ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO

PMR2550 – PROJETO DE CONCLUSÃO DE CURSO II

Prof. Edson Gomes

Prof. Lucas Antonio Moscato

Inteligência Artificial Aplicada ao Reconhecimento de Imagens

Orientador: Prof. Fábio G. Cozman

Relatório Final

Guilherme Araújo Costa	5178811
Danillo Paulo Couto	5176485

São Paulo – 2008

Índice

1. Introdução

Diversas técnicas de reconhecimento de padrões e inteligência artificial são utilizadas na resolução de problemas reais com o objetivo de minimizar a interação humana em tarefas meramente repetitivas. No caso de imagens, a idéia é desenvolver sistemas que façam a interpretação das imagens de forma autônoma.

Diferentemente dos sistemas biológicos que contam com um enorme número de informações, sejam visuais ou não, que se cruzam para produzir um resultado final ótimo, os sistemas artificiais, programados com o intuito de emular o cérebro humano no reconhecimento de imagens, trabalham com um número restrito de informações, informações essas contidas na base de conhecimento do sistema programado.

Essa limitação de informações se dá pelo fato de a imagem ser uma entidade que possui uma complexidade muito alta e de difícil representação.

Projetos relacionados a reconhecimento de padrões, visão computacional e processamento de imagens englobam diversas técnicas com uma série de aplicações práticas. Uma possível aplicação dessa técnica se dá no reconhecimento de veículos.

Tal projeto seguirá essa linha, tendo por finalidade a determinação da existência ou ausência de ônibus em uma imagem capturada em via com grande fluxo de veículos.

Diante de tais fatos, o primeiro ponto chave do trabalho será a aquisição de um banco de conhecimento que represente fielmente o ambiente real no qual o software será aplicado.

O segundo desafio do projeto será a seleção de atributos nas imagens base que possam representar algo relevante para a classificação das imagens, ou seja, deve-se determinar em cada foto do banco de conhecimento, características (atributos) comuns entre imagens de mesma classe, fazendo com que essas características sejam usadas para a comparação com cada imagem testada na execução do aplicativo, sendo então possível a classificação da imagem em “Com ônibus” ou “Sem ônibus”.

2. Motivação e Objetivos

Tendo como principal motivação o emprego do software como base para o desenvolvimento de um aplicativo portátil para auxílio a deficientes visuais, o projeto faz uso de técnicas de inteligência artificial para a identificação da existência de um ônibus em uma determinada imagem capturada pelo usuário. Vale ressaltar que o projeto possui caráter exploratório, não tendo como única finalidade tal aplicação, mas o estudo e desenvolvimento de um método de Classificação de Imagens implementado em uma plataforma móvel.

Sendo assim, o projeto visa ao desenvolvimento de um software de reconhecimento de imagem, aplicado à distinção da imagem entre duas classes: “Com ônibus” e “Sem ônibus”. Devido às aplicações intencionadas para o software, é necessário que ele apresente fácil portabilidade para um dispositivo móvel, como celulares e PDAs, o que obriga a solução desenvolvida a contar com pequeno poder de processamento. Outra restrição do projeto reside nas imagens capturadas pelo dispositivo móvel. Além de possuírem baixa resolução, é necessário que o software não assuma muitas premissas sobre elas, uma vez que serão capturadas por um usuário que não terá possibilidade de garantir suas características, como foco, inclinação ou a presença da totalidade do ônibus na foto.

A solução adotada pelo software empregará o uso de Inteligência Artificial, aplicada ao aprendizado de máquina. Essa é uma técnica bastante comum em situações com poucas premissas em que é necessário classificar um conjunto de informações. Seu emprego exige uma base de conhecimento suficientemente grande, com imagens devidamente rotuladas, a partir das quais será gerada uma função classificadora de imagens.

3. Possíveis abordagens

O problema motivador do estudo – de habilitar um deficiente visual a utilizar de forma autônoma os serviços de transporte público – pode ser abordado por diferentes ângulos. Com caráter de pesquisa, o método de captura e classificação de imagens para reconhecimento de veículos consiste numa das possíveis abordagens para o problema. Há também outras alternativas, podendo apresentar algumas delas mais fácil implementação técnica.

Uma proposição viável seria a utilização de etiquetas inteligentes nos veículos do transporte público. Ao se aproximarem de um ponto de parada, sensores detectariam a etiqueta identificadora do ônibus, transmitindo essa informação ao ponto de parada. Através de um aviso sonoro, todos os passageiros seriam informados sobre o ônibus que se aproxima. Esse sistema teria simples implementação e, possivelmente, eficácia de operação maior do que a solução proposta neste projeto. Apesar disso, uma solução como essa depende de infra-estrutura instalada não apenas nos ônibus, mas também em cada um dos pontos de parada do sistema de transporte. Além do investimento para compra dos equipamentos, há necessidade de efetuar contínua manutenção, uma vez que tais equipamentos estariam sujeitos, não apenas ao desgaste natural por exposição a ambientes externos, como também ao vandalismo e degradações comuns em ambientes urbanos.

Uma alternativa mais elaborada dessa solução seria a instalação de dispositivos de localização GPS nos veículos de transporte público e o cadastro de celulares de deficientes visuais. Os usuários cadastrados no serviço fariam uso dele da seguinte maneira: ao se encontrar num ponto de parada aguardando o ônibus, o usuário enviaria um SMS à operadora do serviço indicando a necessidade de receber informações sobre os ônibus. Através de um sistema de localização GPS, são identificados todos os ônibus que se aproximam do usuário, para que um SMS seja enviado ao usuário em tempo hábil, indicando a aproximação dos ônibus.

Percebe-se que essa solução requer instalação de infra-estrutura apenas nos veículos de transporte público, sem necessidade de alteração dos pontos parada. Com isso, essa medida elimina a exposição da infra-estrutura do serviço ao vandalismo, por exemplo. Ao mesmo tempo, nota-se que há algumas restrições para o seu uso: o usuário precisa de um celular previamente cadastrado. Aparelhos de localização GPS tornam essa solução mais cara, não apenas para a instalação nos veículos, mas também para o usuário final, já que celulares com dispositivos GPS integrados são usualmente mais caros do que os normais.

Ambas as alternativas citadas solucionam de modo mais prático, imediato e abrangente o problema. Todas elas, porém, têm contra si o fato de que não fornecem total autonomia aos

seus usuários, ficando esses dependentes de sua implementação em toda a frota de ônibus de todos os municípios nos quais ele venha a estar. A solução proposta por este estudo visa trazer essa autonomia ao usuário, uma vez que, adequadamente não requer a dependência de terceiros, órgãos públicos ou empresas de transporte – a pessoa que a utiliza pode operá-la e obter todas as informações de que necessita.

4. Fundamentação Teórica

4.1. Classificação de imagens em Inteligência Artificial

Cada vez mais utilizados em aplicações que necessitam identificar o conteúdo de imagens, os classificadores se valem de alguma ferramenta matemática para, a partir de um banco de conhecimento fornecido previamente, gerar uma função que recebe como entrada uma imagem e fornece como saída sua classificação.

Em linhas gerais, as diferentes abordagens para classificação de imagens podem ser divididas em dois grupos [7]: classificação supervisionada e classificação não-supervisionada. Na classificação supervisionada, as classes – universo de saídas do classificador – são definidas através do uso de um banco de conhecimento previamente rotulado. Isto é, as classes nas quais se deseja dividir as imagens são identificadas antes da geração da função classificadora. A classificação não-supervisionada, pelo contrário, não assume que as classes sejam conhecidas antes da análise das imagens. O próprio algoritmo classificador procura classes semelhantes através de alguma técnica de agrupamento ou aglomeração. Em alguns casos, é necessário informar o número de classes existentes ou critérios para que o agrupamento sugerido seja considerado matematicamente estável.

