizado um serviço web de hospedagem de projetos que é organizado pelo sistema de controle de versões Git, o GitHub cujo endereço é http://github.com. Existem funcionalidades no estilo rede sociais como *feeds*, seguidores e gráficos diversos, bem como funcionalidades de projeto como visualização de pastas e códigos, gráficos de desempenho por usuário, por equipe, por período de desenvolvimento, frequência de código, histórico de modificações, entre outros (GITHUB, 2013).

Na versão gratuita, a qual foi utilizada neste trabalho, há uma exigência: que o código seja aberto (GITHUB, 2013). Na versão paga, existem planos que permitem a criação de repositórios privados com times de desenvolvimento. Para ambas as versões o número de colaboradores é ilimitado, assim como o número de repositórios públicos.

4.8 TESTES UNITÁRIOS

Com o objetivo de garantir uma aplicação bem testada, optou-se pela abordagem de escrita de testes automatizados. Esta abordagem é largamente aceita pela comunidade de desenvolvedores de software do momento em que este trabalho é desenvolvido (KOSKELA, 2013). Não obstante existe um grande número de desenvolvedores seguidores do estilo “*test-first*” no qual o desenvolvimento se inicia pela previsão de testes automatizados do código, na data da escrita do trabalho (KOSKELA, 2013). Este artifício busca a previsibilidade dos comportamentos esperados pelo sistema, assim como a validação do *design* prévio da aplicação (KOSKELA, 2013).

Baseado n esta metodologia a aplicação foi desenhada conforme apresentaa Figura 15.

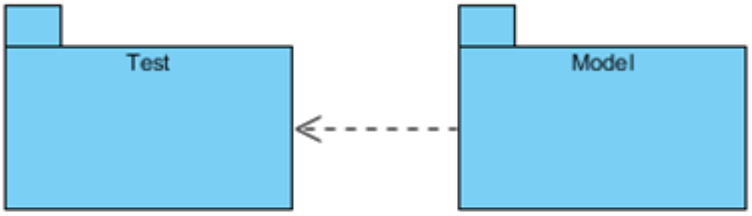


Figura 15 – Diagrama de classe do pacote de testes.

Fonte: Autoria Própria

Neste projeto se fez uso de um *framework* de propriedade da empresa Oracle chamado JUnit que possibilita um suporte para a escrita de testes unitários e criação de relatórios.

4.9 KANBAN BOARD

O *Kanban board* foi a ferramenta utilizada para o auxílio da visualização das tarefas necessárias para o desenvolvimento do projeto. Este instrumento foi e ainda é tradicionalmente utilizado pela indústria automobilística (KNIBERG, 2013).

A equipe de desenvolvimento deste projeto é formada por 3 integrantes que não possuem um local físico unificado que possibilite a visualização e manutenção de um Kanban físico. Por este motivo se optou por uma ferramenta virtual de fácil acesso que auxilie a criação, manutenção e visualização de um Kanban. Trello é uma aplicação de internet que proporciona flexibilidade na criação de um Kanban, facilidade de uso e auxílio no monitoramento de todas as tarefas de um projeto, assim como a visualização do todo (TRELLO, 2013).

Na Figura 16 é possível visualizar um exemplo de estado do Kanban proporcionada pela aplicação Trello.

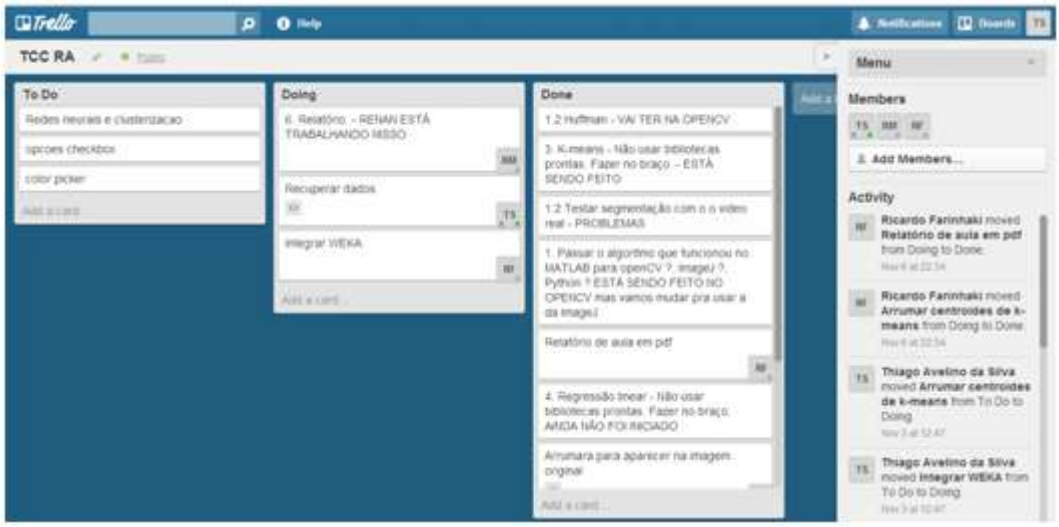


Figura 16 – Exemplo de estado do Trello.

Fonte: Autoria Própria

5. RESULTADOS

Este capítulo apresenta os resultados do desenvolvimento deste projeto.

5.1 ALGORITMO DESENVOLVIDO PARA O PROCESSAMENTO DA IMAGEM

Nesta seção do trabalho apresenta-se detalhes da elaboração e resultados do desenvolvimento do algoritmo que fará a captura do desenho do professor em um quadro branco e extração dos dados da imagem desenhada utilizando técnicas de processamento de imagem.

Primeiramente foi utlizado o software MATLAB para o desenvolvimento do protótipo do algoritmo. Uma vez testadas as técnicas propostas, foi possível a tradução para a linguagem Java.

Como cenário principal de uso, é possível descrever uma sala de aula com o professor à frente, desenhando em um quadro branco com uma webcam que captura as imagens do quadro. Conforme ilustrado na figura 17.

Figura 17 – Artefato de desenho do gráfico.

Fonte: Autoria Própria

O professor desenha um gráfico 2D contendo pontos esparsos e pretende mostrar aos alunos como um algoritmo de clusterização *k-means*, por exemplo, irá classificar estes pontos.

Então, o professor executa o algoritmo:

1) Primeiramente é capturada uma imagem com o gráfico desenhado, conforme a figura 18;

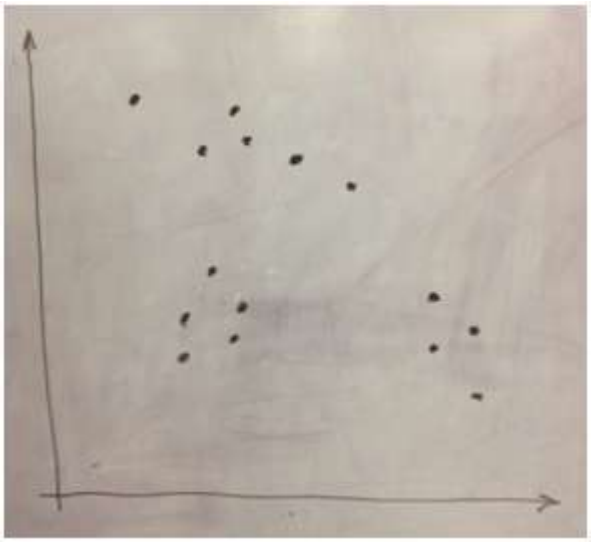


Figura 18: Foto original do quadro branco com os sistema de eixos e os dados representado por pontos.

Fonte: Autoria Própria

2) É realizada a segmentação, que é definida através da transformação da imagem em preto e branco. Desta maneira é possível distinguir o fundo dos elementos que interessam ao algoritmo;

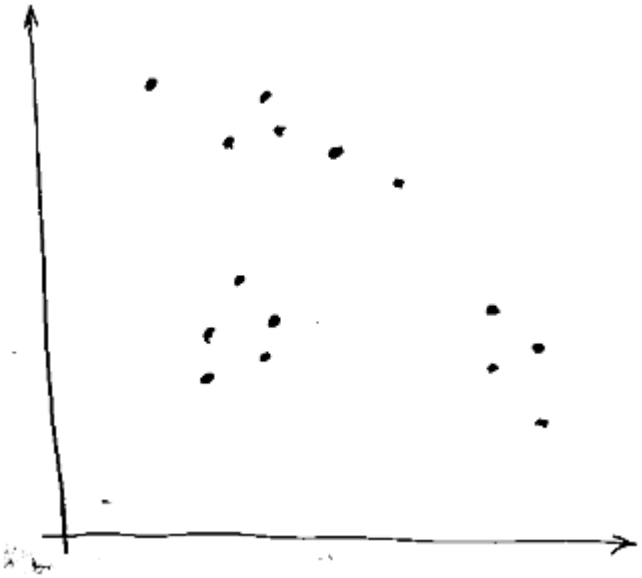


Figura 19 - Gráfico segmentado

Fonte: Autoria Própria

3- A transformação para o formato binário da imagem pode introduzir alguns ruídos que são indesejados. Estes ruídos são ocasionados por não homogeneidades

nas regiões de fundo da imagem capturada, problemas de iluminação, etc. Para eliminar estes ruídos, o sistema aplica o algoritmo de erosão e em seguida o de dilatação. Com a imagem “limpa”, os objetos sofrem uma pequena alteração de morfologia, com objetivo de deixar os pontos perfeitamente preenchidos.

