Segmentação de Vasos Sanguíneos em Imagens do Fundo da Retina

Projeto de Iniciação Científica

Aluno: Emerson Luiz Navarro Tozette
emr@linux.ime.usp.br
Orientador: Dr. Roberto Marcondes Cesar Junior
cesar@ime.usp.br

DCC - Departamento de Ciência da Computação
IME - Instituto de Matemática e Estatística
USP - Universidade de São Paulo

6 de dezembro de 2002

Resumo

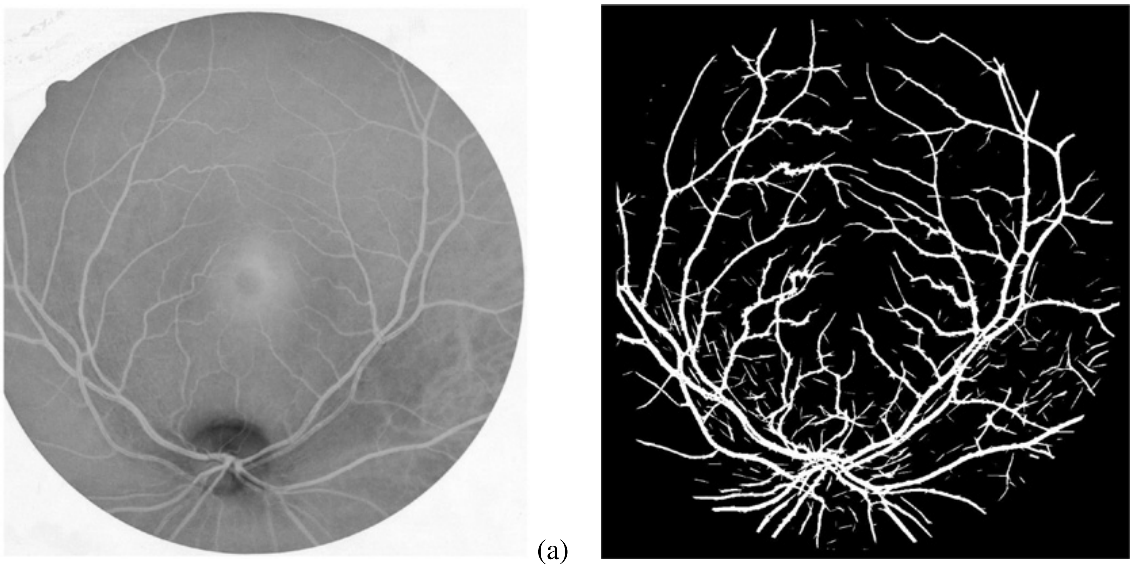
Algumas doenças, como a retinopatia diabética (uma das causas mais comuns de cegueira em adultos), são diagnosticadas pelos médicos através da análise de fotografias do fundo do olho. Essas imagens, porém, são muito complicadas para serem analisadas manualmente e requerem muito tempo e esforço por parte dos médicos. Vários cientistas vêm desenvolvendo métodos computacionais automáticos de análise dessas fotografias. Espera-se que esses métodos ajudem os médicos em seu trabalho, ou seja, que o trabalho seja mais rápido e com menos erros de diagnóstico.

O objetivo deste trabalho é aperfeiçoar o programa de detecção automática de vasos sanguíneos de imagens do fundo da retina proposto em [8]. A segmentação correta dos vasos sanguíneos dessas imagens é uma das primeiras etapas na busca automática de determinadas doenças, como a hipertensão e a retinopatia diabética.

Sumário

1 Introdução

Em oftalmologia, a segmentação automática dos vasos sanguíneos, assim como a detecção de outros componentes do fundo do olho, pode ser de grande interesse para se obter um diagnóstico assistido pelo computador. Veja, por exemplo, o caso da *diabetes mellitus*: uma de suas complicações é a retinopatia diabética, que é uma das causas mais comuns de cegueira e de defeitos visuais na população adulta do ocidente. Essa complicação pode ser moderada se a doença for diagnosticada cedo. O problema é que os defeitos visuais só começam a aparecer em um estado tardio da doença, quando já é tarde para um tratamento óptico. A solução seria examinar em massa todos os pacientes com diabete, mas para isso precisamos de métodos automáticos para a detecção da doença, pois não temos oftalmologistas suficientes para realizar tantos exames. Assim, vários cientistas estão trabalhando em algoritmos que tentam detectar automaticamente a doença a partir de fotografias do fundo do olho. Os algoritmos devem primeiro **segmentar os vasos sanguíneos** (separar os vasos do resto da imagem), produzindo um mapa dos vasos sanguíneos (figura 1). O passo seguinte é **extrair características** desse mapa. Medidas como a área dos vasos sanguíneos, seu comprimento e dimensão fractal, entre outras, são extraídas e analisadas estatisticamente em conjunto (utilizando-se técnicas de aprendizagem computacional, por exemplo). Esperamos que o médico, de posse dos resultados dessa análise, possa decidir com precisão se o paciente está ou não doente.



(a) (b)

Figura 1: **(a)** *Uma fotografia do fundo da retina;* **(b)** *Imagem resultante da segmentação dos vasos sanguíneos.*

Uma das características da diabete é a **neovascularização** (a geração de novos vasos sanguíneos) no fundo do olho. Um paciente com um número muito grande de vasos nessa região deve estar com retinopatia diabética. Assim, esperamos que, para esse paciente, medidas como a área e o comprimento dos vasos sanguíneos se tornem maiores que a média.

Diversas características dos vasos sanguíneos, e de outros componentes do fundo do olho, podem ser analisadas na busca de outras doenças:

- **Microaneurismas** [3, 14] também são indicadores da retinopatia diabética.

- O **disco óptico** [12, 1, 10], a região de origem de todos os vasos sanguíneos, pode ser utilizado no diagnóstico do **glaucoma**.

- A **fóvea** [7, 1] é a região de maior acuidade visual da retina. Quanto mais perto da fóvea, mais grave é uma lesão na retina.

- Algumas doenças podem alterar a cor, a tortuosidade e o diâmetro dos vasos sanguíneos. A **arteriosclerose** pode fazer com que as artérias adquiram uma cor prateada. A **hipertensão** está associada a alterações no diâmetro e no comprimento dos vasos, bem como no ângulo entre os vasos [5].

- Existem áreas da retina onde a percepção visual pode se encontrar completamente destruída. Essas áreas são chamadas de *absolute scotoma* [1].

Este documento apresenta um projeto de iniciação científica em que o aluno estudou e implementou técnicas de segmentação de imagens de retina trabalhando como parte dessa linha de pesquisa desenvolvida pelo orientador.

1.1 Segmentando a retina

Uma grande parte dos algoritmos existentes de segmentação da retina se preocupa com os vasos sanguíneos. A outra grande parte se preocupa com outras estruturas (como lesões, fóvea e disco óptico). Algumas abordagens podem se utilizar da detecção de uma estrutura para detectar outras estruturas. Por exemplo: alguns métodos de segmentação dos vasos sanguíneos utilizam o disco óptico como referência, já que o disco óptico é a origem desses vasos [12]. A localização da fóvea também pode ser estimada a partir do disco óptico [12].

Duas estratégias principais vêm sendo utilizadas para detectar os vasos sanguíneos: *scanning* e *tracking*.

- **Scanning**: o *scanning* [4, 13, 6, 2] varre a imagem inteira procurando pelos pixels de interesse (os pixels que pertencem aos vasos sanguíneos) e os conectando. Em fotografias em tons de cinza do fundo da retina, os vasos sanguíneos normalmente são a estrutura mais clara da imagem, e se contrastam com o fundo da imagem, que é mais escura. O princípio básico do *scanning* é detectar variações bruscas entre tons de cinza. Um local que possui uma variação brusca de cinza escuro para cinza claro (e vice-versa) coincide com a borda de um vaso sanguíneo.

- **Tracking** [11, 2], ao contrário do scanning, começa numa posição do vaso sanguíneo conhecida *a-priori* na imagem. A extração das características desejadas da imagem e o reconhecimento da estrutura de vasos são feitos simultaneamente, explorando a continuidade dos vasos.

Pela sua própria natureza, o *tracking* é computacionalmente mais eficiente que o scanning. Sua desvantagem é que ele não consegue segmentar completamente a rede de vasos quando o vaso sanguíneo desaparece no meio e resurge novamente na mesma direção.

O *scanning*, pelo contrário, pode conseguir uma segmentação completa dos vasos. No entanto, ele requer muito mais tempo e espaço. Ruídos e outras estruturas podem ser confundidas com vasos sanguíneos. Vasos sanguíneos muito pouco contrastantes com o fundo da imagem podem não ser detectados.

2 Objetivos

O objetivo desse projeto é aprimorar o método de segmentação de vasos sanguíneos proposto por R. M. Cesar Jr. e H. F. Helinek [8] . Esse programa utiliza-se da técnica *scanning*: uma função matemática, denominada transformada *wavelet*, é aplicada à imagem. Essa função possui a propriedade que desejamos, que é a de localizar regiões com variações bruscas de níveis de cinza.

O programa possui dois problemas, discutidos a seguir. Um dos problemas foi resolvido com sucesso. O outro ainda necessita de melhorias consideráveis.

2.1 Problema 1: detecção da borda do disco óptico

Um dos problemas é a detecção da borda do disco óptico juntamente com a detecção dos vasos sanguíneos (figura 2). Isso ocorre porque o disco óptico também se contrasta com o fundo da imagem.

O problema, devido a sua dificuldade, não foi totalmente corrigido. A discussão desse problema encontra-se na seção 5.