Nota-se que a resposta das funções classificadoras do primeiro grupo – de classificação supervisionada – é uma das classes previamente existentes nas amostras de treinamento. Todavia, a saída das funções do segundo grupo – de classificação não-supervisionada – é uma classe não rotulada que contém a imagem analisada. Cabe ao próprio analista, ao final do processo, rotular adequadamente as classes criadas pelo classificador.

Não se pode pensar que é impossível usar mais de um classificador na solução de determinado problema. Sistemas multi-classificadores têm apresentado grande sucesso [5], quando comparados ao uso isolado de um método de classificação. É possível combinar algoritmos do mesmo tipo gerados a partir de amostras de treinamento diferentes, método que leva o nome de multi-classificação homogênea. O caso contrário também é usual: a

combinação de diferentes tipos de algoritmos – conhecida por multi-classificação heterogênea.

Generalizar o estudo sobre classificadores é uma tarefa extremamente difícil, dada a heterogeneidade das técnicas envolvidas na classificação de imagens. Existem classificadores que utilizam princípios discriminantes multivariados da estatística, há aqueles que se baseiam em Redes Neurais e até os que se apóiam em princípios da lógica *fuzzy*. Apesar disso, é possível delinear o macro-processo de classificação de imagens de modo que seja possível compreender o funcionamento dos classificadores mais comuns.

O procedimento de classificação supervisionada de imagens é dividido em duas principais etapas: a seleção de atributos da imagem e a classificação da imagem de acordo com os atributos observados. A primeira etapa consiste na tradução da imagem em um número determinado de parâmetros que possuam relevância na definição da classe à qual pertence. A etapa de classificação toma por base os parâmetros observados e infere a classe que possui a maior probabilidade de conter a imagem. A função que calcula essas probabilidades é criada a partir de amostras de treinamento previamente rotuladas.

4.2. Seleção de atributos da imagem

Em Classificação de Imagens, a seleção de atributos de uma imagem é um campo de estudo de importância tão grande quanto o próprio estudo do algoritmo classificador a ser empregado. Isso se dá porque os parâmetros utilizados pelo algoritmo no aprendizado alteram diretamente a função classificadora que será gerada. Sendo assim, se são utilizados parâmetros de baixa relevância para a classificação, a função criada não terá uma taxa de sucesso satisfatória.

De acordo com o resultado que se deseja alcançar, diferentes atributos podem ser encontrados através de uma análise prévia da imagem, a fim de alimentar a função classificadora com informações relevantes para uma divisão em classes mais adequada. Como exemplo, para classificar imagens do sangue em relação à possibilidade de leucemia, é possível utilizar atributos como análise de textura da substância [10].

Uma vez que uma vasta gama de atributos pode ser levantada a partir de uma única imagem, diversos estudos buscam reduzir o número desses atributos, através de uma análise de relevância de cada um deles para o resultado. Mesmo em análises complexas, como a da movimentação de um rosto, algoritmos de redução do número de atributos alcançaram resultados satisfatórios [4]. Com menos atributos para buscar numa imagem, o software que executa a classificação não precisa de um grande poder de processamento.

Para reconhecer o conteúdo de uma imagem, uma técnica comum, porém não muito satisfatória em grande parte dos casos, é utilizar o conteúdo dos pixels da imagem. Para o caso de imagens de alta resolução, é aconselhável fazer uma releitura da imagem, ou seja, dividir-la em conjuntos de pixels e interpretar cada um deles pela média de seus valores.

4.2.1. Histograma de Gradientes Orientados

O objetivo da seleção de atributos é caracterizar e associar padrões entre a base de dados e a imagem a ser testada.

Quando se fala de Histograma de Gradientes Orientados (*Histogram of Oriented Gradients - HOG*), a idéia básica é que tanto as formas quanto as aparências dos objetos contidos em uma imagem podem ser muito bem caracterizadas a partir da distribuição local das intensidades dos gradientes, ou seja, a imagem pode ser representada por meio de um histograma formado pela direção de seus contornos, mesmo sem o conhecimento preciso das posições dos gradientes ou contornos correspondentes [13].

No caso específico de reconhecimento da existência ou não de ônibus, ou outros veículos como automóveis, caminhões ou vans, como é o caso desse projeto, a representação do objeto pode ser muito bem caracterizada somente com a utilização das formas existentes na figura, podendo assim ser ignorada a parcela referente à aparência do objeto [3].

A partir das considerações e conceitos acima, pode-se construir o histograma desejado. Tal histograma será constituído por K colunas, onde cada coluna representa um dos K intervalos de ângulos definidos, e cuja magnitude representa o número de contornos que tem orientação dentro de tal intervalo de ângulos [2].

Para o presente trabalho, o número K adotado será 15 – segundo testes de [13], a partir de 8 intervalos, o incremento marginal com o aumento de intervalos vai diminuindo. Acima de 15 intervalos, o processador é onerado sem o benefício do aumento de desempenho do classificador.

Para a construção de tal histograma, o seguinte roteiro de implementação foi seguido: (I) transformação da imagem em *grayscale* (escala de cinza), (II) aplicação do detector de contornos de Canny e (III) cálculo dos gradientes a partir da Máscara de Sobel.

I) Transformação da imagem em *Grayscale*

Como parte do processo conforme proposto por [3], cada imagem passou por um tratamento, onde a imagem inicial foi modificada para escala de cinza. Essa técnica, como o próprio nome já diz, consiste em transformar uma imagem colorida em tons de cinza. Para isso, foi utilizada uma técnica de conversão por média das componentes de cor vermelha,

verde e azul de cada pixel da imagem. A intensidade resultante de cada pixel foi calculada de acordo com a seguinte média ponderada: $I = 0{,}3R + 0{,}59G + 0{,}11B$; onde R, G e B correspondem às componentes de cor vermelha, verde e azul, respectivamente.

Abaixo segue exemplo do método aplicado para cada uma das classes definidas:

Classe "Com ônibus"

Foto original

Foto em escala de cinza

Figura 1

Classe "Sem ônibus"

Foto original

Foto em escala de cinza

Figura 2

II) Detector de contornos de Canny

A caracterização de contornos é um problema de fundamental importância no processamento de imagens. Os contornos de uma imagem podem ser definidos como áreas que possuem forte intensidade de contraste, ou seja, um contorno pode ser detectado por uma mudança repentina da intensidade de um pixel para outro adjacente.

O detector de contornos é uma ferramenta essencial que tem como finalidade a redução de informações, preservando, entretanto, toda a propriedade estrutural importante para o reconhecimento de uma imagem através da técnica HOG.

Ao final desse processo, o resultado é uma imagem lógica, ou seja, que contém apenas pixels com valores binários, onde somente os pixels relativos aos contornos possuem valor intensidade. Segue, para as mesmas fotos utilizadas no exemplo acima, o resultado da aplicação do detector de contornos de Canny:

Classe "Com ônibus"

Foto original

Foto só com contornos (método de Canny)

Figura 3

Classe "Sem ônibus"

Foto original

Foto só com contornos (método de Canny)

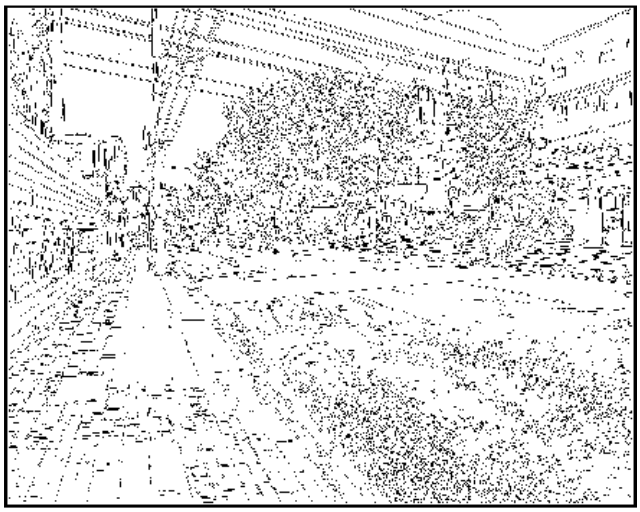

Figura 4

III) Máscara de Sobel

A Máscara de Sobel nada mais é do que uma importante ferramenta usada para estimar os gradientes presentes em cada ponto de uma imagem em *grayscale*.