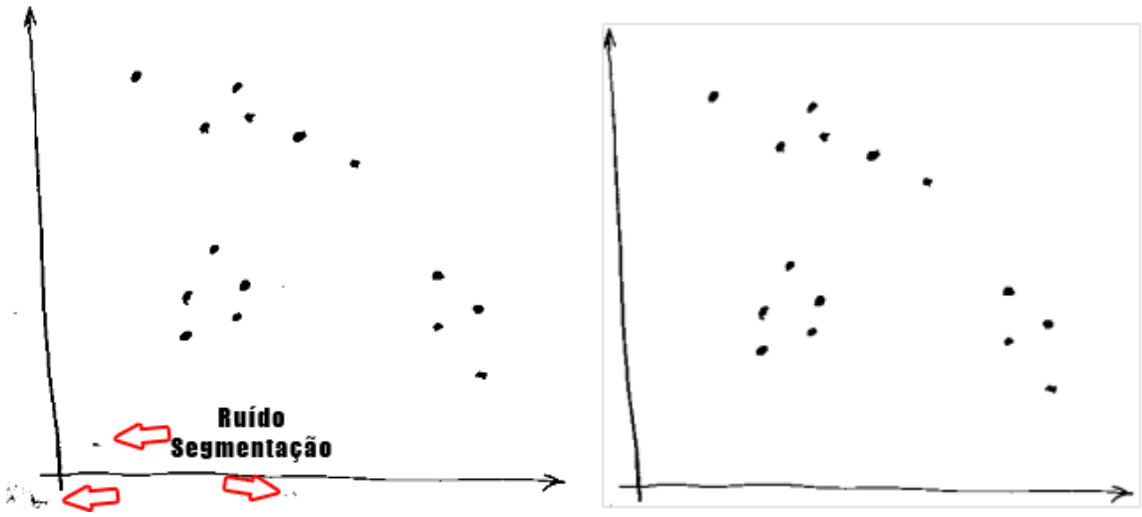


Figura 20 - Gráfico com e sem ruídos.

Fonte: Autoria Própria

4 – Através da imagem segmentada é possível identificar os pontos e eixos do gráfico. Em posse destas informações, o algoritmo é capaz de realizar o procedimento de identificação das coordenadas dos centróides dos pontos em azul.

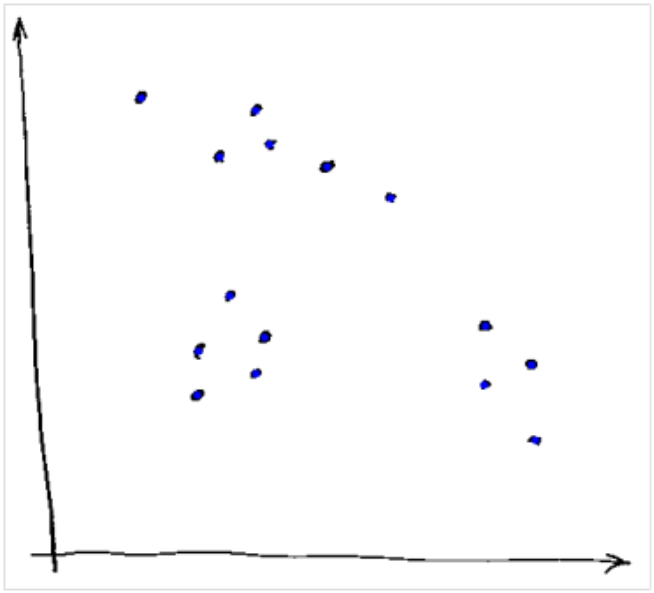


Figura 21 – Gráfico com o centroide de cada ponto em azul.

Fonte: Autoria Própria

5 – Por fim, o algoritmo da transformada de Hough é aplicado para se encontrar as linhas presentes na imagem, a fim de se encontrar os eixos x e y da imagem.

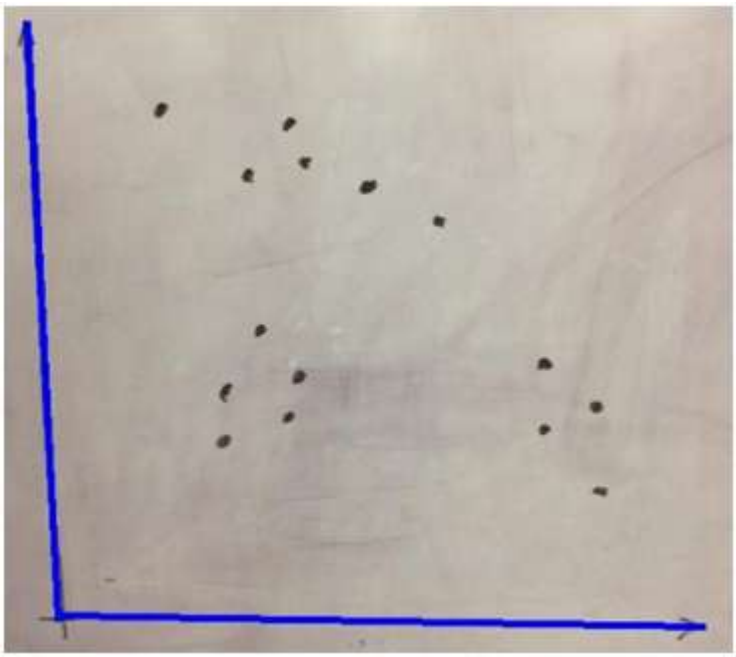


Figura 22 – Linha detectada a partir do algoritmo de Hough e demarcada em azul.
Fonte: Autoria Própria

Em posse das coordenadas dos centróides dos pontos dos gráficos e posição dos eixos é possível aplicar os algoritmos de clusterização e regressão propostos pelo projeto.

Este procedimento se demonstrou muito eficiente em diversas condições de luz e formas de desenho. As falhas encontradas na maioria dos casos são relacionadas ao limite de limiarização da imagem, assim como em desenhos que apresentam grandes diferenças em relação ao padrão esperado pelo sistema. Como artifício corretivo para estas falhas percebidas o sistema possibilita o ajuste do limite de limiarização da imagem.

5.2 ALGORITMOS DE APRENDIZAGEM DE MÁQUINA

Após o processamento da imagem se obtém os valores dos parâmetros do gráfico sendo as coordenadas dos pontos e posição dos eixos. Por convenção o aplicativo utiliza como unidade o tamanho de um pixel. Estas variáveis são processadas pelos algoritmos de aprendizagem de máquina. O resultado obtido é representado pela realidade aumentada. A figura 23 exemplifica este resultado.

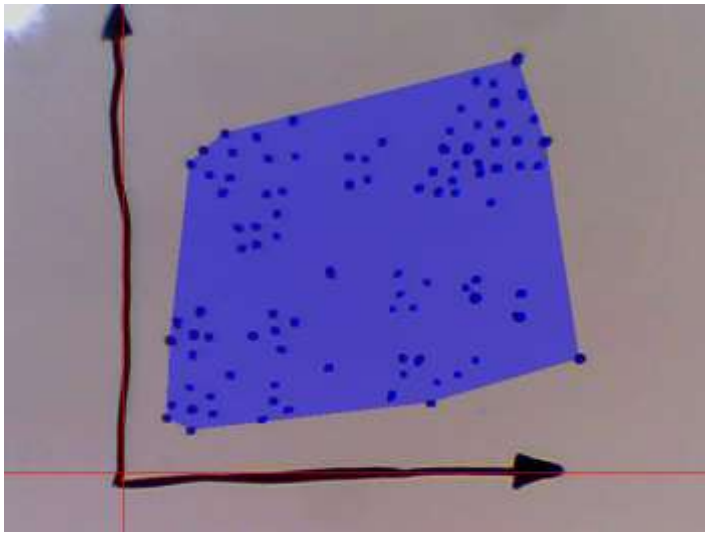


Figura 23 – Imagem com realidade aumentada aplicada.

Fonte: Autoria Própria

5.3 SOFTWARE DESENVOLVIDO

A aplicação foi denominada pela equipe como aplicativo Seta Realidade Aumentada. O desenho de suas telas visa o fácil acesso das funções principais da aplicação, bem como a apresentação de conteúdo informativo para o usuário. Na figura 24 é possível visualizar a tela que é carregada no início do software.

Figura 24 - Imagem de entrada da aplicação.

Fonte: Autoria Própria

Na figura 25 é possível visualizar a tela principal da aplicação. Nesta tela se encontra o painel do vídeo com a realidade aumentada aplicada bem como as funcionalidades da aplicação.

