Universidade de São Paulo

Instituto de Matemática e Estatística

Departamento de Ciência da Computação

Combinação de classificadores

Trabalho de formatura supervisionado

Adolpho Pacheco
Camila Pacheco

Orientadora: Nina S. T. Hirata

Introdução

Técnicas de processamento de imagens podem ser utilizadas para diversas aplicações, como por exemplo, na área médica (em raios-X, tomografias e ressonâncias magnéticas), em aplicações militares (para reconhecimento de alvos) ou na previsão do tempo.

Qualquer operação de manipulação, transformação ou extração de informações das imagens é denominada genericamente de processamento de imagem. A área de pesquisa denominada de Processamento Digital de Imagens estuda técnicas de processamento por computador.

O tratamento de imagens tem principalmente duas finalidades: melhoria na qualidade visual (estética) e extração de informações tais como medidas, estatísticas e classificações de objetos.

Uma das técnicas que vem sendo bastante utilizada no processamento de imagens utiliza a Morfologia Matemática. O estudo morfológico concentra-se na estrutura geométrica das imagens. Morfologia é a forma e estrutura de um objeto ou os arranjos e inter-relacionamentos entre as partes de um objeto.

A base da morfologia consiste em extrair de uma imagem desconhecida a sua geometria através da utilização da transformação de uma outra imagem completamente definida, ou seja, consiste em extrair as informações relativas a geometria e a topologia de um conjunto desconhecido (no caso uma imagem) pela transformação através de outro conjunto bem-definido, chamado elemento estruturante. Com isso torna-se importante ao contexto a utilização de teoria dos conjuntos, pois esta é a base utilizada na morfologia.

Uma imagem digital pode ser definida por um conjunto de pixels (picture elements), sendo que cada um possui um valor associado que representa uma cor na imagem. A quantidade de cores possíveis numa imagem é determinada pelo tamanho em bits desse valor. Numa imagem em preto e branco (binária) a cor de cada pixel é armazenada em um bit. Numa imagem em tons de cinza a cor é armazenada em um byte, totalizando 256 possíveis tons de cinza. Numa imagem colorida, tipicamente, a cor é representada por um conjunto de três bytes que guardam o tom vermelho, verde e azul de cada ponto.

Dependendo do tipo de imagem sendo processada (preto e branco, tons de cinza ou colorida) a definição das operações da morfologia muda, assim cada tipo deve ser considerado separadamente. Neste trabalho, estamos considerando apenas imagens binárias.

Um operador morfológico binário, ou classificador, é uma função que pode ser interpretada como uma transformação de imagens binárias. Uma técnica bastante comum é a utilização de algoritmos de aprendizado que, a partir de exemplos de treinamento, são capazes de inferir classificadores que são utilizados para identificar alguns padrões. Em geral, vários classificadores são projetados e aquele com melhor desempenho empírico é escolhido.

Este trabalho visa experimentar técnicas de combinação de classificadores no projeto de operadores morfológicos possibilitando melhor capacidade de generalização e resultados mais estáveis que os classificadores individuais, aproveitando as informações fornecidas por todos os classificadores individuais.

O método utilizado denomina-se *stacked generalization*. S*tacked generalization* refere-se a qualquer técnica de aprendizado que utiliza mais de um nível de treinamento. O nível 0 utiliza exclusivamente os dados de entrada, enquanto os níveis superiores podem utilizar resultados de níveis abaixo deles.

Durante o projeto, foi feita uma implementação desse método com dois níveis de treinamento, em linguagem C no sistema Khoros, um ambiente integrado para o desenvolvimento de software voltado para processamento de informações com visualização dos resultados, muito propício para o processamento de imagens.

Fundamentos Teóricos

Operadores de imagens

A Morfologia Matemática é uma metodologia para a descrição de operadores. Ela representa um quadro muito amplo para o estudo e desenvolvimento de algoritmos de análise de imagens. Conciliando uma formulação matemática elegante com um amplo espectro de aplicações práticas, a Morfologia Matemática é uma das mais promissoras metodologias para análise de imagens.

Vamos considerar que as imagens encontram-se definidas sobre um conjunto não vazio $E \subseteq \mathbb{Z}^2$, isto é, $E = E_1 \times E_2$, onde $E_1, E_2 \subseteq \mathbb{Z}$. As imagens binárias são representadas por funções do tipo $f: E \rightarrow \{0, 1\}$.

Uma função $\Psi : P(E) \rightarrow P(E)$ será denominada operador morfológico binário, ou simplesmente operador, e pode ser interpretada como uma transformação de imagens binárias.

Toda a teoria se fundamenta no uso de elementos estruturantes, os quais são caracterizados como conjuntos definidos e conhecidos (forma e tamanho), que são comparados ao conjunto desconhecido da imagem. Representam o tipo e a natureza da informação a ser extraída da imagem.

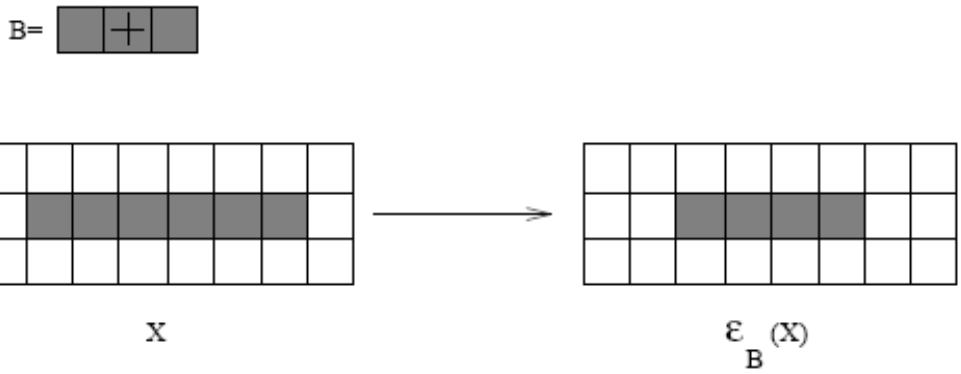
A Morfologia Matemática é constituída a partir de dois operadores básicos, a erosão e a dilatação, que são suficientes para representar qualquer outro operador.

