Projetos de Pesquisa do Sistema de Acompanhamento dos Alunos da Pós-Graduação (SAAP)

Título do Projeto de Pesquisa:

Representação de Imagens Digitais

através de Codificação de Vizinhança.

Aluno: **Tiago Buarque Assunção de Carvalho.**

Orientador: **Tsang Ing Ren.**

Conteúdo

Justificativa

A representação de imagens tem um papel crucial no processamento de imagens digitais, a imagem só pode ser instanciada por um software se estiver representada de alguma maneira na memória da máquina. A questão da representação é um problema de várias áreas da computação: um conjunto ordenado pode ser representado por um *array* ou por uma lista ligada; um grafo pode ser representado tanto como uma lista de adjacência ou como uma matriz de adjacência [Cor02]. Cada maneira de representação também está ligada a um tipo específico de estrutura de dados que se adéqua melhor a um ou outro problema de programação.

Os problemas em processamento de imagens são vários: melhoramento, restauração, reconhecimento, análise multiresolução, operações morfológicas etc. Para cada problema pode existir uma técnica de representação que otimize determinado aspecto, exemplo: menor custo de memória e/ou processamento; maior facilidade de implementação por parte do programador; maior compatibilidade com determinado dispositivo. Mais ainda uma representação pode ter mais de um objetivo, ex.: maior taxa de compressão e maior taxa de reconhecimento.

A representação de imagens binárias é um caso específico que tem grande importância[1] para representação de documentos digitais/digitalizados e reconhecimento de forma [AKF08]. No caso de documentos escaneados a compressão e análise multiresolução são fatores importantes, pois estas imagens geralmente são muito grandes [RMB09]. O reconhecimento de formas é uma tarefa que ganha evidência com os novos padrões de vídeo que estão sendo desenvolvidos, com isto descritores de forma estão em destaque [LLE00] [BYL+10].

[1] Outro ponto importante sobre as imagens binárias é sua ampla utilização em dispositivos eletrônicos voltados para área de documentos, como: impressoras e *e-readers.*

Revisão da Literatura

Representação de imagens é uma técnica usada para descrever uma imagem digital que tem duas abordagens principais: compressão e descrição de conteúdo [Par02]. O objetivo da compressão é economizar recursos de armazenamento e transmissão de dados. Enquanto a descrição de conteúdo visa indexar e recuperar imagens, um exemplo disso é o reconhecimento de formas.

A representação (ou codificação) de uma imagem pode ser realizada de duas maneiras distintas: com perdas, onde é representada uma aproximação da imagem original, e sem perdas, onde a imagem representada é exatamente igual à imagem original. Os métodos de segunda geração para codificação de imagens [RMB97] baseiam-se em características do sistema visual humano. Essa abordagem permite que a imagem seja representada com perdas em troca de uma maior taxa de compressão. A codificação de vizinhança é um método de representação de imagens sem perdas, portanto nesta seção só serão revisados métodos de representação sem perdas.

Representação de imagens sem perdas

Antes de abordar as demais representações de imagens, vale relembrar a representação canônica de uma imagem binária: uma imagem binária é uma matriz binária $P = [p_{xy}]$, de dimensões $w \times h$, em que cada posição $(x, y) \in N^2$ da matriz só pode assumir dois valores, 0 ou 1, $p_{xy} \in \{0, 1\}$. Quando o valor p_{xy} de uma posição (x, y) da matriz é 0 também é chamado branco, pixel branco ou pixel de fundo, quando esse valor é 1 é chamado preto, pixel preto ou pixel da imagem.

A codificação sem perdas de imagens binárias tem sido estudada nos últimos anos sob o problema de representação de formas. As três maiores classes de representação de formas são representação por bitmap, representação interna e representação de contorno [JBB+98]. Cada uma dessas classes é explanada a seguir.

As técnicas de representação por bitmap operam diretamente sobre os dados, isto é, operam diretamente na matriz de bits sem usar qualquer representação intermediária,

essas técnicas também são técnicas de compressão e serão vistas com mais detalhes adiante. Entre elas se destacam a MH (*Modified Huffman*), MR (*Modified* READ), MMR (*Modifiead* MR) e CAE (*Context-based Arithmetic Encoding*). As técnicas MH, MR e MMR são usadas nos algoritmos de compressão CCITT Group 3 e Group 4. A idéia de cada um desses métodos é representar uma seqüência de pixels de uma mesma cor, em cada linha da imagem, por uma cadeia de bits menor do que aquela que está sendo representada. O CAE codifica cada pixel da imagem em função daqueles que já foram codificados anteriormente. Para tanto, faz uma predição do próximo valor do pixel a partir de um conjunto de templates. Este método é usado no algoritmo de compressão JBIG, que é o estado da arte em compressão de imagens binárias [RMB09].

A representação interna visa representar regiões da imagem, opondo-se à representação do contorno. Duas técnicas principais podem ser destacadas nessa classe de representação: esqueletização [Dou94] e decomposição por *quadtree* [JBB+98]. A primeira extrai um esqueleto da imagem enquanto a segunda divide suas regiões de forma hierárquica. A esqueletização representa uma imagem inteira através de um esqueleto. Esse esqueleto é um eixo simétrico que pode ser definido como o centro de todos os círculos de raio máximo inscritos dentro de uma forma. Uma forma pode ser descrita como o conjunto desses centros e os respectivos raios de cada centro. Este método pode tanto ser usado para compressão como para reconhecimento de formas. Exemplos de esqueletizações podem ser vistos na Figura 1. O esqueleto do círculo é apenas um ponto, pois ele é suficiente para tangenciar todos os pontos da borda da forma. Para o quadrado, círculos de raio menor são necessários para representar os pontos perto dos cantos.

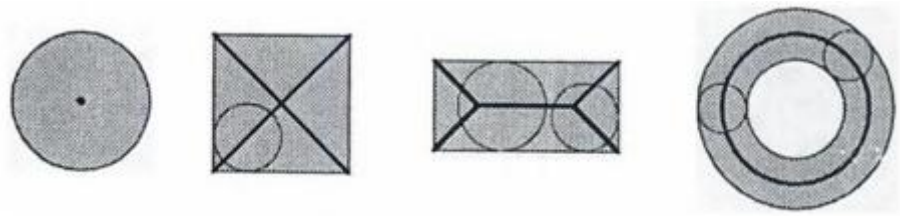


Figura 1. Exemplos de esqueletizações [Dou94]. O círculo é representado apenas por um ponto. O quadrado é representado por eixos que cruzam suas diagonais. As linhas pretas marcam os esqueletos do retângulo e da coroa.

Uma *quadtree* divide cada região da imagem em quatro regiões de tamanhos iguais, a região dividida se torna um nó e cada uma das quatro regiões formadas uma folha da quadtree. A decomposição por quadtree inicia a imagem como uma folha, então vai subdividindo cada folha até que a região representada por cada folha seja uma região homogênea, como pode ser visto na Figura 2

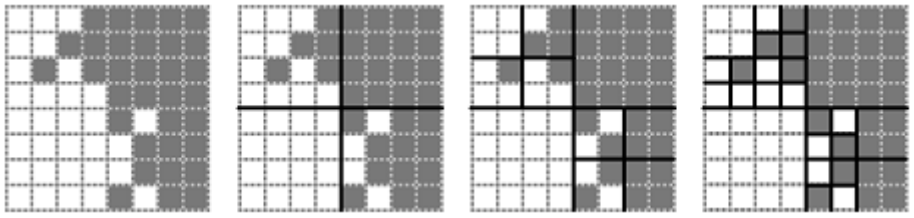


Figura 2. Exemplo de decomposição por quadtree. Cada região é dividida em quatro novas regiões até que todas as regiões se tornem uniformes.

O método para codificação de contorno é a Codificação de Cadeia (*Chain Code*). Esta codificação interpreta o contorno de uma forma como uma cadeia de pixels. Para representar um contorno é necessário um ponto inicial. Cada ponto consecutivo da cadeia é representado através de um código que indica a posição em relação ao pixel atual. Cada ponto da cadeia é codificado até que se chegue ao ponto inicial ou ao fim de um contorno aberto. O código relativo atribuído a cada ponto pode seguir diversas conectividades, como pode ser visto na Figura 3. O exemplo de um contorno codificado pode ser visto na Figura 4, a conectividade usada para essa cadeia foi a F8 (cadeia de Freeman de oito direções), Figura 3(b).

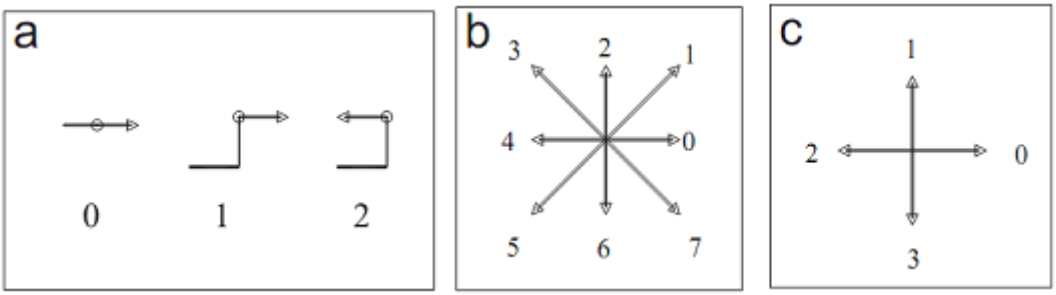


Figura 3. Exemplos de códigos conectividades usados em [SCBRD07]: (a) 3OT, direções ortogonais de cadeias de três símbolos (orthogo- nal chain directions of three symbols); (b) F8, cadeia de Freeman de oito direções; (c) F4, cadeia de Freeman de quatro direções.

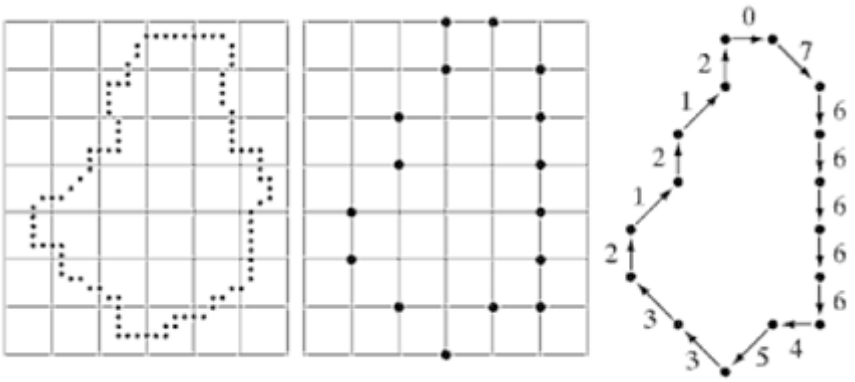


Figura 4. Exemplo de codificação de cadeia [GW10] de um contorno usando a conectividade F8 (cadeia de Freeman de oito direções) da Figura 2.3 (b).