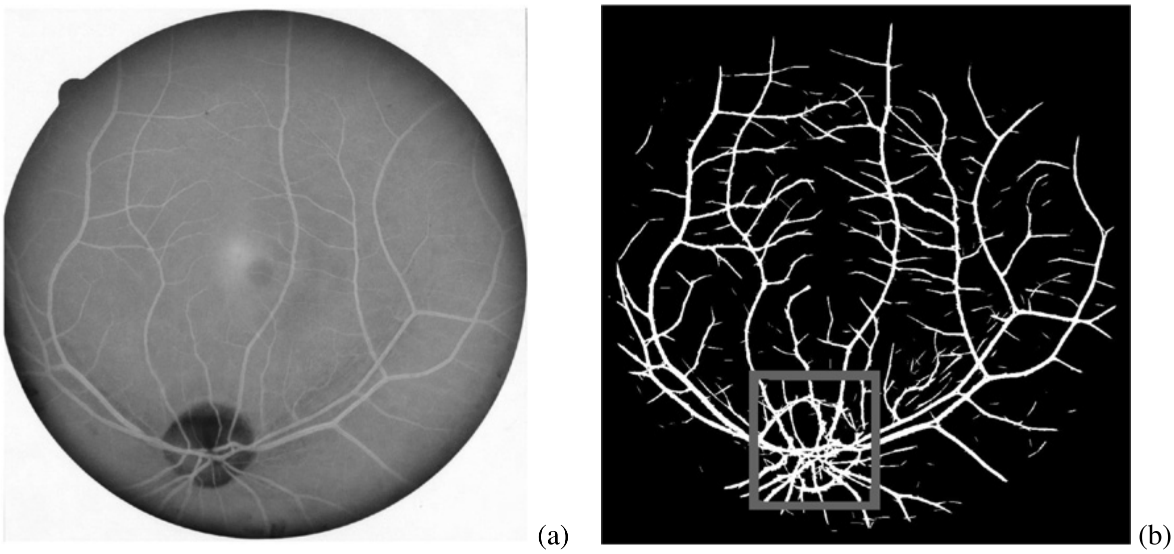


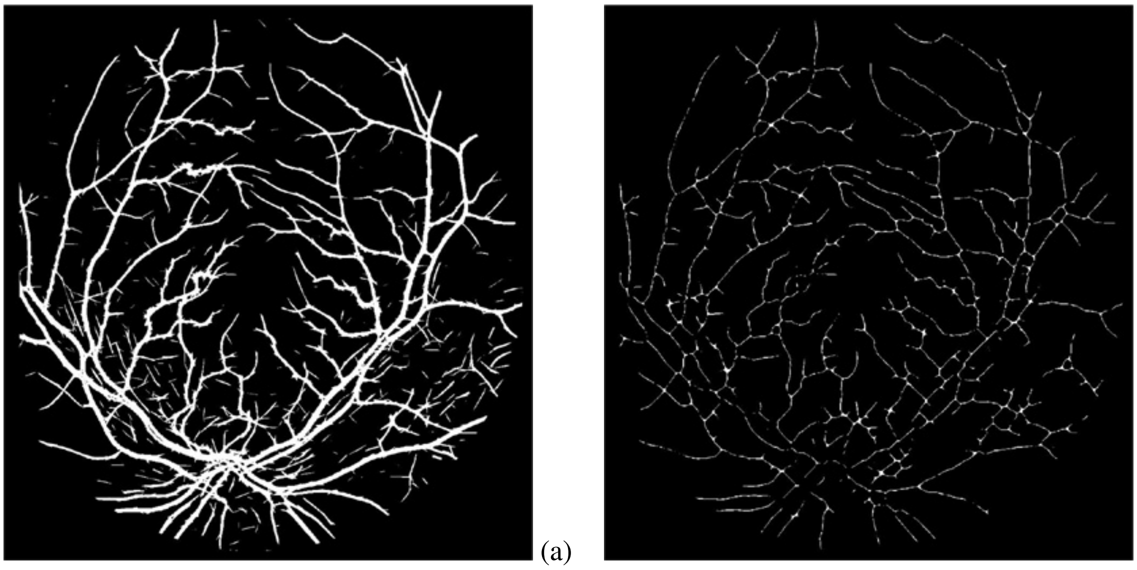
Figura 2: **(a)** *Uma fotografia do fundo da retina;* **(b)** *Imagem resultante da segmentação dos vasos sanguíneos. Na área em destaque podemos perceber uma região circular, que corresponde à borda do disco óptico.*

2.2 Problema 2: falhas na detecção do esqueleto

Após a fase de segmentação, segue a fase de extração das medidas, como a área e a dimensão fractal dos vasos sanguíneos. A extração dessas medidas não é feita diretamente dos vasos segmentados, e sim do **esqueleto** desse vasos. O esqueleto é uma estrutura de um pixel de largura que percorre o interior dos vasos sanguíneos, sendo extraída diretamente dos vasos segmentados (figura 3). Podemos pensar no esqueleto como sendo o mapa dos vasos sanguíneos, mas sem as variações no diâmetro dos vasos [1].

Extrair as medidas do esqueleto elimina alguns problemas práticos. No caso da área dos vasos sanguíneos, por exemplo, a área de um paciente saudável pode ser igual à área

[1]Na prática, os esqueletos extraídos de imagens podem possuir pequenas variações de diâmetro.



(a) (b)

Figura 3: **(a)** *Vasos sanguíneos segmentados;* **(b)** *Esqueleto obtido.*

dos vasos de um paciente doente se o paciente saudável possuir poucos vasos (ausência de neovascularização) e vasos muito dilatados, e se o paciente doente possuir muitos vasos (neovascularização) e vasos muito finos. Esse problema desaparece quando a área é extraída do esqueleto, ao invés de ser extraída diretamente dos vasos segmentados.

O programa em questão extraía o esqueleto dos vasos utilizando um algoritmo baseado em técnicas de morfologia matemática. Esse algoritmo possui alguns problemas, como mostrado pela figura 4: alguns segmentos do esqueleto estão atravessando a parte de fora dos vasos, e alguns vasos não possuem nenhum esqueleto atravessando o seu interior.

Este problema foi resolvido com sucesso implementando-se um novo algoritmo de detecção do esqueleto. Este algoritmo chama-se *Space-Scale Skeletonization*, e é proposto em [9]. A diferença deste algoritmo para aquele outro é que este é um algoritmo **multiescala**, ou seja, o algoritmo é capaz de extrair múltiplos esqueletos da mesma imagem, cada esqueleto com mais ou menos detalhes. O esqueleto extraído depende da **escala** utilizada (que é um parâmetro ou valor de entrada do algoritmo). Este algoritmo é discutido em detalhes na seção 4.

3 Materiais e Métodos

Todos os algoritmos foram implementados na linguagem **C** e no software **Matlab**. O Matlab foi usado devido à sua praticidade, e a linguagem C devido à sua eficiência. O algoritmo antigo de extração de esqueletos, por exemplo, faz parte de uma biblioteca de morfologia matemática disponível para o Matlab (o Matlab oferece diversas ferramentas matemáticas ligadas ao processamento de matrizes e de imagens). O Matlab, porém, é mais lento quando comparado com programas escritos na linguagem C. Uma solução é integrar o Matlab com programas C. O novo algoritmo de extração de esqueletos foi escrito em C e integrado e usado conjuntamente com algoritmos escritos no Matlab (o algoritmo da transformada *wavelet*, por exemplo, está escrito em Matlab).

Todas as imagens do fundo da retina foram cedidas pelo professor Herbert F. Jelinek, da Escola de Saúde Pública da Universidade Charles Stuart, da Austrália, colaborador desse projeto. Dispomos atualmente de aproximadamente 60 imagens.

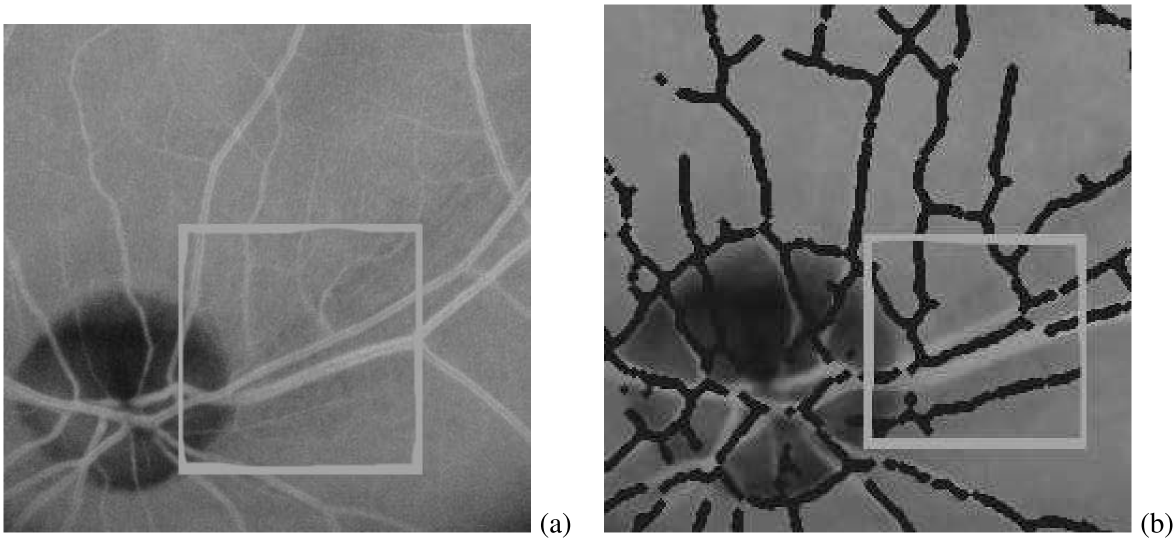


Figura 4: **(a)** *Detalhe de uma fotografia do fundo da retina. A região em destaque mostra dois vasos sanguíneos paralelos;* **(b)** *Esqueleto da imagem anterior sobreposto sobre a imagem. Note que na região em destaque o esqueleto passa pelo meio dos vasos sanguíneos, quando na verdade deveria passar um esqueleto pelo interior de cada vaso. Note também que existem trechos de vasos sanguíneos onde não foram detectados nenhum esqueleto.*

Todo o desenvolvimento do projeto foi executado no laboratório de Visão do IME. O sistema operacional utilizado no laboratório é o **GNU/Linux**.