O procedimento de erosão consiste em verificar se, para cada pixel branco, existe um número de vizinhos brancos menor que um limiar, *N*; caso exista tal pixel é invertido. Este procedimento elimina objetos finos ou pequenos e objetos maiores tem sua área reduzida.

A dilatação, por sua vez, realiza a operação inversa, verificando se o número de vizinhos brancos excede o limiar *N*, quando então o pixel é invertido. Este procedimento elimina buracos finos ou pequenos, unindo objetos. Os objetos, por sua vez, têm sua área aumentada.

Sejam $X, B \subseteq E$. A **erosão** de X por B (elemento estruturante) é definida por:

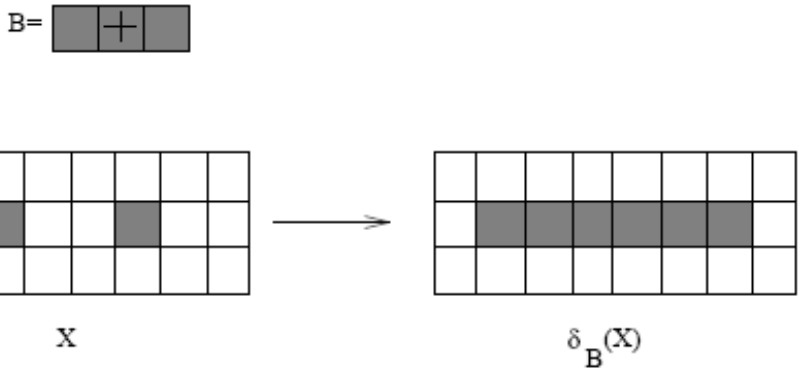
$$\varepsilon_B(X) = \{ x \in E: (B + x) \subseteq X \}.$$



Erosão de X por B. (O sinal + indica a origem de B)

Sejam X, B $\subseteq$ E. A **dilatação** de X por B é definida por:

$$\delta_B(X) = \{ x \in E: (B^t + x) \cap X \neq \varnothing \}.$$



Dilatação de X por B. (O sinal + indica a origem de B)

Estas operações por si só, na maioria das vezes, causam distorções nas áreas dos objetos. No entanto, sua combinação gera resultados muito mais interessantes.

Projeto de operadores

A característica de decomposição dos operadores morfológicos em termos de operadores elementares introduziu mais uma área de pesquisa na Morfologia Matemática, relacionada ao projeto de operadores.

Seja W $\subseteq$ Z^2 uma janela com n pontos. Os padrões (distribuição dos pontos) observados na janela conforme ela se move numa imagem podem ser vistos como elementos de P(W). Então, funções da forma $\psi : P(W) \rightarrow \{0, 1\}$podem ser usadas para definir um operador.

Se associarmos uma variável booleana a cada ponto da janela, então a função $\psi : P(W) \rightarrow \{0, 1\}$ pode ser vista como uma função booleana, isto é, um mapeamento de $\{0, 1\}^n$ em $\{0, 1\}$. As funções booleanas, por sua vez, podem ser vistas como classificadores de padrões binários.

O projeto de um operador deve ser compatível com os requisitos da aplicação (densidade de dados, tempo de resposta aceitável, custo final, etc.), assim como com os requisitos dos processos de desenvolvimento (tempo e metodologia de desenvolvimento, tecnologia de implementação física, etc).

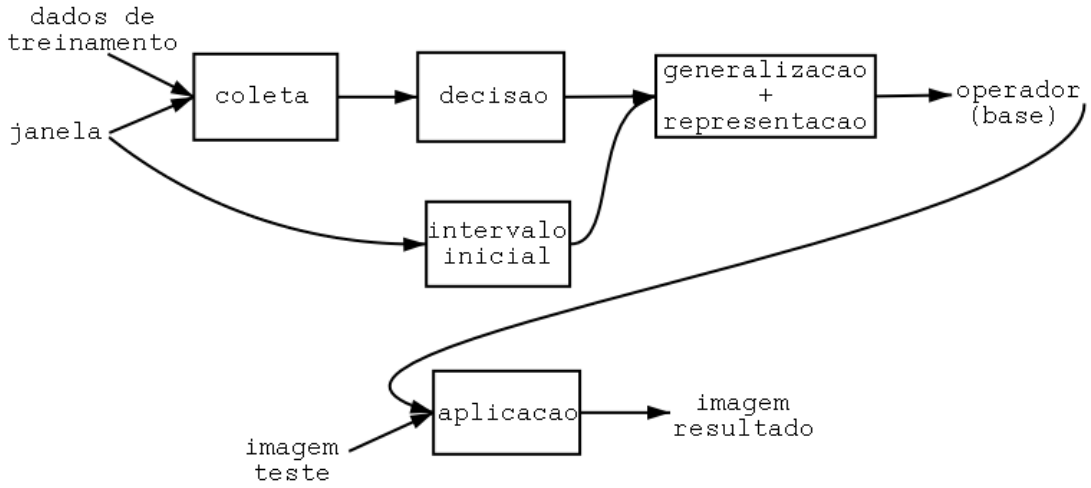
Uma técnica bastante utilizada em projeto de operadores é a aprendizagem computacional. Com isso, um operador é gerado automaticamente através de imagens de exemplo.

Informalmente, qualquer processo capaz de "aprender um conceito" é denominado de "aprendizado". O objetivo de um processo de aprendizado é produzir um conceito que seja uma boa aproximação de um conceito alvo, que queremos aprender.

