MAC 499 - Trabalho de Formatura Supervisionado
Sistema de Reconhecimento Automático de Placas de Veículos

Elcio Koiti Nakashima N.o USP: 3466110 e-mail: elcio at linux.ime.usp.br
Orientadora: Profa. Dra. Nina S. T. Hirata

IME - USP
6 de dezembro de 2004

Sumário

1 Introdução **1**

2 Fundamentos **2**
2.1 Reconhecimento de imagens 2
2.2 Limiarização 3
2.2.1 Método de Otsu 3
2.2.2 Método de Otsu local 4
2.3 Extração de características 6
2.4 Aprendizado computacional 7
2.4.1 Redes Neurais 8
2.4.2 Máquinas de suportes vetoriais 9

3 Sistema desenvolvido **11**
3.1 Módulo de localização de caracteres 12
3.2 Módulo de classificação 16

4 Experiência pessoal **17**

5 Agradecimentos **18**

Resumo

Este trabalho é baseado em uma iniciação científica orientada pela Professora Doutora Nina S. T. Hirata, com o objetivo de desenvolver um protótipo capaz de reconhecer caracteres contidos em placas de veículos utilizando técnicas de processamento de imagens, visão computacional e aprendizado de máquina.

1 Introdução

O grande volume de veículos trafegando nas grandes metrópoles, a necessidade de vigilância constante e o emprego comercial de automóveis, caminhões e ônibus, principalmente no Brasil, nos leva a procurar por soluções automatizadas que possam diminuir custos de operações envolvendo veículos automotores.

Uma atividade que pode ser automatizada é a identificação de veículos, utilizando as placas desses veículos. Em geral, essa atividade é desempenhada por pessoas, de forma não automática. Uma abordagem para esse problema é processar uma imagem de um veículo contendo a imagem da placa, capturada por uma câmera fotográfica ou de vídeo. O resultado desse processamento, devem ser os caracteres da placa.

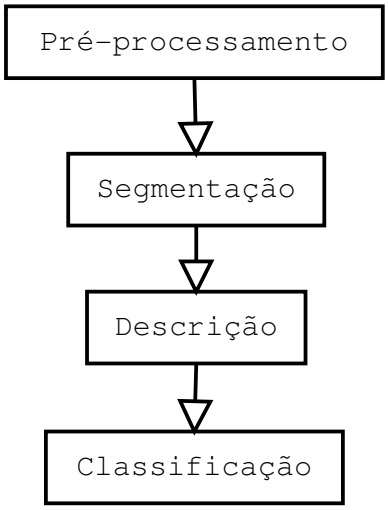

Figura 1: Processo de classificação de imagens

Existem diversas aplicações para sistemas deste tipo. Garagens, estacionamentos e condomínios podem ter um grande fluxo de entrada e saída de veículos, e uma solução automatizada poderia agilizar o controle de acesso a esses locais, e até mesmo diminuir a necessidade de interação humana nessa atividade. Com o maior número de câmeras espalhadas pelas grandes cidades e estradas é possível fazer a monitoração do tráfego de forma automática, com a possibilidade de agilizar o processo de identificação de infratores. Com essas mesmas câmeras é possível auxiliar na segurança urbana, possibilitando a localização de veículos roubados em qualquer ponto monitorado da cidade.

Esta monografia apresenta os fundamentos teóricos estudados com o objetivo de implementar um protótipo de um sistema de reconhecimento automático de placas de veículos. Além disso, são apresentados o sistema desenvolvido e os resultados obtidos até a presente data. A maior parte de todo o processo, desde o tratamento da imagem até a classificação de padrões, foi implementada para este projeto, com a mínima utilização de bibliotecas prontas.

2 Fundamentos

Esta seção contém os fundamentos teóricos básicos, de processamento e descrição de imagens e classificação supervisionada, utilizados no desenvolvimento do projeto.

2.1 Reconhecimento de imagens

O processamento de uma imagem em um sistema de visão computacional envolvendo reconhecimento de padrões pode ser dividido em quatro etapas realizadas em sequência: pré-processamento, segmentação, descrição e classificação. O processo completo está esquematizado na figura 1.

O pré-processamento envolve todas as operações com o objetivo de melhorar as características da imagem que serão utilizadas. Por exemplo, para aplicar operadores que utilizem formas, o pré-processamento deve remover o ruído que possa modificar a forma verdadeira dos objetos e transformar uma imagem colorida em uma imagem binária, contendo apenas as formas dos objetos da imagem original. Entre outras operações nesta etapa podem estar ajustes de contraste, equalização do histograma de cores, filtragem de frequências e restauração.

Na etapa de segmentação, objetos de interesse contidos na imagem devem ser localizados e separados, para que seja possível a análise desses objetos. O desempenho nesta etapa, provavelmente uma das mais difíceis em todo o processo, depende muito de boas escolhas na etapa de pré-processamento e da utilização de características das imagens do problema em questão. Exemplos de como esse conhecimento específico define o processo de segmentação serão dados nas seções que descrevem a implementação do sistema.

Na terceira etapa, de descrição, algumas características sobre o objeto são medidas, e essa informação deve ser utilizada pela etapa seguinte. Essas características são chamadas de atributos. A escolha de quais características utilizar é muito importante para o sucesso da fase de classificação.

A última etapa consiste em atribuir o objeto a uma classe, utilizando os atributos extraídos na etapa de descrição. No caso do sistema de reconhecimento de placas, ou de um sistema de reconhecimento de caracteres impressos, cada classe consiste em um dos possíveis caracteres que podem ser encontrados nesses domínios. Isso é realizado por um classificador estatístico, que através de um processo de treinamento supervisionado, aprende através de exemplos dados por um operador humano. A seleção de um número muito pequeno de características, na etapa de descrição, pode impossibilitar o sucesso do processo de classificação, pois essas poucas características podem não ser suficientes para diferenciar as classes de objetos. Um número muito grande de atributos pode dificultar o treinamento, pois com uma dimensão maior, um conjunto maior de exemplos de treinamento é necessário para se obter uma boa estimativa estatística. Além disso, temos o custo computacional relacionado à esta dimensão, envolvido tanto na classificação quanto no treinamento.