A partir de um par de máscaras, são estimadas as duas componentes x e y dos gradientes de uma imagem 2-D, ou seja, cada uma das máscaras é usada para a estimativa em cada uma das direções.

Para o projeto em questão, foram utilizados os operadores de Sobel de dimensão 3X3. Como as matrizes operadoras possuem dimensão 3X3, ou seja, muito menor que a dimensão da matriz de pixels da imagem, uma mesma máscara deverá percorrer toda a extensão da matriz de pixels, manipulando uma matriz quadrada 3X3 de pixels por vez. As máscaras utilizadas para cada direção foram as seguintes:

$$M_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \qquad M_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Equação 1 **Equação 2**

Com as matrizes acima definidas, para se chegar aos gradientes desejados, primeiramente sobrepõe-se a máscara na posição extrema superior e à esquerda da matriz de pixels da imagem, assim como mostrado na ilustração abaixo. O pixel contido na posição central da sub-matriz sobreposta pela matriz máscara terá seu valor modificado de acordo com a equação exposta na ilustração. Feito isso, a matriz máscara será posicionada no próximo pixel à direita, e o mesmo cálculo será realizado até se atingir o final dessa linha, quando então parte-se para a linha seguinte, repetindo todo o procedimento.

Vale notar que pixels localizados nas primeiras e últimas linhas e colunas permanecerão inalterados. Isso se dá pelo fato de só o pixel central da sub-matriz sobreposta ter seu valor modificado, ou seja, se posicionarmos o elemento central da matriz máscara em um dos pixels das primeiras ou últimas colunas ou linhas, parte da matriz máscara extrapolará os limites da matriz de pixels.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1m} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2m} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nm} \end{bmatrix} \quad \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} \quad \begin{bmatrix} b_{11} & b_{12} & b_{13} & \cdots & b_{1m} \\ b_{21} & b_{22} & b_{23} & \cdots & b_{2m} \\ b_{31} & b_{32} & b_{33} & \cdots & b_{3m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & b_{n3} & \cdots & b_{nm} \end{bmatrix}$$

$$b_{22} = (a_{11} \times m_{11}) + (a_{12} \times m_{12}) + (a_{13} \times m_{13}) + (a_{21} \times m_{21}) + (a_{22} \times m_{22}) + (a_{23} \times m_{23}) + (a_{31} \times m_{31}) + (a_{32} \times m_{32}) + (a_{33} \times m_{33})$$

Equação 3

Calculadas as matrizes Gx e Gy, pode-se então calcular os módulo e ângulos dos gradientes de cada um dos pixels da imagem a partir das seguintes equações:

$$|G| = |G_x| + |G_y|$$

Equação 4

$$\theta = \arctan \frac{G_y}{G_x}$$

Equação 5

Classe "Com ônibus"

Foto original

Matriz Gradientes de Orientação

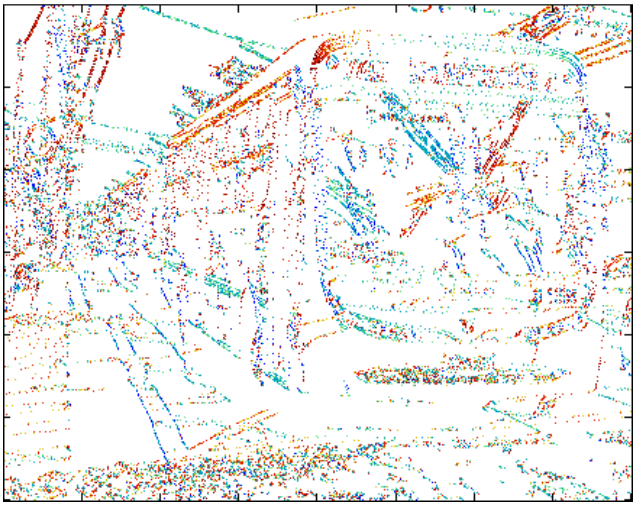

Figura 5

Classe "Sem ônibus"

Foto original

Matriz Gradientes de Orientação

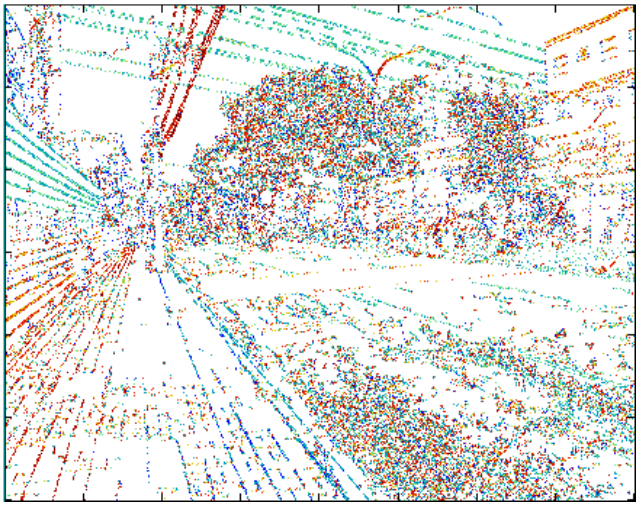

Figura 6

Classe “Com ônibus”

Foto original

Matriz Intensidade de Gradientes de Orientação

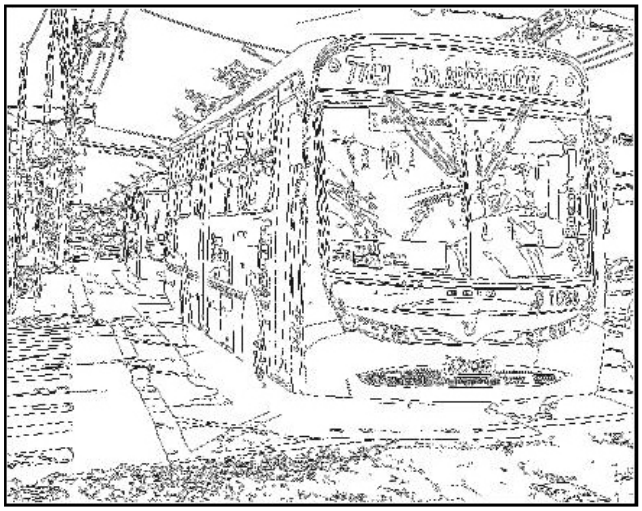

Figura 7

Classe “Sem ônibus”

Foto original

Matriz Intensidade de Gradientes de Orientação

Figura 8

A partir dos cálculos e métodos anteriormente apresentados e discutidos pôde-se construir o histograma para cada uma das imagens exemplo. O histograma é construído levando-se em consideração a freqüência dos gradientes ao longo dos pixels da imagem, contudo, o peso de cada pixel não é uniforme, mas sim equivalente à intensidade relativa ao pixel, calculada pelo filtro de Sobel.

Depois de construído o histograma da imagem, ela é classificada (para posterior aprendizado supervisionado). Sendo assim, ao compilar todas as imagens, o resultado é uma base de dados com 16 atributos (15 intervalos do histograma mais a classe).