4 Detectando o esqueleto: algoritmos e resultados

Com relação a detecção do esqueleto, optamos por implementar o algoritmo multiescala proposto em [9], chamado *Scale-Space Skeletonization (SSSkel)*, que calcula tanto o esqueleto interno como o externo de um objeto de uma imagem binária.

Esta seção detalha o funcionamento do algoritmo. Para entendê-lo são necessários conhecimentos básicos de processamento de imagens, como conectividade e dilatação.

Este algoritmo recebe uma imagem binária, onde pixels com valor 0 pertencem ao *fundo* da imagem enquanto pixels com valor 1 pertencem ao objeto contido na imagem, e devolve uma **imagem diferença**. Para cada *limiar* aplicado na imagem diferença, um esqueleto diferente é obtido (i.e., o *limiar* é a escala do algoritmo). Duas etapas são necessárias antes do cálculo da imagem diferença, o que implica que o algoritmo deverá executar três etapas:

1. **Detectar o contorno** do objeto e **rotular os pixels do contorno** com números inteiros consecutivos.

2. **Propagar os rótulos** utilizando **dilatação exata**.

3. **Calcular a imagem diferença**.

Após a detecção do contorno, rotular e propagar os rótulos dos pixels é uma tarefa simples. Os maiores problemas estão na detecção dos contornos dos objetos e no cálculo da imagem diferença. O algoritmo precisou ser generalizado para trabalhar com imagens que possuem mais de um objeto, e para trabalhar com objetos que possuem zero ou mais buracos.

As próximas subseções explicam o que feito para generalizar o algoritmo.

4.1 Detectando os contornos de uma imagem binária

Para detectar todos os contornos de todos os objetos, incluindo os contornos dos buracos dos objetos, utilizamos o *tracking algorithm* proposto em [9], que assume que a imagem possui apenas um objeto e que o objeto possui no máximo um buraco de um tipo especial. O algoritmo precisou ser generalizado. Este algoritmo extrai o **contorno externo** de objetos, e possui basicamente três etapas:

1. **Encontrar o primeiro pixel** do contorno do objeto: procurar linha por linha (da esquerda para direita, de cima para baixo) um pixel que pertença ao *fundo* da imagem e que possua como vizinho (i.e., à sua direita) um pixel que pertença ao objeto contido na imagem.

2. Dado o primeiro pixel, **encontrar o segundo pixel** do contorno do objeto.

3. Dados o primeiro e segundo pixels, **encontrar os pixels restantes**.

O primeiro passo para generalizar o algoritmo é impedir que um objeto interfira na extração do contorno de outro objeto, o que pode acontecer quando objetos diferentes estão separados pela distância de apenas um pixel. A solução é **rotular os objetos da imagem** utilizando algum *labelling algorithm* [9]. Desse modo, a nossa imagem binária inicial é transformada em uma imagem de números inteiros onde zero representa o *fundo* e qualquer pixel de valor maior que zero pertence a algum objeto. Assim, o primeiro pixel do contorno de algum objeto deve pertencer ao *fundo* e possuir um pixel de valor i, $i > 0$, à sua direita. A etapa seguinte será encontrar o segundo pixel e os pixels restantes utilizando o valor i encontrado. Como i é diferente para cada objeto, um objeto não deve interferir na extração do contorno de outro.

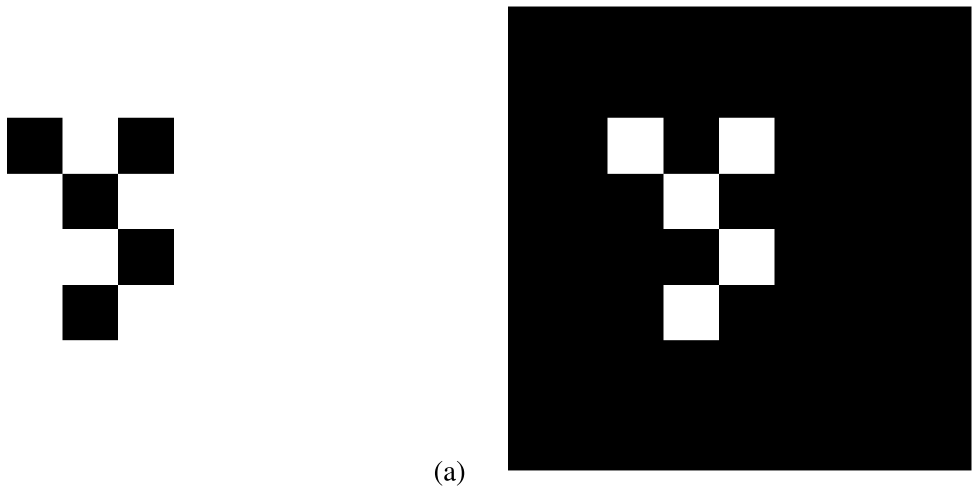
Os contornos detectados pelo *tracking algorithm* são **8-conexos**, o que implica que o *labelling algorithm* deverá detectar objetos **4-conexos** [2]. Para detectar o primeiro e os demais pixels dos contornos de **buracos** de objetos, o próprio *tracking algorithm* pode ser utilizado. A diferença é que o interior dos buracos não precisa ser rotulado, pois um buraco nunca interfere na extração do contorno de outro, sejam buracos pertencentes a um mesmo objeto ou a objetos diferentes. No entanto, utilizar este algoritmo para detectar buracos implica que os buracos serão **8-conexos**, ao invés de 4-conexos como os objetos [3]. Veja a figura 5. Na figura 5a serão extraídos 5 contornos (todos contornos externos), um para cada quadrado. Na figura 5b será extraído apenas um contorno, que passa pelo interior de todos os quadrados.

Ainda resta fazer uma pequena modificação na segunda etapa do algoritmo (encontrar o segundo pixel). O algoritmo que encontra o segundo pixel (dado o primeiro) **sempre** encontra algum pixel, mesmo que o objeto seja composto de apenas um pixel (nesse caso, o contorno do objeto possui 4 pixels que circundam o objeto). No caso de buracos, se o buraco for composto por apenas um pixel, o contorno do buraco será o próprio buraco. Assim, a segunda etapa deve estar preparada para retornar vazio, algo que não aconteceria no *tracking algorithm* original. Caso não retorne vazio, passamos para a próxima etapa (extrair os pixels restantes).

A função que encontra o primeiro pixel varre a imagem inteira, linha por linha, em busca do primeiro pixel de todos os objetos e buracos. Quando esse pixel é encontrado, o segundo e demais pixels são imediatamente extraídos, e a função é reiniciada do ponto onde parou. Tudo isso se repete até que o final da imagem seja encontrado.

[2] Existe um teorema que diz que se um **objeto** é 4-conexo seu contorno será 8-conexo, e vice-versa.

[3] Desse modo, o que é um buraco? Um buraco é um conjunto 8-conexo de pixels de valor 0, cercado por pixels 4-conexos de rótulo i, $i > 0$. Note que não só o buraco é 8-conexo, o seu contorno, extraído pelo *tracking algorithm*, também o é.



(a) (b)

Figura 5: **(a)** *Cinco objetos 4-conexos;* **(b)** *Um buraco 8-conexo.*

Para impedir que um mesmo contorno seja extraído repetidamente, vamos **marcar em uma matriz** M **auxiliar** cada contorno que seja detectado. Se o objeto possui rótulo i, seu contorno será marcado com o valor i. No caso de buracos, se o buraco pertence ao objeto i, seu contorno será marcado com o valor i.

Devemos notar que no caso de objetos separados por um pixel de distância, os contornos desses objetos se sobrepõe. Se nossa matriz M for simplesmente uma matriz de inteiros a marca de um contorno pode sobrescrever a marca de outro, e este mesmo contorno acabaria sendo extraído mais de uma vez. Assim, a matriz M será uma matriz de estruturas. Cada estrutura guardará quatro números inteiros, que é o número máximo de contornos que podem se sobrepor sobre um mesmo pixel.

A função que encontra o primeiro pixel está em busca de um pixel p de valor igual a zero (i.e, que pertence ao *fundo* da imagem) que possua um vizinho p' de rótulo i à sua direita, $i > 0$. Seja (l, c) a posição de p. Se o valor i pertence a $M(l, c)$, então o contorno do objeto de rótulo i já foi extraído. Caso contrário, o contorno ainda não foi extraído e deve ser extraído agora.

Como o contorno de um objeto pode passar mais de uma vez por um mesmo pixel, é importante impedir que uma dada posição da matriz M seja marcada mais de uma vez com o mesmo valor.

Em resumo, esse é o **novo *tracking algorithm***:

1. **Encontrar o primeiro pixel** do contorno de um objeto ou buraco: procurar linha por linha (da esquerda para direita, de cima para baixo) um pixel p de posição (l, c) que tenha rótulo zero, que à sua direita possua como vizinho um pixel p' de rótulo i, $i > 0$, tal que i não pertence à $M(l, c)$.

2. Dado o primeiro pixel **e o rótulo** i, **encontrar o segundo pixel** do contorno do objeto. Devolver `vazio` caso esse pixel não exista.

3. Caso o segundo pixel exista, e dados o primeiro e segundo pixels **e o rótulo** i, **encontrar os pixels restantes**.

4. **Marcar**, **na matriz** M, o contorno encontrado (marcar com o rótulo i).

5. Continuar a busca pelo primeiro pixel do ponto onde a busca parou (i.e., repetir 1, 2, 3 e 4). O programa termina quando esta busca alcança o fim da imagem.

Todos os pixels de todos os contornos encontrados são salvos em sequência em uma mesma **lista ligada**. Para a próxima etapa do algoritmo *SSSkel*, não é necessário uma lista ligada diferente para cada contorno, nem é necessário diferenciar os contornos um dos outros.