O modelo estudado e utilizado neste projeto, como base para a implementação do *stacked generalization,* foi o modelo PAC (Probably Approximately Correct), que é um modelo de aprendizado que considera apenas os algoritmos que satisfazem um determinado critério de qualidade.

É um algoritmo que produz uma hipótese, tal que, com alta probabilidade, o erro entre a hipótese e o conceito alvo seja menor que um certo parâmetro ε pré-determinado. Além disso, um algoritmo é PAC se garante que quanto maior a amostra de treinamento, menor o erro da hipótese inferida. Esse modelo permite considerarmos apenas os algoritmos de aprendizado que, ao receberem mais exemplos, acabam criando uma hipótese melhor (com erro menor).

O aprendizado tem a seguinte estrutura:



Coleta:

São dados como entrada pares de imagens representando respectivamente as imagens anterior e posterior à transformação desejada. A coleta é feita por meio de uma janela que percorre a imagem anterior, e associa o padrão dos pontos (configuração) dessa janela ao valor do pixel na imagem ideal, correspondente ao centro da janela. A partir dessa coleta, é gerada uma tabela de padrões e suas freqüências (números de 0´s e 1´s observados na imagem ideal).

Decisão:

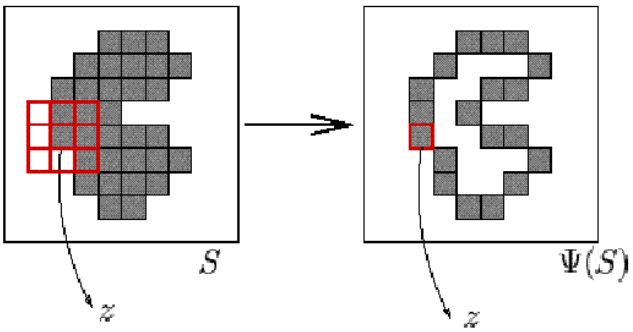
A partir dessa tabela é escolhido, para cada padrão observado, o valor (0 ou 1) que possui a maior freqüência.

Generalização:

Nem todos os possíveis padrões são observados na coleta de dados. Assim, algoritmos de aprendizado podem ser utilizados para associar valores (0 ou 1) a estes padrões. Um algoritmo tem capacidade de boa generalização se é capaz de associar a classificação correta a esses padrões.

Validação:

Aplicar o operador projetado sobre uma imagem de teste e analisar o desempenho.



$$\Psi(S)(z) = \psi\left(\right)$$

Aplicação do operador Ψ sobre o conjunto S

Stacked Generalization

Se a janela considerada é muito grande, o operador gerado é pouco preciso, pois muitos dos padrões possíveis não são observados nas imagens de treinamento. Além disso, o tempo gasto durante o treinamento do operador é muito grande, pois é exponencial no tamanho da janela utilizada (2 elevado ao tamanho da janela), podendo levar muitos dias mesmo nos mais modernos computadores.

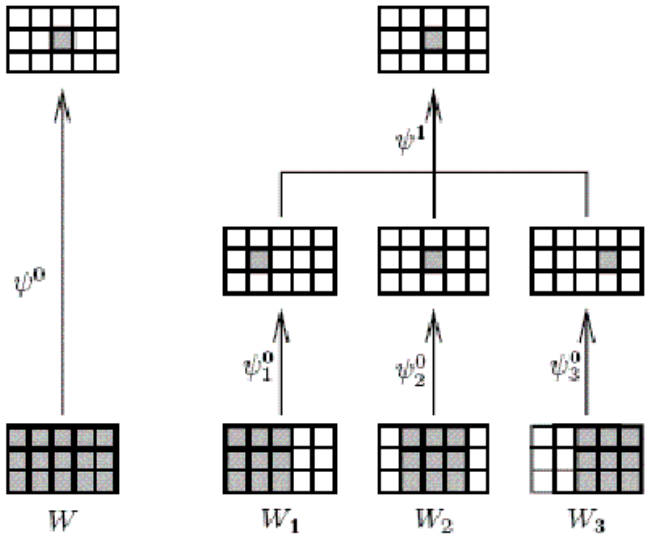
Por outro lado, se a janela escolhida é muito pequena, perdemos a capacidade de discriminar formas relativamente grandes, pois não conseguimos observar uma vizinhança razoável para reconhecer uma forma, podendo assim cometer erros.

Uma possível abordagem para contornar esse problema consiste em considerar várias subjanelas da janela original, projetar um operador para cada subjanela e combinar o resultado desses operadores para obter assim o resultado final.

Para isso podemos usar o método *stacked generalization. Stacked generalization* refere-se a qualquer técnica de aprendizado que utiliza mais de um nível de treinamento. O nível 0 utiliza exclusivamente os dados de entrada, enquanto os níveis superiores podem utilizar resultados de níveis abaixo deles.

A idéia básica por trás do *stacked generalization* é que tendo k operadores (f_1,..., f_k) treinados sobre um conjunto de imagens A, e aplicando-os sobre um exemplo x de um outro conjunto B, obteremos as imagens f_1(x), ..., f_k(x). Como eles não foram treinados sobre B, é possível as imagens obtidas não sejam a ideal y. Podemos agora usar os elementos do tipo ((f_1(x), ..., f_k(x)), y) como dados de treinamento para treinar um novo operador f que combina as informações de f_1,..., f_k.

Cada classificador da fase 0 pode ter como alvo um mesmo pixel ou diferentes pixels na imagem de saída. A figura abaixo mostra a abordagem mais genérica de treinamento em duas fases em comparação com o treinamento em uma única fase.