Abaixo segue exemplo dos histogramas de cada uma das imagens, onde o eixo horizontal representa os quinze intervalos de ângulos definidos:

- $1 \Rightarrow 0° \leq \alpha \leq 12°$
- $2 \Rightarrow 12° < \alpha \leq 24°$
- $3 \Rightarrow 24° < \alpha \leq 36°$
- $4 \Rightarrow 36° < \alpha \leq 48°$
- $5 \Rightarrow 48° < \alpha \leq 60°$
- $6 \Rightarrow 60° < \alpha \leq 72°$
- $7 \Rightarrow 72° < \alpha \leq 84°$
- $8 \Rightarrow 84° < \alpha \leq 96°$
- $9 \Rightarrow 96° < \alpha \leq 108°$
- $10 \Rightarrow 108° < \alpha \leq 120°$
- $11 \Rightarrow 120° < \alpha \leq 132°$
- $12 \Rightarrow 132° < \alpha \leq 144°$
- $13 \Rightarrow 144° < \alpha \leq 156°$
- $14 \Rightarrow 156° < \alpha \leq 168°$
- $15 \Rightarrow 168° < \alpha \leq 180°$

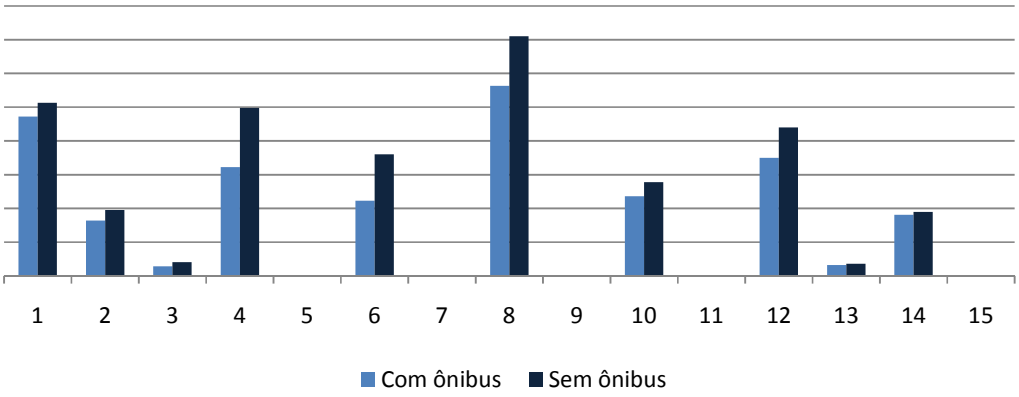

Figura 9 – Histograma final gerado pela seleção de atributos (comparação entre duas imagens de classes diferentes)

4.3. Algoritmos de Classificação

A proposta deste projeto é desenvolver um software capaz de classificar uma imagem em uma das seguintes classes: “com ônibus” ou “sem ônibus”. Dentro do procedimento para alcançar tal resultado, o classificador possui a função de identificar a qual classe a imagem pertence a partir dos valores de cada atributo previamente medidos na imagem a ser classificada.

Durante o desenvolvimento do projeto, fez-se necessário construir pequenos softwares para testar e avaliar a abordagem matemática do problema. Nessa etapa, o algoritmo de classificação *Multilayer Perceptron* apresentou os resultados mais satisfatórios. Contudo, após estudar a portabilidade da aplicação, dadas as restrições de recursos computacionais num dispositivo portátil, a utilização do algoritmo *Naive Bayes* de classificação tornou-se muito mais viável. Apesar de ambos os classificadores cumprirem sua função adequadamente, a abordagem realizada por cada um deles é muito diferente.

4.3.1. Multilayer Perceptron

O algoritmo *Multilayer Perceptron* é uma evolução do algoritmo *Perceptron*, um dos mais simples tipos de Redes Neurais *feedforward* [14]. Uma rede neural consiste em técnicas computacionais que usam um modelo matemático inspirado na estrutura neural de organismos inteligentes. Similarmente aos neurônios do cérebro humano, uma rede neural artificial é composta por elementos chamados de *neurons*. Esses elementos podem possuir conexões entre si e com o meio externo, de onde partem os estímulos.

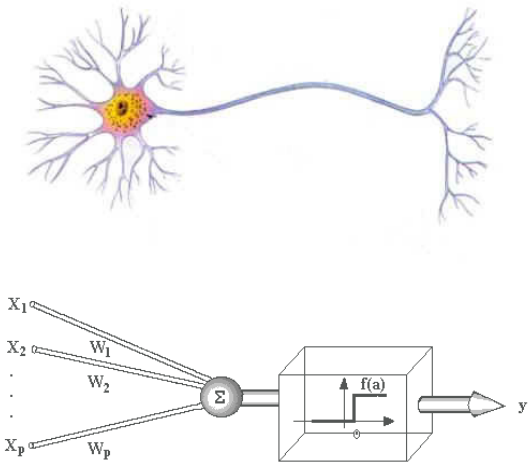

Figura 10 - Comparação entre um neurônio humano e o *neuron* artificial que compõe uma rede neural.

O *neuron* nada mais é do que uma função binária cuja saída depende apenas da comparação da soma ponderada dos estímulos medidos com um valor de referência (conhecido como *Threshold*). Durante as diversas iterações necessárias para o aprendizado da rede neural, os pesos da soma ponderada são atualizados de acordo com a comparação da saída da função com a saída esperada.

Na rede neural do tipo *Multilayer Perceptron*, existe mais do que uma camada de *neurons*, ou seja, ao mesmo tempo em que *neurons* estão ligados a atributos medidos na imagem a ser classificada, existem alguns *neurons* que estão ligados à saída de outros *neurons*. A Figura 11 facilita o entendimento desse tipo de rede neural artificial.

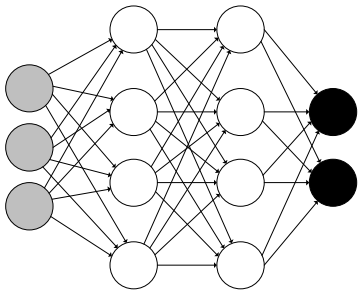

Figura 11 - Rede Neural com várias camadas, em que os *neurons* em cinza se conectam aos atributos medidos enquanto os demais se conectam apenas à saída de outros *neurons*.

Pode-se perceber que em redes neurais com mais de uma camada, é possível capturar os efeitos das relações entre diferentes atributos. Relações complexas podem ser contempladas pelo modelo apenas aumentando o número de camadas. Por outro lado, os recursos computacionais necessários para treinar o algoritmo também crescem com a inclusão de novas camadas.

4.3.2. Naive Bayes

A função classificadora pode ser criada com base em diversos métodos, contudo, a restrição de poder de processamento e de memória disponível num dispositivo portátil privilegia algoritmos simples e com pequena demanda de memória alocada dinamicamente. Um dos métodos mais simples e bem sucedidos utilizados atualmente em Classificação é o de *Naive Bayes*.

O classificador *Naive Bayes* procura inferir as probabilidades de uma imagem pertencer a cada uma das classes, ou antes, de encontrar a classe que apresenta maior probabilidade de representar a imagem [15]. O cálculo dessas probabilidades é feito a partir das observações dos valores de cada atributo, ou seja, $P(C = c_j | A_1 = a_1 \cap A_2 = a_2 \cap \ldots \cap A_{15} = a_{15})$ é a

probabilidade de uma imagem pertencer à classe c_j dado que o seu atributo A_1 possui valor a_1, o seu atributo A_2 tem valor a_2 e assim por diante, até o atributo 15.

Apesar de um tanto simplista, o classificador *Naive Bayes* pode ser treinado de maneira eficiente, especialmente em situações de aprendizado supervisionado. Esse algoritmo assume a hipótese de independência entre os atributos dos dados a serem classificados, o que raramente ocorre em situações reais. Como em diversos outros algoritmos, seu modelo probabilístico é montado a partir da probabilidade *a posteriori* do objeto pertencer a cada classe, uma vez observados os atributos. Essa probabilidade não seria facilmente inferida, a não ser pela hipótese da independência dos atributos assumida. A partir do modelo de probabilidades construído com as amostras de treinamento, o classificador utiliza uma regra de decisão para escolher a classe a que pertence cada objeto. Uma regra usual é a conhecida por MAP, ou Máximo *a posteriori*, que consiste em escolher a classe que maximiza a probabilidade calculada a partir dos atributos observados.