2.2 Limiarização

Seja F uma imagem bidimensional em escalas de cinza em que $f(x, y)$ representa a escala de cinza na coordenada (x, y). Tipicamente, a escala de cinza é representada pelos inteiros entre 0 e 255. Seja t, um valor permitido de escala de cinza que chamaremos de limiar e (b_0, b_1) um par de níveis de cinza. A imagem limiarizada é dada por:

$$g(x,y) = \begin{cases} \mathrm{b}_0 & \text{se } f(x,y) < t \\ \mathrm{b}_1 & \text{se } f(x,y) >= t \end{cases}$$

A função $g(x, y)$, que chamaremos de imagem binária, nos dá uma imagem G que possui apenas os dois níveis de cinza b_0 e b_1. Com isso, podemos aplicar sobre a imagem, diversos algoritmos de processamento de imagens binárias. O desafio em desenvolver um bom algoritmo de limiarização está no cálculo de um limiar que preserve as características da imagem que queremos analisar.

Os algoritmos de limiarização podem ser divididos em algoritmos locais e globais. Os algoritmos globais são aqueles que levam em consideração características da imagem inteira para calcular um valor de limiar, e aplicam esse limiar para gerar a imagem binária. As técnicas locais utilizam informações de regiões da imagem, gerando um limiar para cada uma delas. Diversos algoritmos e uma avaliação de seus desempenhos podem ser encontrados em [2].

2.2.1 Método de Otsu

O método de binarização de Otsu [4] é um método para cálculo automático de limiar, que consiste em separar os níveis de cinza em duas classes $C_0 = \{0, 1, \ldots, t\}$ e $C_1 = \{t + 1, \ldots, n\}$, onde n é o nível de cinza máximo, e uma das classes deve receber o nível de cinza de objeto e a outra o nível de cinza do fundo (para os quais, por convenção, são utilizados usualmente preto e branco, respectivamente). Seja p_i a frequência de pontos na imagem com o nível de cinza i. O limiar ótimo é calculado minimizando σ^2, que é a variância entre as classes C_0 e C_1 e que é calculada da seguinte forma:

$$\sigma^2 = \omega_0\omega_1(\mu_1\mu_0)^2,$$

Figura 2: Imagem original.

$$\omega_0 = \sum_{i=0}^{t} p_i, \qquad \omega_1 = 1 - \omega_0,$$

$$\mu_T = \sum_{i=0}^{n} ip_i, \qquad \mu_t = \sum_{i=0}^{t} ip_i, \qquad \mu_1 = \frac{\mu_T - \mu_t}{1 - \omega_0}, \qquad \mu_0 = \frac{\mu_t}{\omega_0}$$

2.2.2 Método de Otsu local

Os métodos de limiarização globais tem como desvantagem principal o fato de serem muito sensíveis em relação a áreas da imagem com variação nos tons de cinza muito maiores ou menores que o resto da imagem, que pode ser causado, por exemplo, por iluminação não uniforme. Ou seja, dependendo das cores da imagem original, o resultado da limiarização pode ser muito diferente do desejado. Isso pode ser corrigido através de duas formas: aplicando um pré-processamento na imagem ou utilizando um algoritmo local.

Esse problema é exemplificado pela figura 4, que é o resultado da limiarização da mesma imagem limiarizada anteriormente, porém com a árvore na diagonal direita superior coberta por uma mancha branca. Podemos notar diferenças entre detalhes que são perdidos na nova imagem. Apesar de artificial, esse exemplo demonstra como o algoritmo global pode ser sensível a características de objetos da imagem que não são necessariamente o que estamos estudando. As cores dos carros e do fundo, por exemplo, podem causar uma grande diferença no resultado da limiarização dos caracteres na placa.

Neste trabalho, devido ao problema acima citado, foi utilizado um algoritmo local como sugerido em [9]. Para cada ponto da imagem, devemos encontrar dentro de sua vizinhança os pontos com maior e menor intensidades de tons de cinza, denotadas respectivamente por $i_0(x, y)$ e $i_1(x, y)$. Seja t_0 um limiar pré-estabelecido. A imagem limiarizada é dada por:

$$l(x,y) = \begin{cases} \text{b}_0 & \text{se } |i_1(x,y) - i_0(x,y)| >= t_0 \text{ e } |f(x,y) - i_0(x,y)| <= |f(x,y) - i_1(x,y)| \\ \text{b}_1 & \text{se } |i_1(x,y) - i_0(x,y)| >= t_0 \text{ e } |f(x,y) - i_0(x,y)| > |f(x,y) - i_1(x,y)| \\ \text{g(x, y)} & \text{caso contrário} \end{cases}$$

Figura 3: Resultado da aplicação da binarização de Otsu.

Figura 4: Imagem binarizada por Otsu, em que um pedaço da imagem foi retirado manualmente. Pode-se perceber que esse pedaço em branco fez com que o algoritmo escurecesse outras partes da imagem.

Figura 5: Resultado da limiarização pelo Otsu modificado.

onde $f(x, y)$ é a intensidade de cor no ponto (x, y), $g(x, y)$ é calculado utilizando o método de Otsu e b_0 e b_1 são os valores de intensidade de cinza da imagem limiarizada.

A função $l(x, y)$ acima indica que caso a diferença entre a maior e a menor intensidade seja maior que o limiar t_0 dado, o valor de l será determinado da seguinte forma: caso o valor de intensidade no ponto esteja mais próximo da intensidade do ponto de maior intensidade, a imagem limiarizada deve receber nesse ponto o valor b_0, de maior intesidade. Caso esteja mais próximo de i_1 deve receber o valor b_1. Caso a diferença entre os vizinhos de maior e menor intensidade seja menor que o limiar dado, o valor da função é o mesmo que resultaria de uma limiarização utilizando o método de Otsu.