4.2 Extraindo o esqueleto: dilatação exata e cálculo da imagem diferença

Assumindo que nossa imagem possui apenas um objeto (e um contorno), nessa etapa devemos:

1. **Rotular os pixels do contorno** com números inteiros consecutivos.
2. **Propagar os rótulos** utilizando **dilatação exata**.
3. Calcular a **imagem diferença**.

Precisamos generalizar este algoritmo para tratar múltiplos contornos.

Em primeiro lugar, precisamos entender como o algoritmo funciona. Considere a figura 6a, contendo um objeto simples. Uma vez detectado o contorno desse objeto, os pixels correspondentes serão primeiramente rotulados com números inteiros consecutivos. Podemos visualizar os pixels rotulados em uma imagem em tons de cinza, onde os pixels de menor rótulo são os mais escuros, e os pixels de maior rótulo são os mais claros (figura 6b). A etapa seguinte é propagar esses rótulos utilizando dilatação exata. As figuras 6c a 6f mostram uma série crescente de dilatações exatas. Note que a medida que a dilatação aumenta, o espaço interno do contorno vai diminuindo até fechar completamente. Vamos chamar o local onde o espaço interno finalmente se fecha de **local de fechamento** (figura 6g). Note que o local de fechamento é um **esqueleto interno** do meu objeto.

A idéia do algoritmo é dilatar os rótulos até que o interior do contorno se feche completamente (vamos chamar a imagem gerada por essa dilatação de **imagem de rótulos**). Podemos ver na figura 6f e 6g que o local de fechamento é um ponto de **variações bruscas de tons de cinza**. Isso ocorre porque nesse ponto rótulos de valores muito diferentes se encontram.

Como é calculada a **imagem diferença**? Para cada pixel de posição (i, j) da imagem de rótulos, a **maior diferença** d entre esse pixel e seus quatro pixels vizinhos [5] é calculada. O valor d é então atribuído ao pixel de posição (i, j) da imagem diferença.

Fica claro que os pixels de maior valor da imagem diferença pertencem ao local de fechamento do objeto em questão. Aplicando um *limiar* suficientemente alto a esta imagem, conseguimos obter uma imagem binária que contenha todos os pixels do local de fechamento, isto é, que contenha um **esqueleto interno** da imagem. Aplicando *limiares* menores, obtemos outros esqueletos (figura 6h). Assim, o *limiar* não é nada mais que uma **escala** que permite obter esqueletos com mais ou menos detalhes. O local de fechamento é apenas um dos possíveis esqueletos do objeto.

Este algoritmo na verdade calcula tanto o esqueleto interno como o externo. Os esqueletos da figura 6g e 6h foram obtidos utilizando-se a figura 6a como máscara, de forma que o esqueleto externo (que não interessa em imagens do fundo da retina) fosse eliminado.

Agora vamos generalizar o algoritmo para tratar objetos com buracos e imagens com múltiplos objetos. Considere o objeto da figura 7a. Trata-se de um disco com um buraco no meio. Dois contornos serão extraídos dessa figura: o contorno do disco e o contorno

[5] Os quatro vizinhos são o pixel acima, abaixo, à esquerda e à direita (i.e., são pixels 4-conexos).

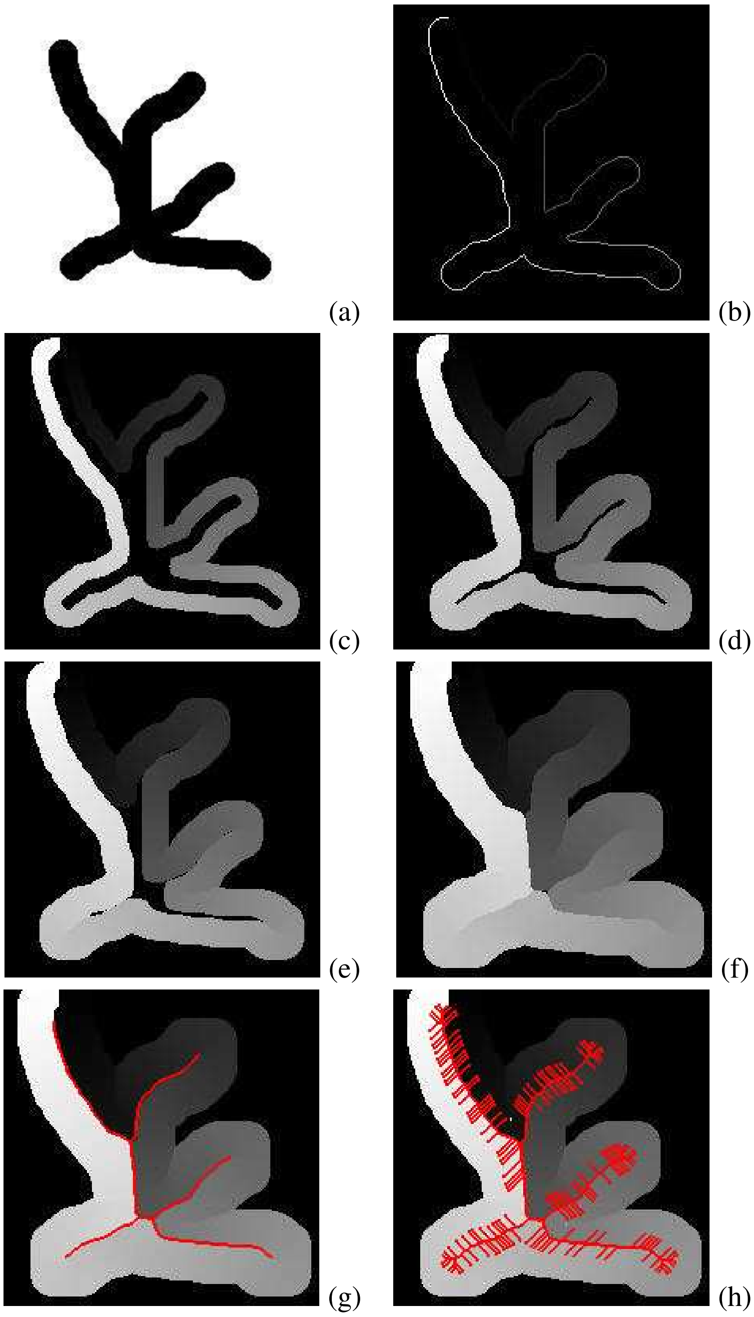
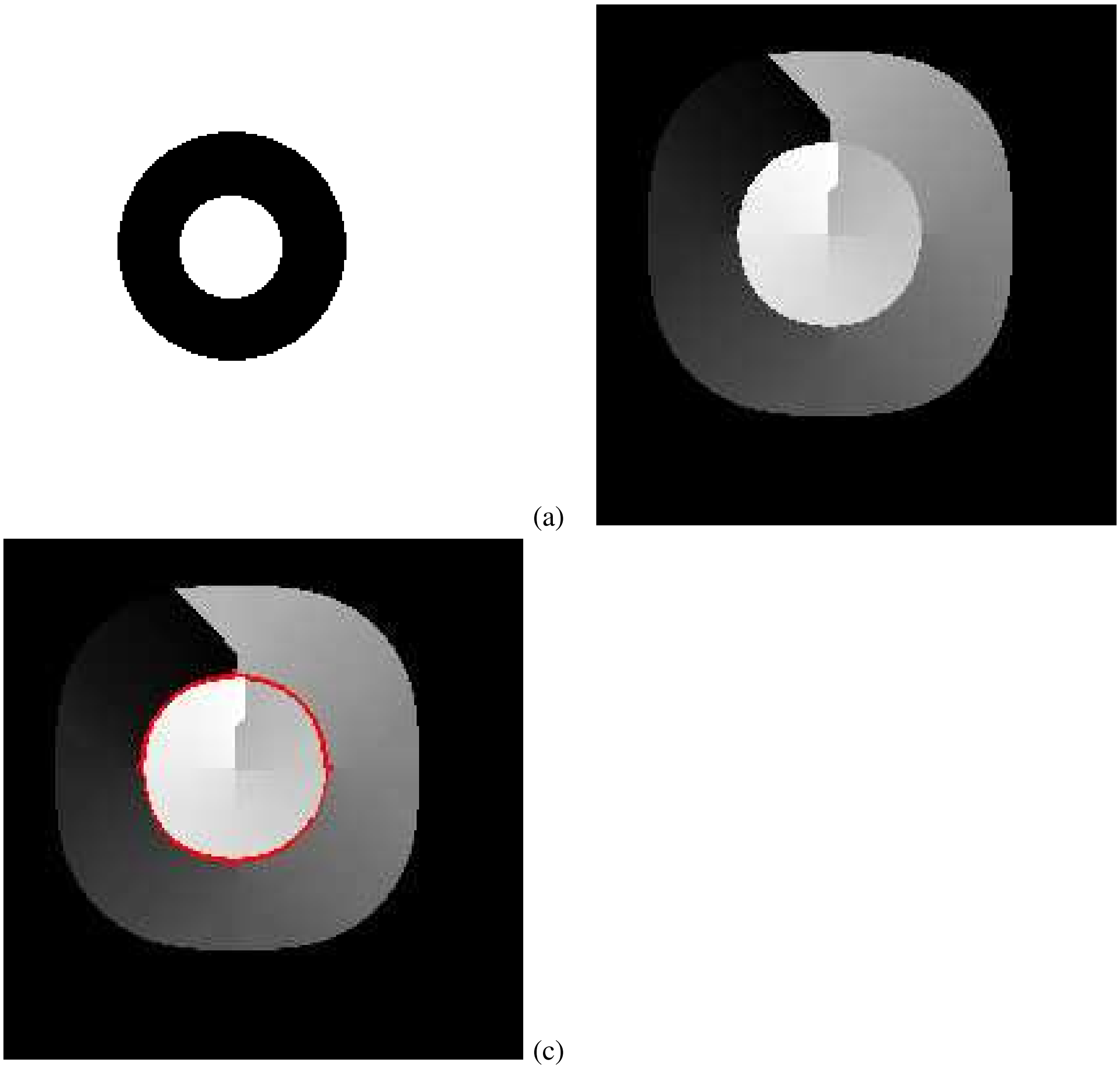