Como citado anteriormente, é notável o freqüente sucesso desse classificador mesmo com hipótese tão simplista como a de independência entre os atributos. Estudos atribuem essa característica ao fato de que, apesar de a independência entre os atributos não ser verdadeira na maior parte das situações reais, a distribuição de dependências entre os atributos pode ser uniforme e causar certa compensação [12].

O modelo matemático utilizado durante o treinamento do classificador *Naive Bayes* consiste numa rede Bayesiana na qual todos os atributos são apenas dependentes da classe da imagem, e não possuem relação alguma de dependência entre si, como mostrado na Figura 12.

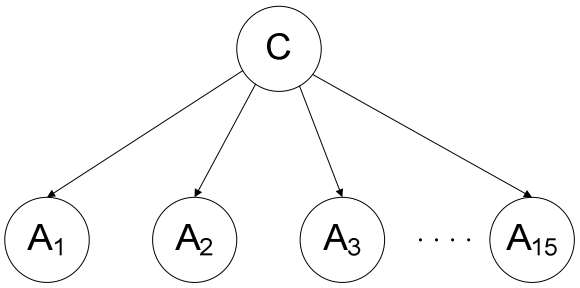

Figura 12 - Rede Bayesiana que representa o modelo matemático utilizado pelo Classificador Naive Bayes.

A probabilidade de a imagem pertencer à determinada classe dada a observação dos atributos é conhecida como probabilidade a posteriori. O Teorema de Bayes calcula esse tipo de probabilidade como enunciado na equação abaixo, onde $\boldsymbol{A}$ e $\boldsymbol{a}$ representam respectivamente o conjunto de todos os atributos e os seus valores na imagem a ser classificada.

$$P\big(C = c_j \big| \boldsymbol{A} = \boldsymbol{a}\big) = \frac{P\big(C = c_j\big).P(\boldsymbol{A} = \boldsymbol{a}|C = c_j)}{P(\boldsymbol{A} = \boldsymbol{a})}$$

Equação 6

A partir da amostra de treinamento fornecida ao Classificador, é simples calcular a probabilidade a priori de cada classe $P(C = c_j)$. Além disso, $P(\boldsymbol{A} = \boldsymbol{a})$ envolve apenas os valores observados para cada atributo, ou seja, em uma operação de classificação, em que esses valores são fixos, tal probabilidade se torna uma constante normalizadora. Todavia, o cálculo de $P(\boldsymbol{A} = \boldsymbol{a}|C = c_j)$ pode se tornar bastante complexo, uma vez que requer o conhecimento de como o valor de um atributo afeta o valor de outro. Ao assumir a hipótese da independência dos atributos, entretanto, o classificador *Naive Bayes* torna o cálculo dessa probabilidade bastante simples, como mostrado na equação abaixo.

$$P\big(\boldsymbol{A} = \boldsymbol{a}\big|C = c_j\big) = \prod_{i}^{15} P\big(A_i = a_i\big|C = c_j\big)$$

Equação 7

Dessa forma, apenas resta o cálculo de $P\big(A_i = a_i\big|C = c_j\big)$ para cada atributo i. Para tanto, é preciso conhecer a distribuição de probabilidade de cada um deles. Como descrito na seção anterior, para este projeto, os atributos de uma imagem são os valores encontrados no seu histograma de gradientes de orientação. Cada um deles foi representado como uma distribuição normal com média e desvio padrão inerentes à cada uma das classes.

O valor da probabilidade de uma variável aleatória real com distribuição contínua em um determinado ponto é igual a zero. Apesar disso, para realizar esse cálculo, utiliza-se o valor da probabilidade da variável dentro de um pequeno intervalo Δ, que pode ser calculada segundo a equação que segue.

$$P(a_i \leq A_i \leq a_i + \Delta) = \int_{a_i}^{a_i+\Delta} \frac{1}{\sigma_{ij}\sqrt{2\pi}} e^{-\frac{(a_i-\mu_{ij})^2}{2\sigma_{ij}{}^2}} \, da_i$$

Equação 8

Pela definição de derivada,

$$\lim_{\Delta \to 0} \frac{P(a_i \leq A_i \leq a_i + \Delta)}{\Delta} = \frac{1}{\sigma_{ij}\sqrt{2\pi}} e^{-\frac{(a_i - \mu_{ij})^2}{2\sigma_{ij}{}^2}}$$

Equação 9

Para uma constante Δ suficientemente pequena,

$$P(a_i \leq A_i \leq a_i + \Delta) = \frac{1}{\sigma_{ij}\sqrt{2\pi}} e^{-\frac{(a_i - \mu_{ij})^2}{2\sigma_{ij}{}^2}} . \Delta$$

Equação 10

Percebe-se que o fator Δ aparece no numerador da equação de $P(C = c_j | \boldsymbol{A} = \boldsymbol{a})$. Todavia, ao substituir o cálculo do denominador da equação dessa probabilidade por uma normalização com os resultados dos cálculos de probabilidade das demais classes, o fator Δ é cancelado. É possível, portanto, calcular $P(A_i = a_i | C = c_j)$ através da aplicação da equação abaixo.

$$P(A_i = a_i | C = c_j) = \frac{1}{\sigma_{ij}\sqrt{2\pi}} e^{-\frac{(a_i - \mu_{ij})^2}{2\sigma_{ij}{}^2}}$$

Equação 11

Sendo assim, o classificador *Naive Bayes* infere, a partir das amostras de treinamento, os parâmetros necessários para a classificação de uma imagem, tais como as médias e desvios-padrão dos atributos para cada classe, bem como as probabilidades a priori de cada classe. Ao classificar uma imagem, esses parâmetros são utilizados no cálculo de $P(A_i = a_i | C = c_j)$, como mostrado acima. Depois disso, $P(C = c_j | \boldsymbol{A} = \boldsymbol{a})$ é calculado para cada classe j, ignorando-se o denominador do Teorema de Bayes, uma vez que é constante para todas as classes, dados os valores dos atributos. Apenas com essas informações, já é possível inferir a classe mais provável a que pertence a imagem, contudo, a probabilidade a *posteriori* pode ser calculada com a regra de normalização abaixo.

$$\sum_j P(C = c_j | \boldsymbol{A} = \boldsymbol{a}) = 1$$

Equação 12

5. Desenvolvimento do projeto

5.1. Ferramentas utilizadas

Antes do desenvolvimento do software classificador de imagens para uma plataforma portátil, um software com as mesmas funções foi desenvolvido para execução num computador pessoal. A intenção dessa etapa inicial foi testar diferentes abordagens de seleção de atributos e de classificação, bem como o uso dos recursos de memória e processamento.

O software desenvolvido para PC fazia uso de todas as funções pré-construídas e pré-testadas no domínio público. Dessa forma, foi possível garantir o foco do desenvolvimento no problema de seleção de atributos. Mesmo a etapa de classificação lançou mão de bibliotecas disponíveis no domínio público. Isso só foi possível graças ao software WEKA, desenvolvido pela Universidade de Waikato da Nova Zelândia [11].

O WEKA permite ao usuário realizar operações de aprendizado de máquina e classificação através de diversos algoritmos classificadores. É possível, inclusive, medir o desempenho desses algoritmos através dos resultados estatísticos das funções classificadoras. Com uma interface bastante amigável, a filosofia do WEKA é dar ferramentas de aprendizado de máquinas a especialistas de quaisquer assuntos, em vez de simplesmente implementar algoritmos para serem usados por estudantes de sistemas de classificação. Além disso, o WEKA possui implementações dos classificadores em bibliotecas Java, que foram importadas ao software para PC desenvolvido durante este projeto.

