# 3
# Afinamento topológico digital

Como foi dito na introdução, o algoritmo de reconstrução apresentado neste trabalho baseia-se na subdivisão do espaço em células retangulares de dimensões idênticas. Para construirmos uma imagem digital a partir dessas células e dos pontos de amostragem, cada uma das células do espaço será classificada como "cheia" ou "vazia"[1]: as células que possuem pontos de amostragem em seu interior serão marcadas como cheias, e as demais, como vazias. O conjunto de todas as células e seus respectivos estados forma uma **imagem digital**. Como o conjunto das células cheias representa a curva que queremos reconstruir, chamamos este conjunto de **objeto da imagem**, e ao conjunto das células vazias, chamamos **fundo da imagem**.

Um problema deste processo é que o ruído presente na amostragem dos pontos pode acarretar em um objeto digital com espessura maior que uma célula. Como vamos construir a curva "ligando" pontos nas células cheias, só teremos sucesso na reconstrução se a espessura do objeto for de apenas uma célula. Por isso, vamos processar esta imagem digital através de operações de afinamento topológico, de modo que a imagem obtida seja homotópica à imagem original e tenha espessura unitária.

Uma maneira intuitiva de ver o afinamento é como uma operação que "remove uma camada de células" de um objeto digital (ou mais especificamente, de um conjunto de células cheias), transformando células de bordo do objeto em células de fundo. O afinamento topológico preserva a homotopia do objeto, isto é, preserva o número de componentes do objeto e do fundo da imagem original, bem como a relação de interioridade dos componentes. Sucessivas operações de afinamento resultarão no que chamamos de "esqueleto" do objeto, que é uma boa representação da forma do objeto com espessura de um pixel e que mantém as características topológicas do mesmo. Operações de afinamento e "esqueletonização"são amplamente utilizadas em processamento de imagens digitais, e possuem uma vasta gama de aplicações. Dentre estas,

[1]Usualmente, os trabalhos sobre imagens digitais utilizam a terminologia "*pixels* pretos"e "*pixels* brancos", denotando células cheias e células vazias. Porém, neste trabalho, utilizaremos o termo "célula", para não criar confusão com os pontos de amostragem.

podemos destacar processamento de imagens médicas e biológicas, aplicações em biometria e digitalização de mapas topográficos. Neste trabalho, o afinamento topológico digital é utilizado na reconstrução de uma curva a partir de seus pontos de amostragem, como veremos neste capítulo.

Neste trabalho são comparados três implementações para o algoritmo de afinamento topológico digital. A primeira implementação é uma implementação tradicional, encontrada nos principais livros sobre processamento e morfologia de imagens, como em [21] e [8]. A segunda leva em consideração o número de pontos no interior de cada célula, removendo primeiro as células com menos pontos em seu interior. Esta abordagem possibilita selecionar células com maior probabilidade de interseção com a curva a ser reconstruída, tendo mostrado bons resultados e podendo ser implementada de maneira eficiente.

## 3.1 Definições, notações e conceitos básicos

Nesta seção, serão apresentadas as definições e notações utilizadas nas demais seções do capítulo.

### Esqueleto de uma imagem

A definição formal de esqueleto é: seja um conjunto $X$ e sua fronteira $\partial X$. Um ponto $x \in X$ pertence ao esqueleto de $X$ se a distância euclideana de $x$ até o complementar de $X$ é atingida em pelo menos dois pontos de $\partial X$:

$$x \in \mathbf{esq}(x) \iff \exists y_1, y_2 \in \partial X, y_1 \neq y_2, d(x, y_1) = d(x, y_2)$$

Esta definição formal de esqueleto pode ser entendida como a união dos centros dos discos maximais contidos no objeto, onde um disco maximal é um disco contido no objeto que toca seu bordo em no mínimo dois pontos. Outra maneira de entender o esqueleto de uma imagem é através da frente de propagação do fogo, iniciando nos bordos dos componentes do objeto, e se movendo-se de forma isotrópica. O esqueleto é constituído pelos pontos onde as frentes de propagação se encontram, e o fogo é extinto. A figura 3.1 ilustra este dois conceitos.

Quando o problema de esqueletonização de uma imagem é visto sob a forma digital, o esqueleto passa a ser um conjunto de pixels que formam uma curva digital. A extensão de algoritmos de esqueletonização para tratar imagens digitais não é direta, já que as noções de propagação da frente ou de

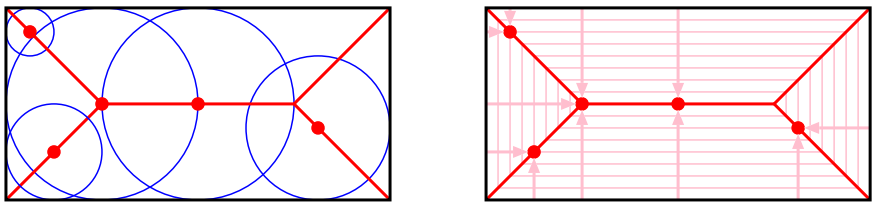


Figure 3.1: Esqueleto de um objeto

discos maximais não possuem um equivalente discreto [21]. Outro ponto é que a curva digital não é infinitamente fina como uma curva contínua, mas possui espessura equivalente a uma célula. Imagine o caso de um retângulo digital, com altura de duas células e largura maior que duas células: não é possível obter um conjunto de células que possua as propriedades de esqueleto vistas anteriormente, pois qualquer pixel escolhido estará sobre o bordo do objeto. Logo, os algoritmos de "esqueletonização digital" encontrados na literatura são no máximo aproximações para o esqueleto real do objeto. De qualquer modo, a "esqueletonização digital" permite extrair boas informações sobre a forma e a topologia do objeto.