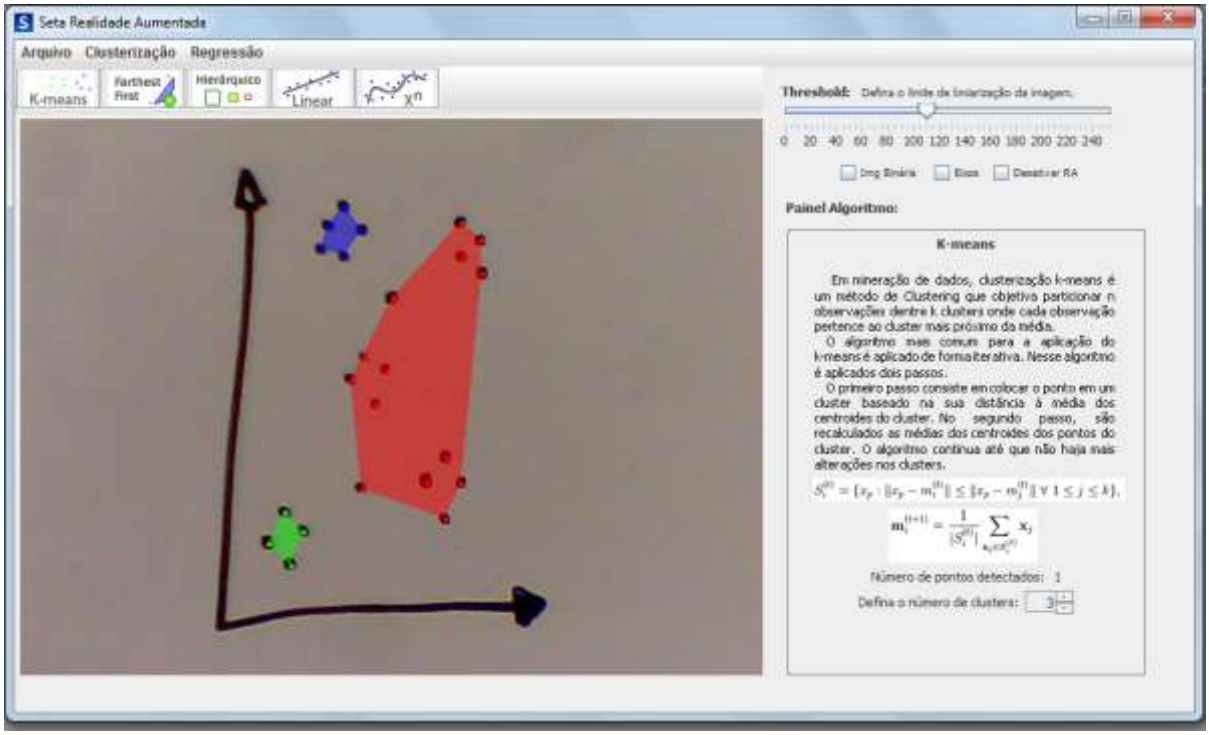


Figura 25 – Imagem da tela principal da aplicação SETA.

Fonte: Autoria Própria

Na figura 26 é possível verificar com mais detalhes a barra de ferramentas principal na qual se pode escolher os algoritmos que se deseja aplicar na imagem. A barra foi desenvolvida em duas partes, a primeira em formato texto e a segunda com botões com ícones desenhados.

Figura 26 – Imagem da barra de ferramentas da aplicação SETA.

Fonte: Autoria Própria

Na figura 27 é apresentada a barra de regulagem do limite de limiarização da imagem que está sendo processada. Desta forma o usuário pode ajustar entre os valores de 0 à 255.

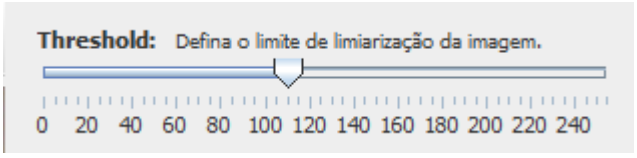


Figura 27 – Imagem da barra de ferramentas da aplicação SETA.

Fonte: Autoria Própria

Na figura 28 é possível verificar o painel de controle do algoritmo selecionado. O objetivo deste painel é de informar o usuário sobre o algoritmo selecionado e também habilitar o ajuste dos possíveis parâmetros.

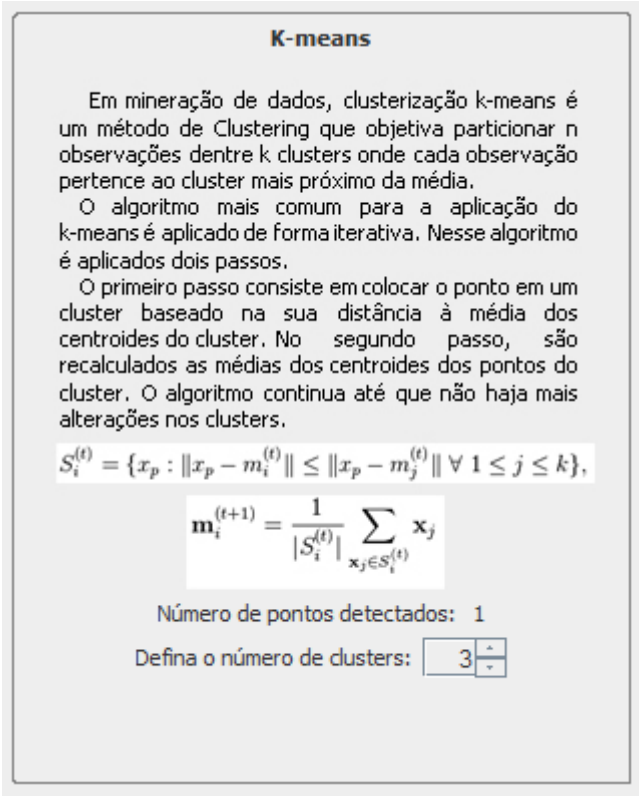


Figura 28 – Imagem do Painel do Algoritmo.

Fonte: Autoria Própria

Na Figura 29 é possível visualizar as opções que podem ser aplicadas ao aplicativo e imagens. No estilo Liga/Desliga se pode definir as preferências dos usuários.

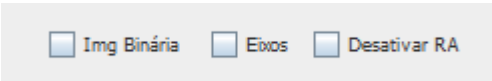


Figura 29 – Imagem da barra de ferramentas da aplicação.

Fonte: Autoria Própria

Na figura 30, pode-se ver um exemplo da aplicação do algoritmo de regressão linear. Nessa imagem, o resultado é mostrado aplicado na imagem capturada da webcam.

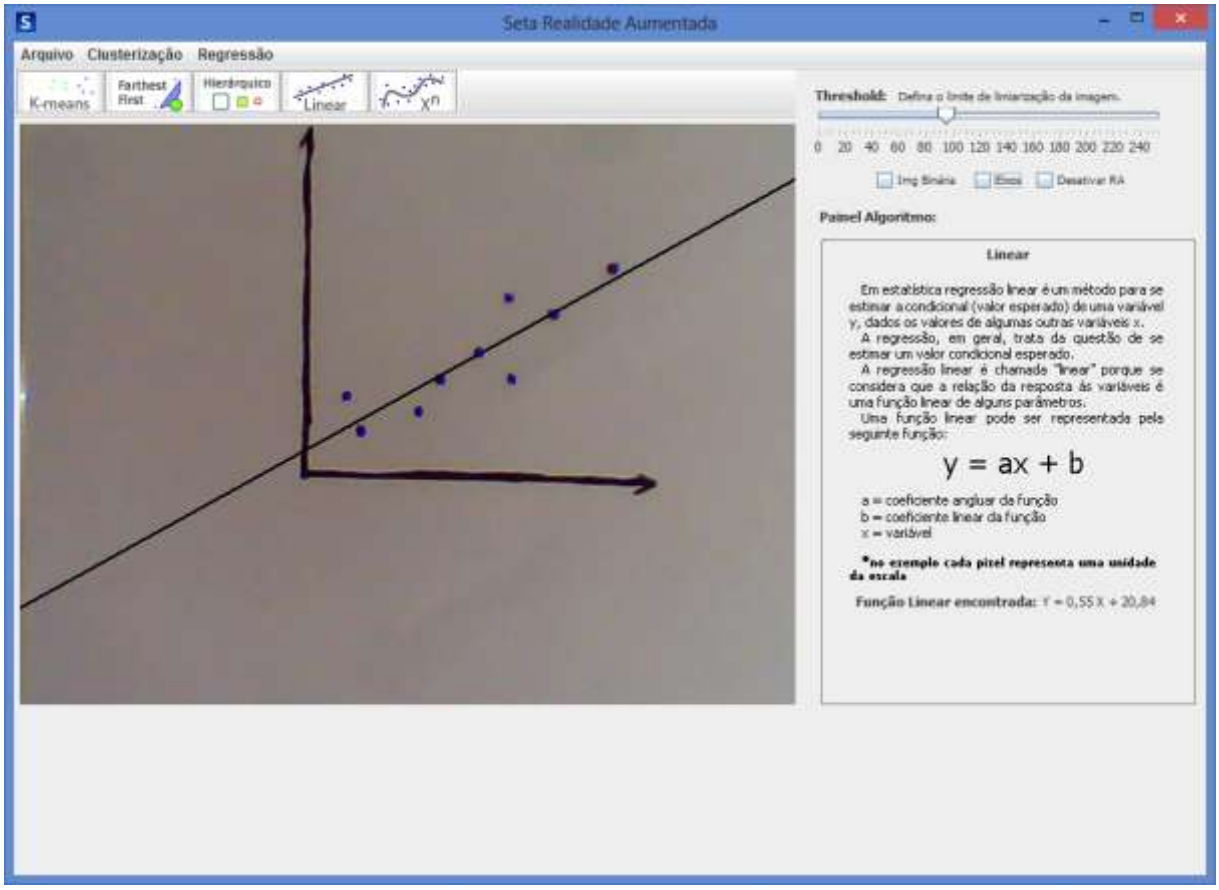


Figura 30 - Aplicação do algoritmo de regressão linear.

Fonte: Autoria própria

A figura 31 apresenta a aplicação do algoritmo de regressão linear aos mesmos pontos da figura 30, mas com a imagem mostrada em formato binário, após a execução da segmentação e, também, com a detecção dos eixos do gráfico.