Na figura 5 temos um exemplo do resultado do algoritmo acima, utilizando $t_0 = 40$. Pode-se notar que muitos detalhes, como por exemplo os caracteres da placa, foram melhor preservados que na limiarização global.

2.3 Extração de características

A etapa de extração de características consiste em obter da imagem informações que possam descrevê-la. Em um problema de reconhecimento de padrões é importante que essas informações possibilitem a discriminação entre objetos pertencentes a cada uma das classes. Outro fator importante a ser considerado é a dimensionalidade do vetor de características, problema já mencionado.

A robustez do processo de classificação também é afetada por possíveis variações das imagens dos objetos segmentados. Por exemplo, os objetos podem ter tamanho variável, estar transladados ou apresentar inclinações. Esse problema pode ser resolvido através de métodos de extração de características que sejam invariantes a essas transformações. No caso de reconhecimento de caracteres, que se aplica a este trabalho, alguns dos caracteres são diferenciados justamente pelo ângulo de rotação, como '6' e '9', e portanto métodos invariantes a rotação não podem ser utilizados para diferenciar estas classes.

O método mais simples de extração de características é utilizar a própria imagem como vetor de características, conhecido como mapa de bits. O mapa de bits tem o problema de não apresentar nenhum tipo de invariância embutida no processo de extração, além de ser muito sensível a variabilidade na forma dos objetos. Isso pode

ser corrigido através de métodos de segmentação ou classificação que levem esses problemas em consideração. Outro problema é que um bom mapa de bits possui uma dimensão muito grande.

Outros atributos facilmente computados são as projeções horizontal e vertical, que são utilizados para imagens binárias. A projeção horizontal $y(x_i)$ é o número de *pixels* de objeto (*pixels* iguais a 1, por exemplo) que possuem a coordenada x igual a x_i. A projeção vertical é similar, porém para as coordenadas y. Esses atributos porém não são capazes de capturar muitas informações sobre alguns aspectos importantes da imagem, como a forma do objeto, e são normalmente utilizados com outros objetivos, como correção de ângulo e segmentação de imagens binárias. Mesmo assim, projeções e mapas de bits são algumas das características utilizadas em [5].

Atributos muito utilizados são os momentos invariantes da imagem, medidas que, teoricamente, tem a propriedade de permitir a reconstrução da imagem, dado um número suficientemente grande desses atributos. Além disso, esses atributos são calculados de forma que sejam invariantes a alguns tipos de transformações, existindo até momentos invariantes a transformações lineares em geral. Como exemplo, temos abaixo um exemplo de momento de ordem $(p + q)$ invariante a translação:

$$\mu_{pq} = \sum_{i=1}^{M} f(x_i, y_i)(x_i - \bar{x})^p (y_i - \bar{y})^q$$

onde o somatório é realizado sobre todos os M pontos da imagem, $f(x, y)$ é o nível de cinza do ponto (x, y), e $\bar{x}$ e $\bar{y}$ são dados abaixo:

$$\overline{x} = \frac{\sum_{i=1}^{M} f(x_i, y_i) x_i}{\sum_{i=1}^{M} f(x_i, y_i)}$$

$$\overline{y} = \frac{\sum_{i=1}^{M} f(x_i, y_i) y_i}{\sum_{i=1}^{M} f(x_i, y_i)}$$

Para encontrar mais detalhes sobre as características acima, e outras que também podem ser utilizadas com o mesmo propósito, consulte [1] e [3].

2.4 Aprendizado computacional

Na etapa de classificação de um sistema de reconhecimento de padrões, podemos aplicar modelos e algoritmos de aprendizado computacional. O processo de aprendizado supervisionado consiste basicamente em encontrar uma boa aproximação para uma função utilizando exemplos de entradas e saídas para esta função. Para o caso de imagens, os dados de entrada são atributos que extraímos para descrever a imagem, enquanto os dados de saída são as classes de objetos que desejamos reconhecer. O processo de aproximação da função utilizada é chamado de treinamento.

Por exemplo, uma máquina de aprendizado computacional poderia ser utilizada para classificar células em cancerígenas ou saudáveis, dadas diversas características dessas células. Para isso seria necessária uma grande quantidade de dados de exemplo, e para cada um destes exemplos, um rótulo indicando se é uma célula saudável ou não. O resultado do treinamento da máquina, utilizando esses dados, deve ser uma função capaz de classificar células nas duas classes consideradas, com o mínimo de erro possível.

Para o caso do sistema de reconhecimento de placas, os dados de entrada podem ser características como as citadas na seção sobre extração de características. Como dado de saída podemos utilizar o código do caracter

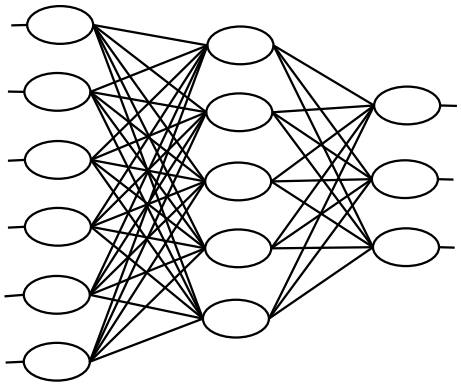

Figura 6: Rede neural artificial

representado por aquela figura.

2.4.1 Redes Neurais

Uma rede neural artificial é um modelo computacional inspirado no funcionamento dos neurônios no cérebro humano e é um dos métodos mais utilizados em aprendizado computacional. A unidade básica de uma rede neural é chamada de *perceptron*. Cada perceptron recebe um ou mais dados de entrada x, e possui pesos w para cada entrada e uma função de ativação $f(\cdot)$, que é aplicada sobre a soma dos valores ponderada pelos pesos ($\sum wx$). Esses perceptrons são interconectados, e os valores calculados são propagados, através dessas conexões, como entradas em outros neurônios. O processo de aprendizado consiste em ajustar os pesos w de forma que o erro cometido pela rede neural seja minimizado.