Figura 6: **(a)** *Um objeto simples;* **(b)** *o contorno desse objeto devidamente rotulado;* **(c-f)** *dilatações (cada vez maiores) do rótulo do contorno;* **(g)** *local de fechamento (um dos esqueletos internos do objeto);* **(h)** *outro esqueleto interno, obtido usando-se um limiar menor;*

do buraco. Como rotular os pixels desses dois contornos? Dado que o contorno do disco possui N pixels, e o contorno do buraco M pixels, uma abordagem seria rotular os contornos de modo **independente**. Desse modo, os pixels do contorno do disco seriam rotulados de **1 a N**, e os do contorno do buraco seriam rotulados de **1 a M**. Outra abordagem seria rotular os contornos de modo **dependente**. Desse outro modo, o contorno do disco seria rotulado de **1 a N**, e o contorno do buraco seria rotulado de **N+1 a N+M**. Como queremos que no local de fechamento se encontrem pixels com rótulos bem diferentes, de modo que o valor da imagem diferença nesse local seja alto, a melhor solução é utilizar **rótulos dependentes**. As figuras 7b e 7c mostram, respectivamente, uma dilatação exata do disco usando essa solução, e o local de fechamento do disco. Note como no local de fechamento temos uma variação brusca de tons de cinza.

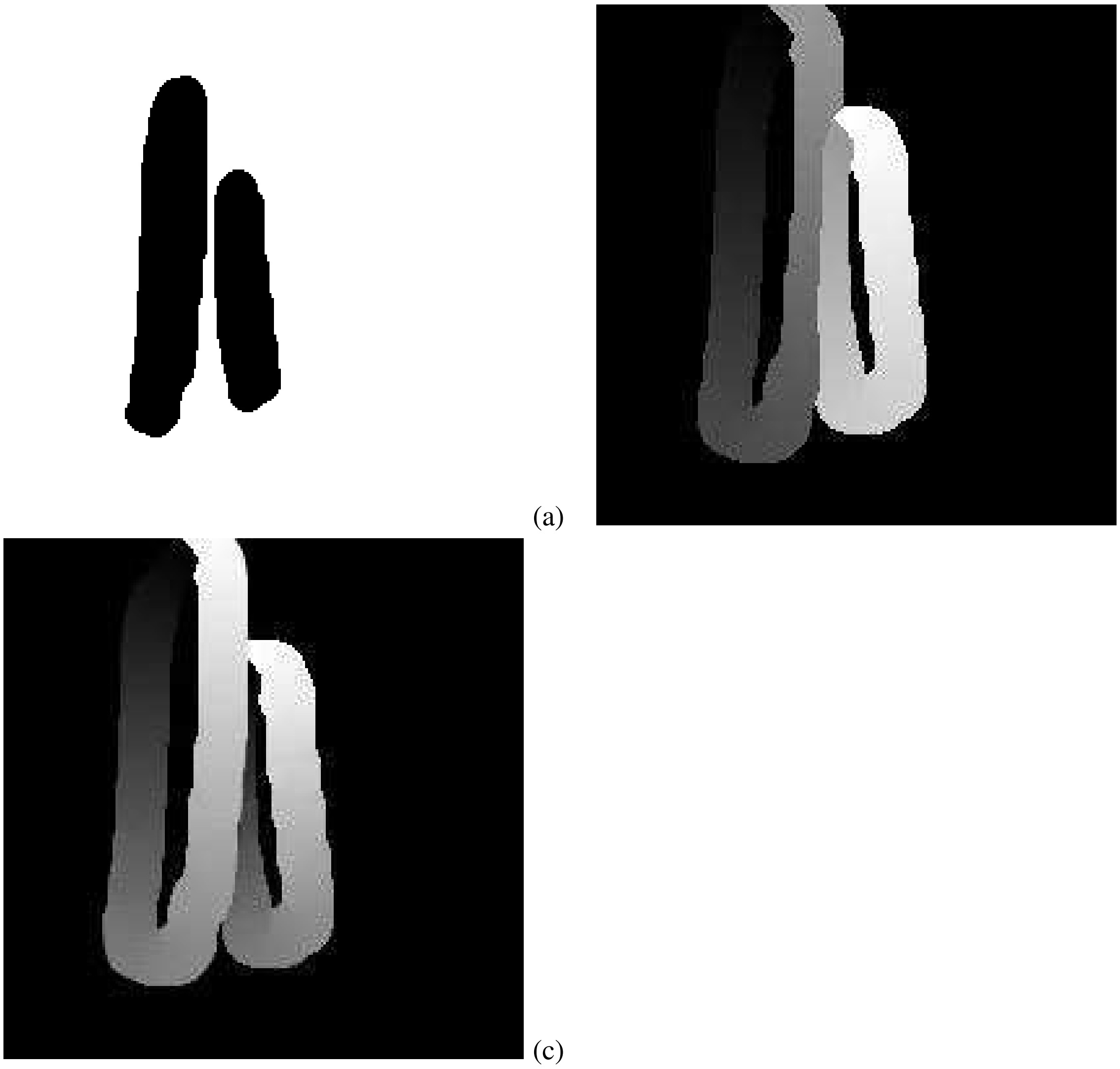


(a) (b) (c)

Figura 7: **(a)** *Um disco com um buraco;* **(b)** *uma dilatação exata dos contornos do disco;* **(c)** *local de fechamento ;*

Esta abordagem de rótulos dependentes pode ser utilizada sem problemas para rotular todos os contornos de todos os objetos e de todos os buracos de uma imagem. As dilatações dos contornos dos buracos de um objeto devem se chocar entre si e devem se chocar também com a dilatação do contorno do próprio objeto, produzindo um local de fechamento, como nós desejamos. As dilatações de buracos pertencentes a objetos diferentes nunca se chocarão entre si. No entanto, dilatacões de objetos podem se chocar externamente (figuras 8a e 8b). Como esqueletos externos serão desprezados, isso não tem importância.

Como a dilatação exata deve ser executada? O algoritmo de detecção de contornos os salva em sequência em uma mesma lista ligada. Seja D o número de dilatações exatas necessárias para fechar o espaço interno de todos os objetos. Então, será necessário percorrer a lista ligada D vezes, cada vez executando uma dilatação. Se decidirmos primeiro



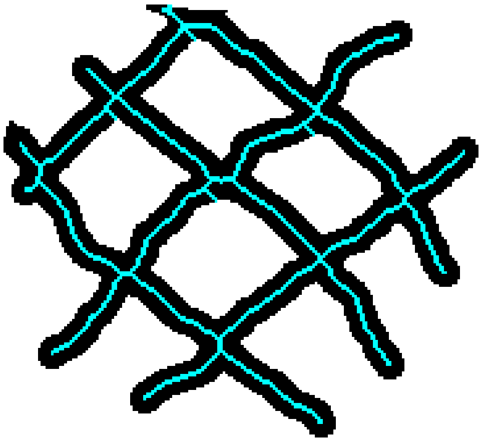
(a) (b) (c)

Figura 8: **(a)** *Dois objetos;* **(b)** *uma dilatação exata dos contornos desses objetos. Os contornos se chocam externamente;* **(c)** *Algo que não deve acontecer: dilatações que se sobrepõe;*

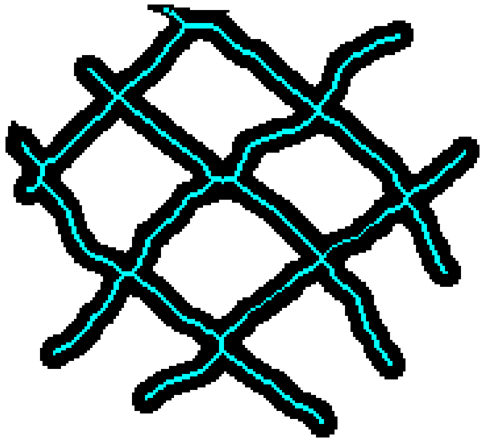
dilatar D vezes apenas um contorno, depois dilatar D vezes o próximo contorno, e assim por diante, acabaremos fazendo com as dilatações dos contornos se sobreponham umas às outras, ao invés de se chocarem (figura 8c), o que não deve acontecer.

O algoritmo de dilatações exatas proposto em [9] utiliza uma estrutura de dados auxiliar, chamada **SEDR** (*Sorted Euclidean Distance Representation*). Esta estrutura salva, para cada distância euclidiana exata, a **posição relativa** [6] de cada um dos pixels que se encontram a essa distância. Esta estrutura pode ser armazenada em uma matriz ou em um conjunto de listas ligadas. Nós adotamos uma abordagem diferente em nosso algoritmo, e não utilizamos nem uma matriz nem uma lista ligada, mas sim um **vetor** $SEDR$ **de** D **posições**, uma posição para cada dilatação exata. A posição $SEDR(k)$, $0 <= k <= D$, armazena a posição relativa de apenas um dos pixels que se encontram a distância exata k. Os outros pixels podem ser encontrados por *8-way symmetry*. Desse modo economizamos o uso de memória.

[6] Relativa a posição (0,0) do plano cartesiano.