Para compreendermos os algoritmos de esqueletonização propostos neste trabalho, são necessárias algumas definições básicas sobre processamento digital de imagens. Estas definições são dadas a seguir.

### Homotopia de uma imagem binária

Uma propriedade importante do esqueleto que queremos obter é a **preservação da homotopia da imagem**. Preservar a homotopia da imagem significa manter o número de componentes da imagem (tanto do objeto como do fundo), bem como a relação de interioridade entre estes componentes. A homotopia de uma imagem é representada pela árvore de homotopia da mesma.

Podemos construir a **árvore de homotopia** de uma imagem a partir das relações de conteúdo entre os componentes. Na figura 3.2, vemos uma imagem, onde os componentes do objeto e do fundo são nomeados e organizados em uma árvore. A raiz da árvore é representada pelo nó $X0$ e corresponde ao plano onde estão inseridos os componentes da imagem. Os componentes do nível seguinte da árvore são os componentes interiores ao componente $X0$, que por sua vez são representados pelos nós $X1$ e $X2$. Os componentes filhos de $X1$ são os componentes cujo bordo conecta-se a $X1$, e são respectivamente $X1'$, $X2'$ e $X3'$.

Duas imagens são homotópicas se as suas árvore de homotopia são

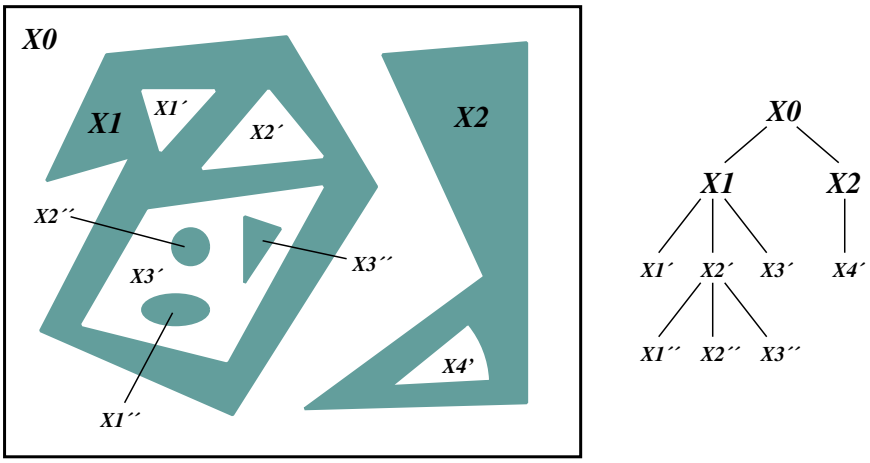


Figure 3.2: Exemplo de imagens homotópicas

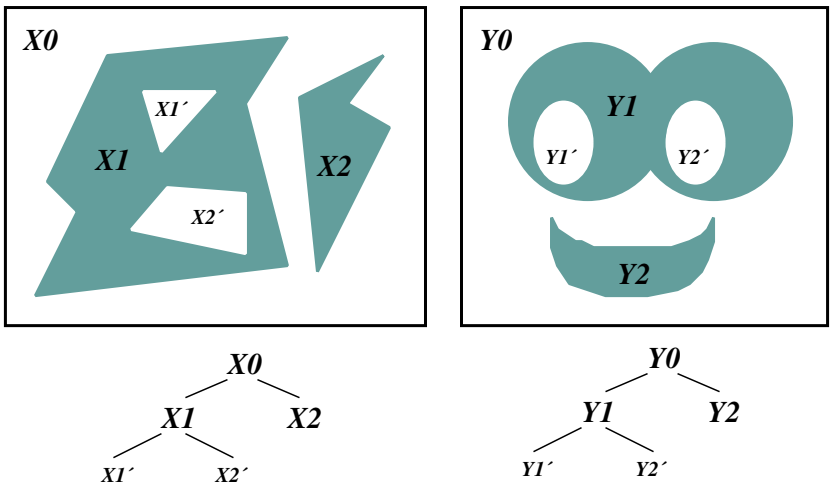


Figure 3.3: Árvore de homotopia de uma imagem

isomorfas. O exemplo da imagem 3.3 mostra duas imagens homotópicas. Uma transformação $\Psi$ é homotópica se para qualquer imagem $X$, $\Psi(X)$ é uma imagem homotópica a $X$.