Existem diversos tipos de redes neurais, em que variam a forma como os perceptrons são interconectados, a função de ativação ou a função de erro. Neste trabalho estamos utilizando redes de múltiplas camadas de perceptrons (*multilayer perceptron*), em que a rede é formada de camadas de perceptrons, em que cada perceptron nas camadas seguintes à camada de entrada recebe como dados de entrada as saídas de cada um dos perceptrons da camada anterior, como esquematizado na figura 6.

O treinamento da rede neural é realizado através do algoritmo de retro-propagação de erro. A função de ativação utilizada é a sigmóide $f(x) = \frac{1}{1+e^{-x}}$. Seja a função de erro $E = \frac{1}{2}(y - f(\sum wx))^2$ em que (x, y) é um exemplo (ou instância de treinamento), x os dados de entrada e y é a saída esperada para um exemplo dado. A idéia básica, para um perceptron na camada de saída(a última camada da rede), é ajustar os pesos para compensar o erro gerado para cada instância de exemplo. Na camada de saída, o ajuste do peso $w_{j,i}$ (peso da conexão entre a saída do perceptron j e entrada do perceptron i) da entrada a_j (saída do perceptron j) pode ser feito utilizando a seguinte fórmula:

$$w_{j,i} = w_{j,i} + \alpha \frac{\partial E}{\partial w_{j,i}} = w_{j,i} + \alpha * Erro * f'(entrada_j) * a_j = w_{j,i} + \alpha * a_j * \Delta_j$$

onde $\Delta_j = Erro * f'(entrada_j)$, $entrada_j = \sum w_{j,k} a_k$, $Erro = (y - f(entrada_i))$ e α é a taxa de aprendizado, e controla a magnitude da influência do ajuste sobre o erro.

Para as outras camadas não temos como medir o erro utilizando um valor y esperado. Vamos utilizar então a contribuição para os erros calculados na camada seguinte, ponderado pelos pesos das conexões entre o perceptron atual e os da camada seguinte. Para isso vamos utilizar para as outras camadas:

$$\Delta_j = f'(entrada_j) * \sum_i w_{j,i} \Delta_i$$

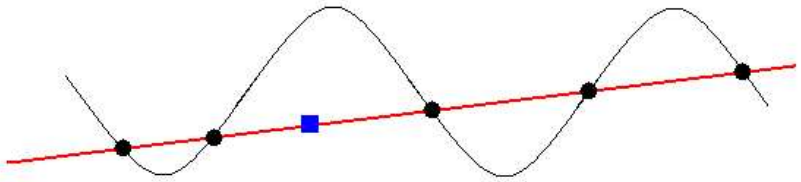

Figura 7: Exemplo de *overfitting*. A linha vermelha representa a função real que queremos aprender e a linha preta a função aprendida através dos exemplos representados pelas bolas pretas. Como podemos ver, a função encontrada não é capaz de classificar corretamente o quadrado azul, pertencente à função original

As atualizações dos pesos da rede são aplicados para cada instância de treinamento e até que se atinja algum critério. Podemos utilizar como critério o número de iterações, o tempo de processamento ou uma certa taxa de erros.

Não existe uma teoria ou metodologia formalizada para a determinação de parâmetros da rede neural, como o número de camadas ou a taxa de aprendizado. Esses parâmetros são usualmente definidos através de heurísticas e experimentos. Mais detalhes sobre redes neurais podem ser encontrados em [6] ou [8].

2.4.2 Máquinas de suportes vetoriais

A estratégia de aprendizado utilizada pelas redes neurais artificiais consiste em fixar uma estrutura e minimizar o erro de treinamento. O maior problema desta abordagem é que uma máquina que tenha capacidade de aprender funções muito complexas, com uma quantidade não suficiente de instâncias de treinamento, pode apresentar um fenômeno conhecido como *overfitting*. Este fenômeno ocorre quando muitos dos pontos, representados pelas instâncias de treinamento, são corretamente classificados pela função discriminante, porém esta classificação correta seja devido à grande capacidade de representação da função utilizada. Esta função, por ser mais complexa que a função original que queremos aprender, não é capaz de fazer boas classificações em relação a pontos muito diferentes dos pontos contidos no conjunto de treinamento. Uma ilustração deste conceito pode ser visto na figura 7

Uma máquina de suportes vetoriais é um modelo de máquina de aprendizado desenvolvido há poucos anos, em que a estratégia de aprendizado utilizada é diferente das redes neurais, ou seja, o erro de treinamento é mantido fixo e minimizamos um intervalo de confiança. Esse intervalo de confiança é chamado de confiança VC (Vapnik-Chervonenkis), e está relacionado ao erro devido à estrutura das funções contidas no espaço de busca da máquina utilizada.

Um conjunto de dados linearmente separável é um conjunto de dados de forma que existe uma função linear $f(x) = \sum w_i x_{ii} + b$, tal que $f(x) > 0$ para todo $x \in C_1$ e $f(x) < 0$ para todo $x \in C_2$. C_1 e C_2 são os dois conjuntos em que queremos realizar a classificação. Apenas por motivos de simplificação, vamos nos ater a dados linearmente separáveis, e o problema de classificação com duas classes.

Para um conjunto de pontos linearmente separáveis, representantes dos dados de treinamento, podemos ter diversos hiperplanos que separam corretamente esses dados nas duas classes consideradas. Para obter maior generalização do classificador, vamos procurar por um hiperplano que possua a maior margem. A margem é o dobro da distância entre o hiperplano de classificação e os pontos mais próximos, como ilustrado na figura 8.

Esses pontos que tocam a margem são chamados de vetores de suporte.