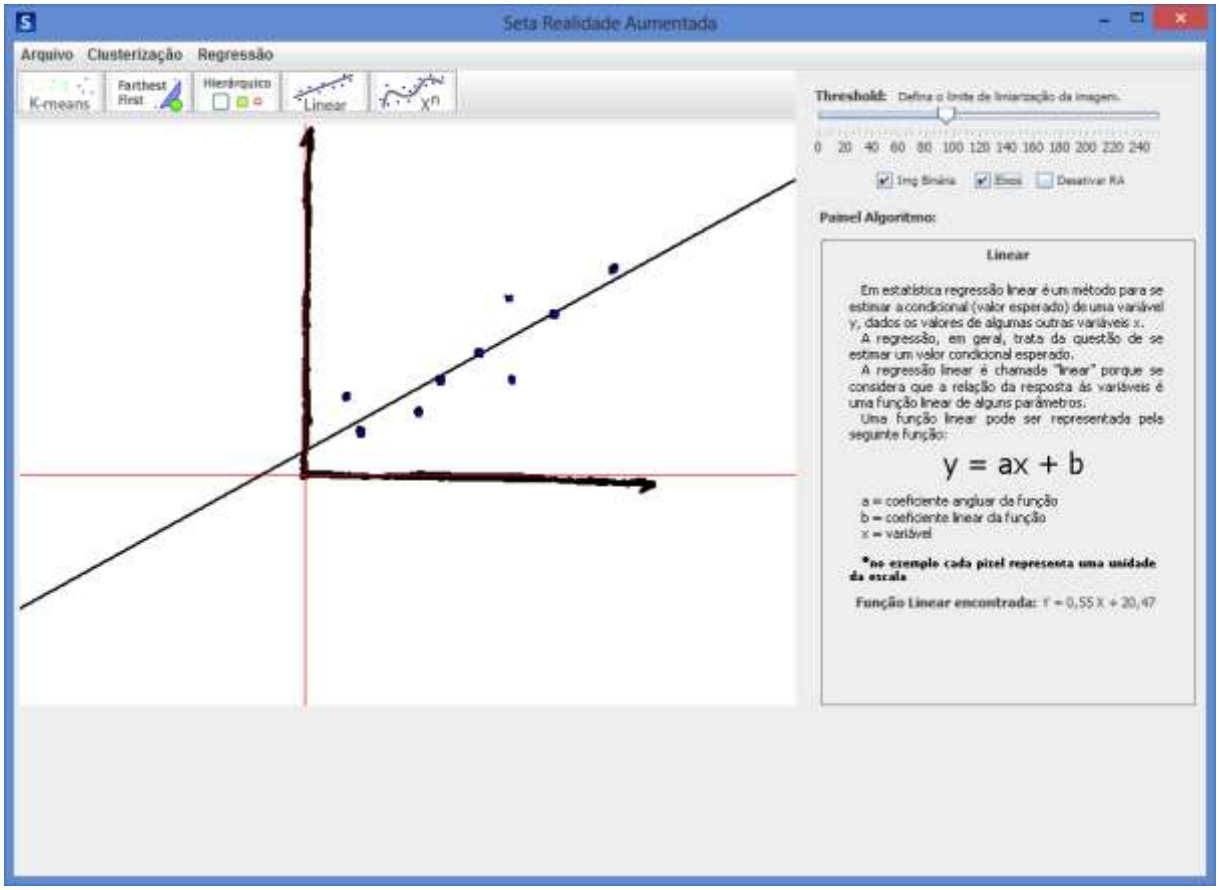


Figura 31 - Aplicação do algoritmo de regressão linear, imagem binária.

Fonte: Autoria própria

6. CONCLUSÕES

O ambiente de RA apresentado neste trabalho tem o objetivo principal de auxiliar os professores no ensino de algoritmos de aprendizagem de máquina, especificamente de clusterização e regressão. Embora a proposta concentre-se no ensino destes algoritmos em particular, é importante mencionar que a arquitetura do software possibilita a fácil integração de outros algoritmos de aprendizagem de máquina, inclusive daqueles implementados em bibliotecas de terceiros, como é o caso dos disponíveis na biblioteca Weka (WEKA, 2013). Além disso, a capacidade do sistema de extrair pontos e retas das imagens capturadas pode ser explorada em outras áreas, como por exemplo álgebra e geometria, apenas para citar alguns.

O algoritmo de segmentação de imagem desenvolvido atendeu às especificações do projeto em termos de robustez e desempenho, isto é, o sistema é capaz de detectar os elementos desejados na imagem em condições de iluminação consistentes com aquelas observadas nas situações reais, demandando um tempo de processamento compatível com o necessário para a apresentação dos resultados em tempo real. Vale ressaltar que em situações em que a iluminação varia de forma não prevista, o processo de segmentação é afetado, o que dificulta a detecção dos elementos chave na imagem.

Com relação ao desenvolvimento, verificou-se uma grande diferença entre as datas de inicio e fim do projeto. Isto se deve ao fato do calendário acadêmico ter sido defasado em função da greve, às dificuldades técnicas encontradas, como aquelas referentes à interface com a webcam e as bibliotecas utilizadas (especialmente Weka e ImageJ), e aos refinamentos nas especificações ocorridas ao longo do projeto. Pode-se destacar também que o período de desenvolvimento do projeto foi o dobro do planejado inicialmente. Possivelmente, isso se decorreu devido ao tempo demandado ao estudo de cada uma das bibliotecas utilizadas, sendo essas WEKA, OpenCV e ImageJ.

Deve-se considerar ainda a importância de alguns fundamentos e conceitos aprendidos em disciplinas ao longo do curso de Engenharia de Computação. Por exemplo, os algoritmos de processamento de imagens, os algoritmos de regressão linear e clusterização respectivamente das disciplinas de Processamento Digital de Imagens, Cálculo Numérico e Sistemas Inteligentes.

Quanto aos trabalhos futuros, sugerem-se testes utilizando outros tipos de imagens de entrada, como as capturadas de um quadro de giz ou de um quadro branco convencionais, situados na parede e a implementação do sistema em dispositivos móveis.

REFERÊNCIAS BIBLIOGRÁFICAS

AYODELE, T. O., **Types of Machine Learning Algorithms,** University of Portsmouth, United Kingdom. Disponível em http://cdn.intechweb.org/pdfs/10694.pdf Acessado em 19/06/2012.

AZUMA R. **A Survey of Augmented Reality**. Teleoperators and Virtual Environments, Hughes Research Laboratories, 1997.

BURGE, W. BURGE, M. **Digital Image Processing: An Algorithmic Introduction using Java**. USA. 2006.

CARNEIRO, T. C. T. V. **Segmentação**. 2013. Disponível em: <http://www2.ic.uff.br/~aconci/curso/binari~3.htm>. Acesso em: 15 de agosto de 2013.

CONSERVANCY, S. F. **Distributed Version Control System**. 2013. Disponível em: <http://git-scm.com/about>. Acesso em: 28 de agosto de 2013.

COOK, J. D. **Three Algorithms for converting color to grayscale**. Disponível em: <http://www.johndcook.com/blog/2009/08/24/algorithms-convert-color-grayscale/>. Acesso em: 21 de julho de 2013.

FREEMAN, Eric; FREEMAN, Elisabeth. **Head First Design Patterns**. O'REILLY. Novembro 2004.

GITHUB, I. **GitHub Social Coding**. 2013. Disponível em: <https://github.com/about>. Acesso em: 15 de agosto de 2013.

JUGARU, G. **5 Top Augmented Reality Apps For Education**. Disponível em: <http://www.hongkiat.com/blog/augmented-reality-apps-for-education/>. Acesso em: 22 de setembro de 2012.

KAUFMAN, L; ROUSSEEUW, R. J. **Finding Groups in Data: An Introduction to Cluster Analysis.** pp. 87 - 125. EUA, 2005.

KNIBERG, Henrik. **Kanban and Scrum - making the most of both**. Disponível em: <http://www.infoq.com/minibooks/kanban-scrum-minibook> Acesso em: 15 de outubro de 2013.

KOSKELA, Lasse. **Effective Unit Testing – A guide for Java developers.** Manning, 2013.

MARTIN, Robert C. **Agile Software Development, Principles, Patterns, and Practices**. Prentice Hall. Outubro de 2002.

MILGRAM P, KISHINO F. **A Taxonomy of Mixed Reality Virtual Displays**. IEICE Transactions on Information and Systems E77-D, 9 (September 1994), pp. 1321-1329.

MITCHEL, T. M., **The Discipline of Machine Learning,** School of Computer Science Carnegie Mellon University. 2006. Disponível em <http://www.cs.cmu.edu/~tom/pubs /MachineLearning.pdf>. Acessado em 19/06/2012.

OMG Object Management Group. **What is UML?** Disponível em: < http://www.omg.org/gettingstarted/what_is_uml.htm >. Acesso em: 19 de agosto de 2013.

RUP. **IBM, Rational Unified Process**. 2013. Disponível em <http://www.ibm.com/developerworks/rational/library/1826.html#N100E4> Acesso em: 20 de outubro de 2013.

SEBER, George A. F.; LEE, Alan J. **Linear Regression Analysis (Wiley Series in Probability and Statistics).** Second Edition, pp.85 - 158. New Jersey, EUA, 2003.

SILVA A. W.; RIBEIRO S. W. M.; JÚNIOR L. E.; CARDOSO A. **Uma Arquitetura Para Distribuição de Ambientes Virtuais de Realidade Aumentada Aplicada à Educação**, Revista Brasileira de Informática na Educação, v. 16, n. 3, 2008.

TRELLO. **Online Kanban Board Tour**.2013. Disponível em:< https://trello.com/tour> Acesso em: 17 de novembro de 2013.

WEKA. Weka 3: Data Mining Software in Java. 2013. Disponível em: <http://www.cs.waikato.ac.nz/ml/weka/>. Acesso em: 19 de agosto de 2013.

WITTEN, I. H; FRANK, E. Data Mining: **Practical Machine Learning Tools and Techniques.** University of Waikato. pp 150 - 235. Hamilton, Waikato, Nova Zelândia, 2005.

ZHANG, D. **APPLYING MACHINE LEARNING ALGORITHMS IN SOFTWARE DEVELOPMENT,** Department of Computer Science California State University. Disponível em http://www.disi.unige.it/person/ReggioG/PROCEEDINGS/zhang.pdf Acessado em 19/06/2012.