### Alguns conceitos básicos de processamento de imagens digitais

Uma célula $p$ é identificada pela dupla $(p_x, p_y)$, representando as coordenadas cartesianas de seu vértice inferior e à esquerda. Trabalharemos com a definição convencional de $\alpha$-**adjacência**, onde $\alpha \in \{4, 8\}$. Duas células $p = (p_x, p_y)$ e $q = (q_x, q_y)$ são **8-adjacentes** se $|p_x - q_x| \leq 1$ e $|p_y - q_y| \leq 1$. Duas células $p$ e $q$ são **4-adjacentes** se são 8-conectadas e $|p_x - q_x| + |p_y - q_y| \leq 1$. Podemos observar, na figura 3.4, a conectividade na vizinhança de uma célula $p$. As células verdes são 8-conectadas a $p$, mas não 4-conectadas. Já as células vermelhas são 4-conectadas a $p$, e por isso são também 8-conectadas $p$.

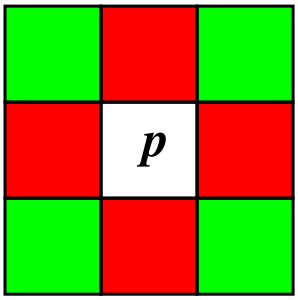


Figure 3.4: Relação de conectividade entre células

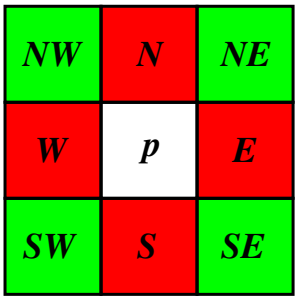


Figure 3.5: Nomenclatura das células em relação às suas posições

Dois conjuntos de células $S_1$ e $S_2$ são $\alpha$-adjacentes quando pelo menos uma célula de $S_1$ é $\alpha$-adjacente a pelo menos uma célula de $S_2$. Um **$\alpha$-caminho** que liga as células $p$ e $q$ é um conjunto de células $p_i$ distintas, $0 \leq i \leq n$, onde $p_0 = p$ e $p_n = q$, e para qualquer $0 \leq i < n$, $p_i$ é $\alpha$-adjacente a $p_{i+1}$. Dizemos que dois pontos $p$ e $q$ são **$\alpha$-conectados** se pelo menos um $\alpha$-caminho ligando $p$ a $q$.

Para facilitar a identificação das células da vizinhança de uma célula $p$, podemos utilizar uma notação simples, que considera a posição da célula vizinha em relação a $p$. Esta nomenclatura pode ser entendida através da figura 3.5. **E(p), W(p), S(p)** e **N(p)** denotam *east*, *west*, *south* e *north*, e representam as direções em relação a $p$ (direita, esquerda, abaixo e acima, respectivamente). De maneira análoga, denominamos as outras quatro células de **SE(p), SW(p), NE(p)** e **NW(p)**. Assim, a célula *S(p)* é a célula que se localiza abaixo de $p$; a célula *SE(p)* é a célula que se localiza abaixo e à direita de $p$; a célula *NW(p)* é a célula que se localiza em cima e à esquerda oeste de $p$, e assim por diante.

Uma **imagem digital** é o conjunto de células de um *grid* retangular, e seus respectivos estados. Cada uma das células classifica-se em **cheia** ou **vazia**. O conjunto das células cheias compõem o **objeto** contido na imagem, enquanto o conjunto das células vazias compõem o **fundo** da imagem.

Para representar uma imagem digital, utilizamos uma quádrupla $(V, \alpha, \beta, B)$, onde $V$ é o conjunto de todas as células do *grid*, $B$ é o con-

junto das células do objeto, $\alpha$ é a conectividade dos componentes do objeto e $\beta$ é a conectividade dos componentes do fundo. Neste capítulo, utilizaremos 8-conectividade para o objeto e 4-conectividade para o fundo. Note que se o objeto for 8-conectado, o fundo deve ser 4-conectado. O contrário também é válido: se o objeto for 4-conectado, o fundo deve ser 8-conectado.

Um **$\alpha$-componente** é um subconjunto maximal do objeto (ou do fundo) onde cada par de células é $\alpha$-conectado. Não se deve confundir o conceito de objeto e de componente: um objeto é o conjunto de todas as componentes de células cheias da imagem. Assim, uma imagem terá sempre apenas um objeto, que por sua vez, pode ser vazio, ou possuir uma ou mais componentes. Assim, como o objeto, o fundo também pode possuir zero ou mais componentes.

## 3.2 Classificações de células através de operações Hit-or-Miss

A configuração de vizinhança de uma célula vai determinar se a mesma deve ou não ser removida na esqueletonização de um objeto. Para verificar se uma célula possui determinada configuração de vizinhança, comparamos a configuração de sua vizinhança com um elemento estrutural. Um **elemento estrutural** é formado por um conjunto de células de coordenadas $(x_i, y_i)$, onde um estado $s_i$ é associado a cada célula $i$. Podemos ver na figura 3.6 dois exemplos de elementos estruturais. O elemento estrutural $A$ possui origem em $p$ e sete células, onde as células $NW(p)$, $N(p)$ e $NE(p)$ devem ser vazias, e as células $p$, $SW(p)$, $S(p)$, $SE(p)$ devem ser cheias[2].

Através da superposição de uma célula $p$ e das células de sua 8-vizinhança pelo elemento estrutural, verificamos se a vizinhança de $p$ possui a configuração desejada. A superposição deve ser feita de tal maneira que a célula $p$ seja superposta pela origem do elemento estrutural. No exemplo da figura, a configuração desejada para o elemento estrutural $A$ é que as três células ao norte sejam vazias, e que tanto $p$ como as três células ao sul sejam cheias. Note que o estado das células $E(p)$ e $W(p)$ não está especificado no elemento estrutural, e portanto o estados das mesmas não importa para a classificação de $p$.