Com o objetivo de facilitar a busca por uma função linear que possua a margem máxima, vamos conside-

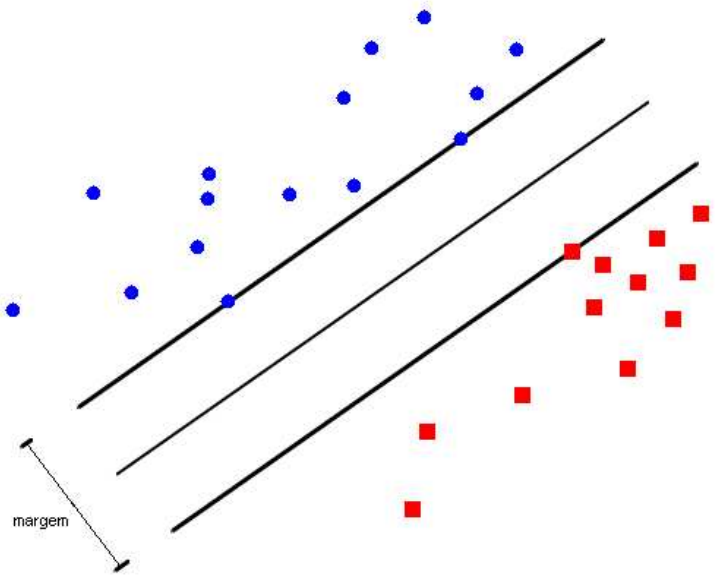

Figura 8: Margem e hiperplano separador para duas classes, representadas pelas bolas azuis e quadrados vermelhos.

rar apenas hiperplanos canônicos. Um hiperplano é dito canônico, para um dado conjunto de treinamento, se $|w^t x + b| = 1$ para todo x pertencente ao conjunto de vetores de suporte.

A distância D de um ponto x_1 para um hiperplano $w^t x + b$ é dado por:

$$D = \frac{|w^t x_1 + b|}{||w||}$$

Seja M a margem, x_1 um vetor de suporte e D a distância desse vetor para o hiperplano de separação. Sabemos que essa distância é dada por:

$$D = \frac{M}{2} = \frac{|w^t x_1 + b|}{||w||}$$

Se considerarmos x_1 um vetor de surporte, temos:

$$\frac{M}{2} = \frac{|w^t x_1 + b|}{||w||} = \frac{1}{||w||} \rightarrow M = \frac{2}{||w||}$$

O que nos leva a conclusão de que para maximizar a margem M, precisamos minimizar $||w|| = \sqrt{w^t w}$. Como a função $\sqrt{\cdot}$ é monotônica, podemos minimizar $w^t w$.

Seja l o número de instâncias de treinamento. Se considerarmos que para cada instância de treinamento x temos que $y = 1$ quando $x \in C_1$ e $y = -1$ quando $x \in C_2$ então para um hiperplano canônico temos as seguintes desigualdades:

$$y_i(w^t x_i + b) \geqq 1,\ i = 1, \ldots, l$$

Temos então um problema de otimização não linear, que pode ser resolvido através da função de Lagrange abaixo:

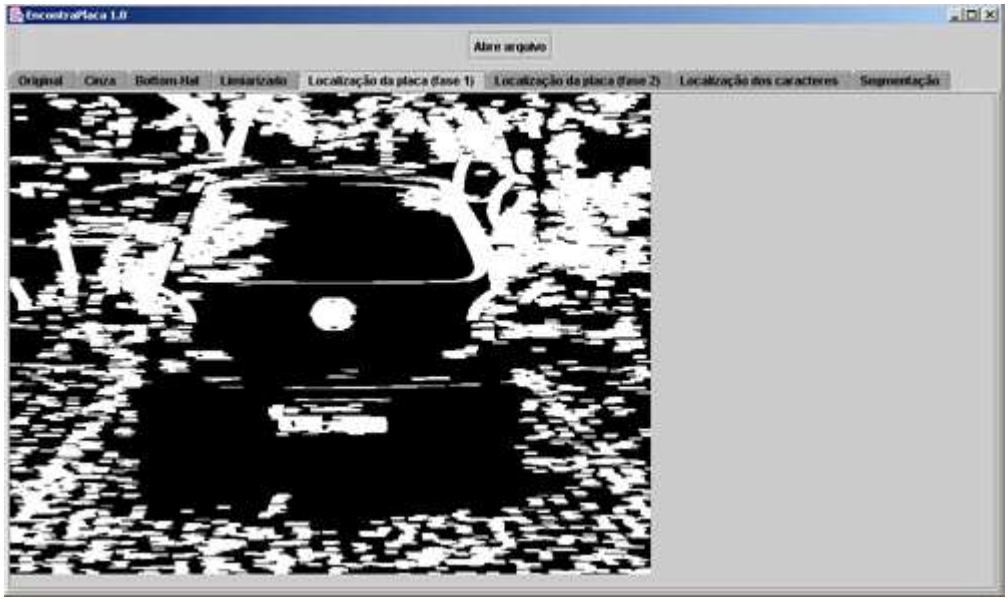

Figura 9: Uma das telas do programa de localização de caracteres, que faz parte do protótipo desenvolvido.

$$L(w, b, \alpha) = \frac{w^t w}{2} - \sum_{i=1}^{l} (\alpha_i (y_i (w^t x_i + b) - 1))$$

Na expressão acima, α é o vetor de multiplicadores de Lagrange.

Igualando a zero as derivadas em relação aos pesos w e a b, podemos reescrever a função de Lagrange em relação aos multiplicadores de Lagrange. Com isso temos a seguinte formulação de um problema de programação quadrática:

Maximizar $L(\alpha) = -\frac{1}{2}\alpha^t H \alpha + I^t \alpha$

Sujeito a $y^t \alpha = 0, \alpha \geqq 0$

Onde $H_{ij} = y_i y_j (x_i^t x_j)$ e $I = [1, 1, \ldots, 1]^t$.

Com isso, o aprendizado neste modelo se resume a resolver o problema acima. Um modo de resolver este problema, é utilizar algum algoritmo de programação quadrática genérico. Porém, é possível utilizar algoritmos que aproveitem a estrutura especial do problema acima. Um exemplo de algoritmo desse tipo pode ser encontrado em [11]. Implementações desse algoritmo podem ser encontradas em [10] e [12].