Na figura, o operador Ψ^0 é o resultado do treinamento de uma única fase. Ele utiliza toda a informação contida na janela W de tamanho 3x5. Os operadores Ψ_i^0 são os operadores gerados na fase 0 e utilizam uma subjanela W_i de tamanho 3x3. Eles não têm todos o mesmo pixel alvo nas imagens intermediárias. O operador Ψ^1 é o operador da fase 1, que combina os pixels alvos de todas as imagens intermediárias gerando assim um pixel de saída final.

Implementação

A plataforma Khoros

O sistema Khoros foi desenvolvido no Departamento de Ciência da Computação e Engenharia Elétrica da Universidade do Novo México, EUA, para proporcionar um ambiente integrado para o desenvolvimento de software voltado para processamento de informações com visualização dos resultados.

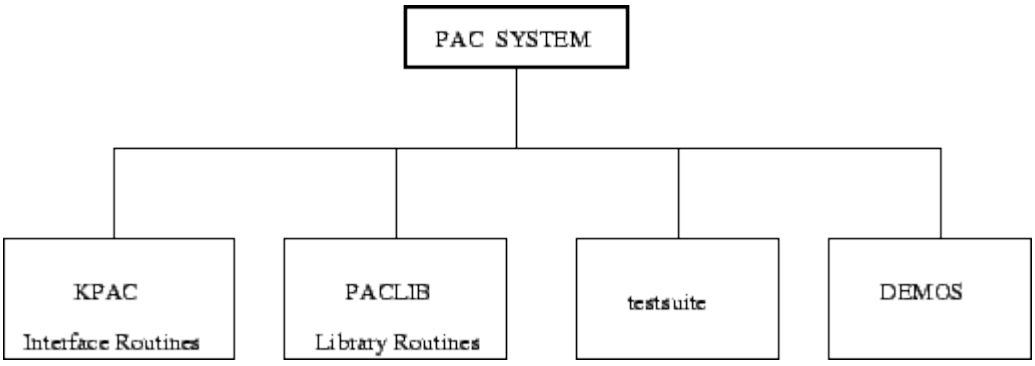
Este sistema possui uma interface para o desenvolvimento de novos programas, o qual gerencia a geração dos códigos fontes, sua documentação e geração dos executáveis. Além disso, possui também uma interface gráfica de alto nível, chamada Cantata, que oferece um ambiente para programação visual.

Um conjunto de programas desenvolvidos neste ambiente pode ser agrupado e organizado como uma "toolbox" (caixa de ferramentas) e pode ser facilmente integrado ao sistema de acordo com a necessidade do usuário. Os programas da toolbox podem ser acessados no Cantata por meio de menus de seleção e ligados entre si de forma que a saída de um deles seja a entrada do outro. Estes programas são representados graficamente por um ícone (glyph) e o fluxo de dados é representado por arestas orientadas ligando estes ícones. Ligados convenientemente, estes ícones representam um programa típico no Cantata, denominado Workspace.

A toolbox PAC

A toolbox PAC foi criada originalmente para fornecer uma ferramenta de transformações em imagens binárias, com geração e aplicação de operadores morfológicos. Posteriormente, ela foi estendida para suportar também a utilização em imagens em níveis de cinza.

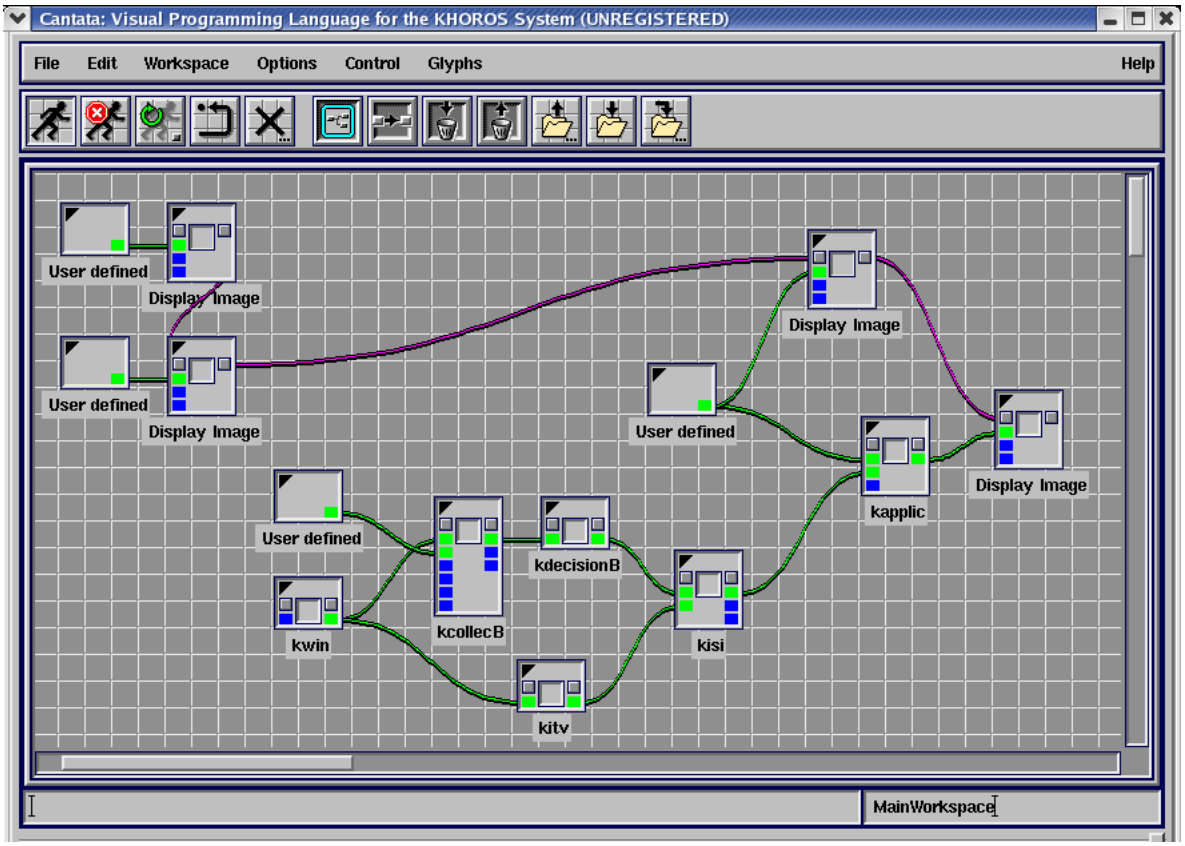
Ela é organizada da seguinte forma:



KPAC é uma toolbox do Khoros com krotinas. Para cada função do sistema, existe uma krotina correspondente. Ela lê parâmetros fornecidos pelo usuário e os passa para a rotina principal. São representadas graficamente no Cantata pelos glyphs. Cada fase do algoritmo PAC visto anteriormente possui uma krotina correspondente, permitindo assim que essas várias fases sejam conectadas no Cantata.

As rotinas principais estão na biblioteca PACLIB, que é outra toolbox do Khoros. Ai estão as funções que implementam as fases do algoritmo PAC, como coleta, decisão e aplicação, bem como outras funções auxiliares.

Abaixo está um exemplo de um workspace no Cantata com glyphs da toolbox PAC que é uma implementação do algoritmo PAC. Ele contém a coleta de padrões em exemplos, decisão, generalização e aplicação. Além disso, ele exibe as imagens geradas e analisa os resultados.



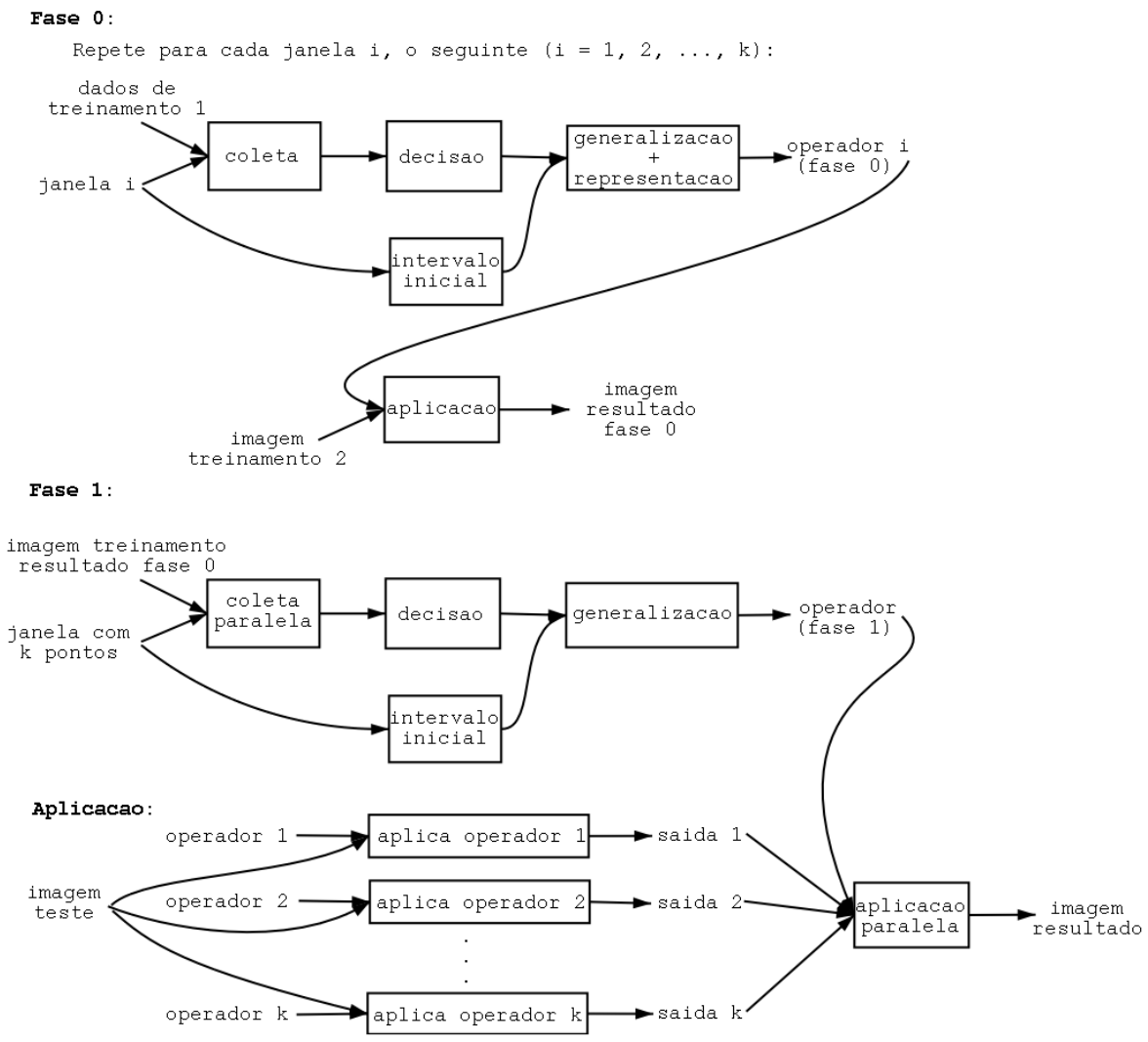
O projeto

O objetivo do nosso projeto foi estender a toolbox PAC para que ela implementasse o *stcked generalization* em duas fases com deslocamento do pixel alvo em relação ao centro da janela.

O projeto foi desenvolvido em dupla, e toda a análise e implementação foram realizadas pelos dois integrantes em conjunto. Foram feitas reuniões semanais com a orientadora, que teve participação intensa na execução do projeto.