Compressão de Imagens Binárias

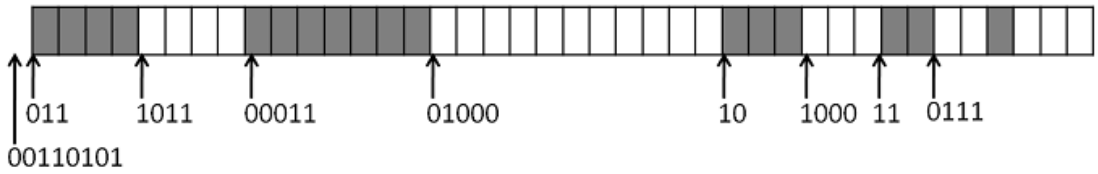
Os métodos de compressão revisados nesta seção têm a característica de serem próprios para a representação sem perdas de imagens binárias. Os algoritmos CCITT Group 3, CCITT Group 4 e JBIG foram criados especificamente para essa tarefa. Já os algoritmos GIF e PNG foram criados para representar imagens sem perdas não se restringindo a imagens binárias, mas esses algoritmos podem ser configurados para obterem uma maior taxa de compressão quando usados para representar imagens binárias. Algumas referências sobre esses métodos de compressão são [WMB99] [Bar06].

A pesquisa sobre compressão de imagens binárias digitais se iniciou com o desenvolvimento de padrões para transmissão de fax. No final dos anos 1970, o CCITT (*Comité Consultatif International Téléphonique et Télégraphiquere*) conheceu o potencial das tecnologias para transmissão de fax e passou a procurar novos padrões e soluções para o mesmo. Em 1980, o CCITT, através da recomendação T.4, normatizou o padrão de transmissão Group 3 [IT03] para a transmissão digital de fax para redes analógicas, que é até hoje um padrão de compressão de arquivo utilizado em máquinas de fax. Apesar de vários outros padrões de imagens binárias comprimidas apresentarem taxas de compressão maiores do que o Group 3, tais formatos são sensíveis a erros durante a transmissão fazendo com que o Group 3 ainda seja o padrão adotado para transmissão de fax via rede telefônica há 40 anos [Bar06]. Este padrão incorpora técnicas simples e rápidas de compressão de dados ao mesmo tempo em que realiza a detecção e correção de erros. O Group 3 substituiu os padrões analógicos Group 1 e Group 2. O Group 1 era não-confiável e era necessário em torno de seis minutos para transmitir cada página, enquanto o Group 2 transmitia a uma taxa de três minutos por página. O método de compressão usado no padrão Group 3 reduziu o tempo de transmissão de um fax para 6 a 30 segundos por página (mais 15 extras para a primeira página).

Em 1984, o CCITT publicou o padrão Group 4 [IT88], que é semelhante ao Group 3 mas é destinado ao uso de redes de dados digitais em vez de circuito de telefone analógico convencional. Ele atinge aproximadamente o dobro da taxa de compressão do Group 3, no entanto, ele não realiza a detecção e correção de erros. O padrão Group 3 se divide em dois métodos de codificação: unidimensional, chamado MH (*Modified Huffman*), e bidimensional, chamado MR (*Modified* READ, em que READ significa *Relative Element Address Designate codes*). O primeiro esquema trata cada linha independentemente enquanto o segundo utiliza MH para transmitir a primeira linha e transmite as outras linhas usando uma codificação relativa à linha anterior. O MH codifica seqüências de bits de uma mesma cor através de um código. Este método é baseado na codificação Huffman [Huf52]. Ele associa códigos com menos bits a seqüências mais freqüentes.

A Figura 5(b) mostra as doze primeiras linhas da tabela de códigos do MH. Essa tabela foi construída a partir de um conjunto de imagens de documentos selecionados pelo CCITT. Nota-se que a tabela de códigos diferencia seqüências de pixels pretos ou brancos. Figura 5(a) mostra os códigos gerados, empregando essa tabela, para uma linha de uma imagem binária. O MH assume que a primeira seqüência de pixels é branco e por isso indica que essa seqüência tem tamanho zero e é representada pelo código binário 00110101. Em seguida a seqüência de quatro pixels pretos é representada pelo código 011. O quatro pixels brancos seguintes são representados pelo código 1011. Os sete pixels pretos consecutivos são representados por 00011. O mesmo se dá para as demais cadeias de pixels da mesma cor.

Na codificação MR (Group 3 bidimensional), a cada k linhas, apenas a primeira é codificada pelo método unidimensional (MH). Todas as outras vão seguir o esquema onde a linha atual faz referência à anterior. Essa abordagem aumenta a compressão. É necessário transmitir uma linha via MH a cada k linhas para diminuir o erro de transmissão. O MMR é o algoritmo de compressão usado pelo padrão Group 4, seu funcionamento é igual ao MR. Porém, por se basear numa rede digital que já garante a integridade dos dados, apenas a primeira linha precisa ser transmitida via MH, as demais linhas são representadas em função da linha anterior.



(a)

White run length	Code word	Black run length	Code word
0	00110101	0	0000110111
1	000111	1	010
2	0111	2	11
3	1000	3	10
4	1011	4	011
5	1100	5	0011
6	1110	6	0010
7	1111	7	00011
8	10011	8	000101
9	10100	9	000100
10	00111	10	0000100
11	01000	11	0000101
...	...	...	...

(b)

Figura 5. Exemplo da codificação MH (Modified Huffman) empregada pelo padrão Group 3. A Figura (a) é uma linha de uma imagem binária codificada pelo algoritmo MH. Cada seqüência de pixels da mesma cor é representada por um código que indica seu tamanho. Cada código está definido na tabela do MH, as doze primeiras linhas dessa tabela estão exibidas na Figura (b). A Figura (a) inicia com um código indicando que o número de pixels brancos no começo da linha é zero pois o MH assume que o primeiro pixel de cada linha é branco.

GIF (Graphics Interchange Format) foi um dos formatos de compressão de imagem sem perda de qualidade mais usado até 1995. O padrão foi introduzido pela CompuServe em 1987 com o objetivo de ser um formato de imagens gráficas que podem ser exibidas em diferentes plataformas. Por isso é amplamente usado na Internet, tanto na forma de figuras fixas como animadas. GIF codifica o arquivo de modo que cada pixel é representado por 8 bits ou menos. Cada código corresponde a uma cor de uma paleta limitada a 256 cores. Essa paleta pode ser especificada para cada imagem ou grupo de imagens que compartilham do mesmo esquema de cor. Imagens binárias utilizam uma paleta reduzida, com apenas duas cores. Cada cor é do tipo RGB e quando gravada no cabeçalho do arquivo consome 24 bits - 8 bits para cada

cor primária: vermelho, verde e azul. A compressão usada pelo formato é o LZW (Lempel-Ziv-Welch). Esta inicializa um dicionário com o alfabeto, cada instância do alfabeto é uma cor, em seguida analisa frases sucessivas da seqüência de entrada. Estas podem ser adicionadas ao dicionário. O esquema de codificação GIF inicia o dicionário com até 256 possíveis códigos para os pixels, mais dois códigos extras, o código "limpar"e "fim". Então processa a codificação LZW para a seqüência de pixels.

O LZW é uma das variações mais populares dos princípios de codificação Ziv-Lempel e foi baseado no algoritmo LZ78 [WMB99]. O método codifica apenas frases numéricas e não tem caracteres explícitos na saída. O algoritmo inicia a lista de frases (dicionário) com todos os caracteres da entrada. Uma nova frase é formada a partir de um elemento já existente anexado ao primeiro caractere da entrada atual. A Figura 6 mostra o exemplo de uma execução do algoritmo LZW. A cadeia de entrada é abaababbaabaabaa. O dicionário é iniciado apenas com os símbolos do alfabeto a=1,b=2. O laço do algoritmo procura a maior cadeia presente no dicionário para codificar os símbolos da cadeia de entrada na posição atual da iteração. A cada trecho da cadeia de entrada codificado, um código é inserido na entrada de saída e uma nova instância é criada no dicionário. A nova instância do dicionário criada a cada iteração é a instância que foi codificada contatenada com o próximo símbolo. No exemplo da Figura 2.6, o símbolo a é codificado pelo código 1 do dicionário, e a instância ab é inserida no dicionário com o código 3. Em seguida b é codificado na cadeia de saída com o código 2, e a instância ba é inserida no dicionário com o código 4. O código a é represento por 1 e a instância aa = 5 é inserida no dicionário. Continuando, algoritmo localiza na cadeia a instância ab do dicionário, que é codificada como 3 na cadeia de saída. O algoritmo repete esse laço até toda a cadeia estar codificada.

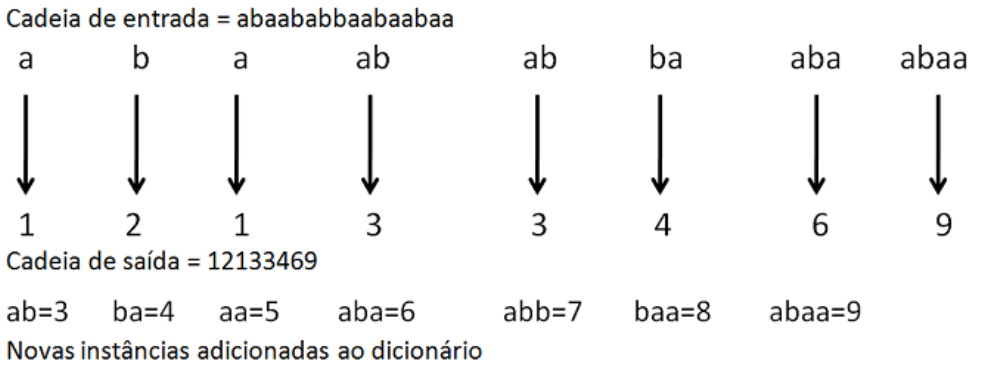


Figura 6. Exemplo da execução do algoritmo LZW. A cadeia de entrada (representada por letras) é convertida numa cadeia de índices para instâncias do dicionário. A cada índice inserido na cadeia de saída, uma nova instância é adicionada ao dicionário.

Com base no GIF foi gerado um novo método de compressão sem perda, o PNG (Portable Network Graphics). Este pretendia substituir o GIF para evitar infringir a patente de Unisys sobre a técnica LZW. O método PNG é similar ao GIF, ambos geram uma seqüência com os valores dos pixels e empregam o propósito geral de compressão Ziv-Lempel. O PNG também tem melhores taxas de compressão e mais recursos. O padrão PNG usa o esquema de compressão gZip e incorpora elementos opcionais, como os "filtros", aplicados antes da compressão. Para cada linha de imagem um tipo de filtro é escolhido, este prevê o valor de cada pixel com base nos valores dos pixels vizinhos. PNG também adiciona uma série de outras melhorias em relação ao GIF. Ele não fica limitado a apenas 256 cores de 24 bits, também suporta paletas de 48 bits, tons de cinza (16 bits) e permite 256 níveis de transparências para cada pixel.