(a) (b)



(c)

Figura 9: **(a)** *Um objeto simples com 4 buracos;* **(b)** *objeto e seu esqueleto interno. Note que o esqueleto possui 4 falhas, cada falha relacionada a um buraco. Cada falha não é nada mais que o local de choque dos rótulos dos pixels que pertencem às extremidades do contorno do buraco;***(c)** *Esqueleto utilizando-se a matriz auxiliar* `maux`*;*

Temos ainda um problema a ser resolvido. Seja um contorno de comprimento N. Seus pixels serão rotulados com valores de 1 a N. Note que a dilatação do pixel de valor 1 (que pertence a uma extremidade do contorno) se chocará com a dilatação do pixel de valor N (pertencente a outra extremidade), produzindo uma região de alto valor numérico na imagem diferença e, consequentemente, criando uma falha no esqueleto. O algoritmo que calcula a imagem diferença resolve isso impedindo que os valores contidos na imagem diferença ultrapassem o valor $N/2$. Vamos chamar essa abordagem de **tratamento de choque**. Utilizando o tratamento de choque, o algoritmo que calcula a imagem diferença assume o seguinte formato:

```
// Esqueleto(): função C que recebe a imagem de rótulos img_lbl
// (de dimensões NxM) de um contorno de comprimento R, e calcula
// a imagem diferença:
```

```
void Esqueleto ( int ** img_lbl, int N, int M, int ** img_dif,
                 int R ) {

  int x, y, max, v[4];

  for ( x = 1; x < N - 1; x++ )
    for ( y = 1; y < M - 1; y++ ) {
      v[0] = abs (img_lbl[x][y] - img_lbl[x+1][y]);
      v[1] = abs (img_lbl[x][y] - img_lbl[x][y+1]);
      v[2] = abs (img_lbl[x][y] - img_lbl[x-1][y]);
      v[3] = abs (img_lbl[x][y] - img_lbl[x][y-1]);
      max = Maximo (v,4);  // máximo do vetor v[0..3]
      if ( max < R/2.0 ) img_dif[x][y] = max;
      else img_dif[x][y] = abs (R - max);
    }
}
```

No entanto, como executar o tratamento de choque para uma imagem que contém diversos contornos, cada contorno com um comprimento diferente? Se esse problema não for resolvido, iremos obter um esqueleto cheio de falhas, como mostrado na figura 9b.

A solução é utilizar uma matriz auxiliar `maux` que **associa, a cada pixel de rótulo P da imagem de rótulos, o comprimento do contorno que contém o rótulo P** [7]. Seja (x, y) a posição desse pixel, e seja N o comprimento desse contorno. Devemos impedir que a posição (x, y) da imagem diferença possua um valor maior que $N/2$. O novo algoritmo de cálculo da imagem diferença terá o seguinte formato:

```
//   Esqueleto_Supremo(): função C que recebe as matrizes img_lbl
// (NxM) e maux (NxM), onde img_lbl é a imagem de rótulos, e maux
// é a matriz que associa cada pixel de rótulo P de img_lbl com o
// comprimento do contorno que contém o rótulo P.
//   Esta função calcula a imagem diferenca img_dif.

void Esqueleto_Supremo ( int ** img_lbl, int N, int M,
                         int ** img_dif, int ** maux ) {

   int x, y, max, v[4], r;

   for( x = 1; x < N-1; x++ )
     for( y = 1; y < M-1; y++ ) {
       r = maux[x][y];
       v[0] = abs (img_lbl[x][y] - img_lbl[x+1][y]);
       v[1] = abs (img_lbl[x][y] - img_lbl[x][y+1]);
       v[2] = abs (img_lbl[x][y] - img_lbl[x-1][y]);
       v[3] = abs (img_lbl[x][y] - img_lbl[x][y-1]);
       max = Maximo (v,4);
       if( max < r/2.0 ) img_dif[x][y] = max;
       else img_dif[x][y] = abs(r - max);
     }
}
```

[7] Nota: lembre-se que pixels diferentes **pertencentes a algum contorno** nunca receberão o mesmo rótulo (estamos rotulando os contornos de modo **dependente**).

Esse algoritmo permite que o esqueleto seja extraído melhor, como mostrado na figura 9c.

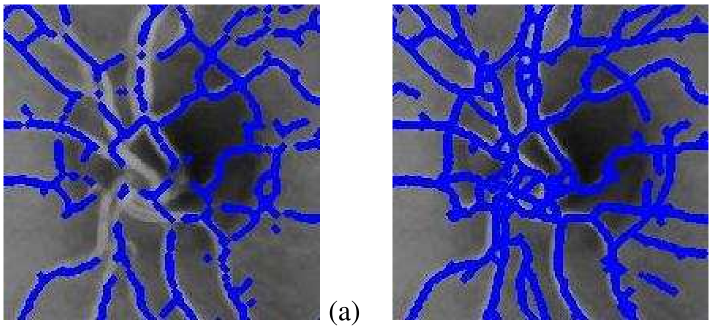
Como calcular `maux`? Inicialmente, cada pixel (x, y) de `maux` deve receber o tamanho t do comprimento do contorno que passa pela posição (x, y) da imagem binária inicial. Depois, cada valor t deve ser propagado em `maux` utilizando-se dilatação exata (o número de dilatações exatas utilizadas para se obter a imagem de rótulos deve ser igual ao número de dilatações exatas utilizadas para se obter `maux`).

4.3 Resultados obtidos

Nas figuras 10 a 14 temos alguns exemplos de resultados obtidos. Cada imagem é um detalhe de uma fotografia da retina com o esqueleto sobreposto em azul. As imagens à esquerda foram obtidas utilizando-se o algoritmo antigo de extração de esqueletos (baseado em morfologia matemática), enquanto as imagens à direita foram obtidas através do novo algoritmo.

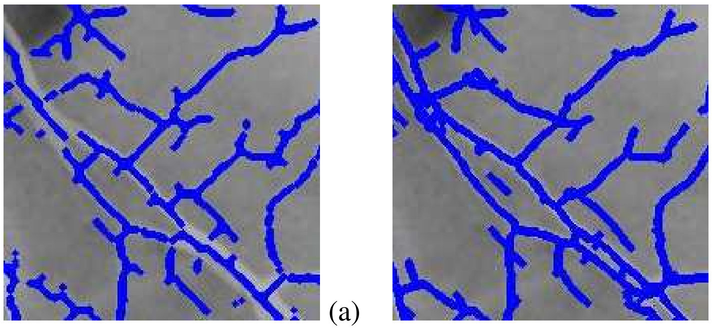
Deve ficar claro que a fase de extração do esqueleto se segue à fase de segmentação dos vasos sanguíneos. Desse modo, falhas (pequenas ou não) na segmentação provocam falhas no esqueleto.

Uma escala (ou *limiar*) entre 10 e 40 tem gerado esqueletos suficientemente eficientes.



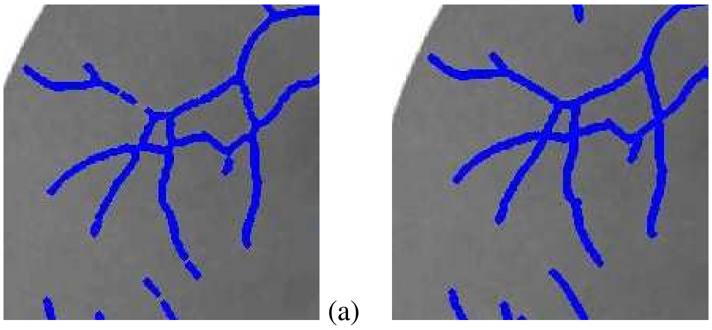
(a) (b)

Figura 10: **(a)** *Antes.* **(b)** *Depois.*



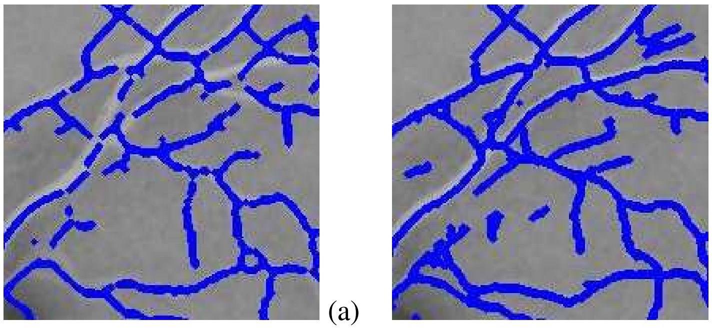
(a) (b)

Figura 11: **(a)** *Antes.* **(b)** *Depois.*



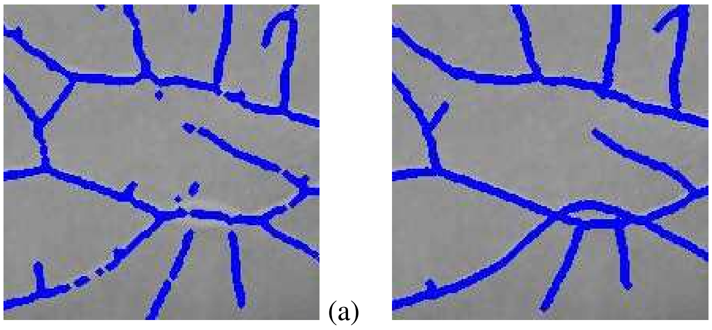
(a) (b)

Figura 12: **(a)** *Antes.* **(b)** *Depois.*



(a) (b)

Figura 13: **(a)** *Antes.* **(b)** *Depois.*



(a) (b)

Figura 14: **(a)** *Antes.* **(b)** *Depois.*

5 Eliminando a borda do disco óptico: algoritmos e resultados

5.1 O algoritmo

Vamos agora analisar o problema da detecção da borda do disco óptico, a abordagem utilizada para solucioná-lo e os problemas dessa abordagem.

Já vimos que a borda do disco óptico pode ser detectada juntamente com os vasos sanguíneos pois, do mesmo modo que os vasos, a borda se contrasta com o fundo da imagem.

O passo seguinte a detecção dos vasos é o cálculo do esqueleto dos vasos. Caso a borda do disco óptico seja detectada no passo anterior, obteremos uma imagem que contém tanto o esqueleto dos vasos quanto o esqueleto da borda do disco (figura 15).