Para mais detalhes sobre máquinas de suportes vetoriais, incluindo extensões do modelo para o caso não linear e para o caso em que os dados são não separáveis, podem ser encontrados em [8].

3 Sistema desenvolvido

O protótipo foi desenvolvido na linguagem Java, e é dividido em dois módulos: localização de caracteres e classificação. O módulo de localização envolve as etapas de pré-processamento, localização da placa e localização e segmentação dos caracteres. O módulo de classificação envolve as etapas de descrição e classificação. Os resultados obtidos em ambos os módulos são descritos em cada uma das subseções. Algumas telas de um dos programas desenvolvidos podem ser vistas nas figuras 9 e 10.

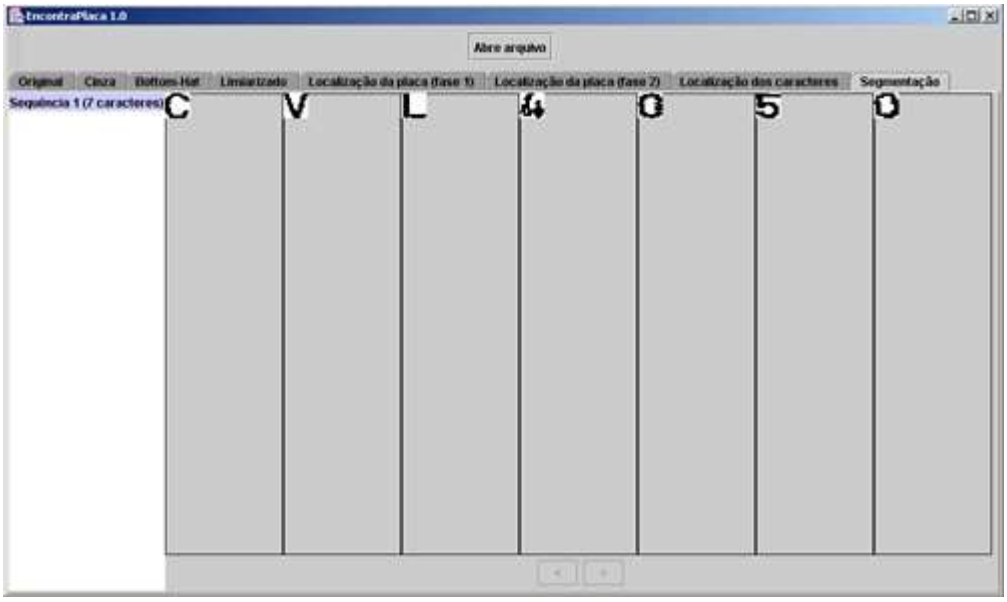

Figura 10: Resultado da segmentação no programa de localização de caracteres.

Foram utilizadas 489 fotos de carros, capturadas para este trabalho ou coletadas na internet. Além dessas, foi coletada uma quantidade maior de fotos, principalmente na internet, que foram descartadas por não atenderem a algumas condições mínimas de qualidade (conforme descrito abaixo, na descrição do módulo de localização de caracteres). Para os testes foram utilizadas 94 fotos.

3.1 Módulo de localização de caracteres

O módulo de localização de caracteres é composto por diversos submódulos, que correspondem às etapas realizadas em sequência sobre a imagem de entrada, com o objetivo de localizar e segmentar os caracteres da placa.

O primeiro passo do processo é transformar a imagem de entrada(figura 11) em uma imagem apenas em tons de cinza (figura 12), para que possamos aplicar um operador de morfologia matemática chamado *bottom-hat* (ou black top-hat). A imagem resultante dessa operação é formada pelos detalhes escuros da imagem em tons de cinza, que são suficientemente finos, como podemos ver na figura 13. Dessa forma, os caracteres são alguns dos objetos que aparecem nessa imagem. Mais detalhes sobre morfologia matemática podem ser encontrados em [7]. O passo seguinte é a limiarização da imagem resultante do *bottom-hat*, utilizando o algoritmo local descrito na seção sobre limiarização(figura 14).

A localização da placa é feita em duas etapas. Primeiro, conforme [10], uma dilatação utilizando um elemento estruturante horizontal mais comprido que a distância entre dois caracteres é realizada (figura 15). Com isso a região da placa, onde estão destacados os caracteres, se transformará em uma região retangular. Para encontrar regiões onde existam candidatos a placa, é aplicado um algoritmo de rotulamento de componentes conexos, e são selecionados apenas aqueles que possuem certos atributos que podem caracterizar uma placa, como a altura, a largura e a razão entre a altura e largura. Desse processamento podem resultar alguns candidatos a placa (figura 16).

Para eliminar falsos candidatos a placa, a rotulação sobre a imagem original limiarizada procura por objetos que tenham atributos de altura, largura e razão entre altura e largura semelhantes a caracteres (figura 17). Desses objetos, o algoritmo seleciona apenas aqueles que estão lado a lado, e em número suficiente para formar os caracteres de uma placa. Caso alguma dessas fileiras de possíveis caracteres tenham uma posição coincidente

Figura 11: Exemplo de imagem de entrada.

Figura 12: Imagem em tons de cinza.

Figura 13: Resultado do bottom-hat.

Figura 14: Figura 13 depois da limiarização.

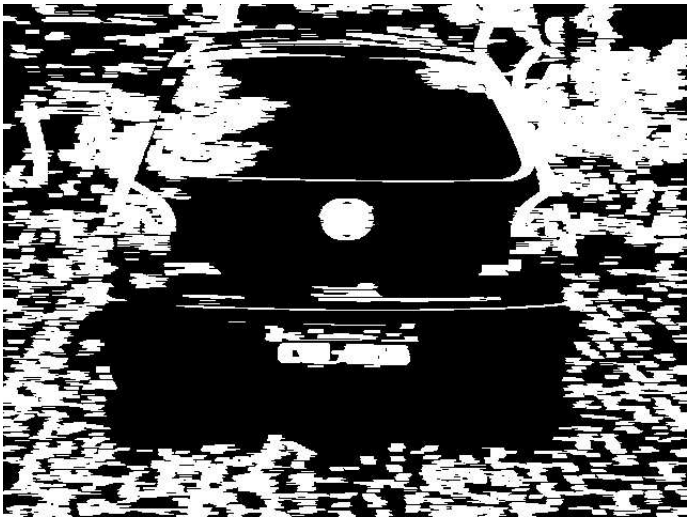

Figura 15: Resultado da dilatação.