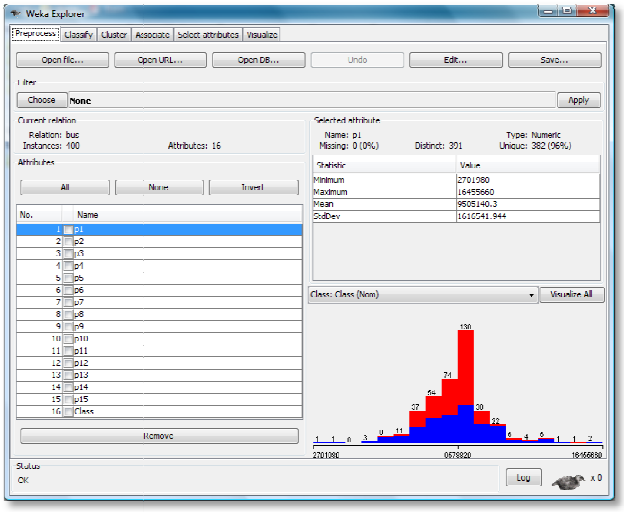

Figura 13 - O software WEKA permite a comparação do desempenho de diferentes classificadores através de uma interface bastante amigável.

Para criar a base de dados com as amostras de treinamento, algumas funções foram desenvolvido na plataforma Matlab, da MathWorks, uma vez que existe grande suporte para

tratamento de imagens nessa plataforma. Além disso, o Matlab permite fácil programação, depuração e execução do código criado. Apesar de apresentar baixa performance, o código executado no Matlab não impactou a performance do software de classificação, uma vez que só foi utilizado na etapa de seleção de atributos para criação da base de treinamento para testes no WEKA. Posteriormente, foi necessário desenvolver métodos que criassem bases de treinamento para cada plataforma ou linguagem de programação utilizada.

O WEKA exige o formato ARFF para as bases de dados com as quais ele se relaciona. Dessa forma, o código em Matlab foi escrito para ler uma imagem, realizar a seleção de atributos e escrever seus resultados num arquivo ARFF, que seria posteriormente utilizado pelo WEKA ou por suas bibliotecas de classificadores. O formato ARFF exige a declaração prévia dos atributos e seus tipos, bem como sua discretização – quando possível. A Figura 14 exemplifica um arquivo desse tipo. Note que após, o identificador "@Data", cada linha apresenta todos os valores dos atributos de uma imagem separados por vírgulas. Os atributos e seus tipos encontram-se declarados anteriormente, cada um antecipado por um identificador "@attibute".

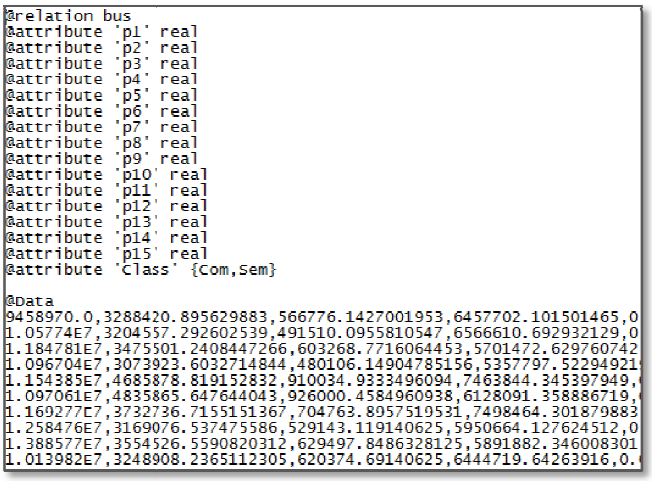

Figura 14 - Exemplo de arquivo ARFF contendo informações sobre os atributos de diversas imagens

Desenvolver softwares voltados para diferentes plataformas – PCs e dispositivos móveis, por exemplo – pode causar a necessidade de reescrever métodos ou mesmo algoritmos inteiros. Numa tentativa de minimizar esses impactos, ambos os softwares utilizaram a mesma plataforma de desenvolvimento, o Netbeans. Disponível gratuitamente na internet, essa plataforma possui funcionalidades de depuração do código escrito, assim como de auxílio à construção de interfaces gráficas e de medição de desempenho do programa criado. Com uma versão customizada para suportar o desenvolvimento de aplicativos móveis, foi possível

desenvolver e testar o software em um emulador de celular no computador antes de implantar o software num dispositivo físico.

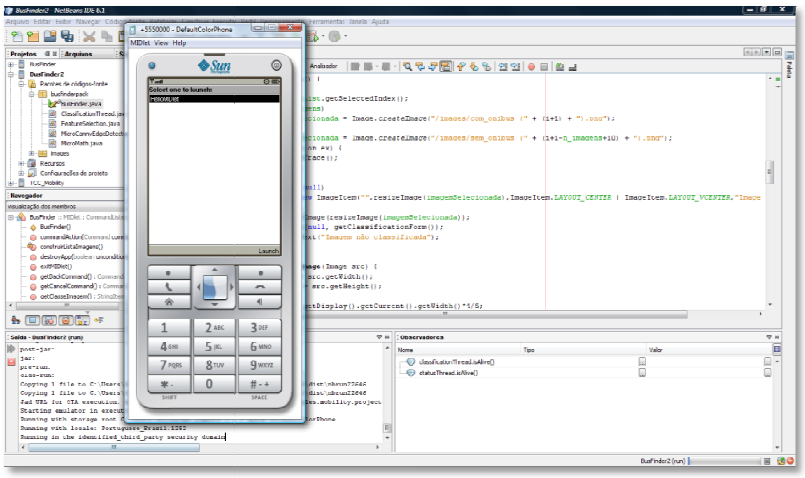

Figura 15 - A plataforma de desenvolvimento Netbeans oferece funcionalidades como um emulador para testes de softwares para dispositivos móveis.

5.2. Aspectos da linguagem de programação

Depois de definido todo o processo de seleção de atributos, assim como o método de classificação de imagem, pôde-se partir para a parte de desenvolvimento do aplicativo, propriamente dito. Tal etapa do projeto foi dividida em três estágios, cada qual com uma determinada plataforma de programação. Os três estágios se utilizaram das seguintes plataformas respectivamente: Matlab, Java SE e, por último, Java ME.

5.2.1. Primeiro Estágio – Matlab

Nessa fase de desenvolvimento, fez-se uso do software Matlab, da empresa MathWorks. Através de funcionalidades providas pelo suplemento Image Processing Toolkit, a extração de atributos foi automatizada, o que possibilitou a preparação de uma nova base de treinamento para testes com diferentes algoritmos de classificação.

O processo de seleção de atributos se inicia com o carregamento da imagem no Matlab, através da função *imread*. Tal função terá como resultado uma matriz 3D, onde a primeira dimensão da matriz são as componentes vermelhas de cada pixel da imagem, assim como as segundas e terceiras dimensões correspondem às componentes verdes e azuis respectivamente. Tendo sida carregada a imagem, o próximo passo foi a criação de uma rotina

“Sobel.m”. Essa rotina recebe a matriz carregada na etapa anterior e tem como resultado duas outras matrizes, a matriz intensidade e a matriz orientação dos vetores calculados ao longo dos contornos da imagem, assim como descrito no tópico Seleção de Atributos. É nessa etapa do programa em que o método de detecção de contornos de Canny é aplicado. Através da função *edge*, que recebe como parâmetros a matriz a ser tratada e o método de detecção utilizado, retornando uma matriz que representa a imagem só com seus contornos.

