EXTRAÇÃO AUTOMÁTICA DE UM GRAFO A PARTIR DE IMAGENS DE MAPAS URBANOS

Mara Franklin Rios

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

Prof. Antônio Alberto Fernandes Oliveira, D.Sc

Prof. Edilberto Strauss, Ph.D.

Prof. Marcos Negreiros, D.Sc.

Prof. Adilson Elias Xavier, D.Sc.

RIO DE JANEIRO, RJ - BRASIL
MARÇO DE 2003

RIOS, MARA FRANKLIN

Extração automática de um grafo a partir de imagens de mapas urbanos. [Rio de Janeiro] 2003

XVIII, 111 p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 2003)

Tese – Universidade Federal do Rio de Janeiro, COPPE

1 - Mapas Urbanos

2 - Processamento de Imagens

3 - Vetorização

4 - Triangulação Restrita de Delaunay

I. COPPE/UFRJ II. Título (série)

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

EXTRAÇÃO AUTOMÁTICA DE UM GRAFO A PARTIR DE IMAGENS DE MAPAS URBANOS

Mara Franklin Rios

Março/2003

Orientadores: Antônio Alberto Fernandes Oliveira
Edilberto Strauss

Programa: Engenharia de Sistemas e Computação

Apresentamos uma solução completa para extração automática de um grafo a partir de imagens digitalizadas de mapas urbanos. O sistema implementado divide o processo de detecção em três etapas. Na primeira, os mapas digitalizados são submetidos a um processamento de imagens visando facilitar o processo de vetorização que é realizado em seguida. Finalmente, por intermédio de uma Triangulação Restrita de Delaunay no mapa vetorizado os elementos de interesse - esquinas, ruas e quarteirões - são identificados. O grafo extraído é uma estrutura G=(V,E), onde V representa o conjunto de esquinas detectadas e E representa o conjunto de ligações entre essas esquinas, correspondendo às ruas e avenidas do mapa.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

AUTOMATIC GRAPH EXTRACTION FROM DIGITAL URBAN MAPS

Mara Franklin Rios

March/2003

Advisors: Antônio Alberto Fernandes Oliveira.
Edilberto Strauss.

Department: Systems Engineering and Computer Science

We present a complete solution for the automatic extraction of a graph from digitized urban maps. The implemented system divides the whole process into three different phases. In the first phase, the digitized maps are preprocessed in order to prepare the image for the next stage, the vectorization phase. The obtained vectorial representation of the map is a set of polygons whose vertices, in the third phase, are inserted into a Constrained Delaunay Triangulation. Then, we can partition the map into regions of interest identifying streets, corners and blocks. The resulting graph is a structure $G = (V, E)$, where V represents the set of all detected corners and E represents the set of connections among these corners, corresponding to the streets and avenues in the urban map.

Agradecimentos

A Deus.

À minha querida mãe Eunice, cujo exemplo de perseverança e gosto pelos estudos me inspirou e motivou. À minha irmã Mariza e meu cunhado Carlos por me acolherem em Niterói. Às minhas sobrinhas Larissa e Marianna por seu carinho. À Márcia, por sua prestatividade quando estou em Fortaleza. Ao meu irmão Marcelo, meu agradecimento.

Ao meu marido, Tibérius Bonates, por estar sempre presente nos momentos decisivos deste trabalho, com seu carinho e seus valiosos conselhos.

A todos os meus parentes, que, de perto ou de longe, de forma manifesta ou silenciosa, estiveram a me desejar sucesso nesta empreitada. Em especial, à minha tia Marlene, com suas dicas de culinária para fazer sucesso nos eventos realizados nas repúblicas, e à minha vovó Geralda, que tornou mais doce minha estada longe de casa, enviando seus deliciosos doces de frutas regionais.

Aos meus sogros, Célio e Margarida Bonates, por todo o apoio, torcida e orações. Muito obrigada.

Ao meu ilustre orientador Antônio Oliveira, por sua generosidade, sua atenção, seus sábios conselhos e, sobretudo, por seu exemplo.

Ao professor Strauss, por sua amizade e por me coorientar neste trabalho.

Aos meus professores de graduação. De maneira bastante especial àqueles que me ajudaram a ingressar nesta Universidade. Aos professores Marcos Negreiros, Wamberto Vasconcelos e Plácido Pinheiro gostaria de agradecer não somente pelos conhecimentos transmitidos e pela ajuda que me ofereceram na época de seleção para o mestrado, mas, também, pela oportunidade de conviver com eles, aprendendo a partir de seus exemplos.

Ao professor Nelson Maculan, por sua amizade e disposição para ajudar e motivar todos os alunos. À professora Maria Helena pela amizade e pelo apoio.

Ao Professor Adilson Xavier, por aceitar o convite para participar da avaliação deste trabalho.

A todos os amigos de Fortaleza e do Rio de Janeiro, recentes e de longas datas. Muito obrigada. Citar os nomes de todos seria estender-me demais, correndo, ainda, o risco de cometer uma injustiça ao esquecer alguém por uma falha de minha memória.

À toda a Ilustre Bancada Cearense, exemplo de pessoas competentes, responsáveis e sobretudo, unidas. A bancada alegra os corredores da COPPE. A prestatividade dos membros da bancada, cearenses ou agregados, são um exemplo de que para crescer é preciso ajudar. Em especial, gostaria de agradecer ao casal Elder e Ana Flávia Macambira, que mesmo estando tão ocupados me ajudaram sendo meus procuradores nos detalhes burocráticos para a conclusão deste curso.

A todos os colegas de estudos, funcionários e professores do Programa. De forma especial, às funcionárias Cláudia Prata, Solange, Mercedes e Gercina, de atenção e paciência inesgotáveis. À Lourdes, cujo cafezinho foi de suma importância para a conclusão deste trabalho. À Fátima, por sua amizade e agradável companhia no almoço e no cafezinho.

Ao pessoal do LCG, em especial ao Antônio Lopes, por sua atenção e paciência para dar as dicas importantes sobre os mais variados assuntos do laboratório e da implementação deste trabalho. Ao Walter, Luís Marcos e Ítalo o meu agradecimento pela ajuda nas horas em que eu esbarrava em alguma dificuldade.

À minha querida amiga Margareth e sua família, que sempre me recebeu com muito carinho em sua casa. Muito obrigada.

À Universidade Estadual do Ceará.

À Universidade Federal do Rio de Janeiro.

À CAPES pela bolsa de estudos concedida durante o período de mestrado.

Ao 485 e ao 998.

Sumário

Lista de Figuras

À minha querida mãe, Eunice.
Ao meu amor, Tibérius.

Capítulo 1

Introdução

Regiões urbanas são caracterizadas por apresentarem uma densa malha viária, constituída em sua maior parte por **ruas**. As ruas podem então ser definidas dentro do contexto urbano como um espaço servindo como passagem e fluxo para veículos, pessoas e animais. Elas são importantes para comunicação entre diferentes regiões as quais se precisa acessar, de modo que as ruas são organizadas como uma rede.

Desta forma, a maioria dos problemas relacionados à infra-estrutura de uma cidade podem ser modelados em uma estrutura conhecida como **Grafo**. Dentre esses problemas, podemos destacar: cabeamento, distribuição de gás, coleta de lixo, planejamento de rotas, entre outros.

Um grafo é definido como uma estrutura $G = (V, E)$, onde V é o conjunto de vértices representando pontos de interesse na região em estudo e E é conjunto de elos, ou arestas, denotando as ligações entre os vértices de V [1].

Neste trabalho buscamos a obtenção de um grafo que seja equivalente a uma dada região urbana, a partir da interpretação de seu mapa digitalizado, a fim de se extrair as ruas e as esquinas. O sistema implementado propõe uma solução completa para a consecução deste objetivo.

1.1 Formulação do Problema

A obtenção de um grafo correspondente a uma dada malha viária, apresentada na forma de uma imagem, não é tarefa trivial. Os primeiros procedimentos para a obtenção de ruas e suas ligações em mapas ou imagens aéreas eram manuais e demandavam muito tempo. Posteriormente, foram surgindo estratégias para a automatização total ou parcial dessa tarefa.

Os mapas confeccionados não seguem um padrão para uso de símbolos e cores e a maioria dos mapas ainda se encontra em papel. Faz-se necessária a digitalização desses mapas.

Ao digitalizarmos um mapa, obtemos dados de imagem no formato *raster*. Dados nesse formato resultam em um *array* de pixels, cuja funcionalidade se restringe puramente à finalidade gráfica de exibição da imagem. Nao há como se extrair informações sobre a imagem estando nesse formato. Uma conversão de raster para vetor é necessária para conseguirmos obter alguma informação em alto nível da imagem gerada.

Dados no formato vetorial representam os mapas digitalizados através

de elementos gráficos com atributos geométricos conhecidos, como pontos, segmentos de reta e áreas. Em nossa aplicação, os elementos obtidos são polígonos fechados representando os quarteirões do mapa.

1.2 Trabalho Proposto

Nosso trabalho se compõe de um conjunto de técnicas aplicadas sobre o mapa, passando pelas seguintes etapas:

1. Processamento da Imagem,
2. Vetorização e
3. Aplicação da Triangulação Restrita de Delaunay sobre o mapa vetorizado

Uma vez obtido o mapa no formato vetorial, partimos para a etapa final do procedimento de extração do grafo. O resultado da etapa de vetorização nos dá um conjunto de polígonos fechados, visto que quadras em mapas urbanos apresentam normalmente esta geometria. Desta forma, podemos particionar o espaço para nos auxiliar na detecção de regiões de interesse, como ruas, esquinas, quadras e ruas sem saída.

Os vértices dos polígonos que representam cada quadra detectada na imagem são inseridos em uma Triangulação Restrita de Delaunay. As características inerentes a essa estrutura de dados foram a motivação para a sua

escolha como ferramenta de apoio para a determinação de esquinas e suas ligações no mapa.

1.3 Estrutura da Tese

O trabalho está dividido como segue. O Capítulo 2 trata do levantamento bibliográfico, destacando as técnicas encontradas na literatura para a resolução deste problema. No Capítulo 3 detalhamos a etapa de preprocessamento e vetorização da imagem. No Capítulo 4 abordamos as características e o uso da Triangulação Restrita de Delaunay para a extração do grafo. O Capítulo 5 descreve as conclusões e os resultados obtidos, bem como indicações de aplicações para o sistema implementado. Detalhes sobre o sistema implementado podem ser vistos no Apêndice A.

Capítulo 2

Extração de Ruas e Estradas

Diversas abordagens para a extração de ruas e estradas em mapas são encontradas na literatura. Este capítulo descreve algumas que consideramos as principais.

A grande quantidade de trabalhos propostos sobre o tema reflete os seguintes fatos: (1) A necessidade de se ter uma representação de regiões urbanas ou rurais no formato vetorial, para facilitar a atualização de informações em Sistemas de Informações Geográficas e (2) o fato de que os trabalhos desenvolvidos e implementados até agora não terem sido capazes de produzir bons resultados para todas as circunstâncias [18]. Busca-se ainda um sistema que possa oferecer total acerto na detecção dos elementos de interesse nas imagens e mapas em quaisquer casos. Os primeiros trabalhos para a solução do problema descreviam em sua maioria sistemas para detecção de estradas em regiões rurais, onde a malha viária é menos densa e o resultado era um conjunto de estradas sem informação sobre suas interseções. Posteri-

ormente, surgiram abordagens para a extração de ruas e estradas onde as suas interseções eram identificadas e a partir daí toda a rede viária representada era levantada para uso em diferentes aplicações.

Aproximadamente, quase todos os métodos para extração de ruas ou estradas seguem um fluxo similar de operações: Pontos iniciais, ou sementes, são detectadas automaticamente ou fornecidos por um operador. O sistema começa por essas sementes e seguem o percurso da rua a que ela pertence o máximo possível, isto é, enquanto for possível prosseguir sem interrupções abruptas e sem perder a similaridade com a semente. Finalmente, os trechos de ruas detectados são combinados para formar uma rua completa e depois toda a rede.

Os trabalhos encontrados constituem abordagens que apresentam variações para esses passos básicos. Cada sistema implementado recebe dados de entrada diferentes, podendo ser mapas vetorizados, mapas escaneados, fotos aéreas e até mesmo modelos urbanos combinados com mapas ou fotos.

Nas seções a seguir resumimos as principais características dessas abordagens.

2.1 Extração de ruas e estradas em fotos aéreas

Grande parte dos trabalhos encontrados adotam como dados de entrada imagens aéreas, em vez de mapas. Em alguns deles, observamos que os mapas são utilizados como dados auxiliares para a etapa de interpretação e validação de resultados. As abordagens variam não só pela metodologia empregada, mas também pelas diferentes informações adicionais que utilizam. Um critério que distingue os métodos aplicados, em particular, nas imagens aéreas é a interação do sistema implementado com um operador, o que classifica a extração em automática ou semi-automática.

2.1.1 Extração semi-automática

As técnicas semi-automáticas surgiram para agilizar a atualização de bancos de dados para Sistemas de Informações Geográficas. Anteriormente, todo o processo era manual, e isso consumia muito tempo.

Nas técnicas semi-automáticas, o operador fornece ao sistema pontos de partida ou um ponto e uma direção para o algoritmo de detecção de estradas e ruas. Além de reduzir o tempo de processamento em relação a um processo inteiramente manual, melhora também nitidamente a acurácia. O operador monitora continuamente o processo e intervém sempre que o sistema encontra uma dúvida de interpretação ou discrepância com a qual não pode lidar.

Como, por exemplo, mudanças bruscas na direção dos *pixels* de uma aresta que esteja sendo seguida ou simplesmente uma quebra na seqüência de *pixels* dessa aresta, possivelmente provocada por sombras ou símbolos na imagem.

Pode ocorrer ainda que um sistema semi-automático necessite de uma série de parâmetros a serem passados em determinada etapa do processo, como se observa no trabalho de Katartzis *et al* [17].

Esse trabalho se caracteriza pelo uso de Morfologia Matemática e Teoria Randômica de Markov para a extração do grafo. A Teoria Randômica de Markov é uma maneira conveniente para modelar entidades dependentes de contexto, como os *pixels* de uma imagem e os objetos aos quais eles correspondem.

O processo é dividido em duas etapas. A primeira etapa recebe a imagem aérea e aplica a ela uma série de filtros morfológicos para facilitar a identificação dos *pixels* de estradas ou ruas, bem como sua orientação. Posteriormente, é obtido o esqueleto do subconjunto da imagem composto pelos *pixels* classificados como pertencentes a ruas aplicando um filtro *watershed*.* Em seguida, toma-se esse esqueleto e orientação encontrada para os seus *pixels* e se aplica um algoritmo de acompanhamento de linhas para a extração de segmentos de reta. A partir de um conjunto de segmentos detectados, de-

*O filtro *watershed* é uma transformação que trata uma imagem em tons de cinza como uma superfície topográfica. Ele faz uma analogia com uma área sendo inundada, onde as regiões mais baixas (regiões cujos *pixels* apresentam menor intensidade, no caso da imagem) desaparecem e, se inundarmos de forma que as águas não se encontrem, os pontos que ficam emersos correspondem, na analogia, aos *pixels* que apresentam maior intensidade no tom de cinza.

notados por S_{det}, é criado um conjunto de segmentos para todas as possíveis conexões - S_{con} - entre os segmentos detectados. Para auxiliar a construção do grafo, um modelo Markoviano é construído, estabelecendo-se um sistema de vizinhança ilustrado na Figura 2.1.

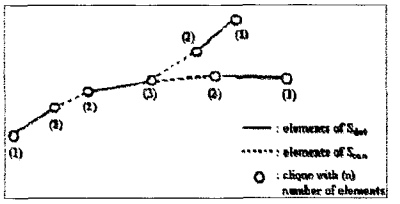


Figura 2.1: Sistema de vizinhança para os conjuntos S_{det} e S_{con}.

Essa abordagem apresenta alta performance, mesmo em imagens altamente texturizadas.

Em Heipke *et al* [6] é apresentado um procedimento para extração semi-automática de estradas a partir de imagens aéreas. Um operador indica um ponto de partida e uma direção inicial e é definida uma área de interesse delimitada por uma janela contendo o ponto de partida, como mostra a Figura 2.2. Dentro dessa área são efetuadas operações de filtragem do gradiente, binarização para identificar *pixels* de borda e redução ao esqueleto do conjunto desses *pixels*.

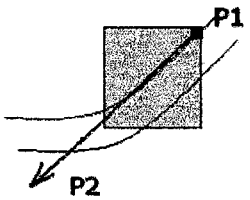


Figura 2.2: Área de interesse delimitada pelo quadrado. P1 é o ponto de partida e P2 é a direção, ambos escolhidos pelo operador.

A esse esqueleto, é finalmente aplicado um algoritmo de simplificação de linhas [11]. O resultado é a extração das bordas das estradas em formato vetorial. Não há informação sobre interseções entre as estradas detectadas.

A abordagem apresentada por Gruen & Li [10] combina a representação da imagem por *wavelets* e um algoritmo baseado em modelos digitais e usando Programação Dinâmica para a extração de estradas em imagens aerofotogramétricas. Em uma primeira etapa, foi criado um *wavelet* particular para identificar *pixels* de estrada, suprimindo detalhes menores na imagem. Uma transformação *wavelet* é então aplicada para destacar *pixels* prováveis de estradas. Em seguida, alguns pontos são fornecidos por um operador para iniciar a fase de detecção e ligação de estradas. Baseado em propriedades geométricas e fotométricas de estradas, as porções detectadas são tratadas como *snakes*[16] e a união dessas porções para a formação da malha viária se dá por meio de Programação Dinâmica. O processo pode ser esquematizado como mostra a Figura 2.3.

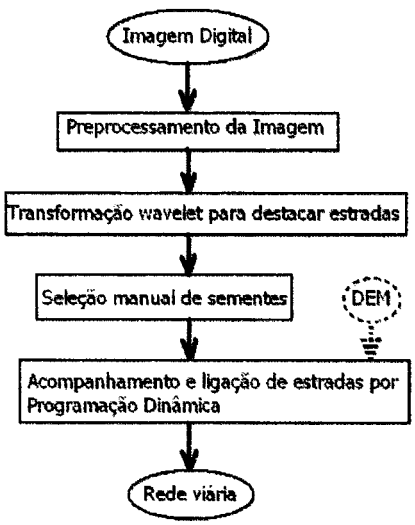


Figura 2.3: Processo de detecção de estradas proposto por Gruen & Li [10].