Figura 16: Os retângulos vermelhos são os candidatos a placa encontrados.

Figura 17: Candidatos a caracteres.

Figura 18: Caracteres segmentados.

com os candidatos a placa, esta fileira é considerada como o conjunto de uma possível placa.

O resultado de todo esse processo é uma coleção de candidatos a placa, com os possíveis caracteres já segmentados. O algoritmo permite que os candidatos a placa possuam mais candidatos a caracter do que o número de caracteres esperado para uma placa, pois a aplicação do *bottom-hat* e da limiarização pode gerar alguns candidatos falsos a partir das bordas da placa.

Atualmente, o módulo de localização de caracteres possui algumas limitações. Caracteres grudados entre si, ou grudados na borda da placa não são localizados pelo algoritmo acima. Isso também acontece com caracteres muito degradados (devido principalmente à má conservação). Outra limitação acontece devido a alguns passos do algoritmo que dependem da distância entre objetos na cena. A dilatação, por exemplo, necessita de um elemento estruturante de comprimento maior que a distância entre os caracteres. O operador *bottom-hat* precisa ter um elemento estruturante de tamanho maior que a espessura dos caracteres. Com isso, o algoritmo não funcionaria para veículos muito próximos ou muito longe do equipamento de captura de imagens.

A taxa de segmentação para o conjunto de testes é de 87,2 %.

O módulo de segmentação pode ser melhorado de diversas formas. Uma delas é utilizar técnicas que combinam diversas estratégias de segmentação. Outra melhoria possível pode ser feita nos casos em que o fundo é praticamente fixo, sendo dessa forma mais fácil de se localizar o carro e a placa.

3.2 Módulo de classificação

O módulo de classificação engloba as etapas de extração de características e classificação. É composto por programas utilizados no treinamento de classificadores e no módulo do protótipo, que utiliza esses classificadores.

Os programas de treinamento recebem os dados das instâncias de treinamento, extraídos por outro programa de extração de características, e gravam o classificador gerado em um arquivo em disco. Esse arquivo deve então ser utilizado pelo protótipo para realizar as classificações.

Além de realizar a classificação propriamente dita, o programa precisa decidir quais candidatos a caractere não são caracteres, pois o módulo de localização de caracteres pode gerar como resultado sequências com mais candidatos que o número esperado de letras e números. Além disso, pode também gerar mais de uma sequência de candidatos. Para resolver este problema, a saída dos classificadores utilizados deve ter alguma relação com a probabilidade daquelas características pertencerem à cada classe, ou seja, o valor do classificador para cada classe deve ser alto quando há uma probabilidade alta daquela instância pertencer a essa classe. Com isso, quando temos uma sequência com mais candidatos a caractere do que o esperado para uma placa, podemos utilizar a saída do classificador para escolher apenas os candidatos com maior probabilidade de serem caracteres. Também podemos selecionar a sequência que tem a maior probabilidade de ser formada por caracteres, entre todas as sequências candidatas fornecidas pelo módulo de localização de caracteres.

Visando aumentar a taxa de acerto, o algoritmo aproveita a estrutura fixa dos caracteres nas placas brasileiras, ou seja, três letras seguidas por quatro algarismos. Para isso, o módulo utiliza dois classificadores, um que considera apenas letras e outro que considera apenas algarismos.

Os primeiros experimentos foram realizados utilizando como vetor de características, o mapa de bits das figuras segmentadas, com 25 *pixels* de altura por 25 *pixels* de largura. Foram realizados os treinamentos para redes neurais e máquinas de suportes vetoriais. Para as imagens segmentadas com sucesso, a rede neural conseguiu uma taxa de acerto de 69,5 %. As máquinas de suportes vetoriais atingiram uma taxa de sucesso de 82,9 %.

Um segundo experimento foi realizado utilizando como características, o número de buracos no caracter segmentado, os momentos geométricos invariantes a escala e translação até o grau 6, e o mapa de bits da figura escalada para dimensão de 4 *pixels* de altura por 4 *pixels* de largura. Apesar de reconhecer até 83 % dos caracteres no conjunto de teste, a máquina de suportes vetoriais gerada só reconheceu 48,7 % das placas. A rede neural reconheceu 52,4 % das placas.

Um experimento utilizando os momentos até o grau 21 e o mapa de bits de 10 x 10, atingiu sucesso de reconhecimento para 94,4 % dos caracteres e 70,7 % das placas bem segmentadas para uma máquina de suportes vetoriais.

Uma grande limitação encontrada no desenvolvimento deste módulo foi o fato do conjunto de treinamento não ser grande o suficiente para que os classificadores pudessem aprender de forma satisfatória alguns dos caracteres. O conjunto de treinamento é composto por 2649 caracteres (letras e números) e alguns dos caracteres aparecem com uma frequência muito baixa, como o u (10 instâncias) e o y (12 instâncias). Por outro lado, os números, que aparecem com frequências muito maiores (todas maiores que 100), tem alta taxa de reconhecimento pelos classificadores gerados. Mesmo assim foi possível gerar um classificador com uma boa taxa de reconhecimento (82,9 %).