Continuando com o exemplo da figura 3.6, observa-se que a célula G3 corresponde à configuração do elemento estrutural $A$: note que, ao transladar o elemento estrutural $A$ de modo que seu centro coincida com a célula G3, todas

[2]Note que o conjunto de células do elemento estrutural não necessariamente contém $p$ nem as células da 8-vizinhança de $p$. Mas como queremos analisar apenas a 8-vizinhança das células cheias, não faz sentido usar elementos estruturais com células que não pertencem à 8-vizinhança da origem

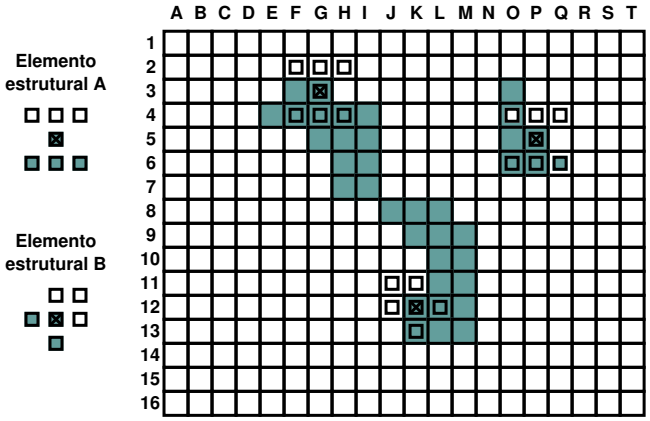


Figure 3.6: Afinamento digital: superposição do elemento estrutural em uma imagem

as células do elemento estrutural $A$ coincidiram com células na imagem de mesmo estado. O mesmo não se pode dizer da célula P5 em relação ao elemento estrutural $A$: as células O4 e Q6 coincidiram com células do elemento estrutural de estados diferentes. Podemos ver também que a célula K12 corresponde à configuração imposta pelo elemento estrutural $B$ rotacionado de $90^o$ no sentido anti-horário.

Operações morfológicas que utilizam elementos estruturais com células cheias e vazias são denominadas **operações de *hit-or-miss***. Este tipo de operação é denominado de hit-or-miss porque o conjunto das células cheias do elemento estrutural devem "acertar" o objeto (*hit*) e as células vazias do elemento estrutural devem "evitar" o objeto (*miss*). A definição formal de uma operação (ou transformação) *hit-or-miss* encontrada na literatura é feita através da interseção de duas operações de erosão, onde a primeira erosão atua sobre o objeto frente às células cheias do elemento estrutural, e a segunda erosão, sobre fundo da imagem frente às células vazias do elemento estrutural:

$$THM_{ES_1 \cup ES_2}(X) = \epsilon_{ES_1}(B) \cap \epsilon_{ES_2}(V - B)$$

$$X = (V, \alpha, \beta, B)$$

$$ES = ES_1 \cup ES_2,$$

onde

- $X$ é a imagem sobre a qual a operação de *hit-or-miss* é realizada
- $THM_{ES}(X)$ é uma transformação *hit-or-miss* sobre a imagem $X$ com o elemento estrutural $ES$
- $E$ é o elemento estrutural
- $ES_1$ são as células cheias do elemento estrutural

Figure 3.7: Afinamento digital: elementos estruturais

– $ES_2$ são as células vazias do elemento estrutural

– $\epsilon$ é o operador de erosão

Maiores detalhes sobre o operador de erosão podem ser encontrados nas principais referências sobre processamento de imagens e morfologia digital, como os livros de Soille [21], Facon [8] e Jain [9].

## 3.3 Afinamento topológico através de operações Hit-or-Miss

Através de elementos estruturais como mostrados na seção anterior, podemos reduzir as espessuras dos objetos de uma imagem até que os mesmos possuam espessura unitária, sem que ocorra alteração na homotopia da imagem. A propriedade de preservação da homotopia é garantida pela escolha adequada dos elementos estruturais.

Uma **transformação de afinamento hit-or-miss** consiste em extrair as células com determinada configuração de vizinhança, onde a configuração é definida por elementos estruturais conforme exemplificado acima. Para realizar o afinamento digital topológico, precisamos utilizar doze elementos estruturais: os dois mostrados na figura 3.7 e as três rotações de cada um dos elementos. O algoritmo de afinamento é mostrado abaixo, de forma simplificada:

**Algoritmo 1** Algoritmo básico de afinamento

```
repita
  para cada célula cheia da imagem faça
    para cada elemento estrutural de afinamento faça
      Remova a célula se sua configuração coincidir com a do elemento
      estrutural.
    fim para
  fim para
até quando estabilidade é alcançada
```

Na figura 3.8 (quadro da esquerda), podemos ver as duas iterações do algoritmo até a estabilidade. As células cor-de-rosa foram removidas na primeira iteração do algoritmo, e as células amarelas foram removidas na segunda iteração do algoritmo. As células, neste caso, foram percorridas da esquerda para a direita, e de cima para baixo, nesta ordem (A1, B1, C1, ...,