Nossa abordagem para solucionar o problema consiste em tentar apagar o esqueleto da

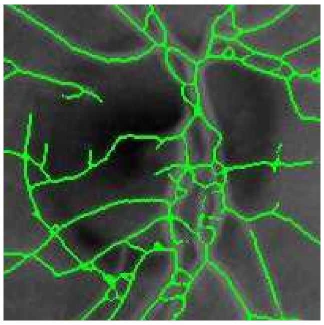


Figura 15: Disco óptico e seu esqueleto (em verde). Note que o esqueleto percorre a borda do disco.

borda do disco. Assim, não tentaremos impedir que a borda do disco seja detectada, nem que o esqueleto da borda seja calculado, mas sim tentaremos apagar este esqueleto caso ele exista.

Vamos agora começar a descrever o algoritmo em questão. Para entendê-lo com precisão são necessários alguns conhecimentos básicos de morfologia matemática.

O primeiro passo do algoritmo consiste em tentar localizar o disco óptico. Como o disco óptico é a parte mais escura da imagem, basta detectar os pixels mais escuros. Isto pode ser feito utilizando-se um *limiar global* suficientemente baixo (figura 16.a). Utilizar um *limiar* global detecta o interior do disco, mas a imagem resultante pode conter pequenos ruídos, como pode ser visto na figura 16.a. Esta imagem também pode ser formada por componentes não conexos, já que o interior do disco contem vasos sanguíneos, e estes não serão detectados pelo *limiar* (por serem muito claros).

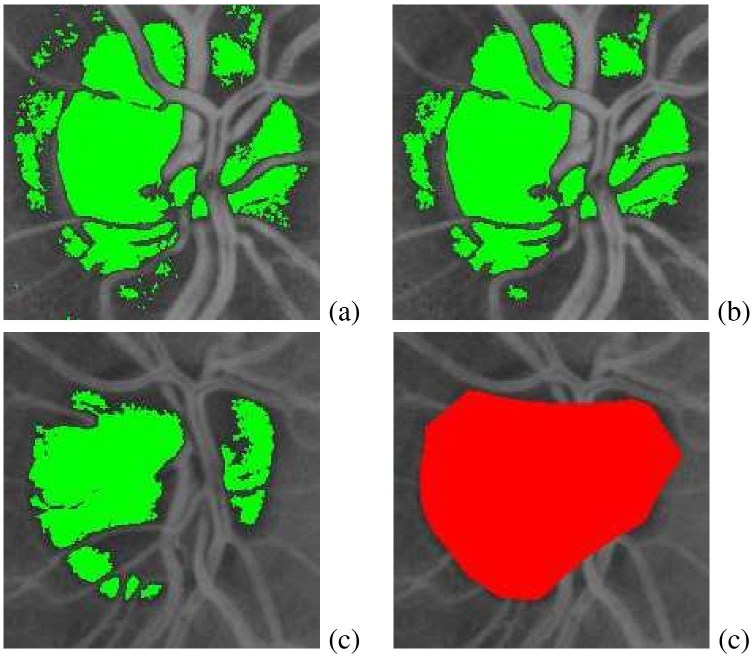


Figura 16: **(a)** *Região detectada através de um limiar global.* **(b)** *Os ruídos são eliminados utilizando-se abertura de áreas.* **(c)** *Outro exemplo com os ruídos já eliminados.* **(d)** *Fecho do interior do disco (c).*

Os ruídos podem ser eliminados utilizando-se **abertura morfológica de áreas**. Abertura de áreas é uma técnica que elimina todos os componentes conexos de área menor que um determinado valor. Como queremos eliminar ruídos, devemos eliminar todos os componentes de área bem pequena (figura 16.b).

O passo seguinte é unir todos os componentes conexos que sobraram em um único componente, de forma que possamos ter uma representação melhor do interior do disco óptico. Isto pode ser obtido utilizando-se **fechamento morfológico**. Optamos por utilizar fechamentos morfológicos que utilizam discos como elementos estruturantes. Aplicando um fechamento morfológico na figura 16.c, utilizando um disco de raio 210 como elemento estruturante, obtemos a forma da figura 16.d. Vamos chamar esta forma de **fecho do interior do disco**.

A idéia do algoritmo é detectar a intersecção entre a **borda do fecho** e o **esqueleto da imagem**, eliminando todas as partes do esqueleto que pertencem a esta intersecção. A figura 17 nos mostra um resultado obtido utilizando essa técnica. Em vermelho temos a borda do fecho, em verde o esqueleto da imagem, e em azul a intersecção entre a borda do fecho e o esqueleto.

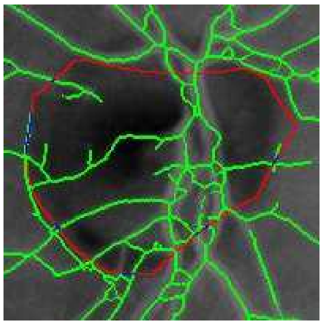


Figura 17: Intersecção entre a borda do fecho e o esqueleto da imagem.

Note que a intersecção intercepta apenas uma pequena parte do esqueleto da borda do disco óptico. É muito difícil que a borda do fecho intercepte totalmente este esqueleto. A solução é **dilatar sucessivamente o fecho do disco**. Para cada dilatação, encontro a sua **borda** e a **intersecção desta borda com o esqueleto**. Cada intersecção encontrada é eliminada do esqueleto. As figuras 18.a a 18.f mostram uma série de intersecções entre a borda de seis diferentes dilatações consecutivas do fecho e o esqueleto. Após uma série de 15 dilatações, se eliminarmos do esqueleto todas as intersecções obtidas, obtemos o resultado da figura 19.

Note que mais do que o esqueleto da borda do disco óptico é eliminado. Isto ocorre por dois motivos:

- a borda do fecho intersecta não apenas o esqueleto do disco, mas também esqueletos de vasos sanguíneos.
- existem vasos sanguíneos que circundam o disco óptico. Os esqueletos desses vasos podem ser “confundidos” com o esqueleto da borda do disco.

Não encontramos uma solução para o segundo motivo. Para o primeiro motivo existe uma solução parcial: após calcular a intersecção entre o esqueleto e a borda do fecho (ou a borda da dilatação do fecho) não devemos eliminar essa intersecção imediatamente do esqueleto. Essa intersecção deve passar por um processamento. Esperamos que a intersecção contenha dois tipos básicos de formas:

- a intersecção da **borda do fecho** com o esqueleto da borda do **disco** deve ter a forma

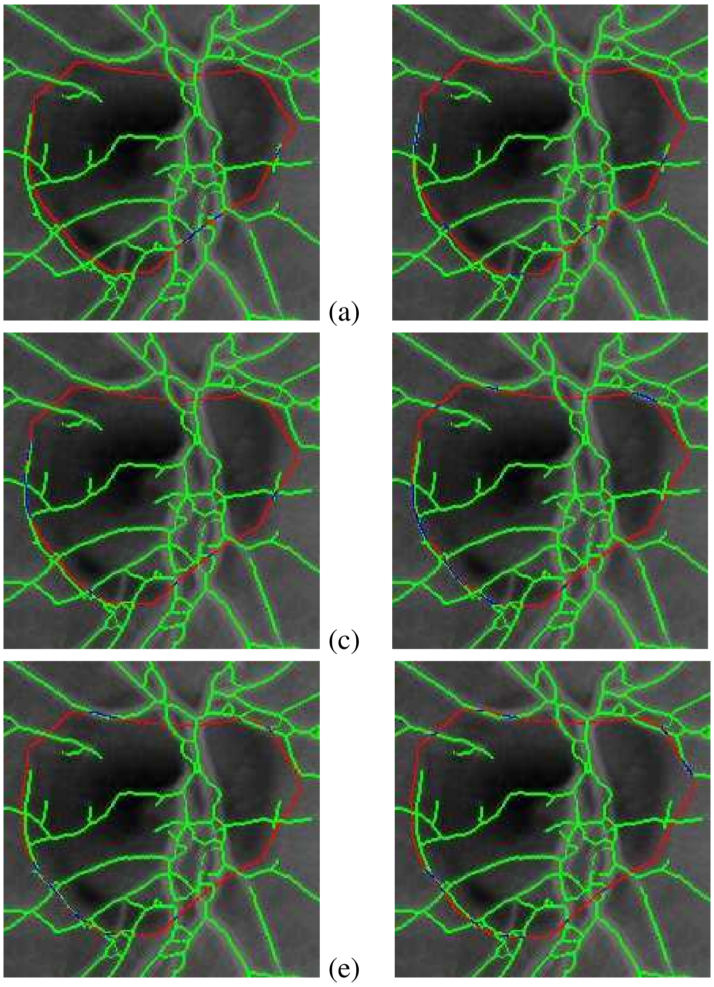


Figura 18: **(a-f)** *Em azul: a intersecção entre a borda de uma dilatação do fecho com o esqueleto da imagem.*

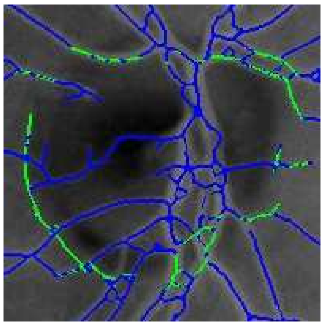


Figura 19: Em azul temos o esqueleto resultante. Em verde as partes do esqueleto original que foram eliminadas.

de um **arco** caso esta borda e este esqueleto se comportem bem, isto é, tenham formas circulares.

- a intersecção da **borda do fecho** com esqueletos de **vasos sanguíneos** deve produzir **pontos** (ou retas muito pequenas), pois esta borda não deve percorrer estes vasos, e sim atravessá-los rapidamente em um único ponto.

O processamento da intersecção consiste em eliminar dela todas as formas de área muito pequena. Esperamos deste modo eliminar os pontos e não eliminar os arcos. No entanto, nada impede que a intersecção entre a borda do fecho com o esqueleto do disco seja um ponto, ou que a intersecção da borda do fecho com o esqueleto de um vaso seja um arco ou uma reta grande.

Supondo que os pontos foram eliminados da intersecção (e que os arcos não foram), basta subtrair esta intersecção do esqueleto para eliminar os arcos deste esqueleto (isto é, eliminar o esqueleto da borda do disco).