O gZip é mais uma variação de compressão Ziv-Lempel e foi baseado no algoritmo LZ77 [ZL77]. O método utiliza o algoritmo de deflação [GA10, GA96] e a Codificação de Huffman para gerar e gravar seus resultados. O algoritmo de deflação encontra seqüências repetidas na entrada. A segunda ocorrência de uma série de caracteres é substituída por um ponteiro para a anterior, sob a forma de uma tupla (distância, comprimento), Figura 7. As distâncias são limitadas a 32 kbytes e representa a extensão do símbolo atual até o primeiro da seqüência que se repete, e os comprimentos são limitados a 258 bytes e significam o tamanho dessa seqüência. Se não for encontrada nenhuma recorrência nesse intervalo, é gerado um literal no lugar

do ponteiro. As séries de símbolos duplicadas são encontradas utilizando uma tabela hash. Os literais e os comprimentos (da tupla) são compactados com a mesma árvore de Huffman, e as distâncias são compactadas com outra árvore. O arquivo de entrada é fragmentado em vários blocos, para cada bloco o algoritmo é aplicado e novas árvores de Huffman são criadas.

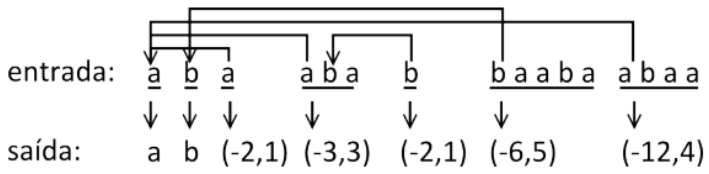


Figura 7. Exemplo do funcionamento do algoritmo gZip, uma série de caracteres é substituída por um ponteiro para a anterior, sob a forma de uma tupla (distância, comprimento).

O JBIG, também conhecido como JBIG1, é um padrão internacional para compressão sem perda de imagens. É destinado a imagens binárias, mais especificamente para fax, porém pode ser usada para tons de cinza comprimindo cada plano de bits separadamente. A compressão pode ser aplicada à imagem inteira, ou pode usar o modo progressivo. No modo progressivo a compressão é aplicada, separadamente, a uma versão de menor resolução da imagem (chamada camada base) e a imagens que guardam a informação complementar para gerar versões em maiores resoluções da mesma imagem (chamadas camadas diferenciais). A codificação é baseada no uso de templates e uma codificação aritmética adaptativa. Esta é baseada na probabilidade de cada bit em relação aos bits anteriores do template, permitindo uma compressão e descompressão em ordem de leitura.

A redução de resolução funciona de forma simples, para dividi-la pela metade, agrupam-se pixels em blocos de 2×2 e faz-se a média dos quatro tons de cada bloco. Porém isso só seria possível para imagens em tons de cinza, no caso de imagens binárias é necessário adaptar esse algoritmo. Uma solução fácil é usar uma função randômica em caso de empate (tantos blocos brancos quanto pretos), como a redução aplicada à imagem da Figura 8. Essa solução gera muito ruído na imagem, a melhor solução seria usar regras mais complexas para associar os pixels. Para cada bloco seria

atribuído um valor calculado em função dos seus vizinhos (cor e posição), o pixel que substituirá os 4 blocos terá a cor em função desses valores.

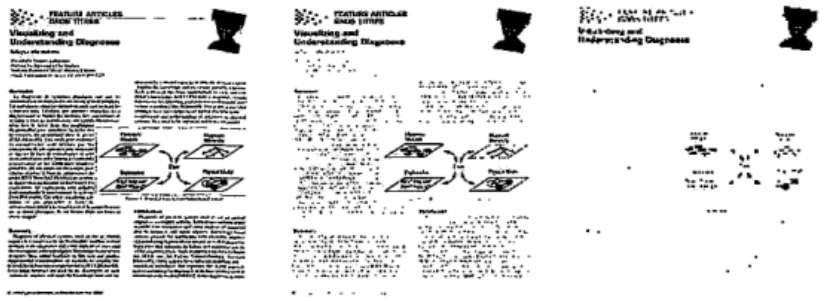


Figura 8. Exemplo de várias resoluções de uma mesma imagem. As duas imagens da esquerda apresentam maior resolução e formam as camadas diferenciais do JBIG. A imagem da direita apresenta menor resolução e forma a camada base do JBIG.

Para o uso de templates o JBIG define dois tipos de contexto, uma para a base e outro para as camadas diferenciais. Para a camada base são usados dois templates diferentes como mostrado na Figura 2.9, ambos com 10 pixels. Para as camadas diferenciais há quatro templates diferentes, todos com 6 pixels e usados dependendo da fase de codificação do pixel. A idéia do template é servir de molde para calcular a codificação aritmética para um determinado pixel, CAE (Context-based Arithmetic Encoding). A probabilidade utilizada pela codificação é calculada para cada pixel em relação aos pixels do template (marcados pelas bolas pretas na Figura 9), isto é, a probabilidade é sensível ao contexto.

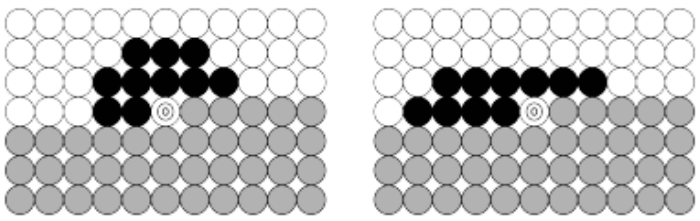


Figura 9. Templates da camada base. As bolas cinzas são os pixels a serem codificados, as pretas e brancas são pixels já codificados e o alvo é o pixel atual. As bolas pretas representam os pixels que fazem parte do template.

JBIG2 é o estado da arte em compressão de imagens binárias. O método é uma otimização do JBIG e faz compressão com ou sem perdas. O padrão foi planejado para ter um desempenho melhor que os métodos existentes na compressão sem perda, e na com perda para atingir taxas muito elevadas de compressão sem prejuízos significativos de qualidade visual. O JBIG2 é útil para aplicação em fax, armazenamento de documentos e arquivamento, codificação de imagens na Internet, transmissão de dados sem fio, fila de espera de impressão, e até mesmo teleconferência. O padrão JBIG2 define explicitamente a decodificação da compressão, ele não define explicitamente o padrão de codificação. Porém é suficientemente flexível para permitir que a concepção de um codificador sofisticado.

Outro diferencial do JBIG2 é a decomposição do documento de imagem em páginas, e essas em várias regiões. Cada uma dessas regiões é tratada de forma diferente, de acordo com seu tipo. Estas podem ser de símbolo, halftone ou genérica. As regiões de símbolo são áreas que contém textos, caracteres pequenos dispostos em linhas horizontais ou verticais. Esses caracteres são chamados de símbolos. O JBIG2 obtém grande parte de sua eficácia reutilizando símbolos individuais que se repetem. Estes são coletados em um ou mais dicionários. Estes indexam um conjunto de bitmaps de símbolos aos números de índice. Uma página pode conter dados em halftone, como imagens geradas por fotografias. Uma região de halftone consiste em um número de padrões distribuídos em uma grade regular. Essas regiões são codificadas com um dicionário de padrão de halftone, os padrões geralmente correspondem a valores de escala de cinza.

Uma página também pode conter outro tipo de dados, como linhas e ruídos, caracterizando a região genérica. Essas áreas podem ser codificadas com dois métodos diferentes, o primeiro é conhecido como *Modified-Modified-Read* (MMR), e é usado pelo Grupo 4 para fax. O segundo método é uma variação do método usado pelo JBIG, faz uso de *templates* e codificação aritmética CAE (*Context-based Arithmetic Encoding*). Este método atinge maiores taxas de compressão e é mais eficaz para regiões com linhas, figuras e gráficos.

Reconhecimento de Forma

Descritores de forma são empregados para comparar silhuetas de objetos bidimensionais de terminando uma medida de similaridade. Estes descritores são úteis e importantes para atividades como recuperação de imagens numa base de dados. Uma definição de forma retirada de [LR01] diz que a forma de um objeto é a característica que permanece invariante sob qualquer translação, rotação e reflexão do objeto. Espera-se, portanto, que descritores de forma apresentem tais características, e em alguns casos ainda espera-se invariância à escala.

Em informação multimídia, objetos bidimensionais geralmente são projeções de objetos tridimensionais. Este fato pode alterar a silhueta dos objetos de três maneiras: mudanças de ponto-de-vista; movimentos não rígidos do objeto (como uma bandeira flamulando, ou um cachorro correndo); ou inserção de ruído durante a digitalização. MPEG-7 é um padrão de interface para descrição de conteúdo multimídia (*Multimedia Content Description Interface*) amplamente utilizado e que incorpora descritores de forma [Mar04]. Para avaliar a qualidade dos descritores de forma sob estas três condições citadas, o MPEG-7 propôs um método de avaliação para os descritores sob um conjunto de base de dados chamado *Core Experiment CE-Shape-1* [LLE00], cada uma dessas bases é formada de imagens binárias de silhuetas de objetos.

A base de imagens *Core Experiment CE-Shape-1* é dividida em três partes: A, B e C. A parte A é subdividida em A1, que avalia a robustez do descritor a variações de escala, e A2, que avalia a robustez do descritor a variações de rotação. A parte B avalia a performance do descritor em relação à recuperação de imagens similares. E a parte C avalia a robustez do descritor em relação a movimentos não rígidos.

A base de imagens A1, que avalia a robustez do descritor a variações de escala, contém 420 formas. São 70 formas apresentadas em seis escalas diferentes cada. As escalas são: 2x, 1x, 0,3x, 0,25x, 0,2x e 0,1x. Exemplos de imagens da base A1 podem ser visualizados na Figura 10. O método de teste padrão para a base A1 é recuperar, para cada uma das 420 imagens, as 6 imagens mais similares, através de uma função de similaridade que utiliza o descritor de cada imagem. Conta-se um acerto cada vez que

uma das seis imagens da mesma classe é recuperada, portanto o número máximo de acertos é 6 x 420 = 2520. E o acerto percentual é 100 x número de acertos/2520.