2.1.2 Extração automática

Sistemas de extração automática de ruas ou estradas utilizando imagens aéreas empregam um conhecimento prévio da região a ser analisada. Este conhecimento pode vir de Sistemas de Informações Geográficas, de mapas ou de Modelos Digitais. Usando este conhecimento temos de ante-mão os objetos prováveis de se encontrarem na imagem e as características como textura, cores, radiometria, que eles devem apresentar. Isto facilita também a identificação de elementos que podem ser obstáculos na detecção das ruas. Por exemplo, imagens aéreas de uma área urbana apresentariam como obstáculos para a detecção de ruas e avenidas as sombras de edifícios ou texturas de árvores, bem como automóveis sobre a pista. Em [4] e [13] encontramos uma abordagem que interpreta imagens aéreas construindo modelos apropriados

para três diferentes contextos: urbano, rural e floresta. Para cada um desses contextos temos um conjunto de características dos elementos prováveis de se encontrar, bem como de possíveis relações entre as porções detectadas de estradas e ruas e os demais objetos contidos na imagem.

Há também estratégias utilizando redes neurais, principalmente para o caso de imagens aéreas. Os modelos criados são construídos usando muitas imagens para treino da rede, resultando assim em um modelo mais completo. Há exemplos de trabalhos onde redes neurais são usadas para identificação de veículos em imagens aéreas. A rede viária pode ser então extraída a partir de veículos cuja posição é colinear. Uma desvantagem do uso de redes neurais é a escolha da arquitetura mais adequada a finalidade a que ela deve ser aplicada, não existindo uma forma padrão para o contexto que estamos considerando [26].

Os métodos mais comuns de identificação de estradas se baseiam na extração de linhas em imagens em escala pequena e na detecção de bordas de ruas ou estradas nas imagens em escala grande. Em Baumgartner *et al*, 97 [4], a extração de estradas se dá pelo uso combinado de imagens em escala grande e pequena para a validação da rede viária obtida. Em Baumgartner *et al*, 99 [13] há uma separação entre as estratégias usadas para a extração de ruas ou estradas, dependendo do contexto aplicado. No contexto rural, a abordagem multi-escala é aplicada, mas no contexto urbano é usado um modelo apropriado a ele.

Outra abordagem para a extração automática de estradas em imagens aéreas foi proposta por Ruskoné *et al* [25]. O processo de interpretação se inicia extraindo de maneira automática as porções de *pixels* correspondentes a estradas que apareçam de forma mais saliente na imagem, segmentando-se a imagem obtida por um filtro gradiente por um método baseado em *watershed*. Essas porções são as sementes que servem como pontos de partida para as etapas seguintes, que são a construção de hipóteses de conexão e validação dessas hipóteses. As hipóteses de conexão são geradas e checadas tendo por base critérios geométricos como distância e direção. A Figura 2.4 mostra exemplos dessas hipóteses de conexão entre porções detectadas.

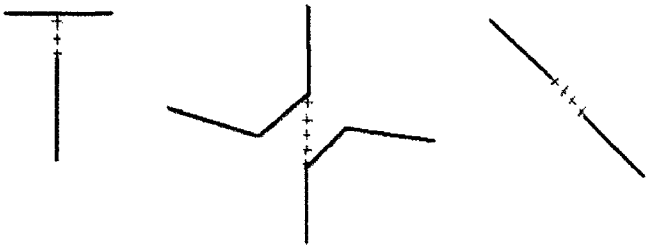


Figura 2.4: Hipóteses de conexão.

O estágio final é o ajuste geométrico da rede obtida. Esse ajuste é feito aplicando-se modelos de contorno ativo sobre as porções detectadas e validadas. Como os valores dos *pixels* no interior das estradas se mantêm praticamente constantes, as porções são ajustadas minimizando a variância dos valores dos *pixels* ao longo de um segmento detectado.

No trabalho de K. Price [23], ruas são extraídas de imagens de alta res-

olução de áreas urbanas. É assumido que as ruas apresentam arestas visíveis sem oclusões significativas. A rede viária é modelada como uma grade regular onde as estradas apresentam largura aproximadamente constante.

O operador fornece ao sistema três pontos correspondendo a três interseções que ele consegue identificar. Com isto a localização, o espaçamento e a direção da grade é inicializada usando esses três pontos. É então definido um modelo inicial contendo quatro segmentos de rua e a interseção definida por esses segmentos. Na Figura 2.5 vemos um exemplo deste modelo inicial.

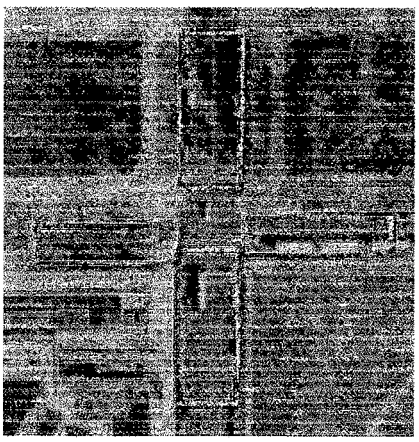


Figura 2.5: Grade inicial.

A partir daí é criada uma grade que será verificada. A Figura 2.6 mostra a estrutura da grade.

A expansão, verificação e refinamento da grade são baseados em informações vindas de DEM[†], de arestas extraídas na imagem e de contextos.

[†]A sigla DEM vem de Modelo de Elevação Digital (*Digital Elevation Model*). Este modelo consiste de um *array* de elevações para posições em um terreno. Estas posições estão normalmente distribuídas em intervalos regularmente espaçados.

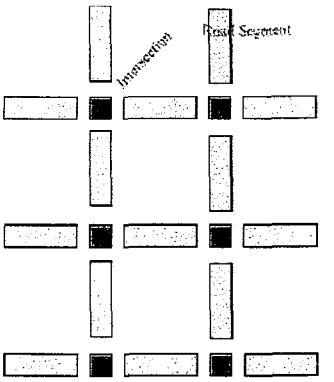


Figura 2.6: Exemplo de grade a ser verificada.

Em Moissinac *et al*, 95 [21] é feita uma interpretação da cena urbana retratada em imagens aéreas. Mapas geográficos são utilizados para auxiliar a análise da imagem. O produto final do trabalho apresentado é uma complexa rede representando todo o cenário urbano exibido na imagem. Esta estrutura complexa é construída a partir da combinação de três redes intermediárias, extraídas em diferentes fases do processo. Essas redes correspondem: à malha viária, sendo um grafo denotado por G_{road}; ao conjunto de quadras e suas relações de vizinhança, representado pelo grafo G_{block} e o terceiro grafo corresponde aos prédios dentro de cada quadra e é denotado por G_{Wblock}. Como nosso estudo está concentrado na extração de uma rede viária, vamos destacar somente a extração do grafo G_{road}.

É obtido um grafo inicial a partir da redução ao esqueleto dos *pixels* classificados como representando ruas no mapa. Este primeiro grafo serve como base para a extração de um segundo grafo referente à malha viária, extraído agora da imagem aérea. Em Moissinac *et al*, 94 [20], cinco algoritmos diferen-

tes foram aplicados para a detecção inicial de ruas na imagem, produzindo cinco resultados diferentes quando aplicados a uma mesma entrada. Esses dois grafos são combinados em um único para se obter o grafo correspondente à malha viária.

Podemos observar que as abordagens que extraem ruas e estradas de imagens aéreas utilizam dados vindos de diferentes fontes, como fotos em diferentes escalas, modelos de contexto, mapas e DEM (*Modelos de Elevação Digital*). Em áreas urbanas a informação fornecida por esses modelos ajuda a remover falsas hipóteses de ruas. Por exemplo, tetos de prédios que apresentam radiometria semelhante à das ruas podem ser em uma primeira análise interpretados como sendo *pixels* de ruas. Usando modelos digitais de elevação, podemos validar o conjunto de *pixels* detectados como sendo de ruas observando os valores de elevação na imagem onde esses *pixels* aparecem.

2.2 Extração de ruas e avenidas a partir de mapas.

Sistemas que utilizam imagens aéreas usualmente se restringem ao uso de imagens em níveis de cinza e os objetos que podem prejudicar a identificação da malha viária podem em grande parte ser modelados *a priori*. Ao contrário, no caso dos mapas urbanos, a imagem fornecida como entrada normalmente é colorida e os objetos que podem constituir obstáculos para a detecção de ruas

podem não apresentar características previsíveis. Os elementos que compõem a versão digitalizada de um mapa dependem das condições de sua confecção e até mesmo de sua conservação e nossas maiores dificuldades provêm do fato de em um mapa serem inseridas legendas e símbolos em posições não padronizadas, e das cores utilizadas para destacar cada elemento variam de mapa para mapa. Assim, um sistema que manipula este tipo de dados de entrada conta com pouco conhecimento prévio para ser utilizado na interpretação dessa entrada e na extração de estruturas de interesse.

A detecção de ruas em mapas apresentam algumas dificuldades. Elas podem se originar, antes de mais nada, do próprio material do mapa de origem e das condições em que foi digitalizado. Mapas em papel podem apresentar sérios problemas com relação à qualidade do papel. E é claro que a qualidade do *scanner* utilizado, a resolução escolhida, ou ainda se a textura do papel interferir no processo de digitalização, também criam dificuldades.

Um segundo problema é a representação de ruas no mapa. Por exemplo, quando as ruas possuem uma única cor específica são facilmente detectadas. Em alguns casos, no entanto, elas podem vir na mesma cor que as quadras, sendo esses dois elementos separados por uma linha marcando o contorno das quadras. Os nomes das ruas impressos ao longo do seu traçado, obviamente, dificultam o processo de se seguir esse traçado. Um outro problema é a presença de nomes de bairros e outros símbolos. Eles não possuem uma posição previsível no mapa, nem um tamanho padronizado. Alguns nomes

de bairros ocludem parte da representação das ruas, dificultando a detecção das mesmas e a junção de componentes conexas já conhecidas.

Em Borghys *et al* [5] é proposto um algoritmo onde inicialmente é construído um modelo de ruas. Este modelo contém um pequeno conjunto de informações, tais como a largura média e as possíveis cores das ruas. Estas informações podem ser obtidas a partir da própria legenda do mapa. Construído o modelo, dois métodos complementares fornecem resultados parciais que são combinados para a detecção de estradas e ruas. O primeiro método toma por base a homogeneidade das cores das estradas ou ruas e o contraste entre as estradas e a região vizinha a elas e se inicia pela busca dos possíveis segmentos de ruas, seguindo na direção em que a imagem apresente menor variância nas cores dos *pixels*. A Figura 2.7 mostra o primeiro passo deste procedimento.

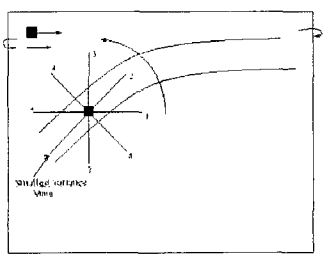


Figura 2.7: Procedimento para busca de um possível segmento de rua.

O segundo método utiliza conhecimento acerca das possíveis cores das estradas e determina a presença de bordas para detectá-las. Os resultados desses métodos são combinados para a obtenção do resultado global. As

linhas de *grid*[‡] e as letras são detectadas à parte para serem levadas em conta na etapa final de fusão dos dois métodos.

Em testes efetuados este algoritmo detectou a maioria das estradas corretamente, no entanto, houve também a detecção de falsas vias.

Uma outra abordagem é apresentada por Cinthia, 2000 [7]. Nesse trabalho, a partir de um mapa urbano se obtém um grafo contendo vértices e suas ligações entre eles. Essas ligações no grafo correspondem às ruas detectadas, enquanto que os vértices correspondem às esquinas. O grafo obtido é não-direcional. Inicialmente, é feita a extração de retas da imagem por meio da aplicação da Transformada de Hough. Em seguida, são obtidos segmentos dessas retas, que são armazenadas em uma estrutura de dados espacial para detectar proximidade e interseções desses segmentos de reta, a fim de se obter os vértices e a ligação entre eles no grafo. O sucesso do algoritmo depende, no entanto, do tipo de imagem que está sendo submetida. Se a imagem fornecida apresentar uma malha viária muito irregular, o índice de erros na detecção de ligações no grafo é aumentado.

Por fim, gostaríamos de destacar o trabalho implementado por Décoret & Sillion [28]. Um grafo representando as ruas de uma região urbana é extraído a partir de um conjunto de polígonos no espaço 2.5D. Os polígonos em questão representam o contorno de prédios e suas respectivas alturas. A estratégia aplicada é a utilização do Diagrama de Voronoi, obtido a partir

[‡]Neste caso o *grid* não representa uma rede viária, e sim, as linhas de *grid* normalmente exibidas em mapas, com o intuito de informar latitude e longitude.

de uma Triangulação de Delaunay não restrita aplicada a pontos que definem os contornos dos polígonos. Esta abordagem efetua previamente um refinamento dos polígonos para que a Triangulação de Delaunay preserve os contornos desses polígonos, de forma que o Voronoi obtido tenha, dentre as suas arestas, aquelas que são de interesse para serem combinadas de forma a representar as ruas e suas ligações, mais facilmente identificadas. Na seção 4.2 descrevemos maiores detalhes sobre este método.

Em nosso trabalho, detectamos as ruas em imagens de mapas. O mapa é processado e vetorizado resultando em um conjunto de polígonos representando as quadras do mapa. As etapas de pré-processamento e vetorização estão descritas no Capítulo 3. Tendo esses polígonos, particionamos o espaço usando uma Triangulação Restrita de Delaunay. A Triangulação Restrita elimina a necessidade de refinamento dos polígonos feito em [28], o que significa também que não temos necessidade de se determinar um ε suficientemente pequeno para preservar o contorno dos polígonos. Associando as faces obtidas por esse particionamento, conseguimos detectar com bons resultados as ruas do mapa e as ligações entre elas, extraindo um grafo correspondente à malha viária do mapa em questão.

Capítulo 3

Processamento e Vetorização da Imagem

Neste capítulo discutimos a importância das técnicas de Processamento de Imagens para sistemas de interpretação e detecção de objetos em imagens. Posteriormente, apresentamos como essas técnicas são aplicadas em nosso sistema. Ao final do capítulo, abordamos a etapa de Vetorização da imagem processada.

3.1 Processamento da Imagem

Vimos no Capítulo 1 que uma imagem digitalizada se compõe de um *array* de *pixels*, cuja funcionalidade é restrita à finalidade gráfica de exibição da imagem. Sistemas para reconhecimento de objetos em uma imagem digitalizada tem como primeira necessidade a interpretação desse *array*, aplicando sobre ele algumas técnicas que auxiliem a extração do conjunto de *pixels* que

correspondam às estruturas a serem reconhecidas.

Processamento de Imagens é o conjunto de técnicas e procedimentos aplicados a uma imagem digital com a finalidade de modificar atributos de seus *pixels*, alterando caracterísiticas da imagem. Essas alterações são normalmente obtidas pela aplicação de filtros sobre a imagem para a consecução dos objetivos aos quais o Processamento de Imagens se destina: melhoria da qualidade da imagem, separação de conjuntos de *pixels* que contenham características comuns, eliminação de ruído, alteração das dimensões de estruturas na imagem e realce de estruturas de interesse, baseado em um conhecimento prévio sobre as características do objeto que se deseja realçar, como textura e radiometria. Por exemplo, para a extração de ruas e estradas, espera-se que os *pixels* de rua apresentem uma textura homogênea e intensidade de tons de cinza de acordo com o material que as compõem, como asfalto ou terra. Desta forma, podemos compreender que um sistema que se proponha a reconhecer objetos em uma imagem digital tenha como primeira etapa o processamento da imagem a ser analisada.

Como exemplo, voltamos ao Capítulo 2, onde vimos que a maioria dos sistemas para detecção de ruas e estradas seguem uma seqüência comum de passos. Uma opção comum é iniciar pela obtenção automática ou semi-automática de sementes que servem como ponto de partida para a extração das estruturas de interesse a que ela pertence, no caso as ruas e estradas. O processamento da imagem fornecida é passo necessário para a obtenção

automática dessas sementes. Em alguns casos, a etapa de pré-processamento da imagem resulta na maior parte do processo de detecção de ruas e estradas, como vimos em Katartzis *et al* [17].

Nosso sistema trabalha com imagens de mapas urbanos. Esses mapas possuem tipicamente uma coleção de símbolos, com o intuito de orientar o usuário sobre informações adicionais, como nomes de logradouros, sentido das principais ruas, destaques para as principais avenidas com o uso de cores diferenciadas, legendas indicando nomes de bairros, parques, praças, entre outros, como se pode ver na Figura 3.1.

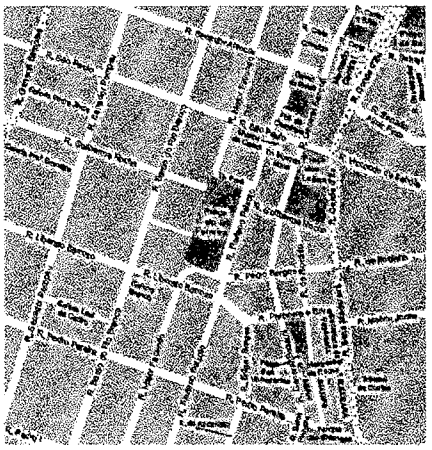


Figura 3.1: Imagem Digital de um Mapa Urbano.

Estas legendas, adicionadas às características incorporadas à imagem quando da sua digitalização, constituem obstáculos para a interpretação desta imagem no sentido de se determinar seu grafo correspondente. A imagem precisa passar por uma série de tratamentos para a a simplificação da mesma para um formato vetorial.

A maioria dos sistemas implementados efetuam basicamente quatro operações: pré-processamento, detecção de arestas, acompanhamento de *pixels* de ruas e agrupamento de sementes. Em nosso trabalho aplicamos técnicas de Processamento de Imagens para atingirmos os seguintes objetivos: Redução da quantidade de cores, eliminação de legendas com o uso de Morfologia Matemática, Binarização e Extração de Contornos dos quarteirões.

3.1.1 Reduzindo a quantidade de Cores

A imagem digital de um mapa urbano pode ser obtida a partir do uso de um *scanner*, caso ele tenha sido disponibilizado em papel. Pode-se também obter imagens digitais prontas de mapas urbanos disponíveis em CD-ROMs ou em páginas *Web*. Quando a imagem é obtida através da digitalização da versão em papel de um mapa, a imagem resultante pode apresentar problemas com relação ao *dithering**.

Em nossas atividades de aquisição de imagens para testes, pudemos observar também que se fizermos a digitalização em resolução muito alta, a imagem pode até mesmo apresentar marcas da fibra do papel onde o mapa está sendo representado, dependendo da qualidade do papel. A Figura 3.2 mostra um detalhe de um mapa em papel que foi digitalizado. Podemos

*O *Dithering* é uma representação de uma cor ou nível de cinza que um programa de computador não possa reproduzir no dispositivo gráfico. Com isso, uma cor que não esteja disponível para o dispositivo é substituída pela mistura de pixels de diferentes cores que, combinadas, aproximam a cor desejada.

observar que as regiões com um mesmo padrão de cor são altamente texturizadas, dada a mistura de cores produzidas pelo processo de *dithering*.