Por último, programou-se uma função “Histogram.m”, onde, recebendo as duas matrizes calculadas pela “Sobel.m”, assim como o número de intervalos de ângulos definidos (no caso K=15), chega-se ao histograma desejado.

5.2.2. Segundo Estágio – Java 2 Standard Edition

A fim de se iniciar o processo de portabilidade do aplicativo, o mesmo código Matlab deveria ser transcrito para uma linguagem que possuísse como característica a fácil portabilidade, além de grande disponibilidade de métodos para tratamento de imagens.

Estudando a literatura, optou-se pela utilização do Java, linguagem essa reconhecida e difundida mundialmente pela sua independência de plataforma. Programas Java não são traduzidos para a linguagem de máquina, como outras linguagens estaticamente compiladas, e sim para uma representação intermediária, chamada de *bytecodes*.

Os *bytecodes* são interpretados pela máquina virtual Java (JVM - Java Virtual Machine), máquina que pode ser instalada em qualquer dispositivo, desde computadores convencionais, até Palm Tops ou celulares.

O primeiro desafio encontrado nessa transcrição foi a ausência de métodos utilizados na primeira codificação. Como exemplo pode-se citar o método de extração de contornos. Diferentemente da plataforma Matlab, que disponibilizava um método específico para essa finalidade, em Java, os próprios desenvolvedores foram responsáveis pelo desenvolvimento de tal rotina.

O segundo desafio encontrado foi a restrição de memória. Devido à máquina virtual utilizada para a execução do programa, que impede que o mesmo cresça demasiadamente, o código teve que ser readaptado, ou seja, teve que se reestruturar o código. Tal reestruturação passou por um processo de divisão das imagens em sub-segmentos, diminuindo assim os tamanhos das variáveis utilizadas, assim como a memória reservada para cada uma delas na compilação do programa.

Uma solução possível solução para o problema acima proposto seria a utilização de parâmetros de execução capazes de expandir a memória utilizada pela máquina virtual. Porém, como o projeto tem como objetivo final sua portabilidade, ou seja, rodar em dispositivos com baixo poder de processamento e memória reduzida, essa solução não seria aplicável.

5.2.3. Terceiro Estágio – Java 2 Micro Edition

O J2ME (Java 2 Micro Edition) é uma plataforma que implementa a linguagem JAVA e é utilizado em dispositivos móveis, como celulares, smartphones, Palm Tops, Pocket PCs, algumas TV´s de nova geração, dentre outros.

Com o intuito de tornar o Java ainda mais portátil, criou-se o J2ME, versão reduzida do J2SE, onde foram feitas remoções e algumas modificações em partes fundamentais do J2SE, possibilitando assim que a linguagem rode em dispositivos com capacidade limitada, ou seja, pouca memória e baixo processamento.

Dadas as condições descritas acima, os desenvolvedores se defrontaram com ainda mais dificuldades.

Assim como na etapa acima, para transcrever o código para Java ME, mais restrições de memória foram encontradas, porém ainda mais graves do que no caso anterior. Além de nova reestruturação do programa, as imagens tiveram que ser tratadas e redimensionadas, passando de resolução 1600 X 1200 para 480 X 360. Isso se deu pela impossibilidade de leitura de uma imagem de alta resolução, já que seria impossível criar vetores que suportassem armazenar todos os pixels dessa imagem.

A escassez de classes e métodos também foi outro ponto de grande dificuldade nessa etapa do projeto. Além de disponibilizar um número restrito de classes e métodos para tratamento de imagens, o Java ME também tem restrições com relação a métodos matemáticos e aritméticos. Um exemplo a ser citado é a inexistência de métodos como arco-tangente e exponencial. Para reverter tal limitação, os desenvolvedores tiveram que criar suas próprias funções através das respectivas séries de Taylor:

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}$$

Equação 13

$$\arctan\ x = \frac{x}{1+x^2}\sum_{n=0}^{\infty}\prod_{k=1}^{n}\frac{2kx^2}{(2k+1)(1+x^2)}$$

Equação 24

6. Resultados

6.1. Desempenho dos classificadores

Como citado anteriormente, o desenvolvimento do software portável para dispositivos móveis foi precedido pelo desenvolvimento de uma versão de testes executável num computador pessoal. Apesar de ambas as versões – a para o PC e a portável – fazerem uso do mesmo modelo de seleção de atributos, devido às limitações dos recursos computacionais disponíveis num dispositivo móvel e das poucas funcionalidades oferecidas de bibliotecas da plataforma Java ME, o desenvolvimento do software portável fez uso de um algoritmo classificador diferente do utilizado na versão de testes do PC. A substituição do algoritmo de classificação apresentou resultados satisfatórios, apesar de causar uma queda no desempenho geral do classificador, medido através do número de imagens classificadas corretamente.

Figura 16 - À esquerda, o software para PC desenvolvido para testes. À direita, o software de classificação desenvolvido para celular.

A medição de desempenho de um software de classificação deve ser executada de forma cautelosa. É necessário atentar para indicadores relacionados com cada classe isoladamente em vez de olhar apenas para a média de acertos global. Isso se dá porque os atributos selecionados para avaliar a imagem podem ser suficientemente bons para diferenciar uma classe, mas não necessariamente todas elas. A avaliação usual do desempenho de um algoritmo de classificação é realizada através das Tabelas de Confusão criadas a partir da classificação de uma base de imagens previamente rotulada e da comparação dos resultados com os valores esperados.

Uma tabela de confusão é composta por colunas representando as classes sugeridas pelo algoritmo e por linhas representando as classes verdadeiras das imagens. Os dados preenchidos na tabela representam a quantidade percentual de imagens em cada uma das situações. Sendo assim todos os valores posicionados na diagonal principal da tabela representam imagens que foram classificadas corretamente.

Os resultados apresentados a seguir referem-se às medições de desempenho realizados no software WEKA sobre as amostras de treinamento utilizadas para construir os modelos matemáticos de cada software: o *Multilayer Perceptron* para o software de PC e o *Naive Bayes* para o software portável. Ambas as bases de treinamento são compostas pelas mesmas 400 imagens – 200 de cada classe –, após passarem por seleção de atributos idêntica (apesar de implementada em diferentes plataformas), como detalhado em seção anterior deste relatório. Para medição do desempenho de cada classificador, a base de imagens foi dividida arbitrariamente de modo que nove décimos da base fossem utilizados para treinar o algoritmo, enquanto o décimo restante foi utilizado para testá-lo. Cada algoritmo foi testado 10 vezes com a mesma base dividida de diferentes maneiras. Esse método é conhecido como *cross-validation*.

6.1.1. Multilayer Perceptron

Utilizado apenas no programa desenvolvido para PC, o algoritmo teve um desempenho geral melhor do que o de *Naive Bayes*. Apesar disso, nota-se que o tempo para construção do modelo (medido pelo software WEKA) foi bastante longo, tendo em vista o tempo de resposta esperado da aplicação.

Tempo para construir o modelo: 2,95 segundos

Classificações corretas: 77,25%

Classificações incorretas: 22,75%

Matriz de Confusão:

		Classe sugerida	
		Com	**Sem**
Classe real	**Com**	77%	24%
	Sem	22%	78%

Tabela 1

6.1.2. Naive Bayes

Com um algoritmo muito mais simples e uma hipótese de independência dos atributos, o bom resultado do classificador chega a ser surpreendente. Implementado no programa desenvolvido para dispositivos móveis, o algoritmo apresentou um desempenho geral menor do que o algoritmo de redes neurais. Apesar disso, nota-se que o tempo para construção do modelo (medido pelo software WEKA) foi bem menor, viabilizando um tempo de resposta satisfatório.