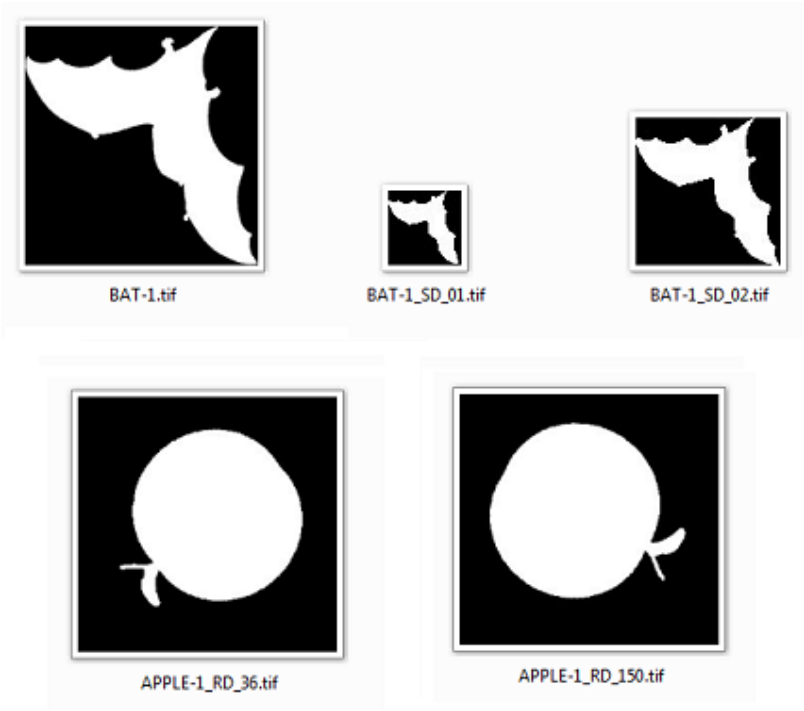


Figura 10. Exemplos da base de imagens binárias de silhuetas de objetos *Core Experiment CE-Shape-1* parte A. A linha de cima contém imagens da base A1, que contém em cada classe uma mesma imagem em escalas diferentes. A linha de baixo contém imagens da base A2, que contém, por classe, uma mesma imagem em diferentes rotações.

A base de imagens A2, que avalia a robustez do descritor a variações de rotação, contém 420 formas. São 70 formas apresentas em seis rotações diferentes cada. As rotações são: 0°, 9°, 36°, 45° (uma rotação de 9° seguida de outra rotação de 36°), 90° e 150°. Exemplos de imagens da base A2 podem ser visualizados na Figura 10. O método de teste padrão para a base A2 é recuperar, para cada uma das 420 imagens, as 6 imagens mais similares, através de uma função de similaridade que utiliza o descritor de cada imagem. Conta-se um acerto cada vez que uma das seis imagem da mesma classe é recuperada, portanto o número máximo de acertos é 6 x 420 = 2520. E o acerto percentual é 100 x número de acertos/2520.

A base de imagens B, que avalia a performance do descritor em relação à recuperação de imagens similares, é a parte mais importante do *CE-Shape-1*. Esta base possui 1400 imagens, sendo 20 imagens por classe, algumas dessa imagens podem ser vistas na Figura 11. O teste para a base B utiliza cada uma das 1400 imagens, compara com

todas, incluindo ela mesma, e recupera as 40 imagens mais similares. O número máximo de imagens corretas recuperadas corretamente é 20 por classe, então o número máximo de acertos é 28000. E o acerto percentual é 100 x número de acertos/28000.

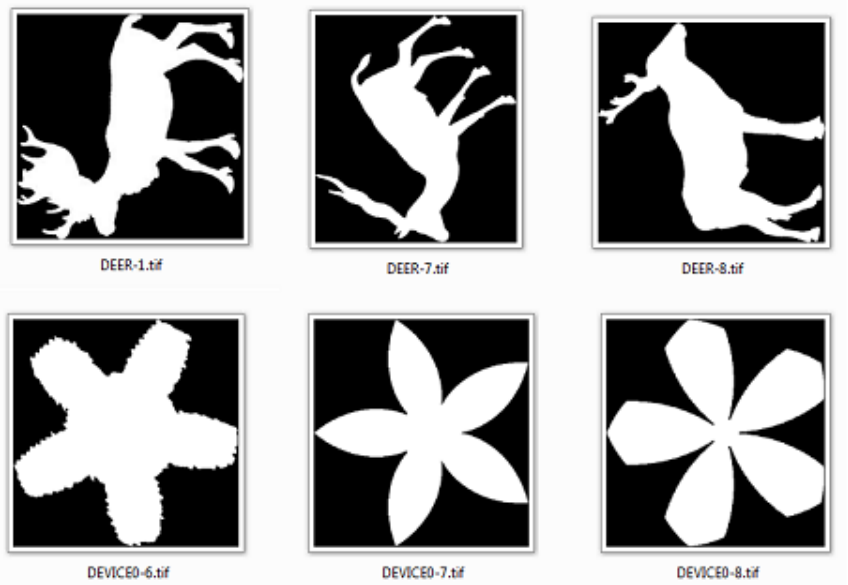


Figura 11. Exemplos da base de imagens binárias silhuetas de objetos *Core Experiment CE-Shape-1* parte B. A linha de cima contém 3 das 20 imagens da classe DEER. E a linha de baixo contém 3 das 20 imagens da classe DEVICE. Percebe-se que os elementos de cada classe apesar de semelhantes representam formas de objetos distintos.

A base de imagens C, que avalia a robustez do descritor em relação a movimentos não rígidos, é composta de 1300 imagens, sendo 200 quadros de um vídeo de um peixe nadando e 1100 imagens de outros peixes e animais marinhos. Exemplos de imagens da base C podem ser visualizados na Figura 12. O método de teste para a base C é recuperar as 200 imagens mais próximas da imagem que forma o primeiro dos 200 quadros do vídeo. Portanto a taxa de acerto percentual é medida como 100 x número de acertos/200.

Alguns dos descritores de forma incorporados pelo padrão MPEG-7 estão listados a seguir:

P298 utiliza o descritor de forma de espaço tangencial baseado na melhor correspondência de partes visuais [LL00]. Um único contorno é usado como um descritor de forma, este contorno é determinado de forma ótima através de um processo de simplificação do contorno chamado evolução discreta de curva;

P320 utiliza o descritor de forma CSS (*Curvature Scale Space*). Um contorno simplificado é obtido através da sua suavização através da convolução com funções gaussianas [dA07];

P517 utiliza autovetores de múltiplas camadas. Os autovetores são calculados para várias regiões da imagem e são usados para formar o descritor [Zib01];

P567 utiliza um descritor de forma *wavelet* onde o contorno é representado em várias resoluções através das funções de aproximação e funções de detalhes *wavelets* [CK96];

P687 utiliza momentos de Zernike, que além de terem um embasamento teórico e experimental bem estabelecidos, apresenta melhores resultados do que momentos convencionais e momentos invariantes [KH90].

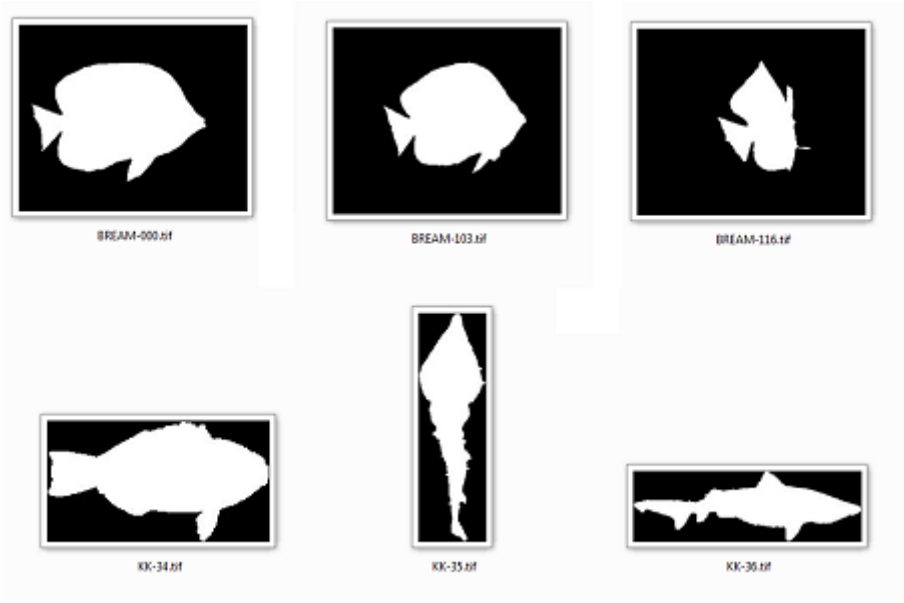


Figura 12. Exemplos da base de imagens binárias silhuetas de objetos *Core Experiment CE-Shape-1* parte C. A linha de cima da figura contém quadros do vídeo de um peixe nadando, a imagem mais à esquerda da primeira linha primeira dos 200 quadros do vídeo. A linha de baixo contém exemplos das 1100 imagens de outros peixes e animais marinhos.

Os resultados dos testes na base *Core Experiment CE-Shape-1* de cada um dos descritores de forma citados anteriormente está descrito na Tabela 1. Os descritores de forma P298 e P320 apresentam melhores resultados do que os outros. Ambos os descritores utilizam versões simplificadas dos contornos, obtidas através de processos de evolução de curvas distintos, para a comparação de formas. Quando as bases do CE-Shape-1 foram testadas com seres humanos, isto é, o classificador das formas é

uma pessoa, a taxa máxima de acerto para as bases A1 e C foi de 93% em cada, a base B também não apresentou 100% de acerto [LLE00].

Tabela 1. Taxas de acerto percentual dos descritores P298, P320, P517, P567 e P687sobre as bases Core Experiment CE-Shape-1 A1, A2, B e C.

Base	P298	P320	P517	P567	P687
A1	88,65	89,76	92,42	88,04	92,64
A2	100,0	99,37	100,0	97,46	99,60
B	76,45	75,44	70,33	67,76	70,22
C	92,00	96,00	88,00	93,00	94,50

Compressão através de Chain Code

Os trabalhos [TS11], [ACRD10], [ACNCRD09], [SCBRD07], [AKF06] e [SCRD05] utilizam o *chain code*, visto anteriormente, para representar comprimir imagens binárias. A base de dados utilizada por eles é de imagens simples e por utilizar *chain code* apenas o contorno das formas é comprimido de fato. Contudo esses trabalhos apresentam grande relevância de utilizar uma técnica de representação diferente da representação por bitmap para realizar compressão. Apesar do escopo reduzido da base de dados os autores afirmam obter taxas de compressão superiores ao estado da arte.

Codificação de Vizinhança

É proposta em [TTD99] uma nova maneira de codificar imagens através de relações de vizinhança. A codificação proposta transforma cada pixel preto de uma imagem binária em um vetor de vizinhança. Este vetor possui quatro dimensões. A primeira posição guarda o número de pixels do pixel preto que está sendo codificado até o próximo pixel branco acima ou no fim (na borda) da imagem na mesma direção, por isso essa posição do vetor é chamada Norte. A segunda posição é chamada Leste e é medida analogamente à primeira contudo noutra direção, à direita. As posições seguintes são Sul e Oeste e são medidas da mesma maneira nas direções abaixo e à esquerda, respectivamente. Portanto um vetor de vizinhança pode ser escrito como V = (n; l; s;o). A Figura 13 mostra um componente conexo de uma imagem binária, os quadrados cinza destacam a vizinhança representada pelo vetor de vizinhaça V = (5,2,9,4) e os quadrados brancos representam os demais pixels do componente conexo. Em [TTD99], esse vetor é então transformado em um código, que é um número inteiro. O método usado para fazer essa transformação gera um número muito grande de possíveis códigos e, como um código é gerado para cada pixel preto da imagem, muitos códigos distintos são produzidos. Portanto os vetores de vizinhança são re-escalados para que o número de possíveis códigos seja no máximo 100. Cada imagem é transformada num vetor de 100 características onde cada característica é a freqüência de cada um dos possíveis códigos. Uma rede neural é treinada com esses vetores de características e aplicada, com sucesso, para o problema de reconhecimento de caracteres manuscritos.