ANEXOS

ANEXO A - Diagrama de Classes

A Figura 32 apresenta o diagrama de classes do projeto. O texto a seguir descreve detalhadamente cada classe do sistema.

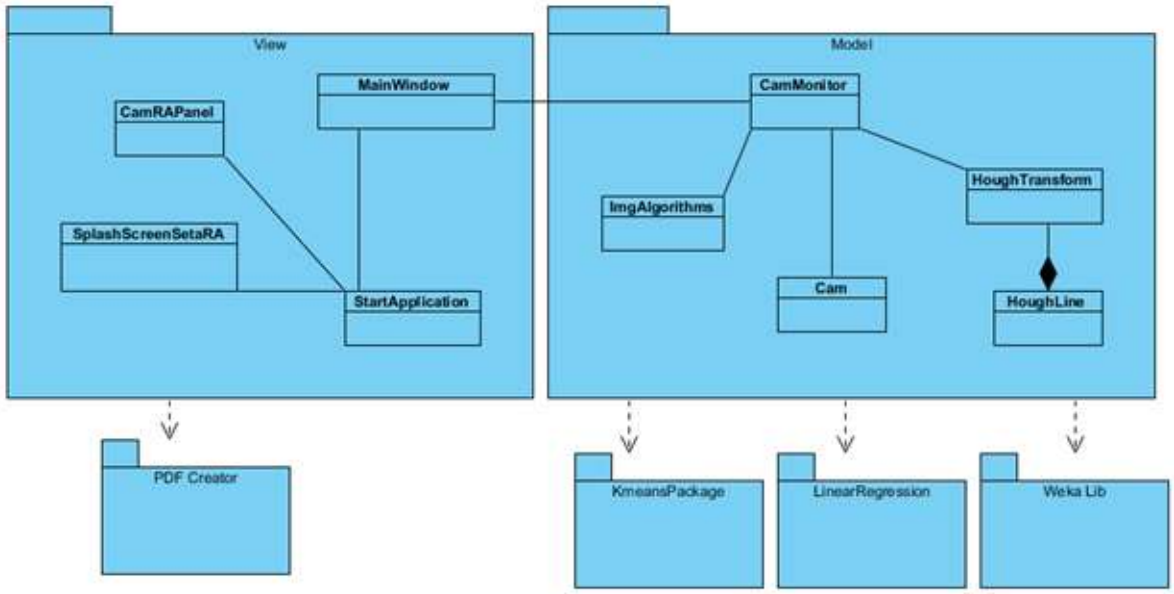


Figura 32 – Diagrama de classe do sistema.

Classe Nome: StartApplication

Descrição: Classe de inicialização do sistema. Ela é responsável por iniciar os objetos necessários para criar o ambiente visual do sistema.

Método	**Descrição**
Public void main(String[] args)	Método de início da aplicação.

Classe Nome: MainWindow

Descrição: Classe responsável por desenhar a tela principal do programa.

Atributos	**Descrição**

private JFrame frmAplicaoDeRealidade;	*Frame* da aplicação.
public final String IMG_FORMAT = ".png";	Constante formato das imagens salvas pela aplicação.
private CamRAPanel RAPanel;	Objeto da classe do painel principal da aplicação.
private JToggleButton tglbtnAcionarAlgoritmo;	Botão responsável por acionar/desativar algoritmos selecionados.
private JSlider sliderThreshold;	*Slide* para o threshold de limiarização.
private boolean algoritmoLigado;	*Boolean* do estado ligado/desligado algoritmo selecionado.
private JCheckBox chckbxImgBinria;	*Checkbox* desativar/ativar imagem binária.
private JCheckBox chckbxDesativarRa;	*Checkbox* desativar/ativar realidade aumentada.
private JCheckBox chckbxEixos;	*Checkbox* desativar/ativar eixos.
private boolean axes;	*Boolean* do estado mostrar/esconder eixo.
private JToolBar toolBar;	*ToolBar* principal da aplicação.
private JButton buttonKmeans;	Botão para acionamento do algoritmo Kmeans.

private JButton buttonFarthestFirst;	Botão para acionamento do algoritmo FarthestFirst.
private JButton buttonHierarchical;	Botão para acionamento do algoritmo Hierárquico.
private JButton buttonLinear;	Botão para o acionamento do algoritmo de regressão linear.
private JButton buttonPolinomial;	Botão para acionamento do algoritmo de regressão polinomial.
private JPanel panelConfig;	Painel de configuração dos algoritmos
private String algorithmSelected;	*String* com o nome do algoritmo selecionado.
private JSpinner spinnerKmeans;	*Spinner* utilizado para parametrização do algoritmo Kmeans.
private JSpinner spinnerFarthestFirst;	Spinner utilizado para parametrização do algoritmo FarthestFirst.
private JSpinner spinnerHierarchical;	*Spinner* utilizado para parametrização do algoritmo Hierárquico.
private JSpinner spinnerPolinomio;	*Spinner* utilizado para parametrização do algoritmo de regressão polinomial.
private JLabel functionLinear;	*Label* que mostra a função encontrada pelo algoritmo de regressão linear.
private JLabel functionPolinomioValue;	*Label* que mostra a função encontrada

	pelo algoritmo de regressão polinomial.
private JComboBox linkTypesComboBox;	*Combobox* com opções utilizadas pelo algoritmo hierárquico.

Método	**Descrição**
public MainWindow()	Método construtor da classe.
private void initialize()	Método responsável pela montagem e criação dos elementos da tela principal.
public void generatePainelInitial()	Método de criação do painel.
public void createToolBar()	Método de criação da barra de algoritmos.
public ActionListener closeFrame()	Método responsável pelo fechamento da tela principal da aplicação.
public void camConfig()	Método responsável pela configuração e interface com a *webcam*.
public ActionListener actionListSalvarImg()	Método responsável por definir a ação do evento de clique na opção salvar imagem.
public ActionListener buttonKmeansAction()	Método responsável por definir a ação do evento de clique na opção algoritmo kmeans.
public ActionListener buttonFarthestFirstAction()	Método responsável por definir a ação do evento de clique na opção algoritmo farthest first.

public ActionListener buttonHierarchicalAction()	Método responsável por definir a ação do evento de clique na opção algoritmo hierárquico.
public ActionListener buttonLinearAction()	Método responsável por definir a ação do evento de clique na opção algoritmo regressão linear.
public ActionListener buttonPolinomialAction()	Método responsável por definir a ação do evento de clique na opção algoritmo regressão polinomial.
public void setFileChooser(JFileChooser arquivo)	Método de acesso atributo.
public JFrame getFrmAplicaoDeRealidade()	Método de acesso atributo.
private void setFrmAplicaoDeRealidade(JFrame frmAplicaoDeRealidade)	Método de acesso atributo.
public void run()	Método de execução da janela principal.
public CamRAPanel getRAPanel()	Método de acesso atributo.
private void setRAPanel(CamRAPanel rAPanel)	Método de acesso atributo.
public JSlider getSliderThreshold()	Método de acesso atributo.
public void setSliderThreshold(JSlider	Método de acesso atributo.

sliderThreshold)	
public JToggleButton getTglbtnAcionarAlgoritmo()	Método de acesso atributo.
public void setTglbtnAcionarAlgoritmo(JToggleButton tglbtnAcionarAlgoritmo)	Método de acesso atributo.
public boolean isAlgoritmoLigado()	Método de acesso atributo.
public void setAlgoritmoLigado(boolean algoritmoLigado)	Método de acesso atributo.
public String getAlgorithmSelected()	Método de acesso atributo.
public void setAlgorithmSelected(String algorithmSelected)	Método de acesso atributo.
public JSpinner getSpinnerKmeans()	Método de acesso atributo.
public void setSpinnerKmeans(JSpinner spinnerKmeans)	Método de acesso atributo.
public JLabel getFunctionLinear()	Método de acesso atributo.
public void setFunctionLinear(JLabel functionLinear)	Método de acesso atributo.

public JLabel getFunctionPolinomioValue()	Método de acesso atributo.
public void setFunctionPolinomioValue(JLabel functionPolinomioValue)	Método de acesso atributo.
public JSpinner getSpinnerPolinomio()	Método de acesso atributo.
public void setSpinnerPolinomio(JSpinner spinnerPolinomio)	Método de acesso atributo.
public JSpinner getSpinnerFarthestFirst()	Método de acesso atributo.
public void setSpinnerFarthestFirst(JSpinner spinnerFarthestFirst)	Método de acesso atributo.
public JSpinner getSpinnerHierarchical()	Método de acesso atributo.
public void setSpinnerHierarchical(JSpinner spinnerHierarchical)	Método de acesso atributo.
public JComboBox getLinkTypesComboBox()	Método de acesso atributo.
public void setLinkTypesComboBox(JComboBox linkTypesComboBox)	Método de acesso atributo.

public JCheckBox getChckbxImgBinria()	Método de acesso atributo.
public void setChckbxImgBinria(JCheck Box chckbxImgBinria)	Método de acesso atributo.
public JCheckBox getChckbxDesativarRa()	Método de acesso atributo.
public void setChckbxDesativarRa(JChe ckBox chckbxDesativarRa)	Método de acesso atributo.
public JCheckBox getChckbxEixos()	Método de acesso atributo.
public void setChckbxEixos(JCheckBox chckbxEixos)	Método de acesso atributo.

Classe Nome: SplashScreen

Descrição: Classe responsável por mostrar tela de aguardo de carregamento.

Atributos: Nenhum.

Método	**Descrição**
static void renderSplashFrame(Graphics2D g)	Método responsável pelo desenho da tela de espera.
public SplashScreenSetaRA()	Método construtor da classe.
public void actionPerformed(ActionEvent ae)	Método de fechamento da tela quando

	finalizado o processo de carregamento.
private static WindowListener closeWindow = new WindowAdapter()	Método de fechamento da tela quando finalizado o processo de carregamento.