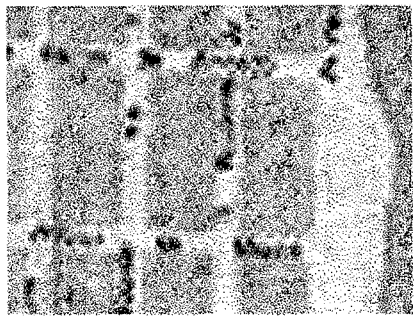


Figura 3.2: Detalhe de uma imagem em papel digitalizada com o uso de um *scanner*.

O primeiro passo em nosso procedimento é portanto a redução da quantidade de cores na imagem, para eliminar os efeitos do *dithering* e texturas. O usuário seleciona dentro da imagem um conjunto de cores que deseja manter. O sistema varre a imagem e vai fazendo as substituições de cores, utilizando critérios de proximidade entre a cor do *pixel* lido e uma das cores escolhidas pelo usuário. A Figura 3.3 mostra um exemplo de resultado desta substituição.

Assim, temos a mesma imagem com um número reduzido de cores e podemos segmentar esta imagem em regiões com o mesmo padrão de cor. Nosso sistema lida com imagens coloridas no sistema padrão RGB. As imagens lidas pelo sistema são imagens digitais no formato PPM. Maiores informações sobre o formato de arquivos PPM podem ser encontrados no Apêndice B.

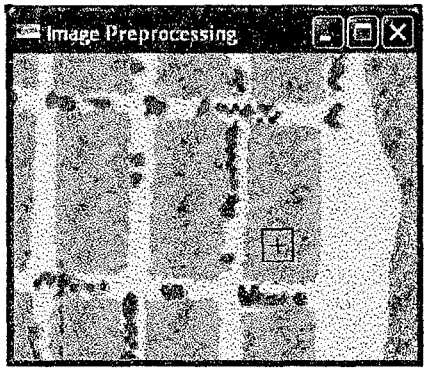


Figura 3.3: Imagem da Figura 3.2 com redução de cores de acordo com cores selecionadas pelo usuário.

3.1.2 Eliminação de Legendas

Legendas indicando nomes de ruas e bairros constituem o principal obstáculo para a interpretação da imagem para o reconhecimento de ruas e quarteirões.

Analisando os operadores morfológicos, vimos que a operação de dilatação expande uniformemente o tamanho de objetos [9]. Isto é feito com o uso de uma matriz denominada *elemento estruturante* (Figura 3.4(a)), que é convoluída com a imagem (Figura 3.4(b)), para adicionar *pixels* ao objeto escolhido. A Figura 3.4(c) mostra um objeto dilatado.

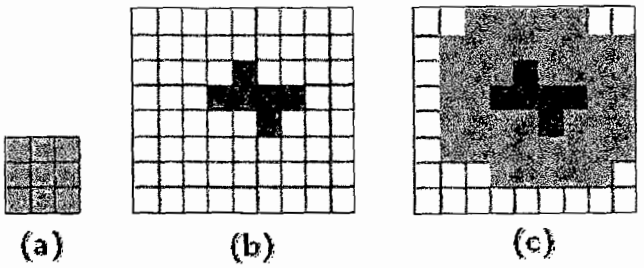


Figura 3.4: Operação de dilatação.

Observamos que nomes de ruas, por exemplo, estão localizados na imagem comumente sobrepostos às ruas como mostra a Figura 3.5. Uma opção para eliminar as legendas é a expansão dos *pixels* de rua de modo que os *pixels* de legenda sejam encobertos pelos *pixels* de rua.

Figura 3.5: Legendas posicionadas sobre *pixels* de rua em imagens de mapas urbanos.

Pudemos observar por meio de experimentos que o operador de dilatação constitui uma boa ferramenta para a expansão dos *pixels* de rua. Isto porque como a dilatação se dá de maneira uniforme sobre um objeto, a direção das ruas não se altera, como vemos na Figura 3.6.

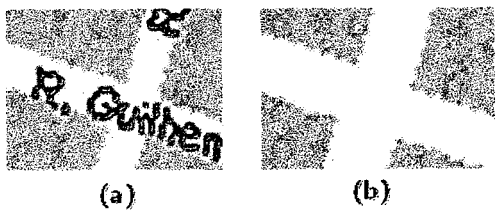


Figura 3.6: (a) *Pixels* de legenda sobrepostos aos *pixels* de rua. (b) Operador de dilatação aplicado sobre os *pixels* de rua.

O estudo de operadores morfológicos mostra aplicações dos mesmos sobre imagens binárias. Assim, os objetos a serem dilatados são identificados por serem constituídos de *pixels* pretos.

Nosso sistema trabalha com imagens coloridas e por isso implementamos um operador de dilatação que pode ser aplicado sobre qualquer um dos padrões de cores selecionados pelo usuário. Desta forma, podemos dilatar determinada região no mapa de forma a eliminar legendas sobrepostas a objetos de diferentes cores em uma mesma imagem. A Figura 3.7 mostra o resultado do procedimento de dilatação sobre pixels de rua aplicados sobre a imagem da Figura 3.3.

Sabemos, no entanto, que os mapas não seguem uma forma padrão para a sua confecção. Alguns apresentam legendas de maior tamanho e a aplicação de dilatação não resolveria. Para esses casos, implementamos uma rotina que possibilita ao usuário trocar a cor de uma determinada região pela cor de uma outra região. Assim, se o mapa apresentar uma legenda muito grande no meio de uma quadra, por exemplo, podemos trocar os valores RGB daquela

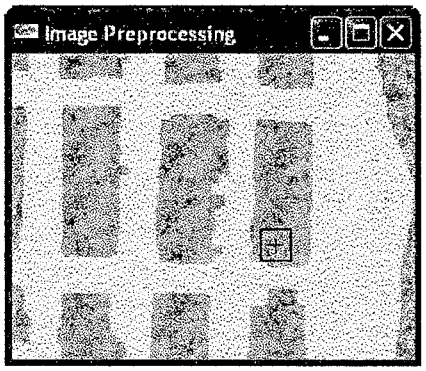


Figura 3.7: Imagem da Figura 3.3 após operação de dilatação sobre pixels de rua.

legenda por valores RGB da própria quadra.

Observe que na Figura 3.8(a) a aplicação direta de dilatação sobre os *pixels* de quarteirão seria insuficiente. Para fazer a legenda desaparecer, seria necessário dilatar o quarteirão repetidas vezes. O problema é que o quarteirão poderia aumentar demais, destruindo também os pixels de rua ao seu redor. Na Figura 3.8(b) vemos o resultado da troca de cores dos *pixels* da legenda pelos *pixels* de quarteirão. Em (c), percebemos que a dilatação dos *pixels* de rua, na cor branca, concluíram a tarefa de eliminação da legenda.

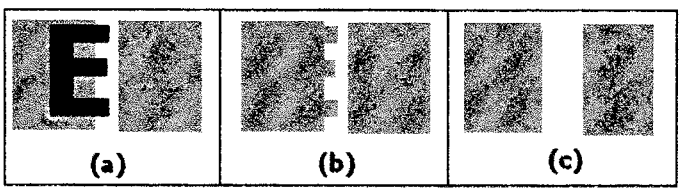


Figura 3.8: (a) Legenda encobrindo grande quantidade de *pixels* de quadra. (b) Resultado da troca de padrões entre os pixels de legenda e os pixels de quadra. (c) Após a troca de padrões, uma dilatação dos *pixels* de rua (cor branca) concluem a eliminação da legenda.

O Capítulo A descreve como o usuário pode efetuar as operações de dilatação e troca de padrões em nosso sistema.

3.1.3 Binarização

Em um mapa urbano, para a extração de um grafo que represente a malha viária da cidade, precisamos separar esta malha viária do restante dos elementos do mapa. Na Figura 3.9 podemos distinguir a malha viária de quadras e praças, estas últimas representadas na cor verde.

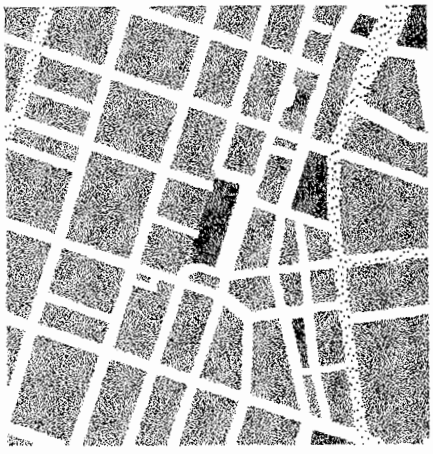


Figura 3.9: Imagem de mapa urbano após a eliminação de legendas.

A binarização consiste na redução da quantidade de cores de uma imagem para duas cores apenas: branco e preto. Este procedimento é comumente aplicado para imagens em tons de cinza e a redução é feita com base em um *threshold*, que determina que os *pixels* cujo tom de cinza esteja abaixo desse limite deverão mudar sua cor para preto e os *pixels* cujo tom de cinza estiver acima deverão mudar sua cor para branco.

Nosso sistema, entretanto, trabalha com imagens coloridas. O *threshold* pode então ser definido para um dos três canais RGB, ou ainda para a soma dos mesmos. Desta forma, independente das cores apresentadas no mapa e selecionadas pelo usuário, sempre haverá uma maneira de se determinar este limite.

O usuário pode defini-lo com base em suas observações sobre os valores RGB encontrados na imagem. Como exemplo, podemos analisar os valores RGB da imagem da Figura 3.10. Os valores estão listados na Tabela 3.11.

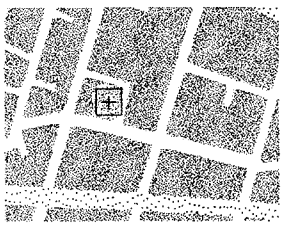


Figura 3.10: Imagem a ser binarizada.

Rótulo	R	G	B	Brilho
quadra	245	197	146	588
rua	255	255	255	765
avenida	255	245	0	500

Figura 3.11: Valores RGB para as cores da imagem mostrada na Figura 3.10.

Portanto, pela análise dos valores RGB na tabela, uma boa opção de *threshold* para separar os *pixels* da malha viária, composta de ruas e avenidas, dos *pixels* de quadra, seria escolher um valor para o canal G igual a 200. A Figura 3.12 mostra o resultado. Observe que o critério escolhido consegue separar as ruas e avenidas das quadras da Figura 3.10.

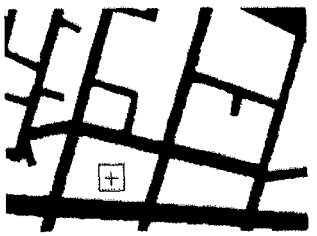


Figura 3.12: Resultado da operação de binarização.

Algumas imagens, entretanto, apresentam cores que dificultam a separação da malha viária, como a imagem da Figura 3.13 (a). Os *pixels* que representam a praia estão com valores RGB entre os valores dos *pixels* de rua e dos *pixels* de avenida. Isso dificulta a determinação de um *threshold* que faça uma separação satisfatória. A Figura 3.13(b) mostra um resultado indesejável se determinássemos um *threshold* limitado às condições apresentadas.

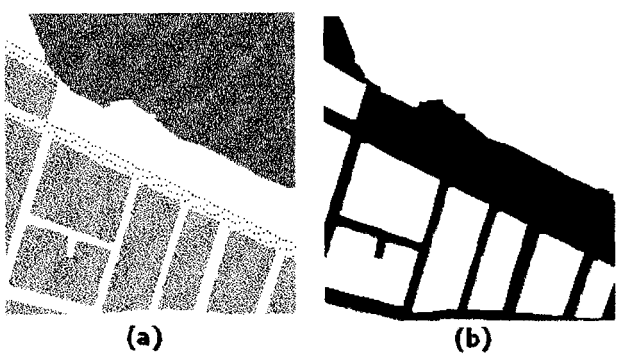


Figura 3.13: (a) Imagem com cores difíceis para se determinar um *threshold* apropriado. (b) Resultado indesejável se escolhêssemos G = 200.

Para evitar esse problema, podemos utilizar o recurso de troca de padrões. A Figura 3.14(a) mostra o resultado da troca efetuada sobre a imagem

mostrada na Figura 3.13(a) e na Figura 3.14(b) vemos que o resultado da binarização corresponde a um resultado desejado.

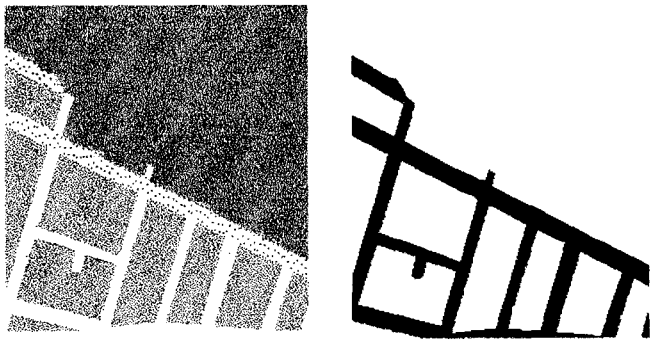


Figura 3.14: (a) Resultado da troca de padrões na Figura 3.13(a) . Em (b) vemos o resultado da binarização.

3.1.4 Extração de Contornos

Com a binarização da imagem, podemos separar no mapa as ruas dos quarteirões. Na maioria dos sistemas para reconhecimento de ruas e estradas em imagens vemos a aplicação de filtros para a detecção de bordas ou para traçados de contornos dos quarteirões. Esta etapa é de grande importância para a detecção da posição das ruas na imagem. No caso de imagens em baixa resolução, aplica-se algoritmos de detecção de linhas; em imagens de alta resolução, aplica-se algoritmos para a detecção de pares de arestas suficientemente paralelas e a uma distância mínima uma da outra, com uma textura homogênea entre elas.

Nosso sistema efetua o traçado do contorno dos quarteirões. O resultado

desta etapa é exibido na Figura 3.15 e mostra que cada quarteirão passa a ser representado na imagem como uma componente conexa de pixels pretos. Esta propriedade é útil para a etapa de vetorização, que vai utilizar essas componentes conexas para a geração de polígonos representando cada quarteirão.

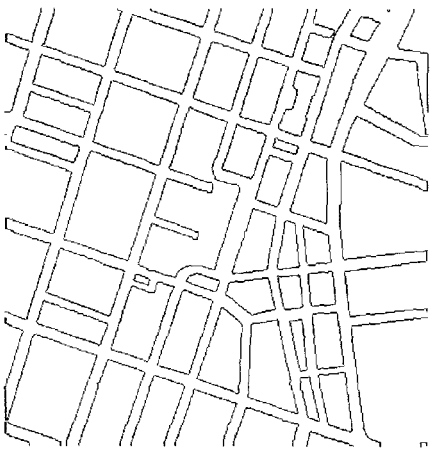


Figura 3.15: Resultado do traçado de contornos dos quarteirões.

3.2 Vetorização da Imagem

A extração dos contornos dos quarteirões na imagem nos dá como resultado um conjunto de componentes conexas de pixels pretos, cada componente com um pixel de largura.

Neste estágio, a imagem encontra-se nas condições ideais para sua conversão para o formato vetorial. Cada componente conexa de pixels pretos é convertida para um polígono fechado, representando cada quarteirão identi-

ficado na imagem.

Para esta conversão, podemos aplicar um algoritmo baseado no algoritmo de simplificação de linhas proposto por Douglas & Peucker [11].

Este algoritmo é um procedimento iterativo que se inicia tomando os pontos extremos de uma cadeia de *pixels* e traça uma linha reta. A cadeia de *pixels* é percorrida seguindo uma ordem determinada e as distâncias entre essa linha e os *pixels* dessa cadeia são calculadas. Se a maior distância encontrada estiver acima de um limiar ε pré estabelecido, o *pixel* correspondente é introduzido como um novo vértice da linha poligonal. A linha poligonal passa a ser constituída por dois segmentos de reta. A cadeia de pixels passa a ser considerada como duas subcadeias representadas por esses dois segmentos de reta. O processo se repete pesquisando os *pixels* dentro das subcadeias formadas sempre na mesma ordem. A condição de parada é quando todos os pixels de todas as subcadeias estiverem a uma distância da linha que une os seus extremos dentro do limiar ε pré-estabelecido. A Figura 3.16 ilustra o procedimento.

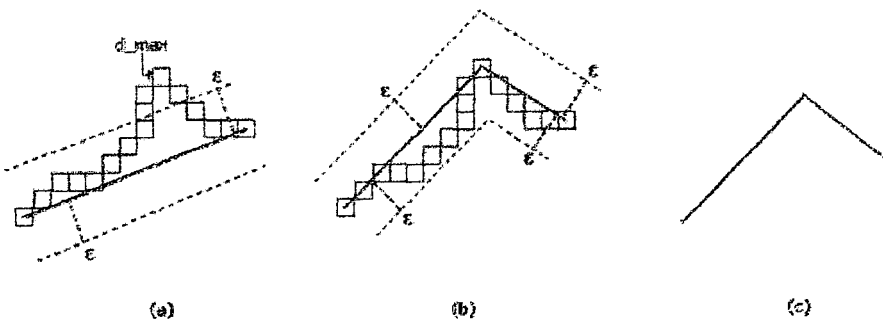


Figura 3.16: Procedimento Iterativo de Douglas-Peucker. A Figura (a) mostra o pixel que mais se distancia da reta e ultrapassa o limiar ε. Na figura (b), vemos o resultado da primeira simplificação. Em (c), o polígono resultante da conversão *raster-vector*.

3.2.1 Considerações sobre a vetorização implementada

Em nosso trabalho, as cadeias de pixels a serem convertidas formam curvas fechadas. Observamos que a aplicação do algoritmo original para a obtenção do polígono geraram resultados insatisfatórios, pois os polígonos resultantes possuíam mais vértices do que o necessário.

A Figura 3.17 mostra os resultados obtidos com a aplicação do algoritmo sobre o contorno de um quarteirão:

Este problema se deve ao fato do algoritmo sempre efetuar a divisão da linha poligonal - e portanto a inserção de novos vértices - a partir do pixel que estiver a uma distância máxima e fora do limiar. A Figura 3.18 detalha o resultado obtido:

Figura 3.17: (a)Cadeia de pixels. (b)Resultado obtido na aplicação do procedimento de Douglas-Peucker.

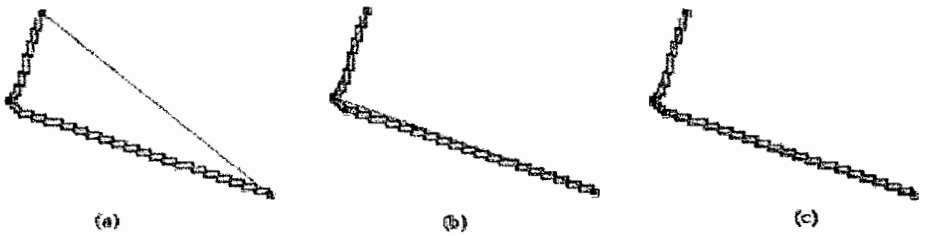


Figura 3.18: Detalhe do funcionamento do procedimento de Douglas-Peucker.