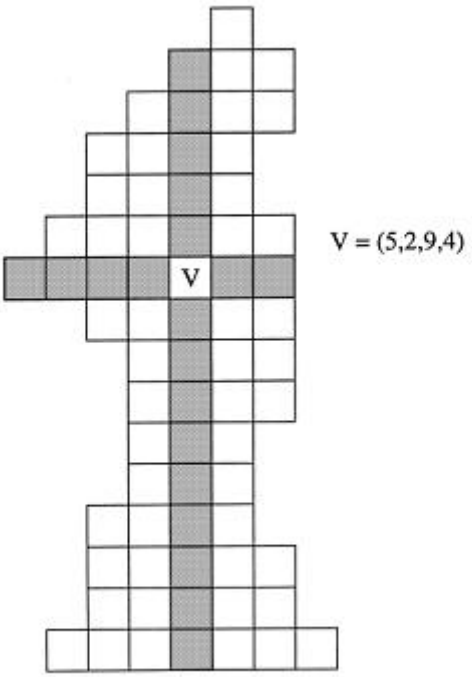


Figura 13. Vetor de vizinhança proposto por [TTD99]. A grade é conjunto de pixels pretos da imagem. Os quadrados cinza são os pixels representados pelo vetor de vizinhança V = (5,2,9,4) – 5 pixels na direção Norte, 2 na direção Leste, 9 na direção Sul e 4 na direção Oeste, V = (n; l; s;o).

Em [TT06a] e [TT06b] outras abordagens são utilizadas para o reconhecimento de padrões empregando os códigos extraídos a partir dos vetores de vizinhança, além disso são propostos os operadores de vizinhaça, que são funções lógicas e matemáticas aplicadas aos vetores de vizinhança que apresentam comportamento semelhantes a operadores morfológicos em imagens binárias. A análise de chi-quadrado é usada em [TT06a] para o reconhecimento de caracteres manuscritos. Em [TT06b] os métodos de agrumamento k-Média e Fuzzy k-Médias são usados para reduzir o número de códigos e novamente empregrados na tarefa de reconhecimento de caracteres manuscritos através de redes neurais. Uma grande limitação dos vetores de vizinhança descritos é o fato de não ser possível reconstruir a imagem original, como é discutido em [TTD99].

Os trabalhos apresentados a seguir são resultado da pesquisa que o autor do projeto desenvolveu no mestrado. Em [dCTT+10] a codificação de vizinhança é reformulada como método de representação de imagem, isto é, com esta nova definição a codificação de vizinhança é capaz de descrever, armazenar e reconstruir a imagem

original sem perdas. Ainda neste trabalho mais recente é proposta uma maneira de reduzir o número de vetores de vizinhança, sem inibir a reconstrução da imagem. Partindo dessa redução é proposto um método de compressão de imagens binárias o qual é comparado com os algoritmos de compressão descritos anteriormente: Group 4, GIF, PNG e JBIG. Ainda neste mesmo trabalho é proposta uma medida de distância que utiliza a codificação de vizinhança como descritor de forma e compara sua performance de reconhecimento na base *MPEG-7 Core Experiment Shape 1 part A2*, descrita anteriormente, com os descritores de forma também descritos anteriormente. Em [TdC+10] a discussão sobre compressão é aprofundada. Finalmente em [dCTT+10b] é dado um embasamento com maior formalismo à codificação de vizinhança descrevendo as vizinhanças através de funções deixando explicita a ligação entre um código de vizinhança e a área da imagem que ele representa, outras funções são utilizadas para demonstrar a reconstrução completada da imagem através dessas funções. Esta formalização é descrita a seguir.

Nova formalização da codificação de vizinhança

Pode-se definir um **código de vizinhança** ou **vetor XC** como um vetor d-dimensional no espaço N^d com d > 2.

$$\varphi = (x, y, c_1, c_2, \ldots, c_{d-2}) \in \mathbb{N}^d,\ d > 2. \quad \text{(eq. 1)}$$

A primeira posição do vetor é chamada x e representa a coluna do centro do XC; a segunda posição, y, representa a linha do centro; as demais posições c_i, 1 < i < d 2, são os comprimentos dos braços do XC, cada c_i está associado a exatamente uma função braço bi (vista adiante). Portanto, para definir um XC é preciso definir quais funções braço estão associadas com cada comprimento.

A codificação de vizinhança gera um vetor XC para cada pixel preto da imagem. Na Figura 14 é destacada a codificação de vizinhança de uma imagem 5 x 7. Esta imagem possui nove pixels pretos. Para cada um desses pixels é gerado um XC, o conjunto dos nove XCs é a codificação de vizinhança dessa imagem. Nesse exemplo esta vizinhança é idêntica àquela (n; l; s; o) proposta em [TTD99].

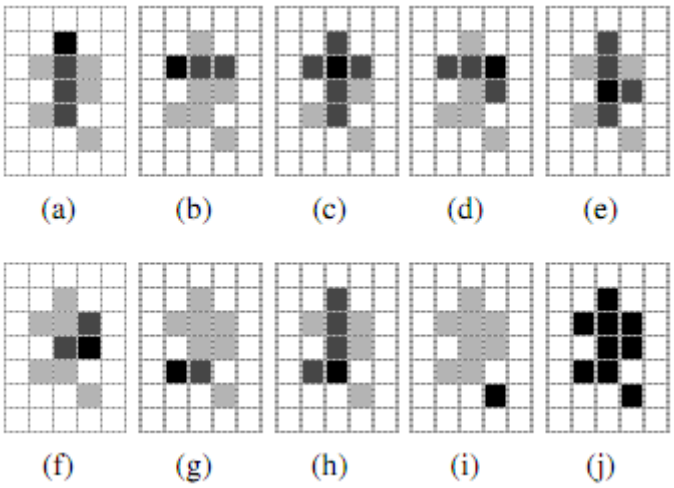


Figura 14.. Codificação de Vizinhança. As Figuras de (a) a (i) destacam os 9 códigos de vizinhança da imagem (j), cada código é representado por um vetor da forma (x; y; n; l; s; o). O centro de cada código está destacado na imagem em preto e as vizinhanças em cinza escuro. Um XC é gerado para cada pixel preto da imagem. Códigos das imagens: a = (2,1,0,0,3,0, b = (1,2,0,2,0,0), c = (2,2,1,1,2,1), d = (3,2,0,0,1,2), e = (2,3,2,1,1,0), f = (3,3,1,0,0,1), g = (1,4,0,1,0,0), h = (2,4,3,0,0,1), i = (3,5,0,0,0,0).

A segunda parte de um XC corresponde à vizinhança de um centro. Cada posição do vetor XC nesta parte de vizinhança é chamada braço ou comprimento do braço. Cada braço codifica um trecho da vizinhança. No trabalho [dCTT+10b] criado o conceito de braços para que a parte de vizinhança do código XC seja versátil. Assim, através da combinação de definições de braços pode-se criar diversas configurações de XC. Uma configuração de um XC é uma especificação de quais funções braços o código usa para representar sua vizinhança.

Uma **função braço** pode ser definida como uma função b no domínio das triplas de números naturais N^3 e contradomínio no conjunto das partes dos pares ordenados de números naturais menos o conjunto vazio $\wp$(N x N):

$$b : \mathbb{N}^3 \rightarrow \wp(\mathbb{N} \times \mathbb{N}). \quad \text{(eq. 2)}$$

Esta função b pode ser interpretada com uma função que tem como parâmetros três números naturais – as coordenadas (x; y) do centro do código e o comprimento do braço – e retorna um conjunto de pontos da imagem binária, onde cada ponto é uma posição na matriz da imagem binária. São utilizadas neste trabalho oito funções braço, que fazem analogia aos pontos cardeais:

Norte

$$b_N(x, y, c) = \{(x, y - k) \mid k = 0, 1, 2, \ldots, c; k \leq y\}, \quad \text{(eq. 3)}$$

Leste

$$b_L(x, y, c) = \{(x + k, y) \mid k = 0, 1, 2, \ldots, c\}, \quad \text{(eq. 4)}$$

Sul

$$b_S(x, y, c) = \{(x, y + k) \mid k = 0, 1, 2, \ldots, c\}, \quad \text{(eq. 5)}$$

Oeste

$$b_O(x, y, c) = \{(x - k, y) \mid k = 0, 1, 2, \ldots, c; k \leq x\}, \quad \text{(eq. 6)}$$

Nordeste

$$b_{NE}(x, y, c) = \{(x + k, y - k) \mid k = 0, 1, 2, \ldots, c; k \leq y\}, \quad \text{(eq. 7)}$$

Sudeste

$$b_{SE}(x, y, c) = \{(x + k, y + k) \mid k = 0, 1, 2, \ldots, c\}, \quad \text{(eq. 8)}$$

Sudoeste

$$b_{SO}(x, y, c) = \{(x - k, y + k) \mid k = 0, 1, 2, \ldots, c; k \leq x\}, \quad \text{(eq. 9)}$$

Noroeste

$$b_{NO}(x, y, c) = \{(x-k, y-k) \mid k = 0, 1, 2, \ldots, c; k \leq min[x, y]\}, \quad \text{(eq. 10)}$$

em que x é a posição horizontal, y a posição vertical do centro (x; y) e c é o comprimento do braço.

Chama-se **configuração do XC** (ou tipo do XC) um vetor de funções braço (eq. 2) que especifica quais funções braço são empregadas para definir a vizinhança de um código XC:

$$\beta = (b_1, b_2, \ldots, b_{d-2}) \in \mathrm{B}^{d-2},\ d > 2, \quad \text{(eq. 11)}$$

em que B é o conjunto de todas as funções braços, $b_i \in B$, $1 < i < d$ 2, e d é a dimensão do vetor XC que utiliza a configuração.

A seguir serão discutidas três configurações. Os nomes desses três tipos de códigos são retirados das peças do xadrez torre, bispo e rainha, porque cada uma dessas configurações possui vizinhança semelhante aos possíveis movimentos de cada uma dessas peças no jogo Figura 15.