Classe Nome: CamRAPanel

Descrição: Classe responsável pelo desenho da tela na qual exibe as imagens da câmera e inserção de elementos virtuais.

Atributo	**Descrição**
private String algorithm	String responsável por guardar o valor do algoritmo selecionado.
private boolean axesOn	Boolean responsável por verificar se está ativado/desativado a inserção dos eixos na imagem.
private BufferedImage master	Imagem que será mostrada na tela.
private ArrayList<Pixel> centroids	Lista de pixels dos centroides dos pontos da imagem.
private ArrayList<Color> colors	Lista de cores.
private ArrayList<Data> dataClusterAlgorithm	Lista com dados dos clusters.
private ArrayList<Data> dataPolinomial	Lista com o resultados do processo de regressão

	polinomial.
private int axeX	Posição do eixo x.
private int axeY	Posição do eixo y.
private Pixel linearRegressionInit	Ponto inicial da regressão linear.
private Pixel linearRegressionFinal	Ponto final da regressão linear.

Método	**Descrição**
public CamRAPanel(BufferedImage master)	Método construtor da classe.
public Dimension getPreferredSize()	Método de definição da dimensão do painel.
protected void paintComponent(Graphics g)	Método responsável pela pintura dos elementos virtuais e imagem da câmera no painel.
public void paintAxes(Graphics g)	Método responsável por desenhar os eixos.
public void paintClusters(Graphics g)	Método responsável por desenhar os clusters.
public void paintDots(Graphics g)	Método responsável por desenhar os pontos identificados na imagem.

public BufferedImage getMaster()	Método de acesso atributo.
public void setMaster(BufferedImage master)	Método de acesso atributo.
public ArrayList<Pixel> getCentroids()	Método de acesso atributo.
public void setCentroids(ArrayList<Pixel> centroids)	Método de acesso atributo.
public String getAlgorithm()	Método de acesso atributo.
public void setAlgorithm(String algorithm)	Método de acesso atributo.
public void CreateArrayColors()	Método de criação da lista de cores.
public ArrayList<Data> getDataClusterAlgorithm()	Método de acesso atributo.
public void setDataClusterAlgorithm(ArrayList< Data> dataKmeans)	Método de acesso atributo.
public int getAxeX()	Método de acesso atributo.
public void setAxeX(int axeX)	Método de acesso atributo.
public int getAxeY()	Método de acesso atributo.
public void setAxeY(int axeY)	Método de acesso atributo.
public Pixel	Método de acesso atributo.

getLinearRegressionInit()	
public void setLinearRegressionInit(Pixel linearRegressionInit)	Método de acesso atributo.
public Pixel getLinearRegressionFinal()	Método de acesso atributo.
public void setLinearRegressionFinal(Pixel linearRegressionFinal)	Método de acesso atributo.
public ArrayList<Data> getDataPolinomial()	Método de acesso atributo.
public void setDataPolinomial(ArrayList<Data> dataPolinomial)	Método de acesso atributo.
public boolean isAxesOn()	Método de acesso atributo.
public void setAxesOn(boolean axesOn)	Método de acesso atributo.

Classe Nome: CamMonitor

Descrição: Classe responsável pelo controle dos algoritmos selecionados e processamento da imagem.

Atributo	**Descrição**
public static final int GETIMAGETIMEMILI = 1000	Constante de tempo de requisição de imagem para a câmera.
public static final int	Constante de número de ciclos para

NUMBERCYCLESGETINFORMATION = 5	o processamento de imagem.
private MainWindow mainWindow	Referência da tela principal.
private ArrayList<Pixel> centroidsTemp	Lista com os centroides temporários.
private BufferedImage imageCamera	Image carregada da câmera.
private Vector<HoughLine> linesTemp	Lista com as linhas processadas e carregadas temporariamente.
private int linearXTemp	Variável da posição do eixo X.
private int linearYTemp	Variável da posição do eixo Y.
private int timeCallAlgorithm	Variável de contagem do tempo dos ciclos.
public static final double THRESHOLANGLE = 0.2	Constante de threshold do algoritmo de Hough
public static final double ANGLEX = Math.PI	Constante de ângulo do eixo x.
public static final double ANGLEY = Math.PI/2	Constante de ângulo do eixo y.

Método	**Descrição**
public CamMonitor(MainWindow mainWindow)	Método construtor da classe.

public void run()	Método de execução da thread.
public PolynomialRegression calculatePolinomial(CamRAPanel camRAPAnel, int degree)	Método que executa e manipula os dados da regressão polinomial.
public PolynomialRegression calculatePolinomialRealFunction(int degree)	Método que executa e manipula os dados da regressão polinomial.
public void defineAxes()	Método que executa e manipula os dados do algoritmo de Hough.
public void getInformation(ImgAlgorithms imgAlgorithms)	Método que executa o processamento da imagem e extrai os dados necessários para a execução dos algoritmos.
public Vector<HoughLine> houghTransform(ImgAlgorithms imgAlgorithms)	Método responsável pela execução do algoritmo de hough.
public void calculateCentroids(ImgAlgorithms imgAlgorithms)	Método responsável pelo calculo das áreas e centroides dos pontos da imagem.
public ImgAlgorithms imageSegmentation(BufferedImage image)	Método responsável pela segmentação da imagem.
public ArrayList<Pixel> getCentroidsTemp()	Método de acesso atributo.
public void setCentroidsTemp(ArrayList<Pixel>	Método de acesso atributo.

centroidsTemp)	

Classe Nome: ImgAlgorithms

Descrição: Classe responsável pelo processamento das imagens.

Atributo	**Descrição**
public final int WIDTH_SCREEN = 640	Constante que define a largura de uma imagem.
public final int HEIGHT_SCREEN = 480	Constante que define a altura de uma imagem.
private BufferedImage image	Imagem manipulada no processamento.
private BufferedImage output	Imagem manipulada no processamento.
private ArrayList<LabelArea> labeledAreas	Lista com as áreas identificadas.
private ArrayList<Pixel> centroids	Lista de com objetos que representam os pixels das centroides.
private int threshold	Variável de *threshold* de limiarização.
private ArrayList<Color> colors	Lista de cores.
private int numberClustersKmeans	Varíavel de número de clusters do algoritmo de Kmeans.
private ArrayList<Ponto> dataKmeans	Lista de dados processados pelo algoritmo *Kmeans*

Método:	**Descrição:**
public ImgAlgorithms(BufferedImage imageTemp)	Método construtor da classe.
public void toGrayscale()	Método que transforma a imagem colorida em tons de cinza.
public void toBinary()	Método que transforma a imagem em formato binário.
public void calculateArrayCentroides()	Método que calcula a lista de centroides.
public void removeBiggerArea()	Método utilizado no processamento de imagem que eliminar o desenho do eixo como ponto.
public void calculateKmeans()	Método que calcula o Kmeans.
public void labeling()	Método que define labels para áreas encontradas na imagem binária.
public void mergeAreas(Pixel pixel, ArrayList<Integer> indexAreas)	Método utilizado pelo algoritmo de detecção de áreas.
public boolean belongsToTwoAreas(ArrayList<Integer> indexAreas)	Método utilizado pelo algoritmo de detecção de áreas.
public boolean belongsToOneArea(ArrayList<Integer>	Método utilizado pelo algoritmo de detecção de áreas.

indexAreas)	
public boolean doesntBelongtoAnyArea(ArrayList<Integer> indexAreas)	Método utilizado pelo algoritmo de detecção de áreas.
public boolean firstArea()	Método utilizado pelo algoritmo de detecção de áreas.
public ArrayList<Integer> indexAreasPixelBelongs(Pixel pixel)	Método utilizado pelo algoritmo de detecção de áreas.
public void createNewArea(Pixel pixel)	Método utilizado pelo algoritmo de detecção de áreas.
public boolean recognizeElement(Color pixel)	Método utilizado pelo algoritmo de detecção de áreas.
public void erosion()	Método responsável pela aplicação da técnica de erosão na imagem binária.
public void CreateArrayColors()	Método responsável pela criação de uma lista de cores aleatórias.
public BufferedImage getImage()	Método de acesso atributo.
public void setImage(BufferedImage image)	Método de acesso atributo.
public BufferedImage getOutput()	Método de acesso atributo.
public void setOutput(BufferedImage output)	Método de acesso atributo.

public int getThreshold()	Método de acesso atributo.
public void setThreshold(int threshold)	Método de acesso atributo.
public ArrayList<LabelArea> getLabeledAreas()	Método de acesso atributo.
public void setLabeledAreas(ArrayList<LabelArea> labeledAreas)	Método de acesso atributo.
public ArrayList<Pixel> getCentroids()	Método de acesso atributo.
public void setCentroids(ArrayList<Pixel> centroids)	Método de acesso atributo.
public int getNumberClustersKmeans()	Método de acesso atributo.
public void setNumberClustersKmeans(int numberClustersKmeans)	Método de acesso atributo.
public ArrayList<Ponto> getDataKmeans()	Método de acesso atributo.
public void setDataKmeans(ArrayList<Ponto> dataKmeans)	Método de acesso atributo.

Classe Nome: HoughTransform

Descrição: Classe responsável pelo desempenho do algoritmo Transformada de Hough.