Para contornar o problema, tomamos por base o trabalho realizado por Janssen *et al* [15] e acrescentamos ao algoritmo de Douglas-Peucker restrições para o tamanho das linhas que formam o polígono e para o ângulo que elas formam entre si.

As restrições com relação ao tamanho determinam que as linhas cujo tamanho seja muito pequeno em relação às demais sejam eliminadas; as restrições de ângulo fazem com que linhas adjacentes com baixa variação angular sejam substituídas por uma única linha.

Deste modo, os vértices que provocam o surgimento de linhas que não satisfazem às restrições acrescentadas ao algoritmo são eliminados. O resultado é um polígono simplificado que não compromete a representação do quarteirão vetorizado. A Figura 3.19 mostra o resultado obtido com o nosso algoritmo, aplicado sobre a cadeia de *pixels* apresentada na Figura 3.17(a).

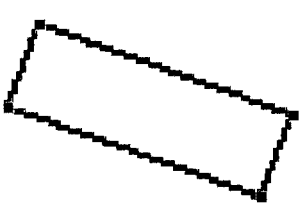


Figura 3.19: Resultado obtido acrescentando-se restrições de tamanho e de ângulo para as linhas de um polígono.

Para a imagem inteira, temos o resultado como se vê na Figura 3.20.

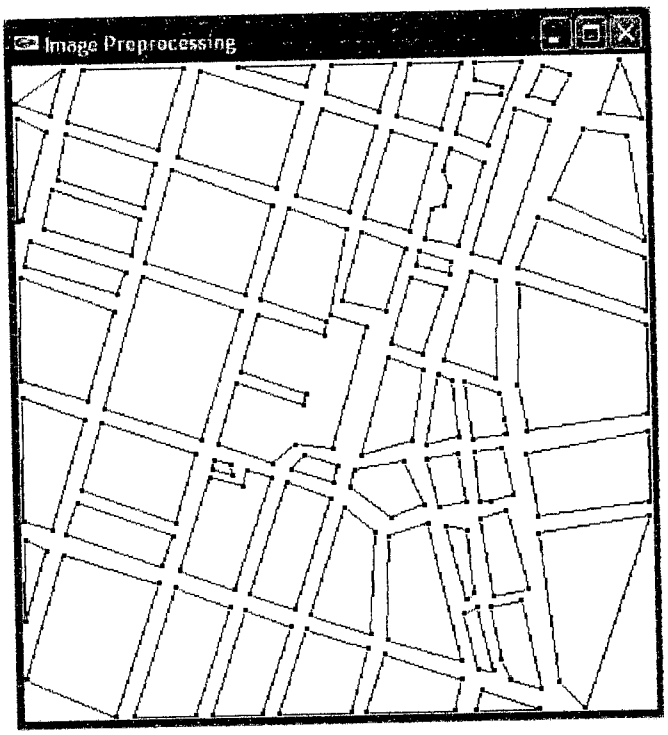


Figura 3.20: Resultado da vetorização aplicado sobre a imagem inteira.

Capítulo 4

Uso de triangulação para a extração do grafo

O resultado da vetorização aplicada sobre a imagem de um mapa urbano em nosso sistema é mostrado na Figura 3.20. Cada polígono fechado representa um quarteirão, e, portanto, o grafo que representa a malha viária deverá estar posicionado no espaço que compreende a região externa a esses polígonos, como mostra a Figura 4.1.

Intuitivamente, uma opção para a obtenção do grafo seria a extração do esqueleto desta região. O esqueleto de uma forma poligonal é composto de linhas retas e parábolas [24]. Entretanto, desejamos em nosso grafo somente linhas retas e portanto, os métodos tradicionais não podem ser aplicados. Em [2] é proposto um novo tipo de esqueletos para polígonos que não é o eixo central, mas é formado somente por linhas retas. Infelizmente esta abordagem também não pôde ser aplicada em nosso trabalho, pois ela ainda mantém alguns dos problemas encontrados nos métodos tradicionais:

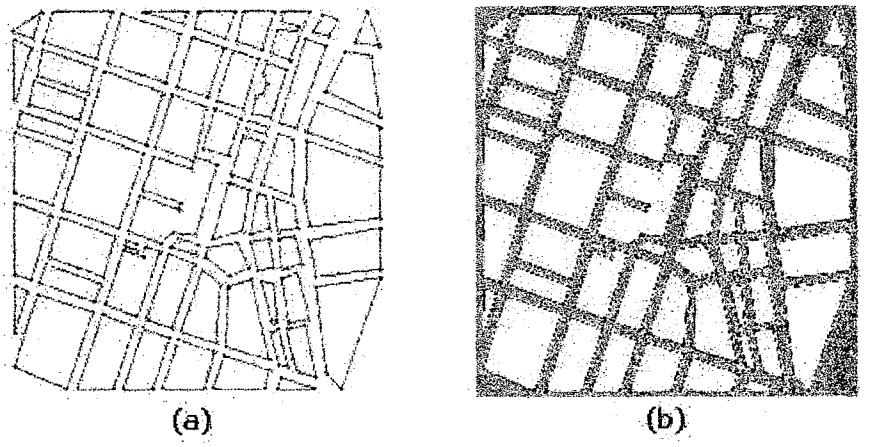


Figura 4.1: (a) Conjunto de polígonos resultante da vetorização. (b) Destacada em cinza, a região onde as arestas e vértices do grafo deverão estar posicionados.

- **Sensibilidade à entrada:** se as bordas do polígono são ligeiramente modificadas, a forma do esqueleto é completamente modificada e se torna desnecessariamente complexa para a nossa aplicação, como mostra a Figura 4.2. Isto se deve ao fato de que o esqueleto tenha uma exata definição em termos de distância à borda, armazenando mais informação do que é necessário para nosso caso;

- **Arestas extras:** o esqueleto apresenta pequenas arestas sempre que a borda apresentar bojos ou perturbações, como mostra a Figura 4.3. Os métodos tradicionais utilizam pós-processamento e filtros para podá-los.

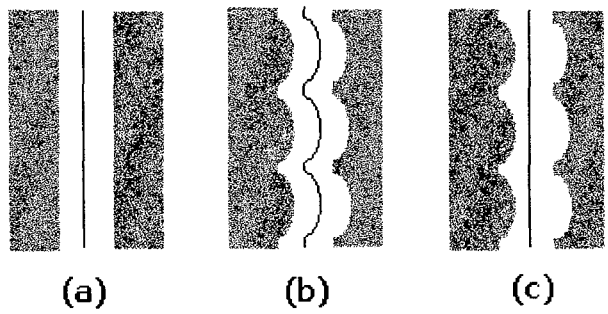


Figura 4.2: (a) Esqueleto extraído entre duas quadras ideais. (b) Quadras com bordas apresentando mais detalhes resultam em um esqueleto mais complexo. (c) Resultado desejado para nossa aplicação.

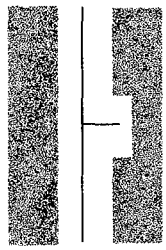


Figura 4.3: Entradas no contorno do quarteirão leva a arestas extras, ao se aplicar métodos de extração de esqueleto.

Uma alternativa ao uso de métodos para extração de esqueletos deve ser baseado em uma divisão do espaço de maneira que seja possível, dentro dessa divisão, identificar as diferentes regiões que compõem o mapa. Uma maneira de se fazer esta divisão do espaço é inserir os vértices dos polígonos em uma Triangulação.

Neste capítulo, discutimos como uma triangulação pode auxiliar na obtenção de um grafo. A seção 4.1 aborda a Triangulação de Delaunay e como esta estrutura de dados pode auxiliar na obtenção de uma boa divisão espacial. A seção 4.2 traz um estudo de caso de extração de um grafo utilizando o Diagrama de Voronoi, obtido a partir do dual de uma Triangulação de Delaunay. Por fim, a seção 4.3 discute a nossa abordagem, baseada em um caso especial da Triangulação de Delaunay: a Triangulação Restrita de Delaunay. Nesta última seção, mostramos as características desta triangulação e as razões que nos levaram à escolha desta estrutura de dados para a etapa final de nosso trabalho. Além disso, mostramos como essa estrutura foi utilizada em nossa aplicação.

4.1 Triangulação de Delaunay

Para se ter uma boa divisão de espaço, não podemos organizar os pontos em uma triangulação qualquer. Precisamos que os triângulos formados por essa divisão tenham uma área interna aberta o suficiente, que mais se aproxime à área de triângulos equiláteros. Essas triangulações especiais usam critérios

para seus triângulos. Uma delas é a **Triangulação de Delaunay**, que usa o seguinte critério:

Definição 4.1 ***Critério de Delaunay:*** *Um certo triângulo t faz parte da Triangulação de Delaunay τ somente se o círculo formado pelos vértices do triângulo não contém nenhum outro vértice de τ em seu interior.*

A Figura 4.4 ilustra esse critério. Podemos observar que, se uma certa aresta atrapalha o critério, podemos trocá-la por outra.

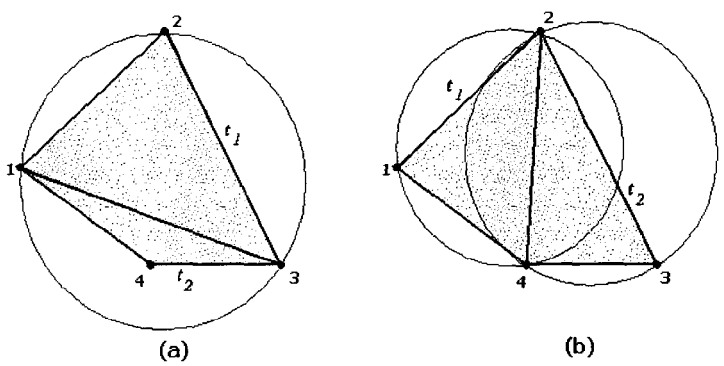


Figura 4.4: Em (a) vemos que o triângulo t2 não pode pertencer à Triangulação de Delaunay e portanto, a aresta 1-3 foi trocada pela aresta 2-4, resultando nos triângulos t1 e t2, em (b), que obedecem ao critério de Delaunay.

4.2 Estudo de caso: Uma abordagem baseada no Diagrama de Voronoi.

O trabalho apresentado por Décoret & Sillion [28] propõe um eixo médio modificado baseado no Diagrama de Voronoi. Os dados de entrada para o

sistema são conjuntos de polígonos. Cada polígono possui um único identificador - *BID* - , e seus vértices mantêm um campo com essa informação.

O sistema refina os polígonos fornecidos, inserindo mais vértices ao longo de suas arestas, para serem inseridos em uma Triangulação de Delaunay. Este refinamento das arestas foi necessário para garantir que as arestas dos polígonos estejam totalmente representadas dentre as arestas da triangulação. A Figura 4.5 ilustra esta situação.

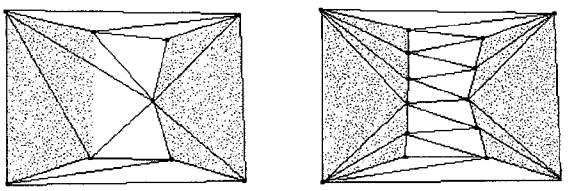


Figura 4.5: Figura (a) mostra que a Triangulação de Delaunay aplicada sobre os vértices originais dos polígonos fornecidos não é suficiente para que as arestas dos polígonos estejam representadas dentre as arestas da triangulação. É necessário acrescentar pontos ao longo das arestas dos polígonos para que as arestas da triangulação correspondam aos contornos dos polígonos, como se vê em (b).

O passo seguinte é a obtenção do Diagrama de Voronoi a partir da triangulação calculada. O Diagrama de Voronoi pode ser obtido pelo cálculo do dual das arestas de uma Triangulação de Delaunay. Como cada vértice possui um campo que identifica o quarteirão ao qual ele pertence, o trabalho desenvolvido por Décoret & Sillion [28] considera no Diagrama de Voronoi apenas as arestas duais das arestas [AB] na Triangulação de Delaunay cujos vértices possuem diferentes valores para o campo *BID* (BID(A)$\neq$ BID(B)).

Estas arestas duais consideradas são as arestas [ab] que indicam a posição das ruas em um mapa, como se vê em destaque na Figura 4.6(b):

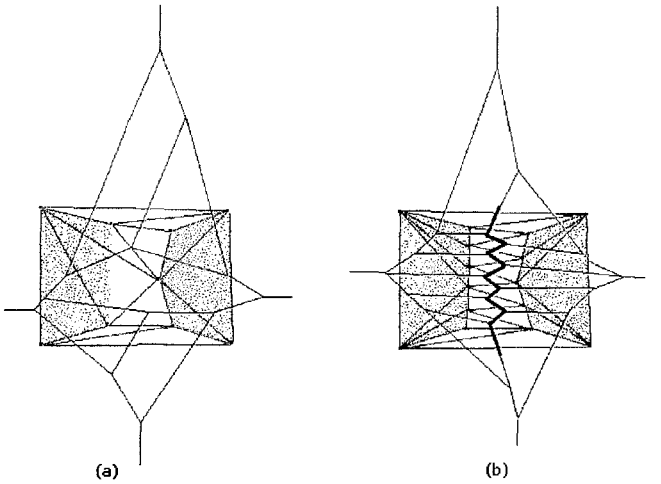


Figura 4.6: Em (a) vemos que as arestas do Diagrama de Voronoi não conseguem dar uma indicação da posição das ruas. Na figura (b), vemos que com o refinamento dos polígonos e o campo *BID* em cada vértice, conseguimos destacar as arestas que correspondem à posição das ruas.

As arestas de Voronoi consideradas são inseridas na estrutura do grafo. Os nodos do grafo mantêm uma lista de BIDs e as arestas mantêm dois campos - *leftBID* e *rightBID* - . O algoritmo para a inserção das arestas está delineado abaixo:

Após a inserção de arestas no grafo, o sistema terá um conjunto de **esquinas** e um conjunto de **caminhos** entre essas esquinas.

- **esquinas** são os nodos no grafo com dimensão igual ou maior que 3.

Algoritmo InsereArestaNoGrafo

1. se $\nexists$ nodo n_a no grafo na posição a
2. então cria nodo n_a
3. adicione *BID(A)* e *BID(B)* à lista de *BIDs* de n_a
4. repita os passos acima para b
5. crie uma aresta orientada ligando a a b
6. *rightBID* da aresta = *BID(A)*
7. *leftBID* da aresta = *BID(B)*

Figura 4.7: Algoritmo para inserir aresta no grafo.

- **caminhos** são definidos como uma seqüência de arestas que possuem os mesmos valores para *rightBID* e *leftBID*. São definidos para um caminho uma esquina inicial e uma esquina final.

Os nodos do grafo identificados como esquinas que estejam ligados diretamente por uma aresta, são contraídos em um único nodo de esquina, como mostra a Figura 4.8.

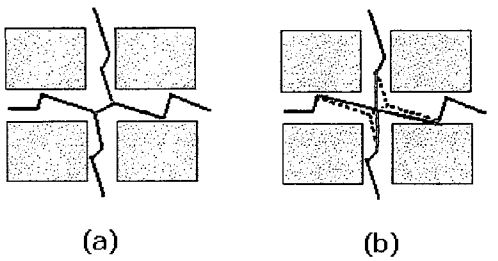


Figura 4.8: Junção de dois nodos de esquina. Figura (a) mostra a aresta que precisa ser eliminada. Figura (b) mostra o resultado da junção, com as arestas incidentes reposicionadas.

As esquinas e os caminhos determinados servem para particionar o espaço.

Esse particionamento considera duas áreas:

1. uma área poligonal em torno de cada esquina, como se vê na Figura 4.9(a).

2. uma área poligonal definida pelas bordas dos quarteirões entre as projeções da esquina inicial e final. Desta forma, são identificadas a linha inicial e a linha final para esta área. A Figura 4.9(b) ilustra essa região. É definida a **largura** l dessa área como sendo a menor distância entre a borda esquerda e a borda direita.

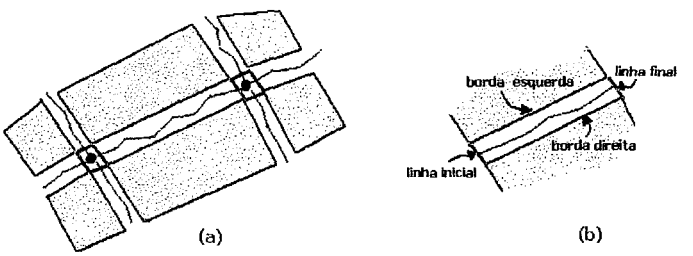


Figura 4.9: (a)Área poligonal definida em torno das esquinas. (b)Área poligonal definida pelas bordas dos quarteirões e pelas projeções das esquinas.

Por fim, para cada caminho, é determinado um esqueleto usando o procedimento descrito abaixo:

1. Para cada vértice da borda, é feita uma projeção desse vértice sobre a outra borda. Toma-se os pontos médios dessas projeções, como se vê na Figura 4.10(a). O ponto médio da linha inicial e o da linha final também são considerados.

2. Toma-se o conjunto de pontos colineares.

3. Será considerado o segmento maior. Segmentos menores são descartados.

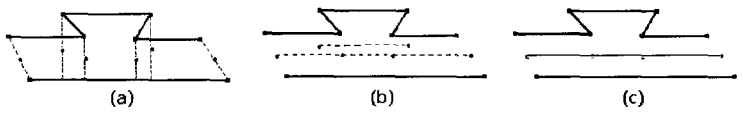


Figura 4.10: (a) Projeções dos vértices sobre a outra borda. Em (b) são tomados os segmentos de reta formados pelos pontos médios. Em (c), vemos que o segmento considerado é o maior.

Desta forma, é obtido um grafo cujas arestas são sempre segmentos de retas e possíveis reentrâncias nos contornos dos quarteirões não são considerados.

4.3 Nossa abordagem.

Tomamos por base a abordagem apresentada na seção 4.2, onde vimos que o grafo é extraído a partir do Diagrama de Voronoi, cujas arestas são calculadas como o dual das arestas de uma Triangulação de Delaunay aplicada sobre os polígonos refinados. Este refinamento de polígonos implicava na determinação de um ε para as distâncias entre os pontos a serem inseridos ao longo das arestas de cada polígono. Este ε não poderia ser maior que a menor distância entre dois polígonos. Nanci Amato [3] propõe uma maneira de se obter a menor distância entre dois polígonos.