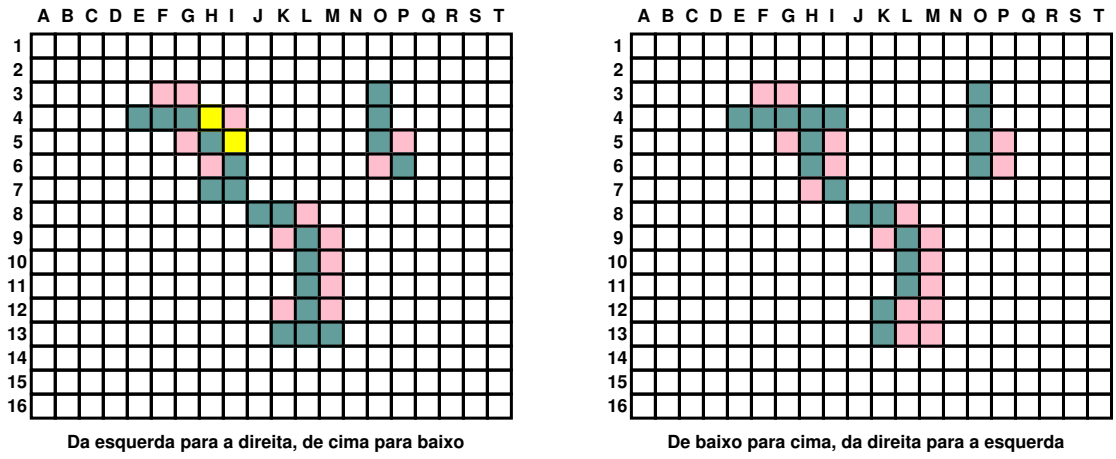


Figure 3.8: Afinamento digital: processo completo

T1, A2, B2, ...). Já na imagem da esquerda, a ordem de percorrimento foi de baixo para cima, e da direita para a esquerda. Somente pela alteração da ordem de percorrimento das células, dois fatos relevantes ocorreram: o número de iterações necessárias para se alcançar a estabilidade, bem como as imagens finais, foram diferentes nos dois exemplos apresentados.

Isto não significa que um dos modos (ou os dois) de percorrimento estejam errados. O processo de esqueletonização digital fornece apenas uma aproximação para o esqueleto real do objeto. É importante ressaltar que a imagem resultante de sucessivas operações de afinamento é apenas um conjunto de curvas digitais, ou seja, o algoritmo garante apenas a espessura unitária do objeto resultante e a homotopia em relação à imagem inicial.

## 3.4 Implementação

### Implementação básica

A primeira implementação para este algoritmo é bastante simples, e a idéia principal está descrita no tópico anterior. A cada iteração, todas as células são percorridas da esquerda para a direita, e de cima para baixo, e as células cuja configuração coincide com uma das oito máscaras de afinamento são removidas. Como a etapa de redução de ruído e ajuste da densidade dos pontos já foi realizada, espera-se que a imagem já possua espessura inicial pequena. Por isso, a ordem de percorrimento não trará efeitos indesejáveis, gerando uma curva digital próxima dos pontos de amostragem.

Um detalhe de implementação importante é a utilização de uma tabela de look-up para verificar se a configuração de um ponto corresponde a um dos elementos estruturais de afinamento. A tabela de look-up é um vetor de booleanos, onde o índice da entrada no vetor representa a codificação da vizinhança da célula e seu valor associado indica se a célula com a configuração de vizinhança representada pelo índice pode ser removida.

A codificação da vizinhança é realizada da seguinte maneira: cada célula da 8-vizinhança de $p$ é representada por um bit. Se a célula a ser representada for cheia, o bit deve ser igual a um, e se a célula representada for vazia, o bit deve ser zero. O bit mais significativo representa a célula NW(p), seguido pelas células N(p), NE(p), W(p), E(p), SW(p), S(p) e SE(p), nesta ordem. A figura 3.9 mostra um esquema para esta codificação, bem como um exemplo. Note que, como estamos realizando um afinamento, já sabemos de antemão que a célula a ser removida deve ser cheia, e por isso não há necessidade de representarmos a célula central na codificação, bastando que testemos apenas a configuração da vizinhança de células cheias.

A montagem da tabela de look-up é realizada antes do processamento do algoritmo de esqueletonização, sendo armazenada em memória até que o algoritmo termine. A tabela de look-up é realizada da montada maneira: para cada valor de codificação da configuração da vizinhança possível (que é um inteiro entre 0x00 e 0xFF), testa-se se o mesmo coincide com a configuração de vizinhança de algum elemento estrutural, armazenando-se o resultado desta pesquisa na posição adequada do vetor. Este teste consiste em um cálculo simples e utiliza, para cada um dos oito elementos estruturais, duas codificações: uma representa as células relevantes do elemento estrutral e a outra representa as células que devem ser cheias no elemento estrutural.