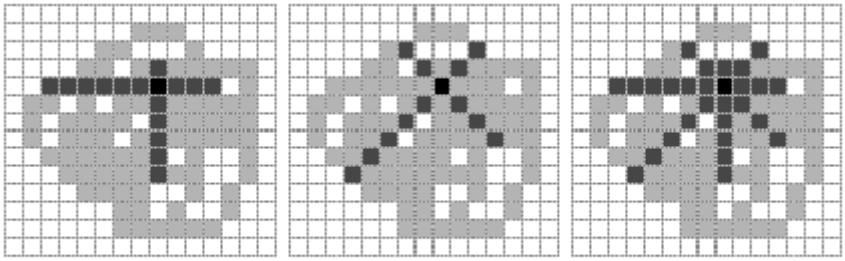


Figura 15. Exemplo de vetores XC torre (a), bispo (b) e rainha (c). A célula preta da grade é o centro do XC, as células cinza escuro são os pontos representados por cada XC e as cinza claro são os demais pontos de pixels da imagem, as células brancas são os pixels do fundo. O vetor XC 'torre = (8; 4; 1; 3; 5; 6) de (a) é do tipo torre. O vetor XC 'bispo = (8; 4; 2; 3; 5; 2) de (b) é do tipo bispo. O vetor XC 'rainha = (8; 4; 1; 2; 3; 3; 5; 5; 6; 2) de (c) é do tipo rainha.

O vetor XC torre utiliza a configuração

$$\beta^{torre} = (b_N, b_L, b_S, b_O). \quad \text{(eq. 12)}$$

O vetor XC bispo utiliza a configuração

$$\beta^{bispo} = (b_{NE}, b_{SE}, b_{SO}, b_{NO}). \quad \text{(eq.13)}$$

O vetor XC rainha utiliza a configuração

$$\beta^{rainha} = (b_N, b_{NE}, b_L, b_{SE}, b_S, b_{SO}, b_O, b_{NO}). \quad \text{(eq. 14)}$$

A função $u : \mathbb{N}^d \times \mathrm{B}^{d-2} \rightarrow \wp(\mathbb{N} \times \mathbb{N})$ é chamada função vizinhança de um XC. Esta função associa um vetor XC e sua configuração de funções braço a um conjunto de pontos numa imagem binária:

$$u(\varphi, \beta) = \bigcup_{i=1}^{d-2} b_i(x, y, c_i), \quad \text{(eq. 15)}$$

em que φ está definido na equação 1, β está definido na equação 11, b_i é cada função braço do vetor , x é a posição horizontal do centro de φ , y é a posição vertical do centro de φ , c_i é o comprimento de cada braço de φ associado à função b_i e d é a dimensão de φ .

Uma imagem binária pode ser reconstruída sem perdas com as informações sobre suas dimensões (altura e largura) e a posição de todos os pixels pretos da imagem. Essas posições podem ser recuperadas a partir do conjunto Φ dos vetores XC da imagem. A função de vizinhança de um conjunto Φ de XCs $v : \wp(\mathbb{N}^d) \times \mathrm{B}^{d-2} \rightarrow \wp(\mathbb{N} \times \mathbb{N})$ gera todos os pontos pretos da imagem através da união dos conjuntos de pontos gerados por $u(\varphi_i, \beta)$ para cada XC:

$$v(\Phi, \beta) = \bigcup_{i=1}^{m} u(\varphi_i, \beta), \quad \text{(eq. 16)}$$

em que m é o número de elementos (vetores XC) de Φ.

Um código XC é gerado para cada pixel preto da imagem. O centro desse código é a posição (x; y) do pixel preto. Para calcular o comprimento dos braços de um XC $\bar{\varphi}_{xy} = (x, y, c_1, c_2, \ldots, c_{d-2})$, dada a posição (x; y) do seu centro, deve-se maximizar $|u(\varphi, \beta)|$:

$$\varphi_{xy} = \arg\max(|u(\varphi, \beta)|), u(\varphi, \beta) \subseteq \Pi \quad \text{(eq. 17)}$$

em que Π é conjunto de todas as posições dos pixels pretos de uma imagem binária P, (x; y) é a posição do centro fixado para φ_{xy}, as variáveis a serem maximizadas (c_1;

c_2; ... ; c_{d-2}) são os comprimentos de cada braço associado à configuração $\beta = (b_1, b_2, \ldots, b_{d-2})$, d é a dimensão de φ_{xy} e $u(\varphi, \beta)$ está expresso na equação 15.

Redução do conjunto de códigos de vizinhança

Muitos pixels pretos de uma imagem representada pela codificação de vizinhança são codificados por mais de um vetor XC, uma vez que cada vetor representa um conjunto de pixels pretos e um vetor é gerado para cada pixel preto. Portanto o conjunto de códigos XC que representam uma imagem é redundante. Algoritmos para a redução desse conjunto de códigos são sugeridos, com a propriedade de ser possível reconstruir, sem perdas, a mesma imagem representada pelo conjunto de códigos completo. Na dissertação em que se baseia este trabalho são propostos três algoritmos para a redução do conjunto de códigos, sendo um deles um algoritmo genético. Um desses algoritmos, chamado RED2 (Redução 2), é exposto a seguir.

O RED2 baseia-se na idéia de poder de codificação do XC, que é o número de pixels representados pelo XC. O RED2 seleciona aqueles vetores XC que representam mais pixels, com uma peculiaridade: o poder de codificação de cada XC no RED2 é atualizado toda vez que um XC é selecionado. No RED2, cada vez que um XC é selecionado, todos os pixels representados por este código são marcados como codificados, e todos os outros códigos XC que representam algum desses pixels têm seu poder de codificação decrementado.

O Algoritmo 1 descreve o funcionamento do RED2. Este método recebe como entrada um conjunto Φ de XCs e uma configuração β de XC. O resultado desse processo é um conjunto $\Phi' \subseteq \Phi$, geralmente menor Φ e suficiente para representar a mesma imagem. Na linha 1 do algoritmo, é criada e inicializada a lista Φ' que irá conter os códigos selecionados pelo algoritmo. Na linha 2, é criada e inicializada a variável r que conta quantos pixels da imagem ainda restam para ser codificados. Esta variável é inicializada com o número de pixels pretos da imagem $|v(\Phi, \beta)|$ (equação 16), pois nenhum pixel da imagem foi codificado ainda. A matriz A = [a_{xy}] é criada e inicializada na linha 3. Esta matriz marca as posições (x; y) dos pixels codificados por Φ'. Quando a_{xy} = 0 indica que o pixel (x; y) ainda não foi codificado, quando a_{xy} = 1 indica que o

pixel (x; y) já foi codificado. Como nenhum pixel foi codificado, A é inicializado com zero. A lista D (linha 4) guarda o poder de codificação di de cada código XC $\varphi_i \in \Phi$. Como nenhum código foi selecionado ainda, d_i é inicializado como o número de pontos $|u(\varphi_i, \beta)|$ (eq. 15) gerado pelo XC φ_i.

Algoritmo 1: Redução do Conjunto de XCs

Entrada: Lista de codigos de vizinhança $\Phi = \{\varphi_i \mid i = 1, \ldots, m\}$;
Entrada: Configuração do código de vizinhança β;
Saída: Lista reduzida de codigos de vizinhança $\Phi' = \{\varphi_i \mid i = 1, \ldots, m'; m' \leq m\}$;

// INICIALIZAÇÃO DAS VARIÁVEIS
1 crie Φ'; inicie Φ' como uma lista vazia;
2 crie r; inicie $r = |v(\Phi, \beta)|$;
3 crie uma matriz $A = [a_{xy}]$; inicie $A = 0$;
4 crie uma lista $D = \{d_i \mid i = 1, \ldots, m\}$;
inicie $d_i = |u(\varphi_i, \beta)|$, $\forall \varphi_i \in \Phi, d_i \in D, 1 \leq i \leq m$;
// SELEÇÃO DOS CÓDIGOS
5 **enquanto** $r > 0$ **faça**
6 $\quad \varphi_l = \varphi_i \mid d_i \geq d_j \forall \varphi_i, \varphi_j \in \Phi$;
7 $\quad$ adicione φ_l a Φ';
8 $\quad r \leftarrow r - d_l$;
9 $\quad$ **para todo** $(x, y) \in u(\varphi_l, \beta)$ **faça**
10 $\quad\quad$ **se** $a_{xy} = 0$ **então**
11 $\quad\quad\quad a_{xy} = 1$;
12 $\quad\quad\quad$ **para todo** $d_k | (x, y) \in u(\varphi_k, \beta)$ **faça**
13 $\quad\quad\quad\quad d_k \leftarrow d_k - 1$;
14 $\quad\quad\quad$ **fim**
15 $\quad\quad$ **fim**
16 $\quad$ **fim**
17 **fim**
18 **retorna** Φ';

O laço iterativo do RED2 continua enquanto r > 0, ou seja, enquanto existirem pixels não codificados na imagem (linha 5). A cada iteração desse laço, é selecionado o código φ_l com o maior poder de codificação d_l na iteração atual, isto é, o código escolhido é aquele que representa o maior número de pixels (linha 6). Este código φ_l é adicionado à lista de resposta Φ' do algoritmo (linha 7). E o número de pixels codificados r é decrementado do poder de codificação d_l do código φ_l, (r ← r - d_l), na linha 8. Então, para cada ponto (x; y) representado pelo código φ_l selecionado na iteração (linha 9), desde que o ponto não esteja marcado como codificado, a_{xy} = 0,

(linha 10), são atualizadas: a matriz A que marca os pixels codificados e o poder de codificação d_k de cada código φk afetado com a inclusão de φl em Φ'. Cada pixel (x; y) representado por φl é marcado como codificado, $a_{xy} = 1$, na linha 11. E o poder de codificação de cada código φk que também codifica (x; y) (linha 12) é decrementado em uma unidade (linha 13).

Compressão de imagens

O método de compressão aqui discutido é dividido em seis partes, como mostrado no diagrama da Figura 16. (1) Reduzir o conjunto de códigos de vizinhança. Essa primeira etapa do método de compressão tem uma importância crucial em eliminar os códigos redundantes de uma codificação de vizinhança. (2) Agrupar os códigos por vizinhança. O vetor XC é dividido em duas partes, a primeira parte é chamada centro e a segunda parte é chamada vizinhança. Esta etapa do método de compressão faz com que os vetores XC que tem a parte de vizinhança com os mesmo valores fiquem em seqüência na memória. (3) Separar a vizinhança dos centros. Os vetores XC são repartidos em suas duas partes (centro e vizinhança). Então todos os centros são movidos para o começo e as vizinhanças para o final da lista de números. (4) Aplicar RLE às vizinhanças. RLE (*Run Length Encoding*) é um algoritmo de compressão de dados que reduz o comprimento de cadeias de caracteres substituindo trechos dessa cadeia onde um símbolo se repete por pares símbolo e contador de repetições. Nessa etapa do método de compressão, o RLE trata cada vetor de braços como um símbolo. Cada série de vetores de braços iguais é substituída por apenas um vetor braço e um contador, que é um número indicando quantas vezes esse vetor se repete. (5) Aplicar RLE aos contadores do outro RLE. Para reduzir a lista de contadores gerados, é aplicado RLE à lista de contadores gerada pelo RLE que foi aplicado aos vetores de braços. (6) Aplicar a codificação Huffman. A codificação Huffman [3] é usada para transformar a cadeia de inteiros que representa a codificação de vizinhança em uma cadeia de bits compacta, utilizando menos bits para representar os inteiros mais freqüentes.