Tempo para construir o modelo: 0,03 segundos

Classificações corretas: 72,00%

Classificações incorretas: 28,00%

Matriz de Confusão:

		Classe sugerida	
		Com	**Sem**
Classe real	**Com**	74%	26%
	Sem	30%	70%

Tabela 2

6.2. Análise de resultados

A avaliação de performance do software desenvolvido para dispositivos móveis é bastante satisfatória, com índices de acerto maiores que 70% para ambas as classes. Esse resultado mostra que a seleção de atributos através do histograma de gradientes de orientação destaca aspectos relevantes para a distinção de imagens proposta neste projeto.

Note que o reconhecimento da presença de um ônibus numa imagem não é tarefa facilmente delineada, uma vez que uma grande quantidade de aspectos pode variar entre diferentes fotos da mesma classe. Como explicado anteriormente, o método de seleção de atributos de uma imagem pelo histograma de gradientes de orientação tende a valorizar as formas da imagem sobre outros aspectos. Observe na Figura 17 exemplos de classificações bem e mal sucedidas.

Figura 17 – Resultados da classificação de diferentes imagens. Percebem-se grandes diferenças mesmo entre imagens de mesma classe

No campo de reconhecimento de imagens através da classificação, estudos semelhantes buscam distinguir imagens entre diversas classes. Para este projeto, o universo de classes disponíveis para classificação pode ser bem caracterizado a partir de informações relativas às formas presentes na imagem. Apesar disso, existem situações em que a forma não é suficiente para diferenciar todas as classes, como em [1]. Com isso, novos atributos necessitam ser selecionados a partir da imagem, destacando informações como a aparência da foto ou a localização e composição das cores presentes nela.

Estudos como [2] conseguiram resultados num intervalo de 70,0% até 81,3% de instâncias classificadas corretamente através do uso de descritores de formas e de aparência da imagem. Os resultados apresentados neste projeto são comparáveis aos de estudos como o citado, mesmo com o uso de algoritmos de seleção de atributos e de classificação mais simples. Isso sugere que há espaço para melhora da performance, através de algoritmos de seleção de atributos e classificação mais elaborados, ressaltando a necessidade de implementação cuidadosa, uma vez que os recursos de um dispositivo móvel são bastante limitados.

7. Possíveis extensões

Como este projeto trata do problema de reconhecimento de imagens através de técnicas de inteligência artificial, possíveis extensões podem ser consideradas.

Uma possível continuidade, seguindo a linha da aplicação escolhida, seria a implementação de métodos capazes de reconhecer não só a existência ou ausência de um ônibus, mas também o seu destino. Para isso, deve ser implementado um método de busca pelo letreiro do ônibus, assim como um método específico para reconhecimento de números e palavras.

Outras inúmeras aplicações podem se utilizar do método em questão. Como exemplo pode-se citar o caso de reconhecimento de placas de automóveis - radares capazes de verificar se um automóvel está transitando em local proibido para determinado dia da semana.

O projeto também pode ser estendido à implementação de um dispositivo capaz de reconhecer a quantidade de veículos presente em uma determinada via. Tal aplicativo poderia ser utilizado, tanto pela engenharia de trânsito, monitorando o tráfego e os congestionamentos, quanto para projeto de semáforos inteligentes. Esses semáforos seriam capazes de determinar o tempo de abertura ou parada de cada via de acordo com o fluxo naquele determinado instante.

8. Conclusão

Analisando o método desenvolvido, assim como seu desempenho em relação a projetos semelhantes, pôde-se perceber que tal projeto apresentou índices satisfatórios, ficando acima da média encontrada em artigos que abordam temas dessa mesma categoria. A partir disso conclui-se que o método escolhido realmente seleciona atributos relevantes, capazes de diferenciar com sucesso cada uma das classes determinadas – Com ônibus ou Sem ônibus.

Também se obteve sucesso na questão relativa à portabilidade do dispositivo. Apesar dos inúmeros obstáculos enfrentados ao longo do trabalho, em especial as limitações de memória e métodos disponíveis na linguagem selecionada para a programação do dispositivo móvel, foi possível chegar ao produto final proposto: um dispositivo portátil voltado ao reconhecimento de imagens, que possui como aplicação a determinação da existência ou não de um ônibus em determinada via.

Assim como mencionado em tópicos anteriores, tal dispositivo pode ser utilizado como parte de uma aplicação para auxílio de deficientes visuais. O software teria a utilidade de informar ao deficiente o momento em que deve solicitar a parada do veículo.

Vale ressaltar que, apesar de haver inúmeras soluções alternativas para tal problema, nenhuma delas apresentaria tal portabilidade, diferencial desse dispositivo em relação aos demais, já que, tendo ele em mãos, pode-se utilizá-lo em qualquer via, não havendo a necessidade de mudança da infra-estrutura de todos os terminais de ônibus.

Outro ponto a ser ressaltado é que, independente da aplicação que venha a ser utilizada, o presente projeto tem como principal enfoque o estudo e o desenvolvimento de um método de classificação de imagens, ou seja, pode ser utilizado como base para diversos outros trabalhos relacionados ao reconhecimento de imagens por inteligência artificial, principalmente para aplicações envolvendo veículos de qualquer categoria.

9. Referências Bibliográficas

[1] Maria-Luiza Antonie, Osmar R. Zaiane, and Alexandru Coman. Application of data mining techniques for medical image classification. In Proc. of Second Intl. Workshop on Multimedia Data Mining (MDM/KDD'2001) in conjunction with Seventh ACM SIGKDD, pp. 2–3, San Francisco, USA, 2001.

[2] A. Bosch, A. Zisermann, and X. Muñoz. Image classification using random forests and ferns. n ICCV, 2007.

[3] A. Bosch, A. Zisserman, and X. Muñoz. Representing Shape with a Spatial Pyramid Kernel. In CIVR, 2007.

[4] Cohen I., Tian Q., Zhou, X. S. and Huang T. S. (2002), Feature Selection Using Principal Feature Analysis. IEEE International Conference in Image Processing (ICIP'02), pp. 4-6, June 2002.

[5] Costa, J. A., Bittencourt, V. G., & Souto, M. C. (2003). Aplicação de Multi-classificadores no Reconhecimento de Classes Estruturais de Proteínas. 2-3.

[6] Demiröz G, Güvenir, HA, Classification by Voting Feature Intervals, Proc. of Ninth ECML, Springer-Verlag, LNAI 1224, 1997.

[7] Erbert, M. (2001). Introdução ao Sensoriamento Remoto. 20-25.

[8] Jacob, A. M., Hemerly, E. M., & Fernandes, D. (2003). SAR Image Classification using Supervised Neural Classifiers.

[9] Maekawa, T., Hara, T., and Nishio, S. Image classification for mobile web browsing. In WWW '06: Proceedings of the 15th international 78 conference on World Wide Web (New York, NY, USA, 2006), ACM Press.

[10] Sabino, D. M., & Costa, L. F. (2004). Feature selection methods applied to multiscale texture.

[11] Witten, I. H., & Frank, E. (2005). Data Mining: Practical machine learning tools and techniques (2nd Edition ed.). San Francisco: Morgan Kaufmann.

[12] H. Zhang, "The optimality of Naive Bayes," in Proc. of the Seventeenth International Florida Artificial Intelligence Research Society Conference, pp. 2-3, 2004

[13] N. Dalal and B Triggs. Histogram of oriented gradients for human detection. In Proc. CVPR, 2005.

[14] D.W. Ruck, S.K. Rogers, and M. Kabrisky, Feature Selection Using a Multi-Layer Perceptron, In J. Neural Network Comput., 2 (2), 1990, pp. 40-48.

[15] John, G. H. & Langley, P. (1995). Estimating continuous distributions in Bayesian classifiers. In Proceedings of the Eleventh Conference on Uncertainty in Articial Intelligence,(pp. 338{345), Montreal, Quebec. Morgan Kaufmann.