Em nossa abordagem propusemos modificações à ideia apresentada na

seção 4.2, usando a Triangulação Restrita de Delaunay. Resultados do procedimento podem ser vistos no Capítulo 5. A Triangulação Restrita de Delaunay está definida abaixo:

Definição 4.2 ***Triangulação Restrita de Delaunay:*** *Uma triangulação restrita é uma triangulação de um conjunto de pontos que precisa incluir dentre as suas arestas um dado conjunto de segmentos ligando alguns pontos. As arestas correspondentes a estes segmentos são definidas como* arestas restritas. *A* ***Triangulação Restrita de Delaunay*** *contém arestas restritas e tenta construir os triângulos tentando ao máximo obedecer ao critério de Delaunay.*

Usando a Triangulação Restrita de Delaunay, fazemos as arestas dos polígonos serem arestas restritas, eliminando a necessidade de refinamento dos polígonos. Desta forma, eliminamos a necessidade de cálculo para a determinação de distâncias entre novos pontos a serem inseridos e nossa triangulação é construída utilizando apenas os vértices originais de cada polígono. Como a quantidade de pontos a serem inseridos na triangulação é menor temos como conseqüência uma diminuição no tempo de cálculo da triangulação.

A seguir detalhamos as etapas de utilização da Triangulação Restrita de Delaunay em nossa abordagem.

4.3.1 Construção da Triangulação

A entrada para a construção da Triangulação é um conjunto de polígonos, resultante da vetorização aplicada à imagem digital do mapa, como se vê na Figura 4.11(a).

Cada polígono é orientado na direção anti-horária e tem um número identificador. Os vértices são armazenados com esse número usando um campo *BlockID* para termos o controle sobre qual bloco contém determinado vértice. Para a construção da triangulação utilizamos a biblioteca CGAL*. A Figura 4.11(b) mostra o resultado da triangulação. As arestas em destaque são as arestas restritas.

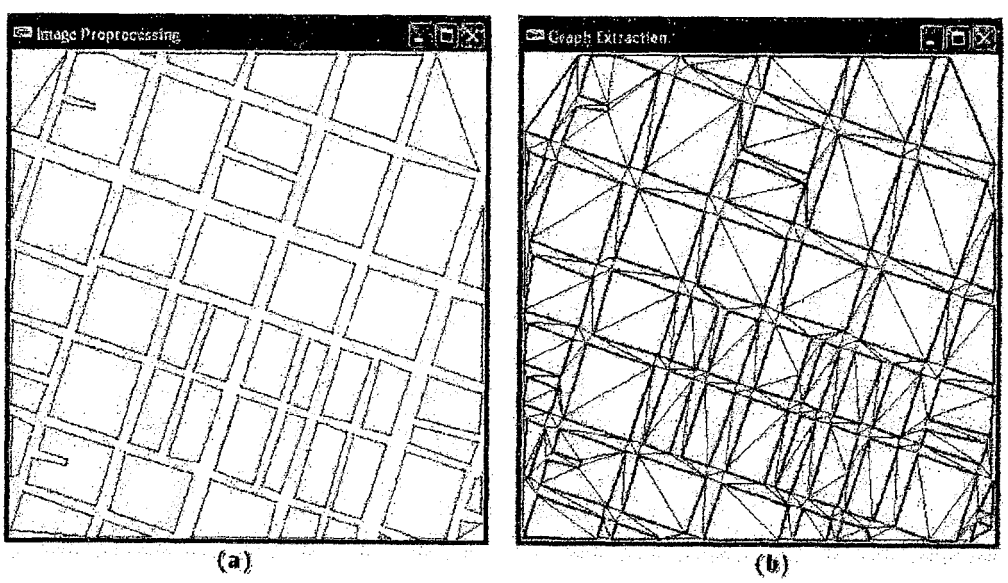


Figura 4.11: (a) Conjunto de Polígonos. (b) Resultado da triangulação restrita sobre o conjunto de polígonos da Figura (a).

*www.cgal.org

4.3.2 Classificação dos triângulos

Os elementos que auxiliam a identificação das regiões onde se localizam as esquinas e becos, as ruas e as áreas internas a quarteirões são o campo *BlockID* de cada vértice e as arestas restritas. Com esses elementos classificamos os triângulos da seguinte forma:

Triângulos Internos

São os triângulos que possuem uma das possíveis características:

1. Possuem exatamente 3 arestas restritas, ou
2. Possuem 2 arestas restritas, ou
3. Possuem 1 aresta restrita e todos os vértices com o mesmo valor armazenado no campo *BlockID*, ou
4. Não possuem arestas restritas e possuem todos os vértices com o mesmo valor armazenado no campo *BlockID*.

A Figura 4.12 ilustra as diferentes situações em que podemos encontrar os triângulos internos a quarteirões. As arestas restritas são as arestas representadas em vermelho. Na Figura 4.12(a), para cada $t_i, i = 0..3$, temos triângulos internos com 0, 1, 2 e 3 arestas restritas, respectivamente. Na Figura(b), o quarteirão q é um exemplo de quarteirão com exatamente 3 arestas restritas.

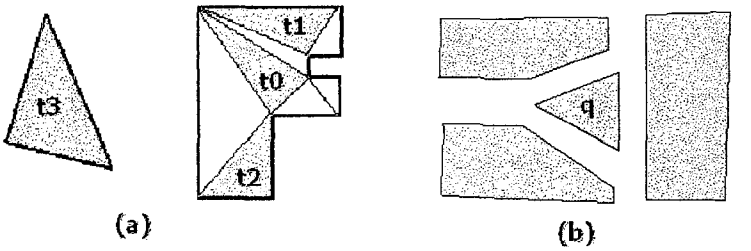


Figura 4.12: (a) Exemplos de triângulos internos. (b) Um exemplo de quarteirão com 3 arestas restritas.

Triângulos de Caminho

Triângulos que correspondem às regiões onde se localizam as ruas e avenidas. São identificados como triângulos de caminho aqueles que possuírem exatamente 1 aresta restrita e cujos vértices não possuem valores iguais para o campo *BlockID*. Na Figura 4.13 vemos a representação de dois quarteirões $q1$ e $q2$ e sua triangulação correspondente. Os triângulos de caminho são os triângulos $t1$, $t2$ e $t3$. Observe que o triângulo $t4$ também possui apenas uma aresta restrita, mas não foi classificado como um triângulo de caminho, pois seus vértices pertencem ao mesmo quarteirão, ou seja, o campo *BlockID* para os seus três vértices são iguais. Nesse caso, $t4$ é classificado como um triângulo *interno*.

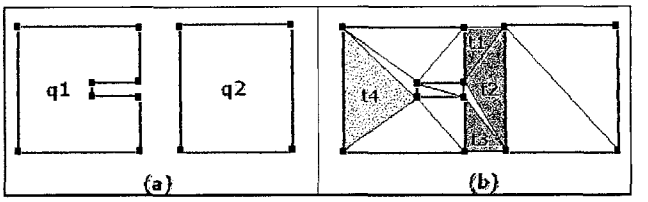


Figura 4.13: Exemplos de triângulos de caminho. Em (a) temos os quarteirões $q1$ e $q2$. Em (b), a triangulação mostra os casos onde os triângulos podem ser classificados como triângulos de caminho. O triângulo $t4$ foi clasificado como um triângulo *interno*.

Triângulos de Esquina

Os triângulos classificados como triângulos de esquina são aqueles cuja posição indica a localização das esquinas no mapa. Eles podem ser identificados por possuírem as seguintes características:

- eles não possuem nenhuma aresta restrita, e
- todos os vértices pertencem a quarteirões diferentes, ou seja, o campo *BlockID* dos três vértices são diferentes.

A Figura 4.14 mostra exemplos de triângulos de esquina. Figura 4.14(a) mostra um conjunto de quarteirões e Figura 4.14(b) mostra a triangulação correspondente, destacando os triângulos de esquina $t1$, $t2$, $t3$ e $t4$. Podemos observar que nenhum vértice desses triângulos terá um valor repetido para o campo *BlockID*.

Em um mapa urbano, encontramos quarteirões em sua grande maioria na forma de retângulos. A Figura 4.14 mostra o caso mais comum de esquina,

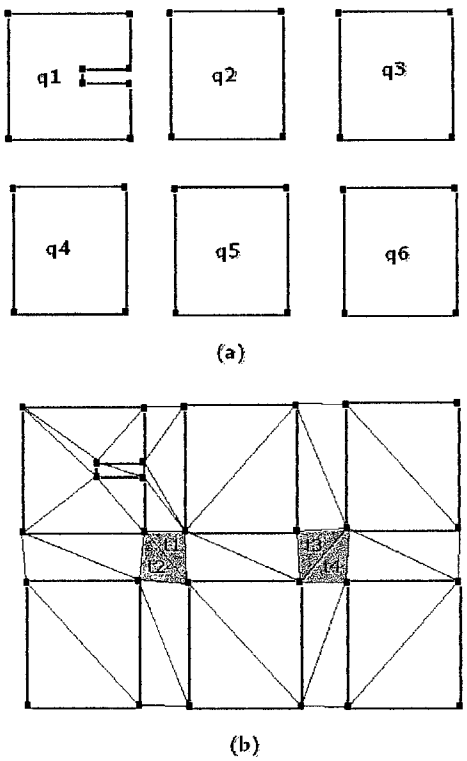


Figura 4.14: Exemplos de triângulos de esquina. (a)Conjunto de quarteirões. Em (b) são destacados os triângulos de esquina.

Nesse caso, as esquinas serão representadas no grafo como vértices de grau 4, ou seja, vértices com 4 arestas incidindo sobre eles. No entanto, existem outros tipos de esquinas. Observamos que há casos em que as esquinas serão vértices de grau 3 apenas. Praças e rotatórias são exemplos que fogem ao caso mais comum. A figura abaixo mostra outros possíveis tipos de esquinas. Figura 4.15(a) mostra conjuntos de quarteirões e a Figura 4.15(b) destaca em cinza os triângulos de esquina:

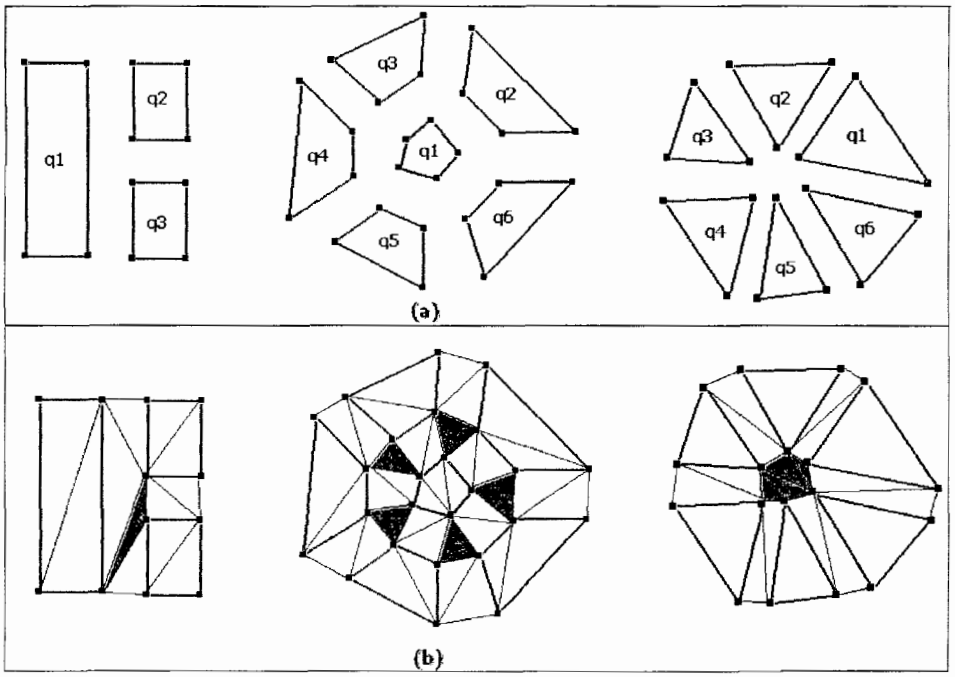


Figura 4.15: Mais exemplos de triângulos de esquina.

Triângulos de Becos

Em mapas urbanos encontramos comumente ruas sem saída, denominadas *becos*. Dentro da triangulação, os triângulos de beco indicam a posição de

entrada para essas ruas sem saída. Eles são um caso especial de triângulo de esquina. Eles também não possuem nenhuma aresta restrita, mas dois de seus vértices pertencem ao mesmo quarteirão. A Figura 4.16 mostra um triângulo de beco t. Podemos observar que seus vértices $v0$ e $v1$ pertencem ao mesmo quarteirão $q1$, enquanto que o vértice $v2$ pertence ao quarteirão $q2$.

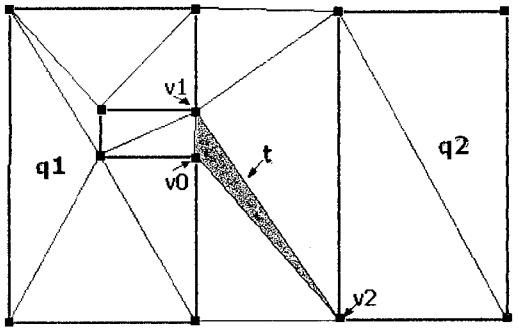


Figura 4.16: Exemplo de triângulos de beco.

4.3.3 Expansão dos triângulos de esquina

Uma vez classificados os triângulos, precisamos de um método para determinar a quais outras esquinas cada esquina deverá se ligar para formar as arestas do grafo. Os vértices do grafo são determinados pelas posições dos baricentros dos triângulos de esquina e de becos.

A triangulação é uma estrutura de dados que mantém relações de vizinhança entre os triângulos. Isto possibilita a elaboração de algoritmos para percorrer a triangulação, ou seja, visitar os demais triângulos, a partir de um

triângulo dado.

Implementamos um método para percorrer a triangulação de forma a encontrar os triângulos de esquina que deveriam se conectar. Precisávamos de uma maneira sistemática, sem repetições, de visitar cada triângulo. Fizemos uma analogia da estrutura da triangulação com um grafo, como mostra a Figura 4.17 e nosso método toma por base o algoritmo de busca em largura em grafos [8], descrito abaixo:

Definição 4.3 ***Algoritmo de Busca em Largura em Grafos:*** *Dado um grafo G=(V,E), o Algoritmo de Busca em Largura é um procedimento que se inicia a partir de um vértice v e encontra quais vértices são alcançáveis a partir de v.*

A Figura 4.17(a) mostra em destaque os triângulos de esquina, em cinza escuro e os triângulos de caminho, em cinza claro. Figura 4.17(b) mostra uma analogia de um grafo correspondente. Nosso algoritmo implementa uma busca em largura adaptada, percorrendo os triângulos de caminho até encontrar um triângulo de esquina.

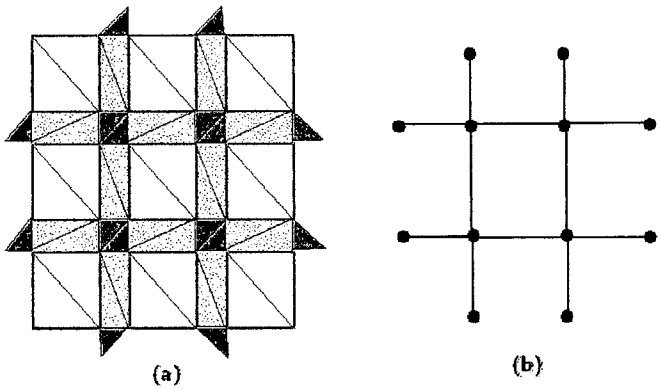


Figura 4.17: Analogia entre a triangulação e um grafo para implementação de uma busca em largura adaptada.

Abaixo, seguem as rotinas para a expansão dos triângulos de esquina:

```
Algoritmo LigaEsquinas

1.  para cada triângulo t de esquina faça
2.    Expandir(t)
```

Figura 4.18: Algoritmo para ligar esquinas.

```
Algoritmo Expandir(triângulo v)

1.  Marcar(v)
2.  para todo vizinho u de v faça
3.    se u é do tipo CAMINHO e não está marcado então
4.      fim = ExpandeCaminho(u)
5.    se fim é do tipo ESQUINA ou do tipo BECO então
6.      Liga(v, fim)
```

Figura 4.19: Algoritmo para expandir um triângulo v.

```
Algoritmo ExpandeCaminho(triângulo w)

1.  Marcar(w)
2.  para todo vizinho z de w faça
3.    se z é do tipo CAMINHO e não está marcado
4.      retorne(ExpandeCaminho(z))
5.    senão se z é do tipo ESQUINA ou BECO e não está marcado
6.      retorne(z)
7.    retorne(VAZIO)
```

Figura 4.20: Algoritmo para expandir caminho de um triângulo w.

O resultado da ligação das esquinas pode ser visto na Figura 4.21.

Podemos observar que, como o algoritmo percorre a triangulação e faz as ligações respeitando critérios de vizinhança dos triângulos, o resultado gera duas esquinas de grau 2, no lugar onde deveria estar apenas uma esquina de grau 4. Para resolver este problema, basta detectar a existência de 2 ou mais triângulos de esquinas que estejam vizinhos e contraímos os vértices em um só, localizado no centro geométrico dos vértices em cada triângulo. Esta operação produz o resultado ilustrado na Figura 4.22.

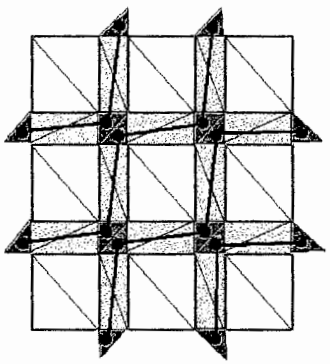


Figura 4.21: Resultado do algoritmo de expansão de esquinas.

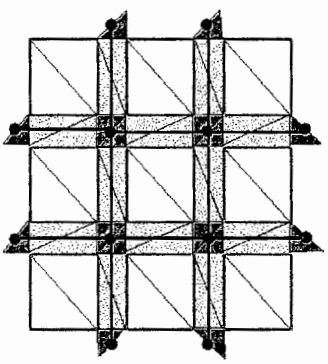


Figura 4.22: Resultado do *merging* de triângulos de esquinas vizinhos.

4.3.4 Ajuste Geométrico

A posição do baricentro de alguns triângulos de esquina, principalmente aqueles correspondentes às esquinas de grau 3, não reflete uma posição esperada para as esquinas em questão. Na figura abaixo vemos como ocorre essa situação. Figura 4.23(a) mostra a aresta e gerada pela expansão dos triângulos de esquina destacados em cinza. Na Figura 4.23(b) vemos o mesmo resultado, mas sem os detalhes da triangulação.