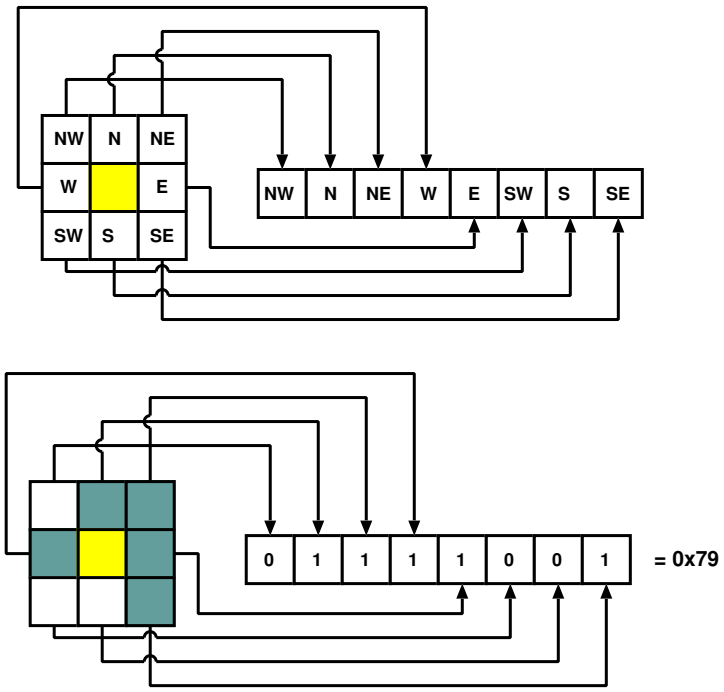


Figure 3.9: Codificação da vizinhança de uma célula

Seja $x$ a codificação da vizinhança que se quer testar, $R_i$ a codificação das células relevantes do $i$-ésimo elemento estrutural, e $F_i$ a codificação das células obrigatoriamente cheias do $i$-ésimo elemento estrutural. Então o valor da $x$-ésima entrada do vetor será dada por:

$$V[x] = \prod_{i=1}^{8} [(x \vee R_i) == F_i]$$

Vamos tomar como exemplo o elemento estrutural visto na figura 3.10. Ele indica que as células relevantes são N(p), NE(p), W(p), E(p) e S(p), dentre as quais as células W(p) e S(p) devem ser cheias. A codificação das células relevantes é o número binário 01111010, que em hexadecimal é igual a 0x7A. Já codificação das células que devem ser cheias é o número binário 00010010, que em hexadecimal é igual a 0x12. Como podemos ver na figura 3.10, uma célula cuja vizinhança seja representada pelo hexadecimal 0x13 coincide com a configuração deste elemento estrutural. Já uma célula representada pelo hexadecimal 0x10 não coincide.

A vantagem da utilização de tabelas de look-up é que são necessários poucos testes para verificar se uma célula pode ser removida. Para obter tal informação, precisa-se apenas calcular a codificação da vizinhança da célula e acessar a posição do vetor relativa à codificação calculada da vizinhança.

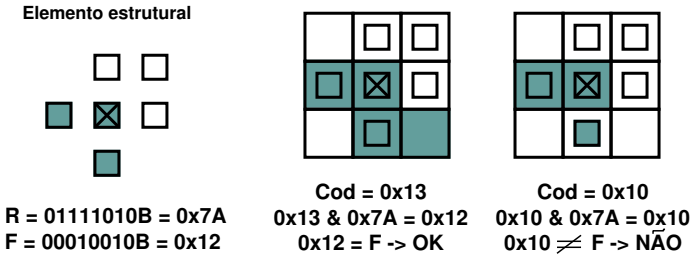


Figure 3.10: Testando a configuração da vizinhança através de sua codificação

### Afinamento de conjuntos de células com "pesos"

A segunda implementação proposta para o algoritmo utiliza como ordem de percorrimento o número de pontos no interior cada uma das células, tentando remover primeiro as células com menos pontos de amostragem em seu interior. Esta medida visa priorizar a remoção das células com menor probabilidade de possuirem interseção com a curva.

Inicialmente, monta-se uma lista ordenada com todas as células cheias da imagem, onde os elementos da lista são estruturas de dados que contém as coordenadas da célula representada e o número de pontos de amostragem no interior da célula. A cada iteração, a lista é percorrida, testando se cada uma das células da lista pode ser removida. Em caso afirmativo, a célula tem o seu estado alterado de CHEIO para VAZIO. Este algortimo possui um problema: suponha que a célula $i$, com $N_i$ pontos de amostragem em seu interior, acabou de ser removida. Devido à sua remoção, a célula $j$, vizinha da célula $i$ com $N_j < N_i$ pontos de amostragem em seu interior, teve a sua configuração de vizinhança alterada, de modo que passou a ser removível. Se continuássemos percorrendo a lista para procurar outras células removíveis, não removeríamos a célula $j$, pois como $N_j < N_i$, então a célula $j$ se encontra em posição anterior à célula $i$. Isto não pode acontecer, pois queremos garantir que se uma célula está sendo removida, então não existe na imagem nenhuma outra célula removível com menos pontos de amostragem em seu interior. Para superar este problema, a solução adotada foi reiniciar o percorrimento da lista sempre que uma célula for removida da mesma. Note que esta solução, embora correta, é uma solução ineficiente em termos computacionais, pois sempre que uma célula é removida, todas as células com menos pontos de amostragem no interior são novamente testadas, e não apenas as células vizinhas àquela que acabou de ser removida.