4 Experiência pessoal

Escolhi fazer este trabalho como projeto de formatura por dois motivos principais. Primeiro por ser um projeto desafiador, o que me motiva bastante. O segundo motivo é que eu queria entrar em contato com uma das áreas de pesquisa do IME que eu não tive oportunidade de conhecer através das disciplinas do curso, o que ocorreu principalmente por incompatibilidade de horários. Posso dizer que esta foi provavelmente minha maior frustração relacionada a este projeto. Por este fato, este projeto exigiu de mim muita disciplina para estudar diversos assuntos por conta própria, o que me acostumei a fazer bastante nos últimos anos.

Para melhorar meu aprendizado, decidi que iria implementar a maioria dos algoritmos de processamento de imagens e de aprendizado computacional. Apesar de não me arrepender dessa decisão, acho que se tivesse escolhido utilizar ferramentas ou bibliotecas prontas, poderia me concentrar em resultados na resolução do pro-

blema em questão e talvez com isso obtido resultados melhores. Uma biblioteca pronta que precisei utilizar foi a *LIBSVM* [10], implementação de uma máquina de suportes vetoriais. Por um lado utilizá-la facilitou muito meu trabalho, pois meus prazos estavam quase estourando. Porém foi uma frustração não poder desenvolver minha própria implementação, assim como fiz para as redes neurais.

Diversas disciplinas foram relevantes de alguma forma para este trabalho. Para a implementação dos algoritmos e modelagem da aplicação, tiveram influência sobre o trabalho as disciplinas Princípios de Desenvolvimento de Algoritmos(MAC 122), Estruturas de Dados(MAC 323), Programação Orientada a Objetos(MAC 441) e Tópicos de Programação Orientada a Objetos(MAC 413). Para uma boa compreensão dos tópicos de aprendizado de máquina, foram importantes as disciplinas Probabilidade e Estatística I e II, Álgebra Linear e Cálculo.

Existem algumas disciplinas não cursadas por mim, que estão relacionadas com aspectos deste trabalho, e portanto poderiam provavelmente ajudar de alguma forma. Entre elas posso citar Visão e Processamento de Imagens(MAC 417), Aprendizado Computacional e Programação Não-linear(MAC 427).

Apesar de meu contato com minha orientadora ter se dado em maior parte através de e-mails, considero que a professora Nina me ajudou muito neste trabalho, contribuindo com idéias e sugestões de como melhorar o trabalho. Por ter trabalhado sozinho neste projeto, não senti nenhuma diferença significativa em relação à forma de trabalho na iniciação científica e dos exercícios-programa do curso. A maior diferença foi a necessidade de estudo de tópicos relacionados.

Acho que o desenvolvimento deste projeto é um bom exemplo de conceitos teóricos sendo utilizados em uma aplicação prática. Sem um bom entendimento da base teórica exigida para realizar este trabalho, provavelmente teria problemas na implementação de algumas partes do protótipo, e na adaptação desses algoritmos às minhas necessidades. Mesmo com programas prontos e funcionando, aplicações de processamento de imagens e aprendizado de máquina exigem um certo nível de conhecimento para que sejam utilizadas como ferramentas.

Por último, gostaria de tecer alguns comentários em relação ao curso do BCC. Quando ingressei no curso, vindo de um curso técnico de Processamento de Dados realizado na Escola Técnica Federal de São Paulo, confesso que pensava que o BCC não iria me acrescentar muito ao que eu havia aprendido. Fico muito feliz de constatar que estava profundamente equivocado. Graças ao curso pude conhecer um mundo que eu não tinha noção que existia, a Ciência da Computação (algo muito diferente de apenas saber programar ou configurar um computador). Fico apenas triste de não ter me dado conta mais cedo, para que pudesse aproveitar mais, realizando mais iniciações científicas, por exemplo.

5 Agradecimentos

Gostaria de agradecer à professora Nina por toda a ajuda prestada durante a realização deste trabalho. Também gostaria de agradecer ao meu colega Fábio Pisaruk, por me ajudar na coleta das fotos dos carros.

Referências

[1] Gonzalez, R. C., Woods, R. E. *Digital Image Processing*. 2nd. ed. Addison-Wesley, 1992.

[2] Sahoo, P. K., Soltani, S., Wong K. C., Chen, Y. C. *A Survey of Thresholding Techniques*. Computer Vision, *Graphics and Image Processing*, volume 41, p. 233-260, 1988.

[3] Trier, O. D., Jain, A. K., Taxt, T. *Feature Extraction Methods for Character Recognition - A Survey*. 1995.

[4] Otsu, N. *A Threshold Selection Method from Gray-Level Histograms. IEEE Transactions on Systems, Science, and Cybernetics*, volume SMC-8, p. 62-66, 1978.

[5] Guingo, B. C. *Reconhecimento Automático de Placas de Veículos Automotores*. Dissertação de Mestrado, UFRJ, 2003.

[6] Bishop, C. M. *Neural Networks for Pattern Recognition*. Clarendon Press - Oxford, 1995.

[7] Dougherty, E. R., Lotufo, R. A. *Hands-on Morphological Image Processing*. SPIE Press, 2003.

[8] Kecman, V. *Learning and Soft Computing*. MIT Press, 2001.

[9] Parker, J. R., Fedeerl, P. *An Approach To License Plate Recognition.* <http://pharos.cpsc.ucalgary.ca/Dienst/Repository/2.0/Body/ncstrl.ucalgary$_cs$/1996 − 591 − 11/*pdf* > $Martn, F., Borges, D. Automatic\ Car\ Plate\ Recognition\ Using\ a\ Partial\ Segmentation\ Algorithm. < http : //www.gpi.tsc.uvigo.es/pub/papers/pfc\_dborges.pdf >$

[10] Chang, C. C., Lin, C. J. *LIBSVM: a Library for Support Vector Machines*. <http://www.csie.ntu.edu.tw/ cjlin/papers/libsvm.pdf>

[11] Platt, J. C. *Fast Training of Support Vector Machines using Sequential Minimal Optimization.* <http://research.microsoft.com/users/jplatt/smo.html>

[12] Ge, X. *Sequential Minimal Optimization for SVM*. <http://www.datalab.uci.edu/people/xge/svm/>