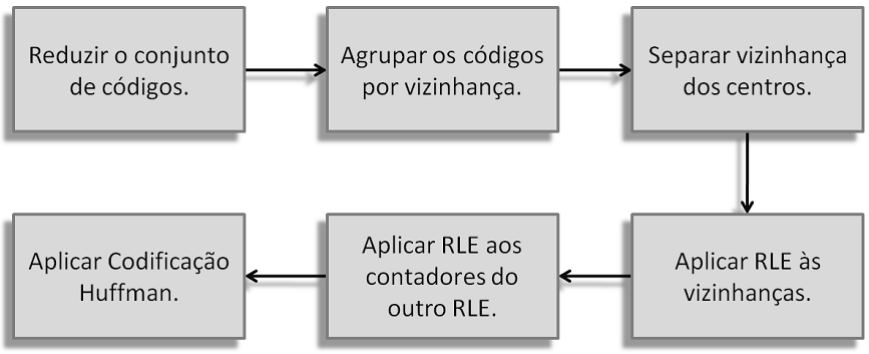


Figura 16. Diagrama de fluxo do método de compressão proposto. O processo é dividido em seis etapas: reduzir o conjunto de códigos, agrupar os códigos por vizinhança, separar a vizinhança dos centros, aplicar RLE às vizinhanças, aplicar RLE aos contadores do outro RLE e, por último, aplicar a codificação Huffman.

A Tabela 2 mostra alguns dos resultados desse experimento. Os resultados são expressos em bpp (bits por pixel), isto é, o número de bits necessário para representar cada pixel da imagem. As imagens utilizadas nos resultados dessa tabela estão expostas na Figura 17. O método proposto, como mostra a tabela, apresentou o segundo melhor resultado em três dos cinco casos. O método se mostrou adequado para compressão de imagens de símbolos e desenhos (five, bell e hand), mas não apresentou bons resultados na compressão de imagens *halftone* (cat) ou imagens de texto (courier12). O método JBIG comprovou ser o estado da arte na compressão de imagens binárias.

Tabela 2. Comparação entre os métodos de compressão propostos utilizando a redução de códigos red2 com as configurações torre (red2t) e rainha (red2r) e os formatos png, gif e jbig. Os resultados estão expressos em bpp (bits por pixel). *melhor e **segunda melhor compressão para a imagem.

Imagens	RED2T	RED2R	PNG	GIF	JBIG
five	0,170**	0,181	0,263	0,184	0,091*
bell	0,528**	0,581	0,831	0,538	0,314*
hand	0,192	0,157**	0,222	0,200	0,103*
cat	8,154	2,565	0,365**	0,473	0,283*
courier12	0,642	0,539	0,353**	0,383	0,192*

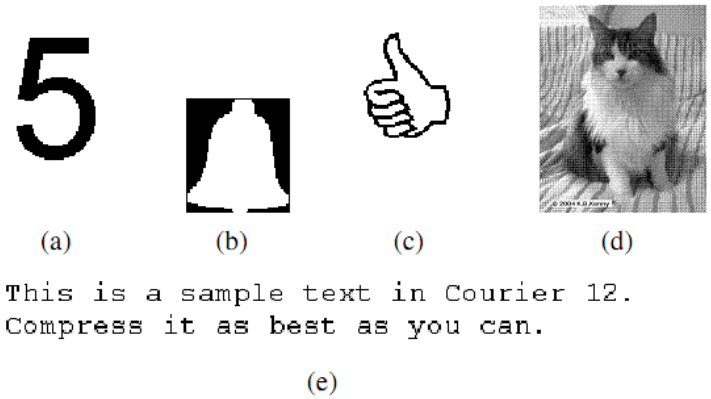


Figura 17. Subconjunto das imagens de teste: (a) five, 128x128; (b) bell,59x64; (c) hand, 129x36; (d) cat, 380x469; (e) courrier12, 374x46.

Reconhecimento de Forma

Esta seção relata a utilização da codificação de vizinhança como descritor de forma e avalia o método proposto utilizando a base *Core Experiment CE-Shape-1 parte A2*. A base de imagens A2 avalia a robustez do descritor em relação a variações de rotação, como descrito anteriormente.

Este método proposto para o reconhecimento de formas extrai os códigos de vizinhança de uma imagem, então reduz o conjunto de códigos extraídos através do algoritmo de redução RED2. E calcula a similaridade entre duas imagens comparando seus conjuntos de códigos através de um algoritmo de *template matching*. Antes da extração dos códigos, a rotação de cada imagem é corrigida por um processo chamado padronização de rotação. Esta padronização consiste em alinhar, com o eixo horizontal, a direçãode maior variância da imagem. A padronização é realizada utilizando a técnica PCA (*Principal Component Analysis*) sobre a posição (x; y) dos pixels pretos da imagem.

O experimento foi realizado utilizando o teste padrão para a base *Core Experiment CE-Shape-1 A2* e compara os resultados com os métodos P298, P320, P517, P567 e P687, descritos anteriormente. Os resultados do experimentos estão expostos na Tabela 3. Percebe-se que o método proposto apresentou um resultado tão bom quanto os outros métodos. A taxa de acerto de 99,52% indica que apenas 12 das 2520 imagens da base de teste foram incorretamente classificadas. Apesar de não se ter amostras suficientes para se fazer um teste de hipótese, é aceitável assumir que não há diferença entre a taxa de acerto do método proposto e as maiores taxas de acerto da mesma tabela.

Tabela 3. Taxas de acertos no reconhecimento de forma do método proposto, sob a base ce-shape-1 a2, comparado com os descritores P298, P320, P517, P567 E P687.

Base	P298	P320	P517	P567	P687	Proposto
A2	100,0	99,37	100,0	97,46	99,60	99,52

Conclusão

O autor do presente projeto desenvolveu no mestrado os formalismos apresentados para a codificação de vizinhança, bem como os algoritmos e técnicas para compressão e reconhecimento de formas. Tudo o que foi construído já aponta que a representação de vizinhança encontra-se no caminho para maturidade. Contudo tanto o formalismo, como as técnicas precisam ser aprofundadas. Mais técnicas utilizando esta representação devem ser desenvolvidas para que a codificação de vizinhança possa estabelecer-se com contribuição definitiva para a área de processamento de imagens. Os objetivos para a extensão, entretanto, serão discutidos em outra seção.

Metodologia

O trabalho de doutorado proposto pretende ser desenvolvido em torno de três pilares: revisão bibliográfica, pesquisa e realização de publicações. A revisão bibliográfica consiste em averiguar os trabalhos relacionados nas áreas de representação de imagens, codificação de imagens, compressão de imagens e reconhecimento de imagens. Esta revisão deve priorizar os trabalhos mais recentes e os periódicos/conferências principais da área.

As realização de pesquisas visa a proposta e desenvolvimento de soluções na forma de teoria com fundamentação matemática sólida, algoritmos eficientes e implementação em software. Cada proposta realizada na pesquisa deve ser analisada e mensurada por meio de experimentos coerentes de modo que seja possível comparar as propostas com os demais resultados recuperados a partir da revisão de literatura.

Objetiva-se que as implementações de *software* não necessitem de nenhum recurso computacional além daqueles disponibilizado nos computadores pessoais contemporâneos. Uma vez construídas e validadas propostas que devem apresentar resultados comparáveis ao estado-da-arte na área de pesquisa, pretende-se publicar os resultados e os métodos desenvolvidos nos principais periódicos/conferências da área.

Entre os principais periódicos da área destacam-se

- *IEEE Transactions on Image Processing* (qualis A1)*,*
- *IEEE Transactions on Pattern Analysis and Machine Intelligence* (qualis A1)*,*
- *Computer Vision and Image Understanding* (qualis A1),
- *Pattern Recognition Letters* (qualis A2),
- *Image and Vision Computing* (qualis A2) e
- *ACM Computing Surveys* (qualis A1).

Entre as principais conferências da área destacam-se:

- ICIP (qualis A2) *IEEE* International Conference on Image Processing,

- ICASSP (qualis B2) *IEEE International Conference on Acoustics, Speech, and Signal*,
- CIVR (qualis B2) *ACM International Conference on Image and Videl Retieval* e
- ICIAP (qualis B2) – *Intenational Conference on Image Analysis and Processing.*

Objetivos

Este trabalho pretende construir um *survey* sobre imagens binárias incluindo os seguintes tópicos: representação de imagens binárias, compressão de imagens binárias e reconhecimento de formas. Além desses tópicos principais, cogita-se explorar também os tópicos de segmentação de *layout* de documentos (reconhecimento de regiões de um documento entre texto, figura etc.), armazenamento de imagens de impressões digitais, mapas de relevância (imagens binárias que marcam a relevância de alguns pontos da imagem). Tem-se a intenção de submeter tal *survey* para *ACM surveys*, um dos periódicos mais conceituados na área de Ciências da Computação. Pretende-se publicar os resultados das pesquisas da teste de doutorado no periódicos *IEEE Transaction on Image Processing*.

Alguns dos objetivos específicos estão descritos abaixo, evidentemente alguns objetivos novo podem surgir, assim como objetivos antigos podem ser suprimidos:

• Análise da complexidade algorítmica de processamento e memória dos métodos propostos. Incluindo os métodos que serão propostos como os já conhecidos para codificação de vizinhança, redução do número de códigos e reconhecimento de formas utilizando esta codificação.

• Propõe-se criar uma taxonomia de imagens binárias que leva em consideração a dispersão dos pixels pretos na imagem, algumas possíveis classes desta taxonomia são: *halftone*, desenho, texto, linhas finas, linhas grossas. Um conjunto de funções braço pode ser definido para representar cada tipo de imagem da taxonomia, dessa forma espera-se alcançar uma maior redução no conjunto de códigos necessário para representar a imagem sem perdas e um aumento na taxa de compressão usando codificação de vizinhança. Uma possível forma de definir essas funções braços é através de um algoritmo genético.

• Testar novas configurações de funções braço. Um estudo preliminar aponta ganho na compressão usando codificação de vizinhança usando diferentes configurações de funções braço para uma mesma imagem.

• Os métodos propostos para a redução do conjunto de códigos [dCTT+10] não são ótimos, espera-se encontrar um algoritmo que seja ótimo e computacionalmente pouco custoso para essa tarefa.

• Segundo [Hol75] um algoritmo genético é capaz de encontrar a resposta ótima para um problema, dado tempo suficiente, portanto acredita-se ser possível reformular o algoritmo genético para redução do conjunto de códigos de forma a alcançar sempre a redução ótima.

• Espera-se também que seja possível encontrar uma forma de estimar o número mínimo de códigos de vizinhança necessários para representar uma imagem. Essa estimativa pode ser usada como critério de parada para o algoritmo genético para redução do conjunto de códigos.