### Afinamento de conjuntos de células com "pesos" - Implementação otimizada

O terceiro algoritmo proposto visa superar o problema de desempenho ao algoritmo anterior, mas a garantia de sempre remover células com menos pontos de amostragem em seu interior deve continuar. A idéia de criar uma lista com as células cheias, ordenada pelo número de pontos de amostragem no interior de cada célula, foi mantida nesta terceira implementação. Porém, quando uma célula removível é encontrada durante o percorrimento da lista, inicia-se um procedimento recursivo de remoção, que verifica se alguma das células vizinhas com menos pontos de amostragem em seu interior passou a ser removível, caso em que a mesma é removida.

Para implementar esta solução, foi necessário acrescentar algumas estruturas de dados às já existentes. Ao elemento da lista que representa a célula $p$, com $N(p)$ pontos de amostragem em seu interior, adicionou-se uma lista ordenada de células vizinhas de $p$ cujo número de pontos de amostragem em seu interior é menor que $N(p)$. Também é necessário uma matriz onde se armazena, em cada elemento $(i, j)$, a posição da célula de coordenadas $(i, j)$ na lista com todas as células cheias ordenadas pelo número de pontos de amostragem em seu interior. Esta matriz visa a recuperação rápida da posição das células vizinhas a uma determinada célula. Células que não estão na lista são marcadas com $-1$. O algoritmo de remoção para esta implementação pode ser visto a seguir.

**Algoritmo 2** RemoveCelula(X, Y)

```
Muda estado da célula (X, Y) para VAZIA
listaVizinhos ← Celula(X, Y).listaVizinhos;

// Percorre ordenadamente os vizinhos da célula
para cada Elemento ∈ listaVizinhos faça
  X ← Elemento.X;
  Y ← Elemento.Y;
  se EstadoCelula(X, Y) = CHEIA ∨ CelulaEhRemovivel(X, Y) então
    RemoveCelula(X, Y);
  fim se
fim para
```

## 3.5 Pós-processamento do esqueleto através de poda

Podemos ver, na figura 3.11 [22], uma imagem e seu "esqueleto digital". Note que a curva obtida apresenta algumas ramificações com poucas células

**Algoritmo 3** AfinaObjeto

```
// Cria a lista ordenada com as células do objeto
listaOrdenada ← CriaListaVazia;
para cada Celula ∈ Objeto faça
  X ← Celula.X;
  Y ← Celula.Y;
  N ← Celula.NumeroPontosInteriores;
  Elemento ← CriaElemento(X, Y, N);

  // Monta a lista de vizinhos da célula
  listaVizinhosOrdenados ← CriaListaVazia;
  para cada Vizinho ∈ 8-Vizinhança(Célula) faça
    V ← Vizinho.NumeroPontosInteriores;
    se V < N então
      listaVizinhosOrdenados.Insere(Vizinho);
    fim se
  fim para
  listaOrdenada.Insere(Celula);
fim para

// Percorre ordenadamente as células da lista,
// removendo recursivamente as células removíveis
para cada Elemento ∈ listaOrdenada faça
  X ← Elemento.X;
  Y ← Elemento.Y;
  se EstadoCelula(X, Y) = CHEIA ∨ CelulaEhRemovivel(X, Y) então
    RemoveCelula(X, Y);
  fim se
fim para
```

de comprimento. Isto ocorre devido a variações na espessura da imagem e à presença de "quinas". Não desejamos que estes artefatos estejam presentes na curva que desejamos reconstruir. Para removê-los, utilizamos o processo de poda (*prunning*, em inglês), que remove células das extremidades de um curva. A cada iteração deste processo, uma célula em cada uma das extremidades da curva é removida. Note que este processo deve ser realizado poucas vezes, sob pena de se diminuir excessivamente o comprimento da curva. No caso do algoritmo implementado neste trabalho, são executadas duas iterações do algoritmo de poda. Este número mostrou-se suficiente para todos os casos testados, sem reduzir a curva excessivamente.

Assim como o afinamento topológico, o processo de poda deve preservar a homotopia da imagem. Para tal, basta escolhermos elementos estruturais adquados. Os elementos estruturais utilizados no processo de poda podem ser visto na figura 3.12. Na figura à direita dos elementos estruturais, podemos ver