A imagens da figura 18 e 19 foram obtidas aplicando-se este processamento na intersecção, e como podemos ver esqueletos de vasos sanguíneos não deixam de ser eliminados.

Segue abaixo um pseudo-código do algoritmo, que recebe uma imagem do fundo da retina e seu esqueleto, e devolve o novo esqueleto obtido.

Algorithm 1 : Borda (imagem, esqueleto)

$disco\_optico = imagem < limiar$ // limiar global.
$disco\_optico =$ Abertura de áreas ($disco\_optico$) // eliminando ruídos.
$fecho =$ Fechamento morfológico ($disco\_optico$)
// o fecho será dilatado N vezes:
for $i = 1..N$ **do**
  $fecho\_dil =$ Dilatação morfológica ($fecho$)
  $fecho\_borda = fecho\_dil - fecho$
  $interseccao_0 =$ Intersecção ($esqueleto, fecho\_borda$)
  // eliminando pontos da intersecção:
  $interseccao_1 =$ Abertura de areas ($interseccao_0$)
  $esqueleto = esqueleto - interseccao_1$
  $fecho = fecho\_dil$
end for
return $esqueleto$

Algumas observações sobre o algoritmo:

- caso o valor de N seja muito alto, podemos acabar eliminando estruturas demais do esqueleto.

- a qualidade do resultado obtido pelo algoritmo depende da precisão com que o disco óptico é detectado. Um valor de *limiar* que detecta o disco com precisão em uma imagem pode não detectar o disco corretamente em outra imagem.

- a dilatacao morfológica do $fecho$ pode ser executada utilizando-se como elemento estruturante um disco de raio 1. A idéia é dilatar o fecho em apenas um pixel. A diferenca entre o fecho dilatado ($fecho\_dil$) e o próprio fecho nos dá a borda do fecho dilatado.

5.2 Resultados obtidos

Alguns resultados são apresentados na figura 20. Para todas estas imagens o disco óptico foi detectado com um *limiar* de valor 65. O fecho do disco foi calculado utilizando-se um disco de raio 200 como elemento estruturante, e foram realizadas 15 dilatações do fecho.

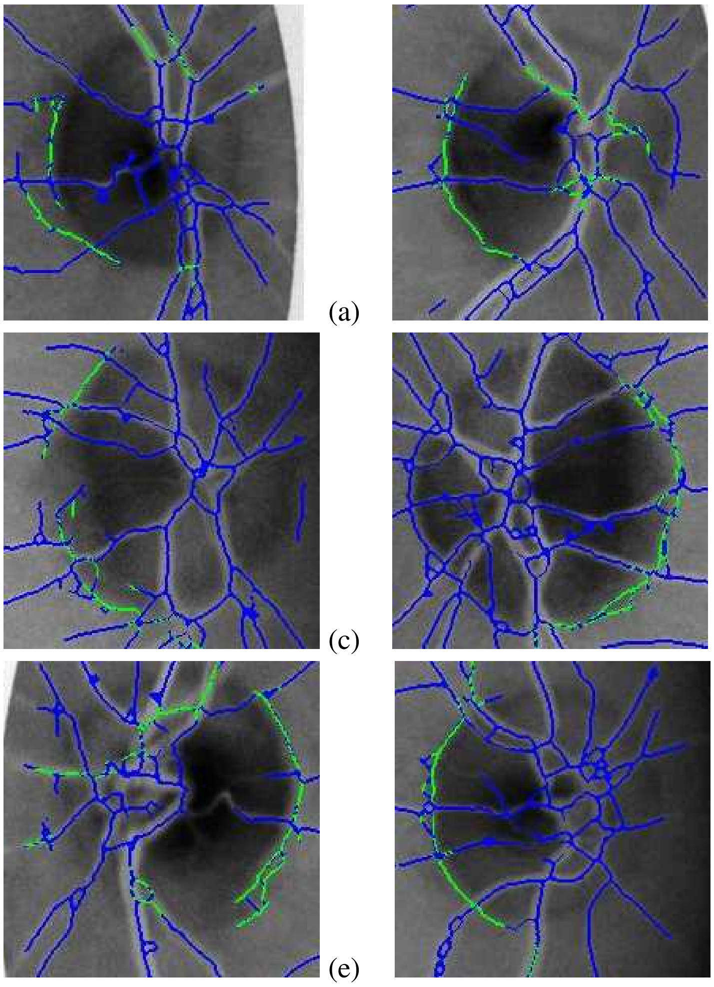

(a) (b) (c) (d) (e) (f)

Figura 20: Em azul temos o esqueleto resultante. Em verde as partes do esqueleto original que foram eliminadas.

6 Conclusão

Dos dois problemas que nos dispomos a resolver um, o cálculo do esqueleto, foi bem solucionado. O outro, o problema da borda do disco óptico, ainda necessita de muito trabalho. Técnicas mais avançadas como aprendizagem computacional poderão se mostrar mais eficientes.

7 Integração com o BCC

De todas as disciplinas do BCC, pelo menos três foram essenciais durante a execução deste trabalho:

- *Introdução à computação*: obviamente esta disciplina é necessária para qualquer trabalho de computação. O interessante é que as estruturas de dados mais complexas utilizadas na execução deste trabalho foram matrizes e listas ligadas, ou seja, quase todas as estruturas utilizadas são do conhecimento de qualquer aluno que cursou esta disciplina.

- *Princípios de desenvolvimento de algoritmos*: disciplina essencial para se conseguir trabalhar com listas ligadas. Esta também é a disciplina em que aprendi a programar em C, linguagem utilizada no cálculo do esqueleto devido a sua eficiência.

- *Análise e reconhecimento de formas*: o trabalho apresentado aqui se encaixa nas áreas de visão e processamento de imagens, duas áreas cobertas por esta disciplina. Os assuntos essenciais para o desenvolvimento deste trabalho e tratados por esta disciplina foram, além dos conceitos básicos de visão e processamento de imagens, análise multi-escala e morfologia matemática.

Visão e processamento de imagens não são duas áreas simples de se aprender nem de se trabalhar. Embora todos os conceitos tenham aplicações práticas, alguns envolvem matemática avançada como cálculo, álgebra linear e estatística. O mais indicado para alguém que gostaria de seguir carreira nesa área seria ingressar em programas de mestrado e doutorado.

A única frustração encontrada na realização do trabalho foi a incapacidade de resolver o problema do disco óptico. No entanto esse problema é um desafio realmente muito difícil, e não tínhamos certeza desde o início do trabalho se conseguiríamos resolvê-lo. Ficou claro ao longo do tempo que o cálculo do esqueleto era um desafio menor, embora bem complexo. Aliás, o mais difícil no cálculo do esqueleto foi a etapa anterior ao seu cálculo, ou seja, a detecção dos contornos dos objetos da imagem.

Este foi um trabalho realizado individualmente. Praticamente quase toda a cooperação que obtive veio da parte do meu orientador.

Referências

[1] A.Pinz, S.Bernogger, P.Datlinger, and A.Kruger. Mapping the human retina. *IEEE TRANSACTIONS ON MEDICAL IMAGING*, 17(4), august 1998.

[2] O. Chutatape, L. Zheng, and S. M. Krishnan. Retinal blood vessel detection and tracking by matched gaussian and kalman filters. *20th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, (EMBS'98), Hong Kong*, pages 3144–3149, 29 October - 1 November 1998.

[3] F.Zana, J.C.Klein, T.Walter, and P.Massin. Automatic segmentation and registration of retinal fluorescein angiographies. *First International Workshop on Computer Assisted Fundus Image Analysis (CAFIA), Copenhagen, Denmark*, may 2000.

[4] L. Gang, O. Chutatape, and S. M. Krishnan. Detection and measurement of retinal vessels in fundus images using amplitude modified second-order gaussian filter. *IEEE*, 2002.

[5] X. W. Gao, A. Bharath, A. Stantona, A. Hughes, N. Chapman, and S. Thom. Quantification and characterisation of arteries in retinal images. *Comput. Methods Programs Biomed.*, 2000.

[6] A. Hoover, V. Kouznetsova, and M. Goldbaum. Locating blood vessels in retinal images by piece-wise threshold probing of a matched filter response. *IEEE Transactions on Medical Imaging*, 19(2):203–210, march 2000.

[7] M. V. Ibañez and A. Simó. Bayesian detection of the fovea in eye fundus angiographies. *Pattern Recognition Letters*, 1999.

[8] H. Jelinek and R. M. Cesar-Jr. *Angiography*, chapter Segmentation of retinal fundus vasculature in non-mydriatic camera images using wavelet. IEEE, 2003 (to appear).

[9] R.M. Cesar Jr. and L.F. Costa. *Shape Analysis and Classification*. CRC Press, 2001.

[10] J.C. Klein and T. Walter. Segmentation of color fundus images of the human retina: detection of the optic disc and the vascular tree using morphological techniques. *Second International Symposium on Medical Data Analysis (ISMDA) October 9-11, Madrid, Spain*, 2001.

[11] M. Lalonde, L. Gagnon, and M. Boucher. Non-recursive paired tracking for vessel extraction from retinal images. *Presented at the Conference Vision Interface, Montréal*, mai 2000.

[12] H. Li and O. Chutatape. Automatic location of the optic disc in retinal images. *IEEE*, 2001.

[13] S.Chaudhuri, S.Chatterjee, N.Katz, M.Nelson, and M.Goldbaum. Detection of blood vessels in retinal images using two-dimensional matched filters. *IEEE Transactions on Medical Imaging*, 8(3):263–269, 1989.

[14] T.Spencer, J.A.Olson, K.C.McHardy, P.F.Sharp, and J.V.Forrester. An image-processing strategy for the segmentation and quantification of microaneurysms in fluorescein angiograms of the ocular fundus. *COMPUTERS AND BIOMEDICAL RESEARCH*, 1996.