Atributo:	**Descrição:**

final int neighbourhoodSize = 4	Constante utilizada para definir o número de vizinhos de um pixel.
final int maxTheta = 180	Constante utilizada para definir o ângulo máximo de theta da transformada de Hough.
final double thetaStep = Math.PI / maxTheta	Constante do passo angular de theta.
protected int width, height	Variáveis da imagem altura e largura.
protected int[][] houghArray	Vetor temporário utilizado pelo algoritmo.
protected float centerX, centerY	Variáveis utilizadas pelo algoritmo.
protected int houghHeight	Variável temporária utilizada pelo algoritmo.
protected int doubleHeight	Variável temporária utilizada pelo algoritmo.
protected int numPoints	Variável temporária utilizada pelo algoritmo.
private double[] sinCache	Variável temporária utilizada pelo algoritmo.
private double[] cosCache	Variável temporária utilizada pelo algoritmo.

Método:	**Descrição:**
public HoughTransform(int width, int height)	Método construtor da classe.
public void initialise()	Método de inicialização das variáveis temporárias

	utilizadas pelo algoritmo.
public void addPoints(BufferedImage image)	Método de desenho dos pontos.
public void addPoint(int x, int y)	Métodos de desenho dos pontos.
public Vector<HoughLine> getLines(int threshold)	Método de execução do algoritmo.
public int getHighestValue()	Método de processamento do vetor temporário retornando o maior valor.
public BufferedImage getHoughArrayImage()	Método de processamento do vetor de Hough na imagem.

Classe Nome: HoughLine

Descrição: Classe responsável pela estrutura de dado de representação de uma reta no Algoritmo de Hough.

Atributo:	**Descrição:**
private double theta	Coeficiente angular da reta
private double r	Coeficiente linear da reta

Método:	**Descrição:**

public HoughLine(double theta, double r)	Método construtor da classe.
public void draw(BufferedImage image, int color)	Método que desenha em uma imagem a linha com cor desejada.
public int getHorizontal(int width, int height)	Definição das coordenadas x dos pontos a serem desenhados.
public int getVertical(int width, int height)	Definição das coordenadas y dos pontos a serem desenhados.
public double getTheta()	Método de acesso atributo.
public void setTheta(double theta)	Método de acesso atributo.
public double getR()	Método de acesso atributo.
public void setR(double r)	Método de acesso atributo.

Pacote: PDF Creator

Descrição: Biblioteca que reúne as funções necessárias para criação de relatórios em PDF.

Pacote: Kmeans

Descrição: Biblioteca que reúne as funções necessárias do algoritmo de clusterização Kmeans.

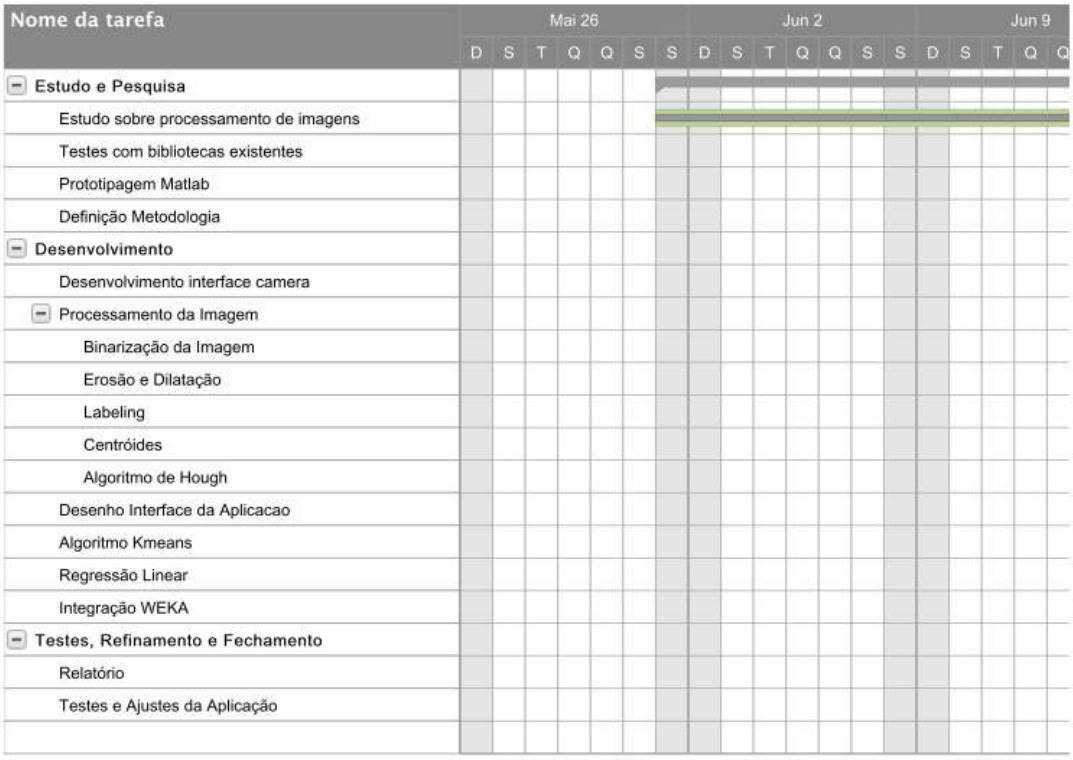
Pacote: Linear Regression

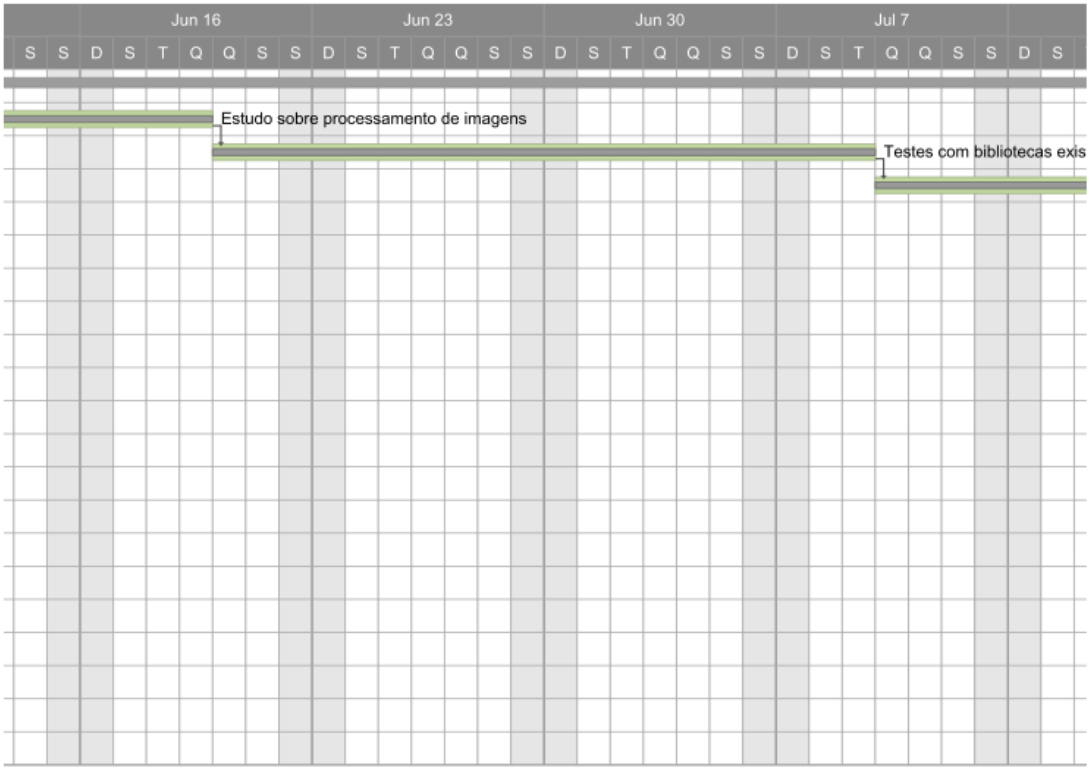
Descrição: Biblioteca que reúne as funções necessárias para execução do algoritmo de regressão linear.