Os prazos estipulados na proposta do projeto não foram cumpridos, principalmente porque a fase de instalação e configuração se estendeu até meados de agosto. Isso deixou pouco tempo para a implementação e quase nenhum para testes.

A figura abaixo ilustra o esquema do algoritmo implementado no trabalho:



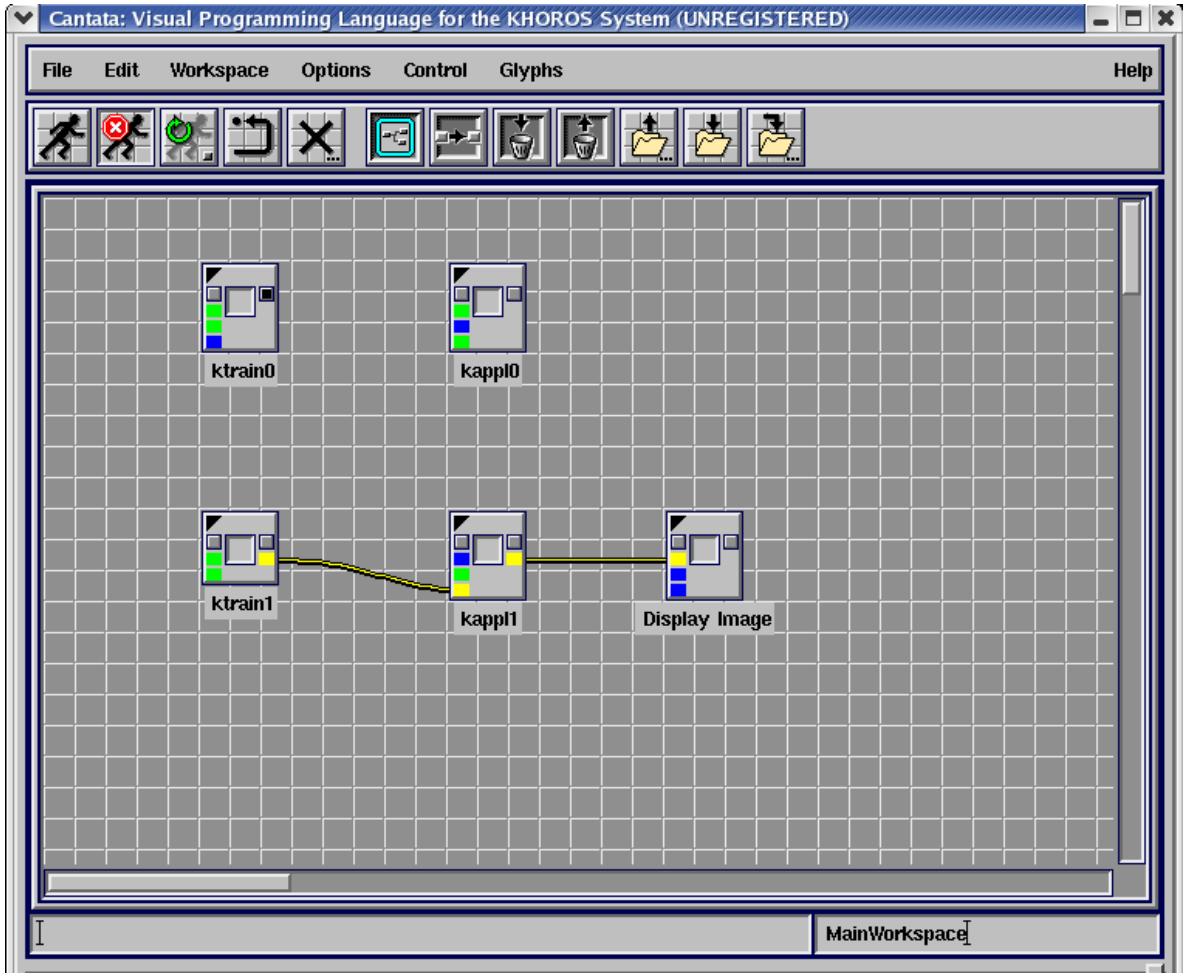
- k subjanelas são utilizadas na fase 0. Para cada subjanela, os operadores da fase 0 são gerados conforme o procedimento descrito anteriormente.
- Na fase 1, a coleta considera as imagens resultantes de cada um dos k operadores da fase 0. As etapas de decisão e generalização são similares à fase 0.

Foram implementadas quatro krotinas no Khoros:

- *ktrain0*: representando o treinamento da fase 0. Inclui coleta, decisão e generalização. Recebe um conjunto de imagens de treinamento e k janelas com seus respectivos deslocamentos. Tem como saída k operadores.
- *kapplic0*: representando a aplicação dos operadores gerados na fase 0. Recebe k operadores com os respectivos deslocamentos e um segundo conjunto de imagens. Tem como saída k imagens resultantes da aplicação dos operadores no conjunto recebido.
- *ktrain1*: representando o treinamento da fase 1. Recebe k imagens resultantes da aplicação da fase 0 e os respectivos deslocamentos. Gera um operador que combina as informações das k imagens recebidas.

- *kapplic1*: recebe k imagens resultantes da aplicação da fase 0 com os respectivos deslocamentos e o operador gerado no treinamento da fase 1. Tem como saída a imagem final.