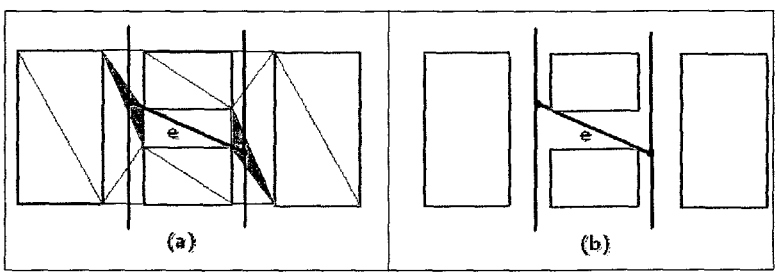


Figura 4.23: Aresta gerada pela expansão dos triângulos de esquina destacados em cinza. Em (b) vemos o mesmo resultado, sem os detalhes da triangulação.

Em muitas aplicações que utilizam grafos, o peso das arestas é um dado importante. No caso do resultado visto na Figura 4.23, observamos que a topologia está correta, mas a geometria resultante não corresponde à geometria esperada. Isto vem a ser um problema, pois o cálculo do peso de arestas como a da figura não resultaria num dado real.

Para resolver este problema, implementamos um ajuste geométrico, que se resume nos seguintes passos:

1. Tomar o menor lado do triângulo de ESQUINA que gere uma esquina com grau 3;

2. Traçar uma reta paralela ao menor lado, passando pelo vértice do triângulo que não pertence ao menor lado;

3. Traçar uma reta perpendicular, passando pelo ponto médio do menor lado;

4. Achar a interseção das retas encontradas nos passos 2 e 3;

A nova posição do vértice será o ponto médio do segmento $\overline{ab}$, ilustrado na figura abaixo. Figura 4.24(a) ilustra o detalhe do procedimento de ajuste geométrico. Figura 4.24(b) mostra o resultado do ajuste.

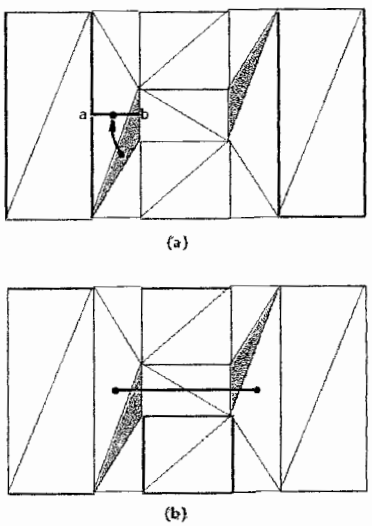


Figura 4.24: (a)Detalhe do procedimento de ajuste geométrico. Em (b), vemos o resultado do procedimento.

Resultados reais podem ser conferidos nas figuras abaixo. Mais detalhes sobre resultados computacionais são descritos no Capítulo 5.

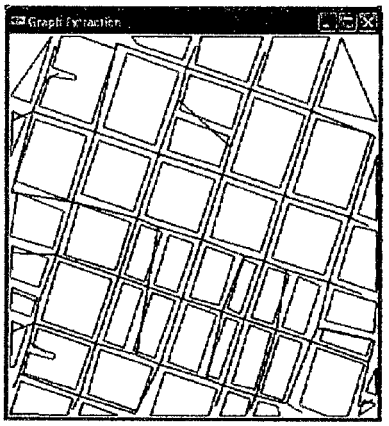


Figura 4.25: Grafo resultante sem o ajuste geométrico.

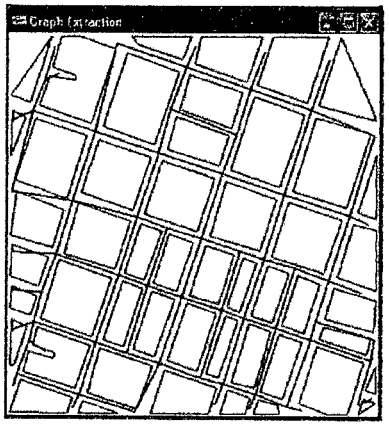


Figura 4.26: Grafo resultante com o ajuste geométrico.

Capítulo 5

Conclusões e Resultados

Sistemas para reconhecimento de ruas e estradas em imagens podem ser divididos em duas categorias: (1) sistemas que apenas realçam os *pixels* correspondentes às ruas e estradas e (2) sistemas que extraem tanto as ruas e estradas quanto as suas interseções, resultando em uma rede viária.

Os sistemas definidos em (1) se baseiam em sua maioria na implementação de operações de processamento de imagens. São encontrados também na literatura abordagens utilizando outros recursos, como a aplicação de redes neurais. Sistemas que extraem informação topológica expressa na forma de um grafo em sua maioria recebem como entrada a representação vetorial da imagem, como é observado em Cinthia [7] e Décoret & Sillion [28].

Nossa proposta de trabalho provê uma solução completa para a extração de um grafo a partir de imagens de mapas urbanos digitalizados. Implementamos diferentes etapas para o processo, desde operações de processamento

de imagens, passando pela fase de extração dos polígonos correspondentes a quarteirões aplicando técnicas de vetorização e finalmente inserindo os vértices desses polígonos em uma Triangulação Restrita de Delaunay. Esta última etapa foi implementada tomando por base o trabalho proposto por Sillion [28], com modificações que reduziram os cálculos necessários para se obter o resultado final. O Capítulo 4 discute esse trabalho e descreve a nossa abordagem.

A implementação dessas diferentes etapas permite ao sistema receber como entrada tanto representações vetoriais de um mapa quanto suas imagens digitais coloridas sem nenhum tratamento.

Os resultados finais alcançaram margens de acertos satisfatórias se comparadas a outros métodos estudados. A seção 5.1 mostra os resultados obtidos em nosso sistema para um conjunto de imagens de mapas reais.

5.1 Resultados Computacionais

As tabelas 5.1 e 5.2 mostram resultados obtidos em 6 imagens de mapas reais. Denotamos como $|V|$ e $|E|$ as quantidades de vértices e de arestas nos grafos, respectivamente. Na tabela 5.1 mostramos os resultados da detecção de vértices, destacando as quantidades de detecção de falsos vértices e as quantidades de vértices que não foram detectados, ao compararmos com o grafo esperado de cada imagem.

Na tabela 5.2 fazemos a mesma análise, mas para as quantidades de arestas. As imagens utilizadas nos testes podem ser vistas no Apêndice C.

	Grafo Esperado	**Grafo Obtido**			
Mapa	$\|V\|$	$\|V\|$	Falsos Vértices	Vértices não detectados	% acerto $\|V\|$
m10	83	79	0	4	95.18
m11	67	63	0	4	94.03
m12	72	71	0	1	98.61
m20	74	71	0	3	95.95
m21	66	66	0	0	100.00
m22	66	64	0	2	96.96
testmap01	154	140	0	14	90.90
testmap02	86	83	0	3	96.51
testmap03	61	56	0	5	91.80
testmap04	67	59	0	8	88.05
testmap05	42	44	2	1	92.85
testmap06	54	52	0	3	96.29
testmap07	122	121	0	1	99.18
testmap08	63	53	0	10	84.12
testmap09	43	43	3	3	86.04
testmap10	46	47	3	2	89.13

Figura 5.1: Resultados obtidos na obtenção de vértices (esquinas) em diferentes mapas reais.

Pela análise das tabelas e das figuras mostradas no Apêndice C, concluímos que nosso método apresenta uma boa margem de acertos, com raros casos em que haja detecção de falsos vértices ou arestas.

	Grafo Esperado	Grafo Obtido			
Mapa	$\|E\|$	$\|E\|$	Falsas Arestas	Arestas não detectadas	% acerto $\|E\|$
m10	100	96	0	4	96.00
m11	83	79	0	4	95.18
m12	91	91	2	2	95.64
m20	96	93	0	3	96.88
m21	92	92	0	0	100.00
m22	88	86	0	2	97.72
testmap01	227	208	1	20	91.18
testmap02	123	118	0	5	95.93
testmap03	85	80	1	6	92.94
testmap04	95	84	0	11	88.42
testmap05	56	58	2	0	96.42
testmap06	73	67	0	6	91.78
testmap07	175	174	0	1	99.42
testmap08	90	80	1	11	87.77
testmap09	62	61	3	4	88.70
testmap10	61	59	2	4	90.16

Figura 5.2: Resultados obtidos na obtenção de arestas (ruas) em diferentes mapas reais.

5.1.1 Complexidades

Nesta seção avaliamos os tempos computacionais para as operações efetuadas nos diferentes estágios do processo de extração do grafo.

As operações efetuadas durante a etapa de processamento da imagem levam tempo $O(n)$, onde n é a quantidade de *pixels* na imagem.

A implementação da etapa de vetorização tomou por base o algoritmo proposto por Douglas & Peucker [11], que leva tempo $O(n^2)$, onde n é a quantidade de *pixels* em uma componente conexa representando um quarteirão. Esse tempo computacional pode ser diminuído aplicando-se o algoritmo proposto por Snoyeink *et al* [12], que leva tempo $O(n \log n)$.

A Tabela 5.3 mostra as complexidades para as operações efetuadas na etapa final de extração do grafo.

Etapa final: Extração do Grafo		
Operação	**Complexidade**	**Observações**
Construção da Triangulação	$O(n \log n)$	n = número de vértices.
Classificar Triângulos	$O(n)$	n = número de faces.
Ligar Esquinas	$O(n + m)$	n = número de triângulos de esquina. m = número de triângulos de caminho

Figura 5.3: Complexidades para as operações efetuadas na etapa de extração do grafo.

5.2 Trabalhos Futuros

O trabalho apresentado serve como ponto de partida para alguns trabalhos futuros, que enumeramos nesta seção.

5.2.1 Unificar os dois programas.

No Capítulo A vemos que o sistema implementado se constitui de dois programas separados. Como ambos foram escritos usando as mesmas linguagens de programação e utilizam a mesma biblioteca para os procedimentos de exibição gráfica, uma primeira sugestão seria a junção dos dois programas em um só, desenvolvendo para o programa único uma interface adequada para o usuário.

5.2.2 Implementar um *mosaico* de grafos.

Muitas vezes um mapa urbano não é fornecido em sua totalidade. Ele é normalmente dividido em partes menores dispostas em imagens separadas que podem posteriormente ser combinadas. A esta união de partes menores de uma imagem para a recuperação de uma imagem maior chamamos de *mosaico*.

Nosso sistema trabalha com uma imagem de cada vez e obtém o grafo correspondente. As imagens de teste utilizadas e exibidas no Apêndice C fazem parte do mapa da cidade de Fortaleza.

Com essas imagens podemos fazer o mosaico do mapa completo. Como exemplo, a Figura C.5(a) mostra uma parte do mapa de Fortaleza que está à esquerda da imagem da Figura C.6(a). Se estas imagens forem colocadas lado a lado, vemos que a Figura C.6 é a continuação da Figura C.5, à sua direita. A Figura 5.4 ilustra esse fato.

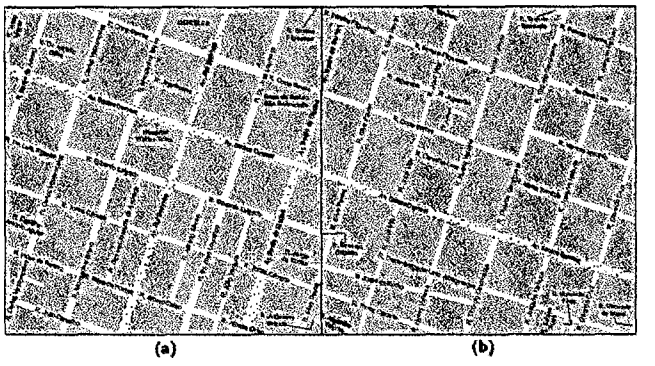


Figura 5.4: Em (a) vemos o mapa m21 que é continuado pelo mapa m22, mostrado em (b).

Seria interessante implementar um *mosaico para os grafos* obtidos. Como podemos obter um grafo para cada imagem, resta obter um grafo que seja o resultado da combinação de dois ou mais grafos já extraídos. Devemos observar que o mosaico deve ser feito para continuações para cima, para baixo, para a esquerda e para a direita.

Um ponto de partida para a implementação desta idéia poderia ser aproveitando a própria triangulação. As arestas que vão se conectar para formar o grafo maior são arestas que têm como um de seus vértices esquinas geradas por triângulos infinitos. A estas esquinas poderíamos chamar de *esquinas especiais*. Para a combinação de dois grafos, precisaríamos estudar

uma correspondência entre esquinas especiais de dois grafos. A Figura 5.5 ilustra a idéia.

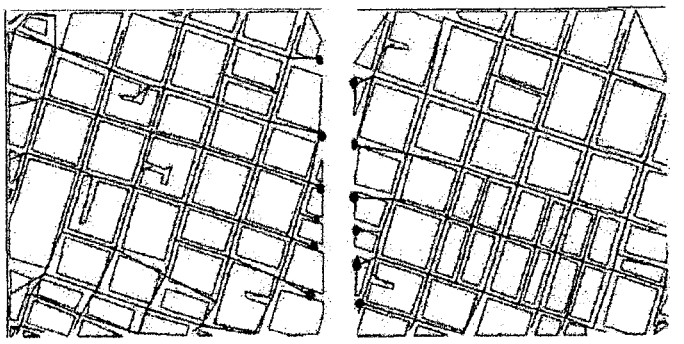


Figura 5.5: Destacadas com círculos as esquinas que deverão ser ligadas ao combinarmos dois grafos.

Uma analogia aos *operadores de Euler*[19] seria uma sugestão para a construção do algoritmo para a união dos dois grafos. Os operadores de Euler são aplicados em modelagem de sólidos e descrevem procedimentos para a criação e destruição de vértices, arestas e faces. Observando a Figura 5.5, vemos que os vértices especiais que fazem correspondência deverão ser combinados em um único vértice. As arestas que chegam a esses dois vértices também deverão ser reajustadas para o vértice que vai ficar. Portanto, seria necessário implementar rotinas para destruição de um dos vértices mantendo a coerência entre as arestas que chegam a eles.

5.2.3 Acrescentar rótulos às arestas.

As arestas em nosso grafo representam ruas e avenidas. Problemas de Otimização Combinatória aplicados a infra-estrutura urbana precisam obter

no grafo correspondente à malha viária informações tais como os pesos das arestas, sentido das ruas e ainda os nomes das ruas. Uma maneira de armazenar essas informações é através do uso de rótulos associados a cada aresta.

Uma outra proposta de trabalho complementar ao nosso seria acrescentar ao grafo extraído os rótulos para as arestas. Encontramos em [22] um estudo sobre a extração dos nomes de ruas em imagens de mapas urbanos. O estudo poderia ser estendido para a extração de outros símbolos específicos como setas indicando sentidos das ruas. A Figura 5.6 mostra um exemplo de mapa que exibe esta informação. Desta forma, seria possível detectar e armazenar dados como sentido das ruas e os nomes das ruas. Os pesos das arestas são facilmente obtidos pelo cálculo das distâncias entre os vértices que as definem.

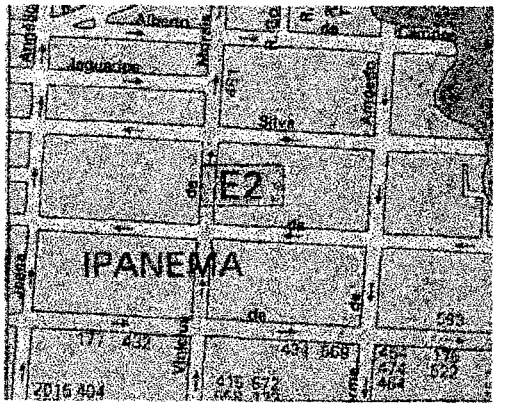


Figura 5.6: Exemplo de mapa urbano que inclui entre seus símbolos setas indicando os sentidos das ruas.

5.2.4 Aprimorar o ajuste geométrico.

O ajuste geométrico, descrito no Capítulo 4 está limitado à situação descrita na seção 4.3.4. No entanto, obesrvamos que há outras situações, menos freqüentes, que necessitam de ajuste geométrico.

Seria interessante a implementação de um procedimento mais genérico, que fosse independente das circunstâncias de determinado tipo de esquina. Uma idéia seria testar as arestas do grafo contra as arestas restritas da triangulação, que são as arestas que representam os contornos dos quarteirões. É necessário, porém, estudar uma forma eficiente de se efetuar esses testes.

5.2.5 Implementar um método mais eficiente de vetorização.

Podemos sugerir a substituição do algoritmo existente, que apresenta tempo $O(n^2)$ por um algoritmo que tome por base a idéia apresentada por Hershberger & Snoeyink [12], que apresenta tempo $O(nlogn)$, que corresponde ao tempo no melhor caso do algoritmo usado em nosso trabalho.

5.2.6 Modificar o algoritmo que faz a junção de esquinas.

Foi observado em nossos testes que o problema de não detecção de vértices é causado pela existência de ruas que apresentam uma quebra em sua direção

ou que terminam muito próximas de onde outras iniciam. A triangulação não acompanha o ponto onde existe essa quebra de direção e detecta somente uma esquina, onde deveria detectar duas. A Figura 5.7 ilustra essa situação. A Figura(a) destaca um caso de rua cuja direção não é contínua. Na Figura(b) vemos o grafo esperado para esta situação. Em (c), vemos como a triangulação é formada, e destacamos em cinza os triângulos de esquina. Na Figura(c) temos o resultado obtido em nosso sistema.

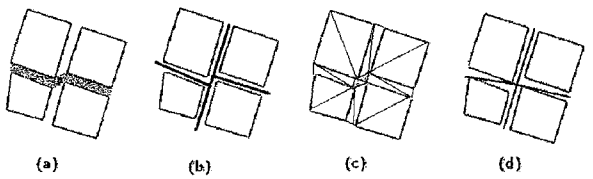


Figura 5.7: Estudo de caso em que há problema na detecção de vértice.

Poderíamos sugerir um estudo sobre como resolver este problema. Uma possibilidade é evitar que os triângulos de esquinas, nesses casos, sejam contraídos em um só, como é descrito na seção 4.3.3 e ilustrado na Figura 4.22. Para casos como o descrito acima, em vez de juntar as esquinas, seria necessário mantê-las separadas e acrescentar uma aresta ligando os baricentros desses triângulos de esquina vizinhos, obtendo assim o resultado esperado, ilustrado na Figura 5.7(b).

5.2.7 Estender o método para obtenção de grafos não planares.

Alguns sistemas que extraem grafo correspondente a uma malha viária utilizam como entrada imagens e informações sobre alturas nos objetos da imagem. Tais informações auxiliam na eliminação de falsas ruas. Como exemplo, em fotos aéreas os tetos de alguns prédios podem ser detectados como ruas, mas se forem utilizadas informações adicionais sobre as alturas em cada ponto da imagem, essa possibilidade de falsa detecção é eliminada, como é observado em Baumgartner, 99 [13]. Em Décoret & Sillion [28] a entrada para o sistema é um conjunto de polígonos no espaço 2.5D, onde cada polígono possui uma informação de altura associada a ele. Essa informação é utilizada para a modelagem da cena urbana, pois os polígonos nesse caso representam contornos de prédios e não de quarteirões.