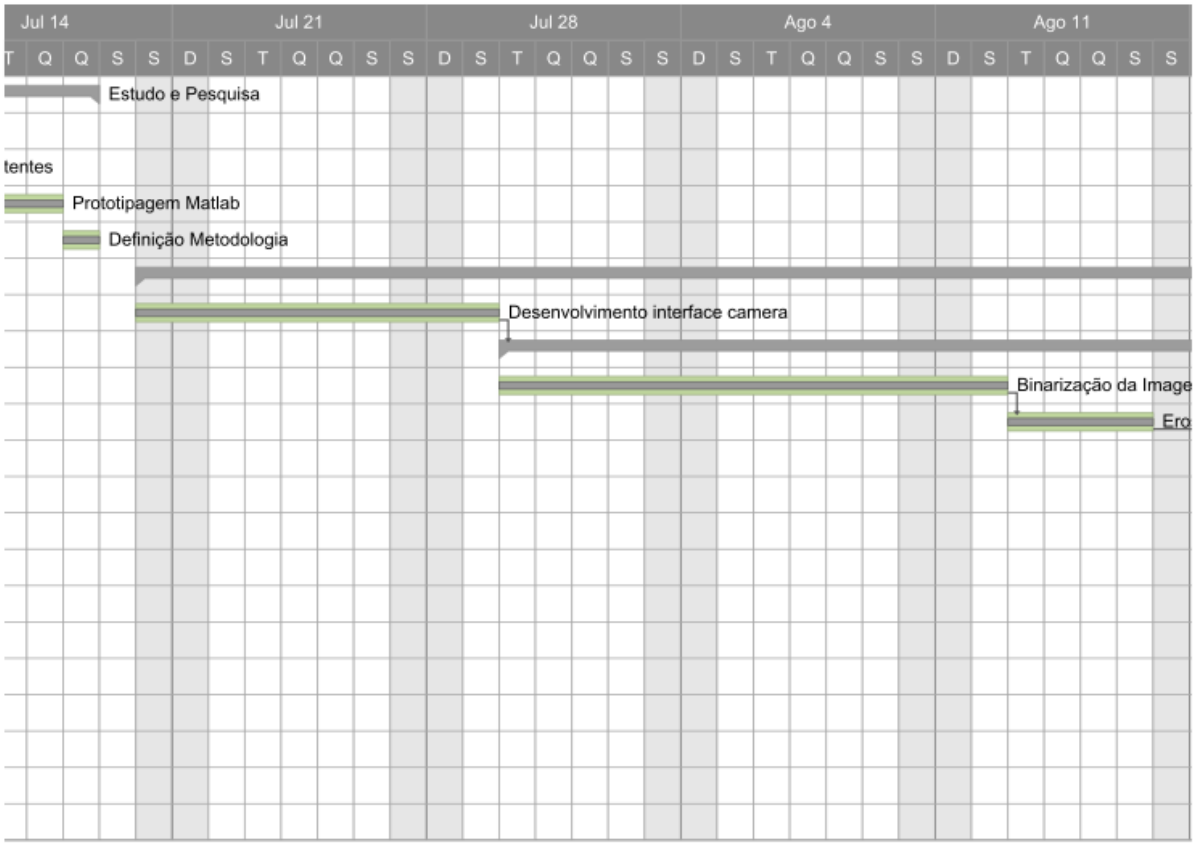
Pacote: Weka Lib

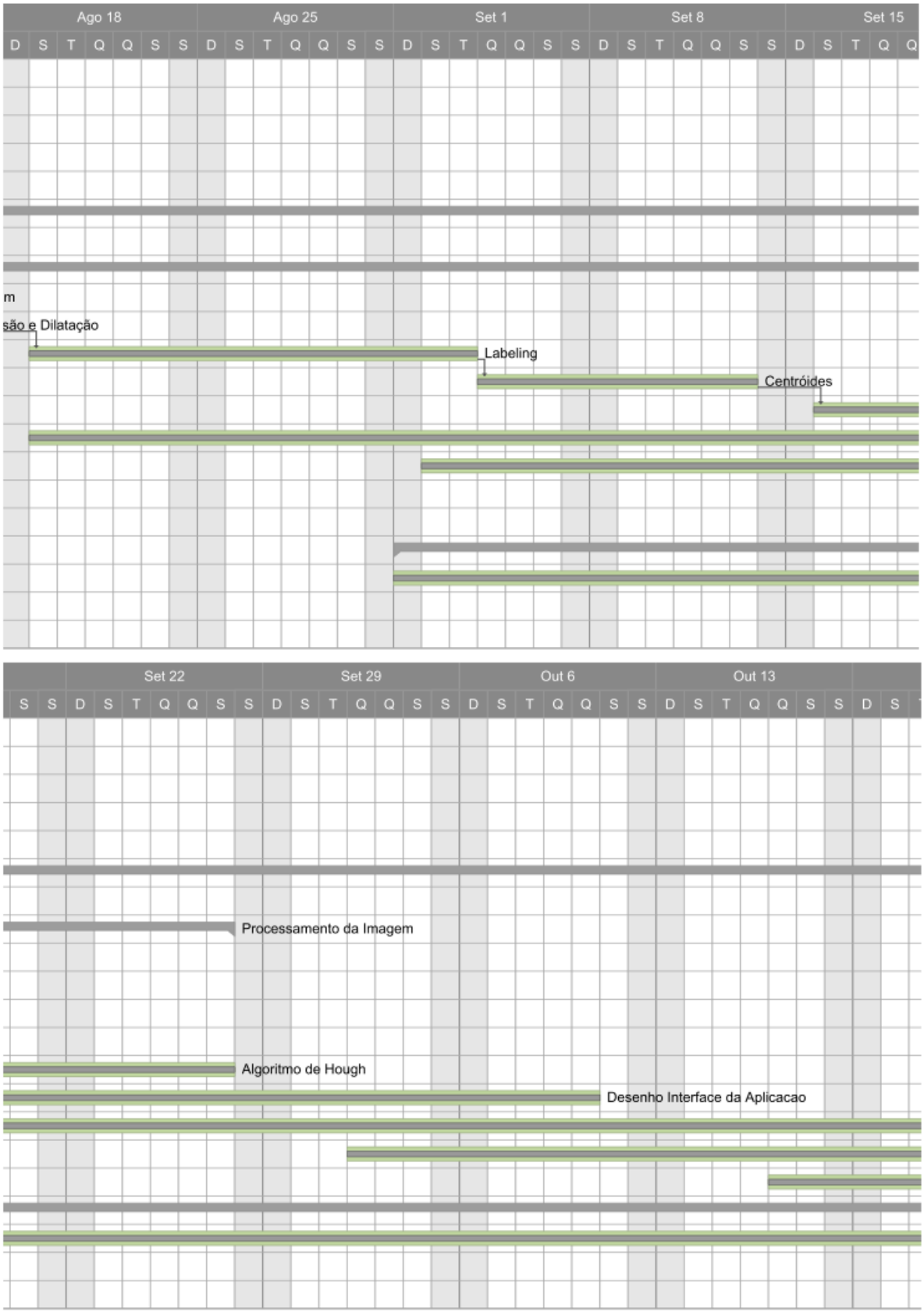
Descrição: Biblioteca que reúne as funções utilizadas pelos algoritmos de clusterização.

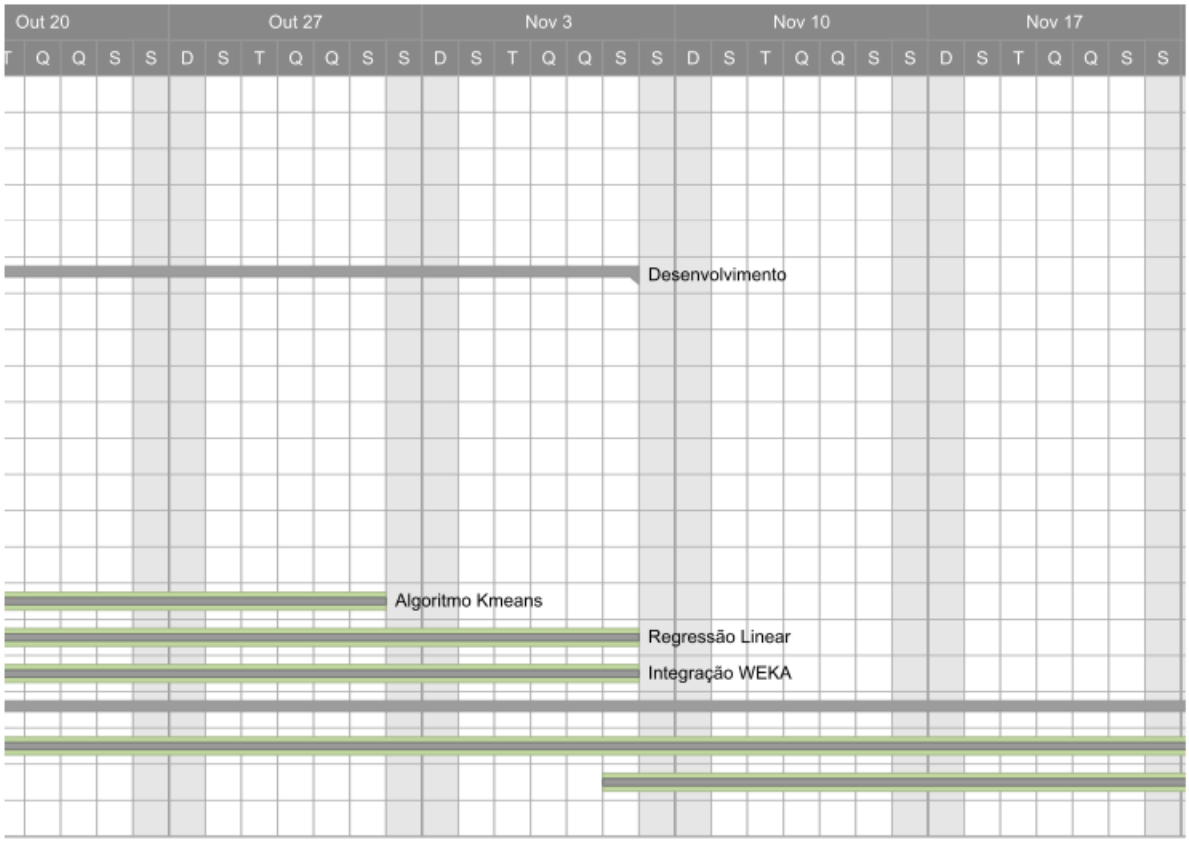
ANEXO B - DIAGRAMA DE GANTT






Jul 14
Jul 21
Jul 28
Ago 4
Ago 11
T Q Q S S D S T Q Q S S D S T Q Q S S D S T Q Q S S D S T Q Q S S
Estudo e Pesquisa
tentes
Prototipagem Matlab
Definição Metodologia
Desenvolvimento interface camera
Binarização da Image
Ero


Ago 18
Ago 25
Set 1
Set 8
Set 15
D S T Q Q S S D S T Q Q S S D S T Q Q S S D S T Q Q S S D S T Q Q
m
são e Dilatação
Labeling
Centróides
Set 22
Set 29
Out 6
Out 13
S S D S T Q Q S S D S T Q Q S S D S T Q Q S S D S T Q Q S S D S
Processamento da Imagem
Algoritmo de Hough
Desenho Interface da Aplicacao


Out 20
Out 27
Nov 3
Nov 10
Nov 17
T Q Q S S D S T Q Q S S D S T Q Q S S D S T Q Q S S D S T Q Q S S
Desenvolvimento
Algoritmo Kmeans
Regressão Linear
Integração WEKA