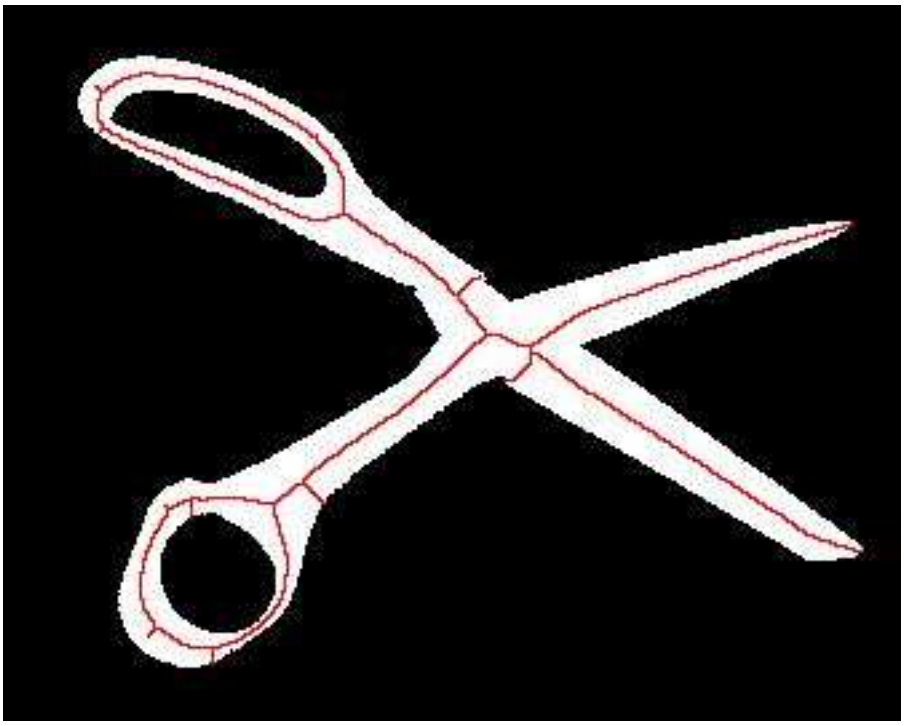

Figure 3.11: Exemplo de esqueleto digital



que as células D4, I2, O6 e Q4 possuem uma configuração que coincide com pelo menos um dos elementos estuturais.

O processo é realizado de maneira análoga ao afinamento: percorre-se as células do objeto, removendo aquelas cuja configuração coincidir com um dos elementos estruturais mostrados na figura 3.12. Estes elementos estruturais são formados de modo que apenas as configurações das células nas extremidades da curva coincidam com os mesmos. Novamente, a ordem de percorrimento das células pode influenciar no resultado do algoritmo, como podemos observar na figura 3.12, imagem de baixo à esquerda. Isto acontece porque, ao remover uma célula de extremidade $c$, uma de suas células vizinhas $v$ pode passar a ser uma extremidade. Assim, dependendo da ordem de percorrimento, se a célula $v$ for visitada depois da célula $c$, ela também será removida. Note que na figura de baixo e da esquerda, apenas duas iterações do processo de poda foram realizados (as células em rosa foram removidas na primeira iteração, e as amarelas, na segunda). Isso não impediu que apenas uma das células originais continuasse na imagem.

Para resolver este problema, podemos utilizar a seguinte técnica: a cada passagem do algoritmo, marcamos as células cuja configuração coincide com um dos elementos estruturais. Depois que todas as células forem percorridas, removemos as células marcadas. Note que a remoção de uma célula marcada implica na alteração da configuração de suas células vizinhas. Logo, é impor-

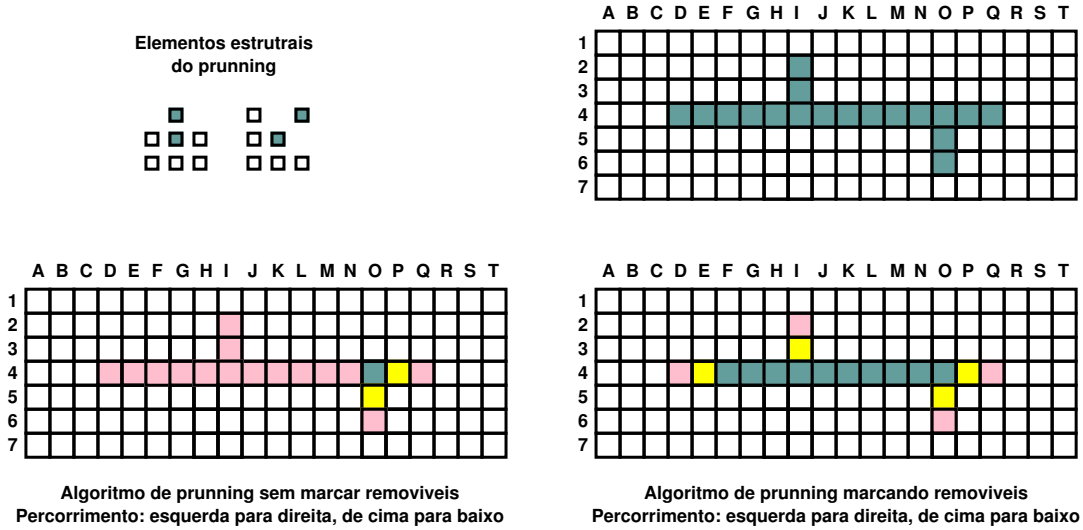


Figure 3.12: Algoritmo de poda com e sem marcação prévia de células removíveis

tante que a configuração da vizinhança das células marcadas seja verificada mais uma vez, imediatamente antes de ser removida. A figura 3.12 mostra o efeito desta técnica, comparando-a com a implementação "ingênua" do processo de poda. A imagem original está acima e à direita. As duas imagens de baixo representam a imagem original processada com o poda de duas maneiras diferentes. As células marcadas em rosa foram removidas na primeira iteração do poda, e as células marcadas em amarelo foram removidas na segunda iteração. A imagem abaixo e à direita é a implementação ingênua. Observe na figura da direita e abaixo que, devido à ordem de percorrimento, várias células foram removidas indevidamente. O problema foi contornado com a marcação prévia das células removíveis, como pode ser observado na figura abaixo e à esquerda.