Seria interessante obter essa informação adicional sobre alturas do terreno mapeado para estudar a viabilidade de extração de um grafo não planar. Uma opção seria utilizar triangulação 3D e fazer uma analogia ao métodos aplicados em nosso trabalho. Uma vantagem de um grafo não planar seria a correta detecção de viadutos e elevados.

Apêndice A

Sistema Implementado

Nossa implementação está dividida em dois programas. O primeiro programa trata das operações relacionadas às etapas de processamento e vetorização da imagem. O segundo programa trata da extração automática de um grafo equivalente ao mapa analisado, utilizando a Triangulação Restrita de Delaunay.

Nosso objetivo neste capítulo é mostrar os modos de operação nos dois programas. A seção A.1 descreve o primeiro programa. Na seção A.2 abordamos as características do segundo programa.

Os dois programas foram desenvolvidos utilizando as linguagens de programação C padrão e C++. Operações de exibição da imagem foram implementadas utilizando o sistema gráfico *OpenGL*.

A.1 Primeiro Programa: Processamento e Vetorização da Imagem

Este programa recebe como entrada uma imagem, que é fornecida como argumento de chamada ao programa. Essa imagem deve estar no formato *PPM* e com ela o programa efetua as operações descritas nas seções A.1.1 e A.1.2. Maiores informações sobre o formato de *arquivos PPM* podem ser encontradas no Apêndice B.

A interface deste programa se compõe de elementos padrões de Interface Gráfica com o Usuário, como botões de rádio, caixas de verificação, botões e campos para entrada de dados. A padronização desses elementos ajuda ao usuário a se familiarizar rapidamente com a operação no sistema implementado [27]. Os comandos na interface estão escritos em língua inglesa, para facilitar o estudo e o uso do sistema por estudantes e pesquisadores de outros países.

O objetivo desta interface é fazer com que o sistema possa se adequar às diferentes condições que uma imagem fornecida possa apresentar, dando ao usuário liberdade para optar por quais operações devem ser aplicadas à cada imagem. Para o desenvolvimento desta interface utilizamos o pacote *GLUI**. A Figura A.1 mostra as telas do sistema: uma contendo a interface gráfica e outra para exibição dos resultados das operações efetuadas sobre a imagem.

* *OpenGL User Interface*

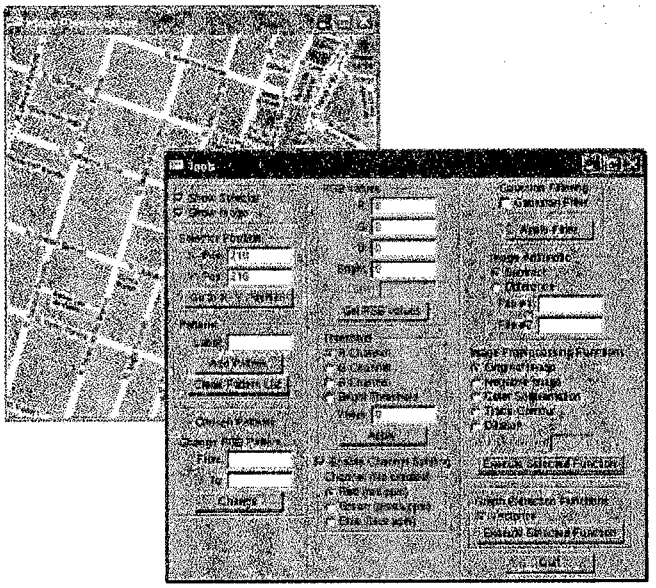


Figura A.1: Telas que compõem a interface do programa.

A.1.1 Efetuando operações de Processamento de Imagens

Vamos descrever a maneira de se efetuar as operações de processamento da imagem na ordem em que elas foram mencionadas no Capítulo 3.

Redução da quantidade de cores na imagem

Para se efetuar esta operação o usuário conta com os seguintes elementos na interface:

- Seletor;
- Painel `Pattern`;

- Painel `Chosen Patterns`;
- Opção `Color Segmentation`, no Painel `Image Preprocessing Functions`.

A forma de utilização de cada um desses elementos está descrita a seguir.

A redução de cores de uma imagem é feita pela substituição de cores dos *pixels* da imagem por aquelas que estão em um conjunto de cores escolhidas pelo usuário. O usuário seleciona quantas e quais cores deseja manter no mapa. O sistema varre a imagem e substitui as cores dos *pixels* que não estão dentro do conjunto de cores selecionadas pelo usuário pela cor que mais se aproxima de uma das cores escolhidas.

O Seletor aparece sobre a imagem, na forma de uma mira, acompanhando o movimento do mouse, quando está com o botão pressionado. Com ele é possível selecionar um determinado ponto na imagem para o armazenamento dos valores RGB daquele *pixel* escolhido. A Figura A.2 mostra uma imagem e o seletor sobre ela.

Cada cor escolhida é armazenada quando o usuário posiciona o seletor sobre um ponto da imagem e clica sobre o botão `Add Pattern`. O usuário precisa neste momento atribuir um rótulo àquela cor escolhida. O uso de rótulos facilita ao usuário fazer uma correspondência entre a cor escolhida e seu significado na imagem. Ainda, o usuário consegue controlar quais cores já foram escolhidas.

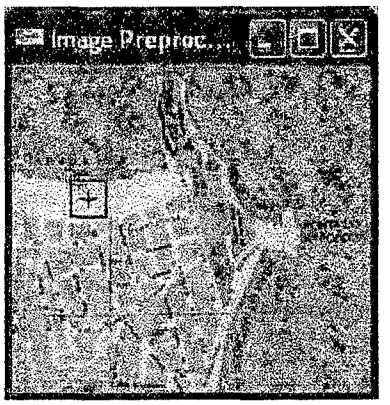
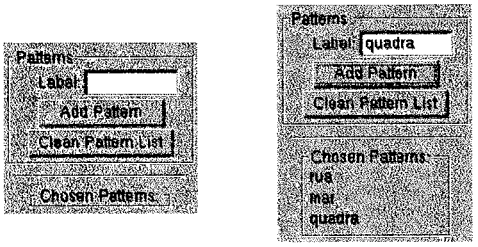


Figura A.2: O seletor é o elemento responsável por auxiliar o usuário na seleção de cores da imagem.

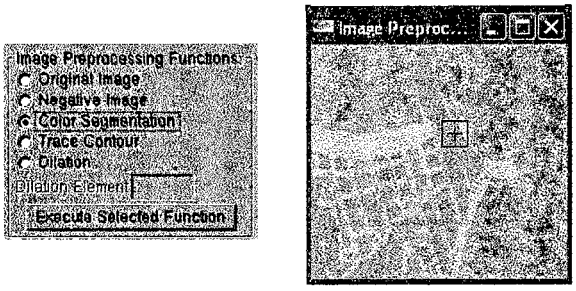
A estrutura responsável pelo armazenamento das cores escolhidas é uma lista chamada de *Lista de Padrões*. O Painel **Chosen Patterns** mostra o conteúdo dessa lista à medida que novas cores vão sendo selecionadas. Na Figura A.3 vemos os painéis **Patterns** e **Chosen Patterns**. No primeiro painel vemos a caixa de edição onde o usuário deverá inserir o nome do padrão e o botão para inserir a cor escolhida na lista de padrões. Um botão para esvaziar a lista de padrões também é fornecido, caso o usuário deseje refazer as escolhas de cores. Podemos observar que quando o usuário não selecionou cores, a lista de padrões está vazia e o painel **Chosen Patterns** também está vazio, como mostra a Figura A.3(a). A Figura A.3(b) mostra como ficam os painéis quando o usuário vai escolhendo e rotulando as cores na imagem.



(a) (b)

Figura A.3: Painéis relativos às operações de seleção de cores na imagem. Figura(a) mostra o painel *Chosen Patterns* vazio, enquanto que em (b) vemos o conteúdo da lista de padrões escolhidos.

Selecionadas todas as cores, o usuário pode efetuar a substituição na imagem inteira, selecionando a operação `Color Segmentation` e clicando sobre o botão `Execute Selected Function` no Painel `Image Preprocessing Functions`. A Figura A.4 mostra em (a) a opção a ser escolhida para se efetuar a operação e em (b) o resultado da operação aplicada sobre a imagem.



(a) (b)

Figura A.4: Em (a) vemos selecionada a opção para efetuar a redução de cores. Em (b), o resultado da operação aplicada sobre a imagem da Figura A.2.

Eliminação de legendas

O usuário pode eliminar legendas do mapa utilizando tanto os recursos do sistema para a aplicação de dilatação sobre a imagem como os recursos para a troca de padrões. Para dilatar uma cor escolhida, o usuário conta com os seguintes elementos na interface do sistema:

- Painel `Chosen Patterns`;
- Painel `Image Preprocessing Functions`.

No painel `Chosen Patterns` o usuário pode ver os nomes associados a cada cor na imagem, como mostra a Figura A.5.

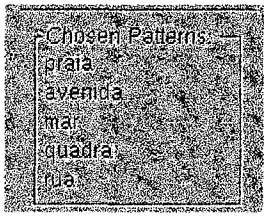


Figura A.5: Painel `Chosen Patterns` mostrando os rótulos associados às cores escolhidas.

Com esses nomes, podemos aplicar a operação de dilatação informando o rótulo da cor que se deseja dilatar. No Painel `Image Preprocessing Functions` devemos selecionar a opção Dilation e informar o rótulo da cor escolhida como indica a Figura A.6.

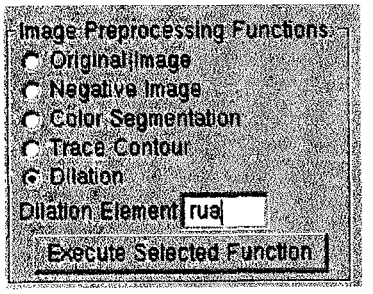


Figura A.6: Opção `Dilation` selecionada, com o campo preenchido com o rótulo associado à cor sobre a qual se deseja efetuar a dilatação.

Uma outra forma de conferir à qual rótulo uma determinada cor está associada é utilizando o Painel **`RGB Values`**. Basta posicionar o Seletor sobre um ponto na imagem e clicar sobre o botão **`Get RGB values`**. O Painel exibe os valores R, G e B de um *pixel* selecionado. O painel mostra também o brilho do *pixel* e seu rótulo associado. Para isso, a cor selecionada precisa já ter sido anteriormente associada a um rótulo. A Figura A.7 mostra um exemplo.

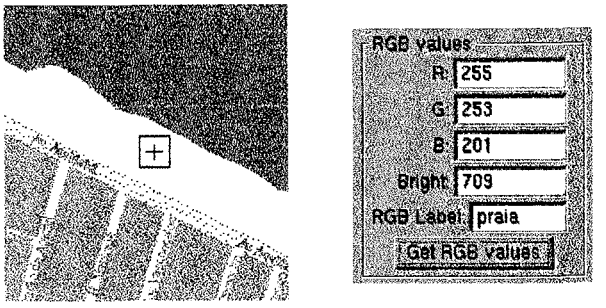


Figura A.7: Uma maneira para o usuário ver o rótulo associado um ponto selecionado na imagem.

A Figura A.8(a) mostra uma imagem com legendas sobre os *pixels* rotulados como sendo de rua. Em (b), temos o resultado da operação de dilatação sobre esses *pixels*.

Figura A.8: (a) Imagem antes da operação de dilatação sobre os *pixels* de rua. (b) Resultado da operação.

Em uma mesma imagem, podemos selecionar mais de uma cor para efetuar a operação de dilatação. Isto é necessário quando se deseja eliminar legendas localizadas sobre regiões apresentando cores diferentes. Ao observarmos a Figura A.8(b) notamos que ainda há legendas a serem eliminadas. Para fazer nova dilatação sobre uma cor diferente, basta selecionar o novo rótulo e executar a função de dilatação novamente, como mostra a Figura A.9(a). Na Figura A.9(b) vemos o resultado de uma nova aplicação de dilatação, desta vez, sobre os *pixels* rotulados como sendo de avenida, na mesma imagem mostrada na Figura A.8(b).

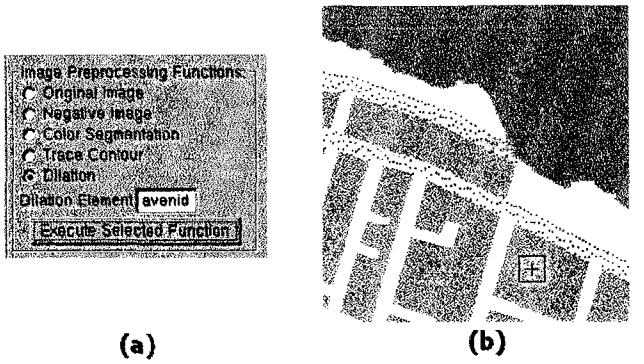


Figura A.9: (a)Selecionando novo rótulo para efetuar nova dilatação sobre a mesma imagem. (b) Imagem após nova operação de dilatação sobre os *pixels* de avenida.

Binarização da Imagem

Para cada imagem, a binarização deve ser aplicada utilizando parâmetros diferentes. Para isto, a interface oferece ao usuário condições de efetuar a operação de binarização de uma maneira que se ajuste a cada imagem. Para realizar esta operação, o usuário utilizará os painéis `RGB Values` e `Threshold`. O Painel `RGB Values` pode ser visto na Figura A.7(b).

O botão `Get RGB values` auxilia o usuário a definir como será o limiar para a binarização da imagem. O limiar pode então ser definido com relação a um dos canais RGB ou à soma de seus valores, com a opção `Bright Threshold`. A Figura A.10 mostra como o usuário deve fornecer os critérios para a definição deste limiar.

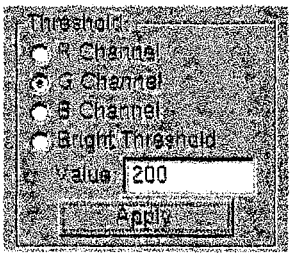


Figura A.10: Escolha de um *threshold* pelo canal G.

Nos casos em que a imagem apresenta regiões com valores RGB entre os valores RGB de estruturas que correspondam à malha viária (veja seção 3.1.3), o sistema oferece uma função para trocar padrões. A Figura A.11 mostra um exemplo de utilização do Painel `Change RGB Patterns` para a troca de *pixels* rotulados como sendo de praia por *pixels* rotulados como sendo de mar.

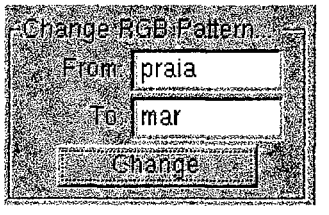


Figura A.11: Painel para a troca de padrões.

Extração de Contornos

Para efetuar a extração de contornos de uma imagem binarizada, basta o usuário selecionar a opção `Trace Contour` no Painel `Image Preprocessing Functions` e clicar sobre o botão `Execute Selected Function`. A Figura

A.12 mostra a opção a ser escolhida no painel para a execução desta operação e na Figura A.12 (b) vemos o resultado obtido.

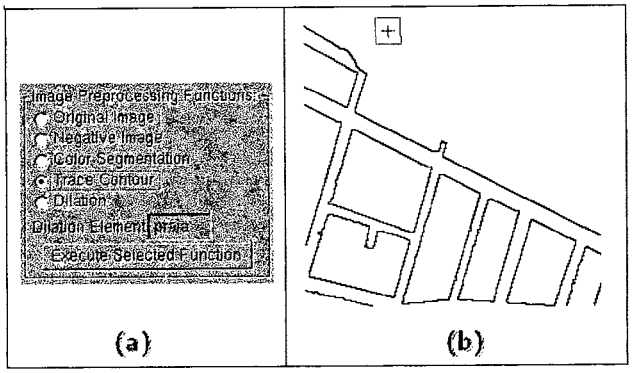


Figura A.12: (a)Opção a ser escolhida para obter o traçado dos contornos dos quarteirões (b) Resultado da operação aplicada sobre a imagem da Figura 3.14(b).

Além das operações já citadas, o sistema executa as seguintes operações:

Separação dos canais RGB

No Painel `Channel` o usuário pode obter imagens correspondentes a cada um dos canais RGB. O painel aparece inicialmente desabilitado. Para habilitá-lo, basta marcar a caixa de verificação `Enable Channel Splitting`. Selecionando um dos botões de rádio do Painel `Channel` o usuário obtém um arquivo de imagem referente ao canal escolhido e vê o conteúdo desse arquivo de imagem na tela de exibição de resultados. A Figura A.13 mostra o Painel `Channel` e a caixa de verificação para habilitá-lo.

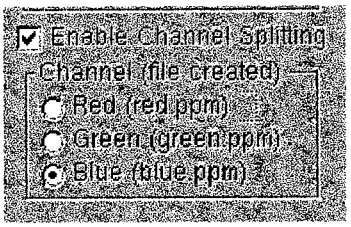


Figura A.13: Painel `Channel` e a caixa de verificação para habilitá-lo.

Aritmética sobre a imagem

As operações de subtração e diferença entre duas imagens são efetuadas no Painel `Image Arithmetic`. Um botão de rádio permite ao usuário determinar qual das duas operações será efetuada e nas caixas de edição de texto o usuário deverá indicar os nomes dos arquivos de imagem que serão submetidos à operação. A Figura A.14 mostra o Painel.

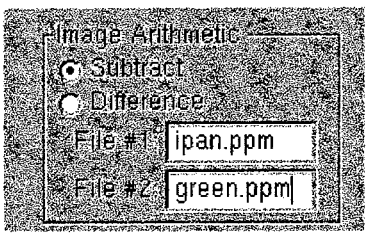


Figura A.14: Painel `Image Arithmetic`.

Obtenção do negativo da imagem

Para se obter o negativo de uma dada imagem, basta escolher a opção `Negative Image` no Painel `Image Preprocessing Functions`. A Figura A.15 mostra onde essa opção é encontrada na interface do sistema. O nega-

tivo da imagem é mostrado na tela de exibição de resultados.

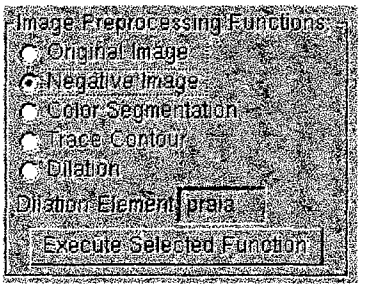


Figura A.15: Opção a ser escolhida para a obtenção do negativo de uma dada imagem.