Abaixo segue a imagem do workspace do Cantata contendo as krotinas desenvolvidas:



Resultados e conclusões

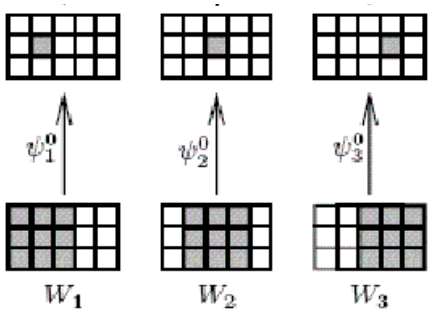
Uma vez feita a implementação, é interessante estudar se o processo em duas fases traz resultados melhores que o processo em uma única fase considerando as mesmas condições. Para isso, nos testes, precisamos usar o mesmo conjunto de imagens de treinamento nos dois casos, ou seja, o conjunto usado para testar o processo de uma fase deve ser a união dos dois conjuntos usados no processo de duas fases. Além disso, a janela usada no processo de uma única fase deve ser a união de todas as subjanelas do processo de duas fases.

Como nossa implementação permite o uso de deslocamentos, surge uma outra questão, sobre a existência de uma configuração de deslocamentos que para qualquer aplicação dê resultados melhores.

Abaixo estão os testes mais significativos realizados. Cada teste consiste de três experimentos:

- O processo em uma única fase
- O processo em duas fases com todos os operadores tendo como alvo o mesmo pixel em relação ao centro da janela (sem deslocamento)
- O processo em duas fases com os operadores tendo como alvo diferentes pixels em relação ao centro da janela (com deslocamento)

Testamos algumas configurações de deslocamentos, mas aqui apresentaremos apenas a que deu o melhor resultado, que é ter como pixel alvo o centro da subjanela considerada, conforme ilustrado na figura abaixo:



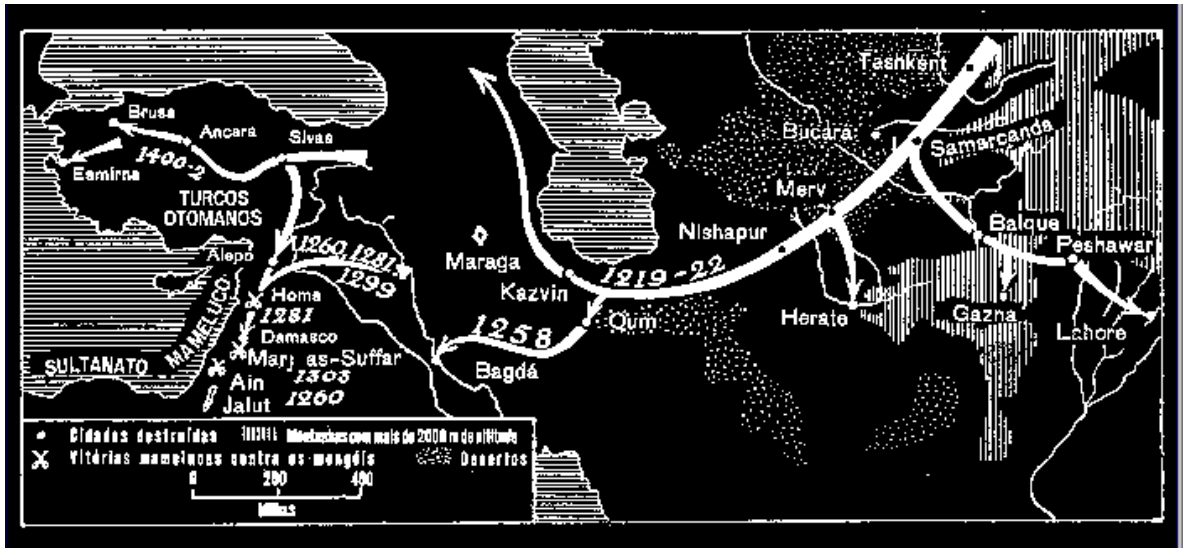
Os valores de erro mostrados nos resultados dizem respeito à porcentagem de pixels diferentes da imagem resultante em relação à imagem ideal.

Teste 1: reconhecimento de textura

Neste teste, o objetivo era treinar um operador que fosse capaz de reconhecer uma textura com listras horizontais.

Janela considerada para uma fase: 3x7.

Janela considerada para duas fases: três subjanelas 3x5.



(Imagem teste)

(Operador de uma fase)

(Operador de duas fases sem deslocamento)

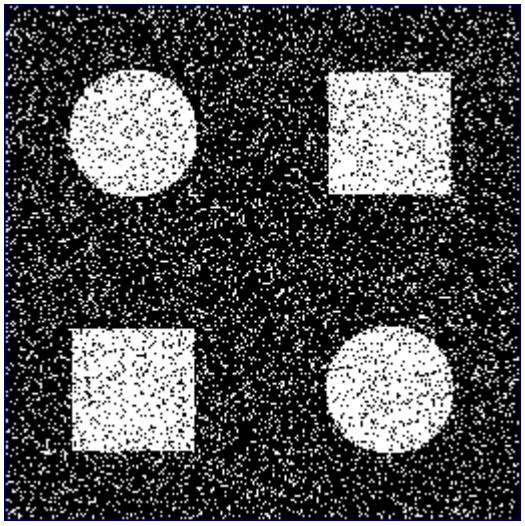
(Operador de duas fases com deslocamento)

Abaixo estão os resultados:

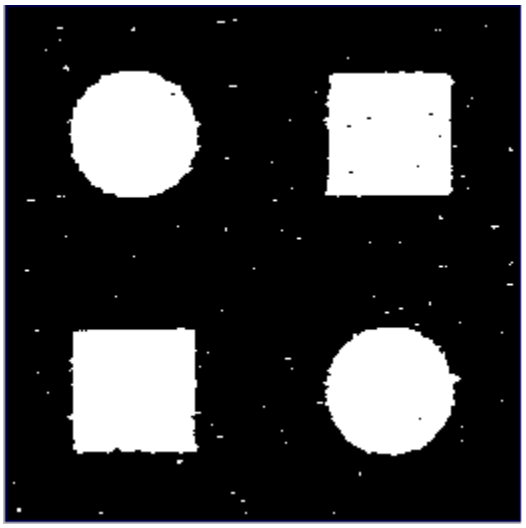
	Erro	**Tempo**
Uma fase	2,75%	33 horas
Duas fases sem deslocamento	2,65%	2 minutos
Duas fases com deslocamento	2,78%	2 minutos