• Os algoritmos para redução do conjunto de códigos leva em consideração apenas o número de códigos gerados, contudo, para a tarefa de compressão os algoritmos para a redução do conjunto de códigos devem levar em consideração a entropia do conjunto reduzido de códigos.

• O método de compressão usando codificação de vizinhança mostrou-se mais adequado para imagens com grandes áreas contínuas, portanto deve ser avaliado em outras bases que possuam imagens com essa característica. Uma base de imagens com essa característica e um método novo que está sendo empregado para compressão desse tipo de imagem é proposto em [ACNCRD09].

• Para atenuar as limitações da compressão com código de vizinhança em imagens grandes, sugere-se o uso de técnicas de análise multi-resolução, com por exemplo, pirâmides de imagem [GW10]. Cada pixel nas subimagens de menor resolução numa pirâmide de imagens representa vários pixels numa subimagem de maior resolução na mesma pirâmide. Se a codificação de vizinhança for empregada nas imagens de menor resolução, cada código de vizinhança representará mais pixels numa subimagem de maior resolução.

• Em [TT06b, TT06a] foram propostos operadores de vizinhança sobre os códigos de vizinhança que apresentam funcionamento semelhante a operadores morfológicos. Essa é um tópico que pode ser explorado com mais profundidade.

• Uma vez que um algoritmo genético pode ser bastante custoso em tempo, é possível propor um formato de arquivo para compressão que seja capaz de ser otimizado com o tempo. Através de um algoritmo genético pode-se definir um método que tenta sempre reduzir o tamanho da imagem comprimida com codificação de vizinhança, fazendo uso da capacidade de processamento ociosa de um computador. Ou mesmo, imagens que possuem partes idênticas, podem trocar informação para reconhecer que possuem a mesma informação e uma pode usar a representação da outra, se essa for mais compacta.

• Estender a codificação de vizinhança para imagens em tons de cinza e imagens multi-espectrais. Uma proposta para usar codificação de vizinhança em imagens em tons de cinza é: realizar segmentação da imagem, representar cada segmentos usando codificação de vizinhança, armazenar a textura de cada segmento.

• A compressão usando codificação de vizinhança supera a compressão MH do padrão CCITT Group 3, que é empregada até hoje para a transmissão de fax via rede telefônica. Esse padrão já é empregado há 40 anos e ainda não foi substituído porque utiliza correção de erros. O codificação de vizinhança é redundante, o que facilita a criação de correção de erros para essa codificação. Uma proposta de trabalho futuro é avaliar a relevância de um novo padrão para compressão de imagens binária, resiliente a erros de transmissão, que pode ser usado para transmissão de fax em redes analógicas.

Cronograma

Março de 2012

- Cursar disciplinas.
- Projeto e implementação de um método de rotação rápida utilizando codificação de vizinhança, este método deve ter um custo computacional menor do que uma rotação em uma imagem binária canônica.
- Finalização de um artigo para *Pattern Recognition Letters.*

Abril de 2012

- Cursar disciplinas.
- Finalização de um artigo para *Pattern Recognition Letters.*
- Submissão deste artigo.

Maio a setembro de 2012

- Cursar disciplinas.
- Levantar artigos recentes e *surveys* relacionados ao tema e escrever um *survey* para *ACM Computing Surveys*.

Agosto de 2012

- Cursar disciplinas.
- Submissão do *survey* para *ACM Computing Surveys*.

Setembro de 2012 a Fevereiro de 2013

- Cursar disciplinas.

- Desenvolvimento de técnicas de processamento de imagens utilizando codificação de vizinhança, visando mostrar a versatilidade deste método de representação de imagens.

Referência Bibliográfica

[AKF06] Alexander Akimov, Alexander Kolesnikov, Pasi Fränti., Lossless compression of map contours by context tree modeling of chain codes, Pattern Recognition (2006)

[AKF08] Naif Alajlan, Mohamed S. Kamel, and George H. Freeman. *Geometry-Based Image Retrieval in Binary Image Databases*. IEEE Transactions On Pattern Analysis And Machine Intelligence, vol. 30, no. 6, june 2008

[ACNCRD09] Sergio Alcaraz-Corona, Ricardo A. Neri-Calderón, and Ramón M. Rodríguez- Dagnino. *Efficient bilevel image compression by grouping symbols of chain coding techniques*. Optical Engineering, 48(3), March 2009.

[ACRD10] S. Alcaraz-Corona and R.M. Rodriguez-Dagnino, *Bi-level image compression estimating the markov order of dependencies*, IEEE Journal of Selected Topics in Signal Processing, vol. 4, no. 3, pp. 605–611, 2010.

[Bar06] Mauro Barni. *Document And Image Compression.* CRC Press, May 2006.

[BYL+10] X. Bai, X. Yang, L. Latecki, W. Liu, and Z. Tu. *Learning context-sensitive shape similarity by graph transduction*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2010.

[CK96] G. Chuang and C. C. Kuo. *Wavelet descriptor of planar curves: theory and aplications.* IEEE Tranasactions on Image Processing, 5:56–70, 1996.

[Cor02] Thomas H. Cormen. *Algoritmos: teoria e prática*. Elsevier. 2002

[dA07] Carlos Wilson Dantas de Almeida. *Recuperação de imagens baseada em uma abordagem híbrida.* Master's thesis, UFPE- CIn, 2007.

[dCTT+10] Tiago Buarque Assunção de Carvalho, D. J. Tenório, I. R. Tsang, G. D. C. Cavalcanti, and I. J. Tsang. *Neighborhood coding for bilevel image compression and shape recognition.* In IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 2010.

[dCTT+10b] de Carvalho, T. B. A. ; TENORIO, D. J. ; CAVALCANTI, G. D. C. ; TSANG, I. R. . *Codificação de Vizinhança para Compressão de Imagens e Reconhecimento de Forma.* In: 23rd SIBGRAPI Conference on Graphics, Patterns and Images, 2010.

[Dou94] Edward R. Dougherty. *Digital image processing methods*. CRC Press, January 1994.

[GA10] J. Gailly and M. Adler. *Compression algorithm (deflate)*, 2010.

[GA96] J. Gailly and M. Adler. *RFC1951 - DEFLATE Compressed Data Format Specification version 1.3*. Aladdin Enterprises, May 1996.

[GW10] Rafael C. Gonzalez and Richard E. Woods. *Processamento digital de imagens.* PEARSON, 3 edition, 2010.

[Hol75] John H. Holland. *Adaptation in natural and artificial systems.* University of Michigan Press, 1975.

[Huf52] D. A. Huffman. *A method for the construction of minimum-redundancy codes.* Proceedings of the I.R.E (Institute of Radio Engineers), pages 1098–1101, September 1952.

[IT88] (CCITT) ITU-T. Recommendation T.6. Facsimile Coding Schemes And Coding Control Functions For Group 4 Facsimile Apparatus, 1988.

[IT03] (CCITT) ITU-T. Recommendation T.4. Standardization of Group 3 Facsimile Terminals for Document Transmission, july 2003.

[JBB+98] Corinne Le Buhan Jordan, S. Bhattacharjee, F. Bossen, F. Jordan, and T. Ebrahimi. *Shape representation and coding of visual objects in multimedia applications - an overview.* Annals of Telecommunications, 53(5-6):164–178, 1998.

[kH90] A. Khotanzan and Y. H. Hong. *Invariant image recognition by Zernike moments.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 12:489–497, 1990.

[LL00] L. J. Latecki and R. Lakämper. *Shape similarity measure based on correspondence of visual parts.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 22:1185–1190, 2000.

[LLE00] Longin Jan Latecki, Rolf Lakämper, and Ulrich Eckhardt. *Shape descriptors for non-rigid shapes with a single closed contour*. In IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), pages 424–429, 2000.

[LR01] Subhash R. Lele and Joan T. Richtsmeier*. An invariant approach to analysis of shapes.* Interdisciplinary Statistic Series. Chapman & Hall / CRC, 2001.

[Mar04] José M. Martínez. ISO/IEC JTC1/SC29/WG11N6828 MPEG-7 Overview (version 10), October 2004.

[Par02] Alvaro Pardo. *Lossless shape representation and coding*. Technical report, Facultad de Ingeniería, Universidad de la República, C.C. 30, Montevideo, Uruguay, April 2002.

[RMB09] Rahul Raguram, Michael W. Marcellin, and Ali Bilgin. *Improved Resolution Scalability for Bilevel Image Data in JPEG2000*. IEEE Transactions on Image Processing, vol. 18, no. 4, april 2009.

[RMB97] M. M. Reid, R. J. Millar, and N. D. Black. *Second-generation image coding: An overview.* ACM Computing Surveys, 29(1), 1997.

[SCBRD07] Hermilo Sánchez-Cruz, Ernesto Bribiesca, and Ramón M. Rodríguez-Dagninoc. *Efficiency of chain codes to represent binary objects.* Pattern Recognition, 40:1660–1674, 2007.

[TdC+10] TENORIO, D. J. ; de Carvalho, T. B. A. ; CAVALCANTI, G. D. C. ; TSANG, I. R. . Lossless Binary Image Compression Using Neighborhood Coding. 2010.

[SCRD05] Hermilo Sánchez-Cruz and Ramón M. Rodríguez-Dagnino. *Compressing bilevel images by means of a three-bit chain code.* Optical Engineering, 44(9), September 2005.

[TS11] Tabus, I., Sarbu, S., *Optimal structure of memory models for lossless compression of binary image contours*, Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on, On page(s): 809 -812, Volume: Issue: 22-27 May 2011

[TT06a] I. R. Tsang and I. J. Tsang. *Image Analysis and Recognition*, volume 4141/2006 of Lecture Notes in Computer Science, chapter Pattern Recognition Using Neighborhood Coding, pages I: 600–611. Springer Berlin / Heidelberg, September 2006.

[TT06b] I. R. Tsang and I. J. Tsang. *Neighbourhood vector as shape parameter for pattern recognition.* In International Joint Conference on Neural Networks (IJCNN), pages 3204–3209, July 2006.

[TTD99] I. J. Tsang, I. R. Tsang, and D. Van Dyck. *Image coding using neighbourhood relations*. Pattern Recognition Letters, 20:1279–1286, 1999.

[WMB99] Ian H. Witten, Alistair Moffat, and Timothy C. Bell. *Managing gigabytes: compressing and indexing documents and images.* TheMorgan Kaufman Series in Multimedia Information and Systems. Morgan Kaufman Publishers, 2nd ed. edition, 1999.

[Zib01] Carla Raquel Faria Lopes Zibreira. *Descrição e procura de vídeo baseadas na forma.* Master's thesis, Image Group - Instituto de Telecomunicações, Instituto Superior Técnico, Av. Rovisco Pais - 1049-001 Lisboa, PORTUGAL, Julho 2001.

[ZL77] Jacob Ziv and Abraham Lempel. *A universal algorithm for sequential data compression.* IEEE Transactions on Information Theory, 23(3):337 – 343, 1977.