Teste 2: filtragem de ruídos

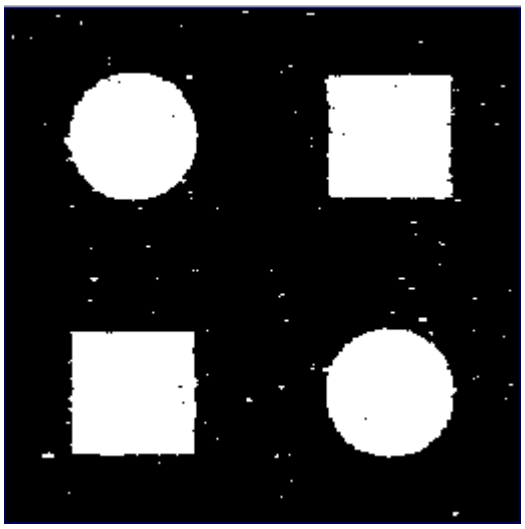
Neste teste, o objetivo era treinar um operador que eliminasse ruídos das imagens.
Janela considerada para uma fase: 3x7.
Janela considerada para duas fases: três subjanelas 3x5.



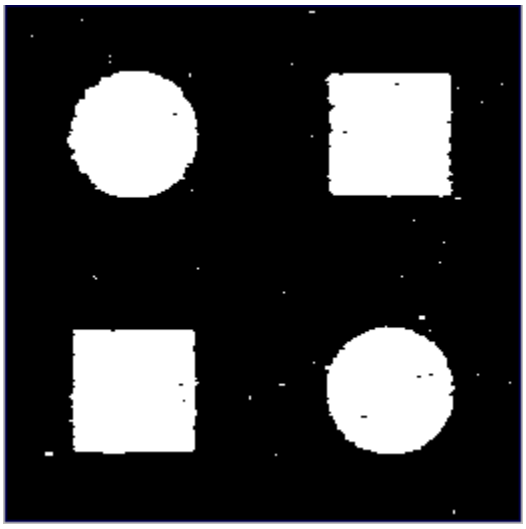
(Imagem teste)



(Operador de uma fase)



(Operador de duas fases sem deslocamento)



(Operador de duas fases com deslocamento)

Abaixo estão os resultados:

	Erro	**Tempo**
Uma fase	0,65%	82 horas
Duas fases sem deslocamento	0,62%	13 minutos
Duas fases com deslocamento	0,45%	13 minutos

Conclusões

O maior problema na geração automática de operadores morfológicos para transformação de imagens é a escolha apropriada do tamanho da janela para o tipo de imagem em questão e o objetivo do operador. Com a utilização do *stacked generalization,* surgem novos problemas, como o número de subjanelas que serão utilizadas e seu formato, e também o pixel alvo dessas janelas na imagem de saída.Dadas todas essas variáveis, seria necessário um número grande de testes e uma estatística mais precisa e elaborada para descobrir qual é a melhor configuração de deslocamentos e subjanelas na maioria dos casos.

Infelizmente o tempo colossal gasto no treinamento de um operador utilizando-se janelas grandes, bem como o atraso nas fases iniciais do projeto, impossibilitou a execução de testes suficientes para chegar a uma conclusão absoluta sobre tais questões. No entanto, nossos testes mostraram que, mesmo se existir uma configuração que é melhor na maioria dos casos, ela pode não ser a melhor em todos os casos. Nos testes realizados, existe um certo equilíbrio entre a acuidade dos operadores de uma única fase, de duas fases sem deslocamento e de duas fases com deslocamento, com a vitória (menor porcentagem de erro) variando entre estes três processos.

É importante notar, no entanto, que a relação da acuidade do operador de uma única fase e do de duas fases depende fortemente do tamanho da janela escolhida (que será particionada no *stacked generalization).* Caso a janela escolhida seja muito pequena, ou seja, menor do que a janela que dá os melhores resultados para o problema em questão, então uma subjanela dessa janela será ainda menor, e nos dará resultados piores. No entanto, se a janela escolhida for um pouco maior que a janela de tamanho ótimo, então as subjanelas podem ter o tamanho ótimo, e sua combinação irá gerar resultados muito melhores do que os apresentados pelo operador treinado em uma única fase com a janela inteira.

Apesar de todas essas dúvidas, fica óbvio que o tempo de processamento gasto no processo com divisão da janela é muito pequeno se comparado ao tempo do processo de uma única fase. Portanto, o *stacked generalization* pode compensar mesmo nos casos que seus resultados forem um pouco piores.

Apesar dos grandes avanços que a área da geração automática de operadores viu acontecer nos últimos anos, ainda restam em aberto essas questões, que hoje em dia são respondidas caso a caso e na base da tentativa e erro. O projeto aqui apresentado mostrou alguns desses avanços e ajudou a inserir mais uma questão que precisa ser respondida por quem deseja criar um operador morfológico ótimo.

Bibliografia

Nina S. Tomita, *Programação automática de máquinas morfológicas binárias baseada em aprendizado PAC*, Dissertação de mestrado, IME-USP, 1996.

D. H. Wolpert, *Stacked Generalization*, Neural Networks, vol.5, pp.242-259, 1992.

N. S. T. Hirata, *Binary Image Operator Design Based on Stacked Generalization*, Proceedings of SIBGRAPI 2005.

Documentação da toolbox PAC: http://www.vision.ime.usp.br/~nina/pac/

Documentação do Khoros: http://www.cab.u-szeged.hu/local/doc/khoros/Tutorial/