Filtro Gaussiano

Para a aplicação de uma suavização Gaussiana na imagem, o usuário deverá usar o Painel *Gaussian Filtering*, marcando a caixa de verificação `Gaussian Filter`. A operação será efetivada quando o usuário clicar sobre o botão `Aplly Filter`. O resultado será a imagem suavizada exibida na tela de exibição de resultados. A Figura A.16 mostra o Painel `Gaussian Filtering`.

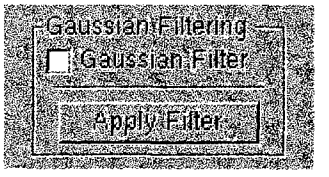


Figura A.16: Painel para a aplicação de filtro Gaussiano sobre uma dada imagem.

A.1.2 Efetuando operações de Vetorização da Imagem

Para se obter uma representação vetorial da imagem, o usuário deverá escolher a opção `Vectorize` no Painel `Graph Extraction Functions` e clicar sobre o botão `Execute Selected Function`. A Figura A.17 mostra o Painel.

Figura A.17: Painel para a vetorização da imagem.

A imagem deverá estar apresentando apenas as bordas dos quarteirões para poder ser vetorizada. Isto é necessário para que a vetorização resulte em um conjunto de polígonos fechados. Os polígonos são exibidos na forma de linhas do OpenGL na tela de exibição de resultados, podendo estar sobrepostos à imagem original ou não. Para optar por visualizar somente os polígonos, o usuário deverá desmarcar a caixa de verificação `Show Image`. Para tornar a ver a imagem, basta voltar a marcar essa caixa de verificação. A Figura A.18(a) mostra uma imagem com as bordas extraídas. Em (b) vemos um resultado de vetorização sobreposto à imagem. Na Figura A.18(c) vemos a exibição do resultado da vetorização quando o usuário desmarca a caixa de verificação `Show Image`.

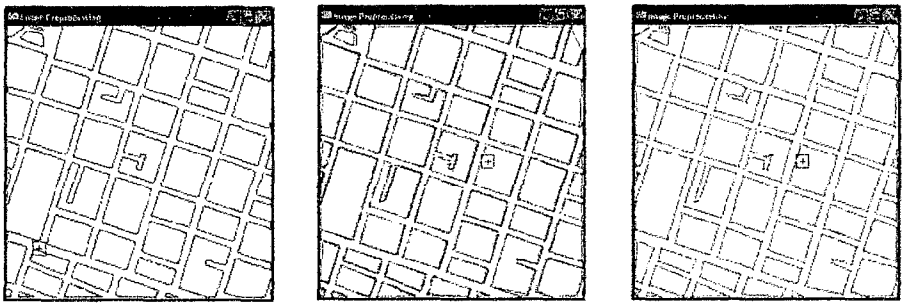


Figura A.18: (a) Imagem com a extração das bordas dos quarteirões. (b)Resultado de vetorização sobreposto à imagem. Em (c) vemos apenas o resultado da vetorização, sem exibição da imagem.

Esta operação gera um arquivo-texto. Cada linha do arquivo contém informação sobre cada vértice dos polígonos originados. O arquivo é organizado em três colunas: a primeira coluna armazena os valores da ordenada x de cada vértice; a segunda coluna armazena os valores de y e na terceira coluna ficam armazenados o número identificador de cada polígono. A Figura A.19 mostra um exemplo de um arquivo-texto contendo dois triângulos.

```
3  3  1
3  0  1
0  2  1
1  3  2
3  5  2
0  4  2
```

Figura A.19: Arquivo-texto contendo informação de dois polígonos.

A.2 Segundo Programa: Extração do Grafo

A implementação deste programa utiliza o sistema gráfico *OpenGL* para a exibição de resultados e conta adicionalmente com a utilização da biblioteca *CGAL*[†] para a implementação da Triangulação Restrita de Delaunay. O programa lê um arquivo texto que pode ser fornecido pelo primeiro programa e constrói o grafo. Como a extração do grafo é automática, não foi desenvolvida uma interface com o usuário para este programa.

Após a leitura do arquivo, a triangulação é construída e todos os procedimentos descritos na seção 4.3 são executados, resultando em um grafo que é exibido juntamente com os contornos dos quarteirões em uma tela semelhante à tela de exibição de resultados do programa 1. A Figura A.20 mostra como é feita a exibição de resultados.

[†]*Computational Geometry Algorithms Library*

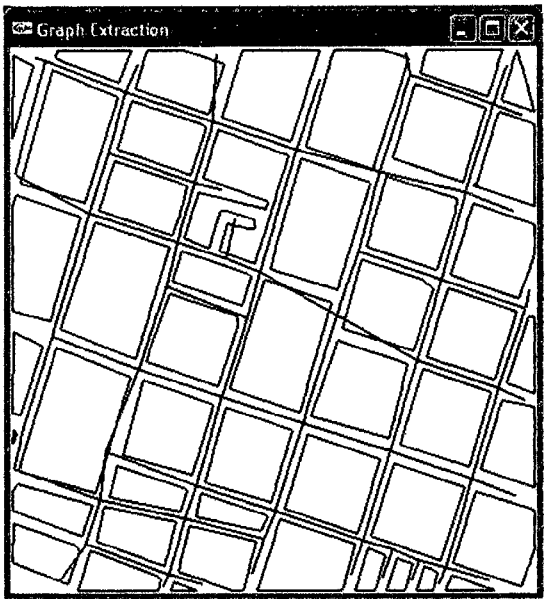


Figura A.20: Tela de exibição de resultados para o programa de extração do grafo.

Apêndice B

Arquivos PPM

Arquivos PPM* são arquivos de imagem em um formato que torna fácil a implementação de operações de leitura e manipulação de seu conteúdo.

Um arquivo PPM consiste de duas partes: um cabeçalho e os valores dos *pixels* que compõem a imagem. O cabeçalho do arquivo consiste de três partes, normalmente separadas por caracteres de nova linha ou espaços em branco. As informações contidas em cada uma dessas três partes são descritas a seguir:

- um identificador, denominado *número mágico*. Este número é composto de dois caracteres, podendo ser P3 ou P6;
- a largura, seguida pela altura da imagem. Estas duas informações são separadas por um espaço em branco;

*PPM significa *Portable Pixel Map*

- o valor máximo que um *pixel* pode assumir na imagem.

Linhas de comentário podem ser inseridas no cabeçalho e são identificadas por inciarem com o caractere #. Um exemplo de cabeçalho pode ser visto na Figura B.1.

```
P3
#Arquivo Exemplo
4  4
15
```

Figura B.1: Exemplo de cabeçalho de um arquivo PPM.

A segunda parte do arquivo contém os valores R, G e B dos pixels da imagem. Se uma imagem possui *largura* × *altura* pixels, então o arquivo conterá 3 × *largura* × *altura* caracteres nesta parte do arquivo. A Figura B.2 mostra um exemplo do conteúdo de um arquivo PPM completo.

```
P3
 4   4
15
 0   0   0   0   0   0   0   0   0  15   0  15
 0   0   0   0  15   7   0   0   0   0   0   0
 0   0   0   0   0   0   0  15   7   0   0   0
15   0  15   0   0   0   0   0   0   0   0   0
```

Figura B.2: Exemplo de conteúdo de um arquivo PPM.

Como podemos ver na Figura B.2 o *número mágico* utilizado foi o P3. Esse número indica que o conteúdo do arquivo está em formato texto. Arquivos PPM identificados com o *número mágico* P6 são arquivos binários.

Os arquivos binários são menores e a operação de leitura é bem mais rápida. O valor máximo possível para um PPM do tipo P6 é 255.

Nosso sistema foi desenvolvido para ler e exibir as imagens neste formato. Imagens fornecidas em diferentes formatos podem facilmente ser convertidas através de algum *software* utilitário específico para a conversão de imagens. As imagens testadas em nosso sistema vieram originalmente no formato *bitmap* e convertidas usando um *software* comercial.

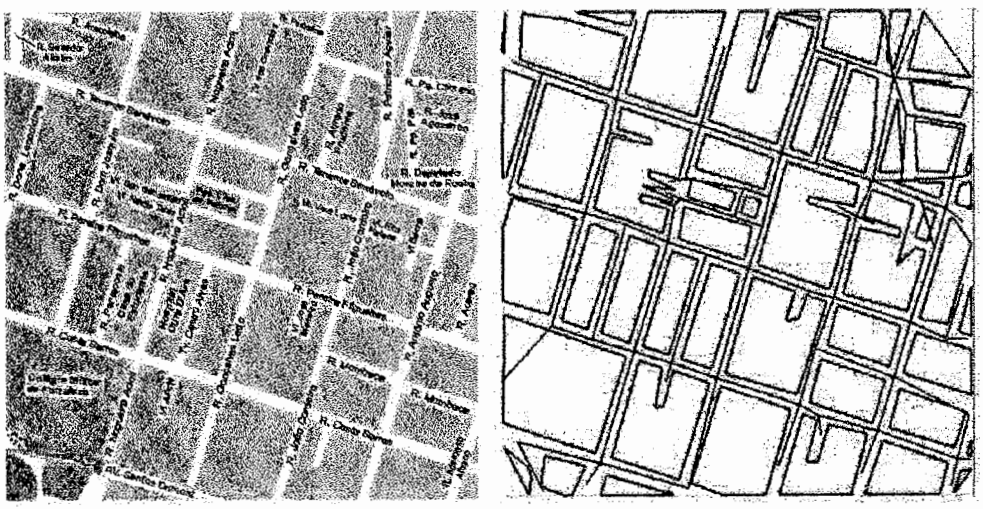
Há outros tipos de arquivos da família PPM. São eles:

- PBM - *Portable BitMap*: formato para imagens monocromáticas.
- PGM - *Portable GrayMap*: formato para imagens em tons de cinza.

Apêndice C

Mapas utilizados nos testes

Apresentamos neste Apêndice as imagens utilizadas nos testes realizados, bem como os resultados obtidos.



(a) (b)

Figura C.1: (a)Mapa m10 e em (b) o resultado obtido.

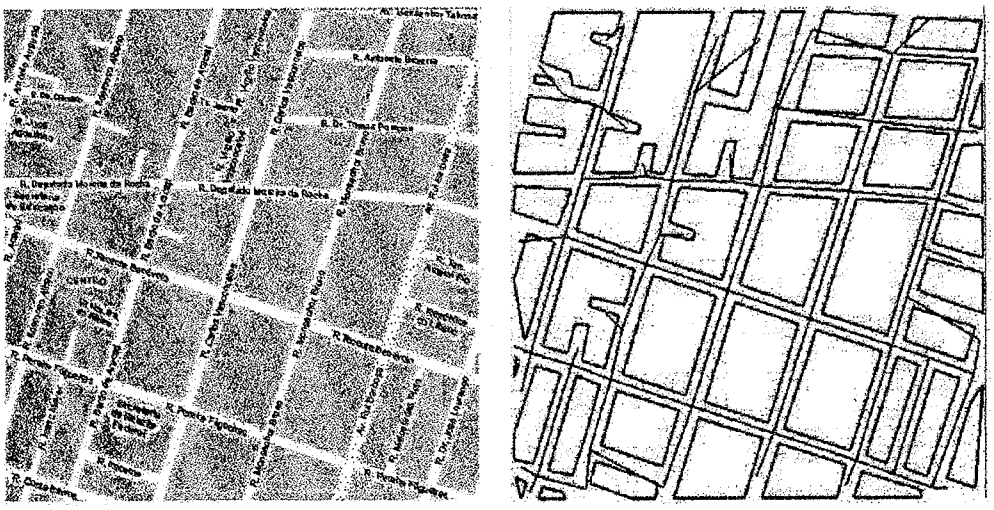


Figura C.2: (a)Mapa m11 e em (b) o resultado obtido.

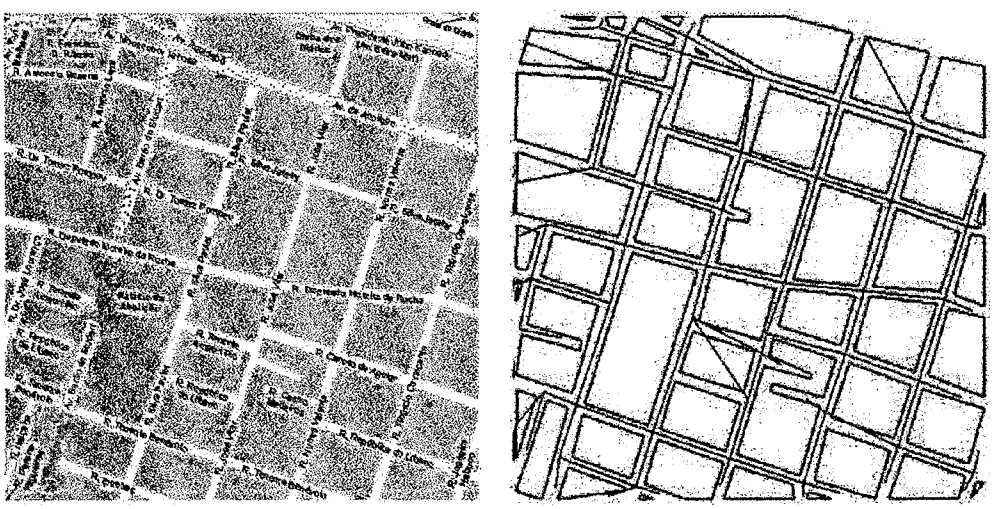


Figura C.3: (a)Mapa m12 e em (b) o resultado obtido.

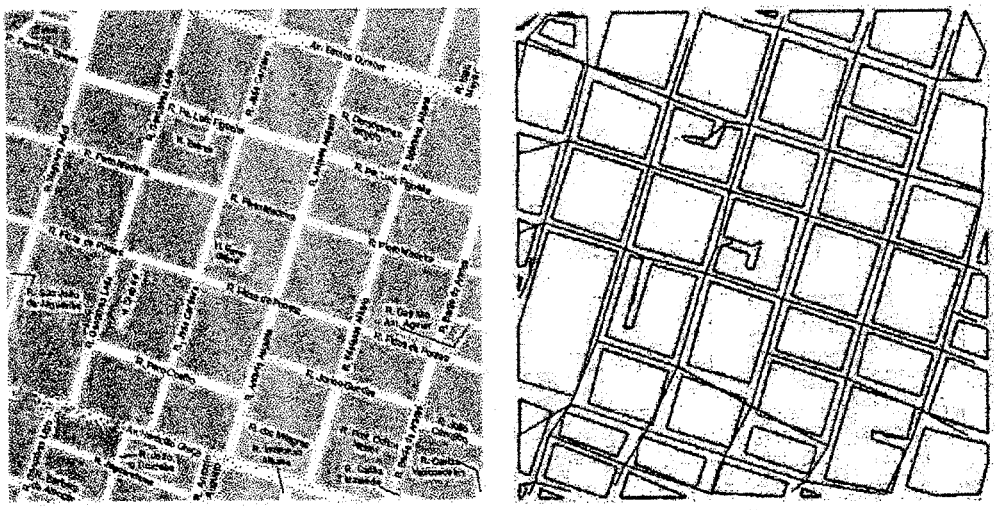


Figura C.4: (a)Mapa m20 e em (b) o resultado obtido.

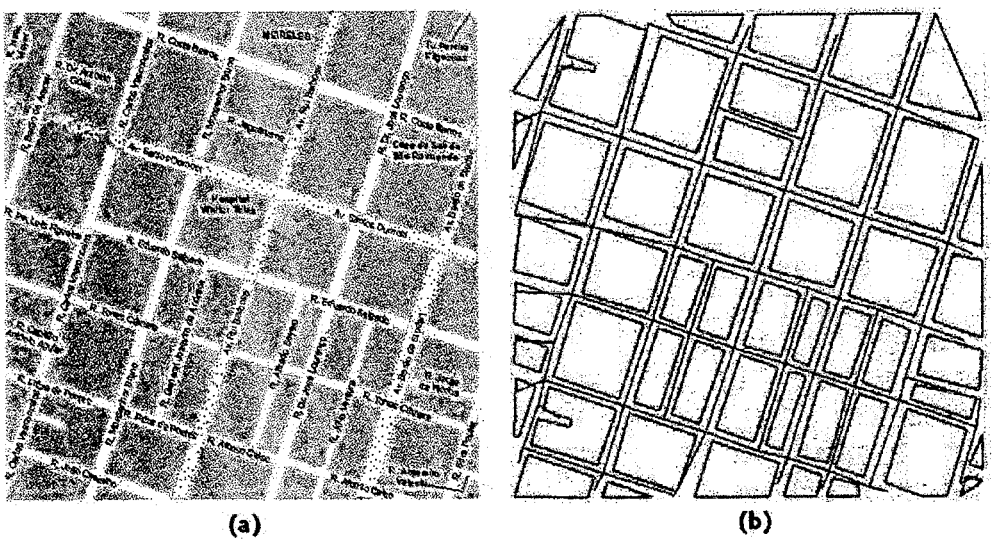


Figura C.5: (a)Mapa m21 e em (b) o resultado obtido.

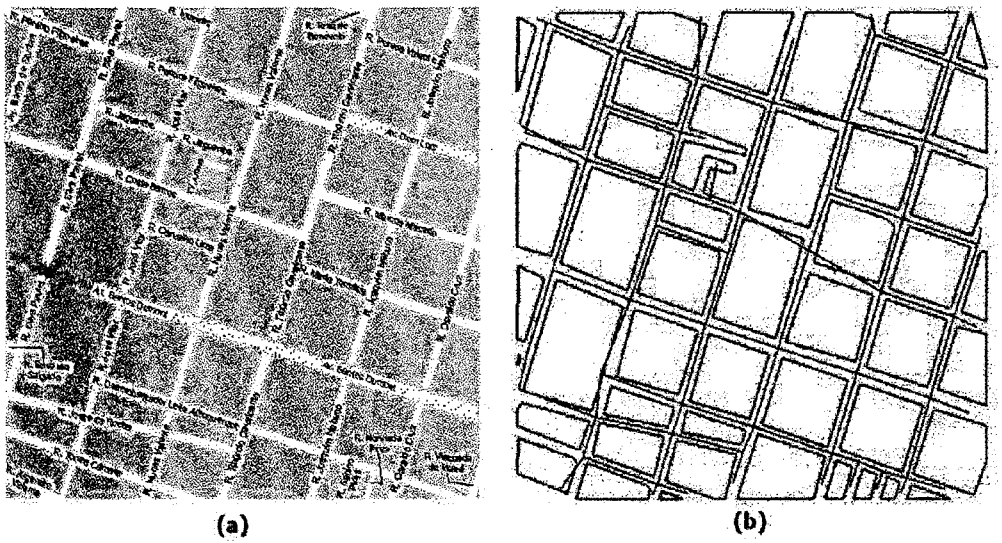


Figura C.6: (a)Mapa m22 e em (b) o resultado obtido.

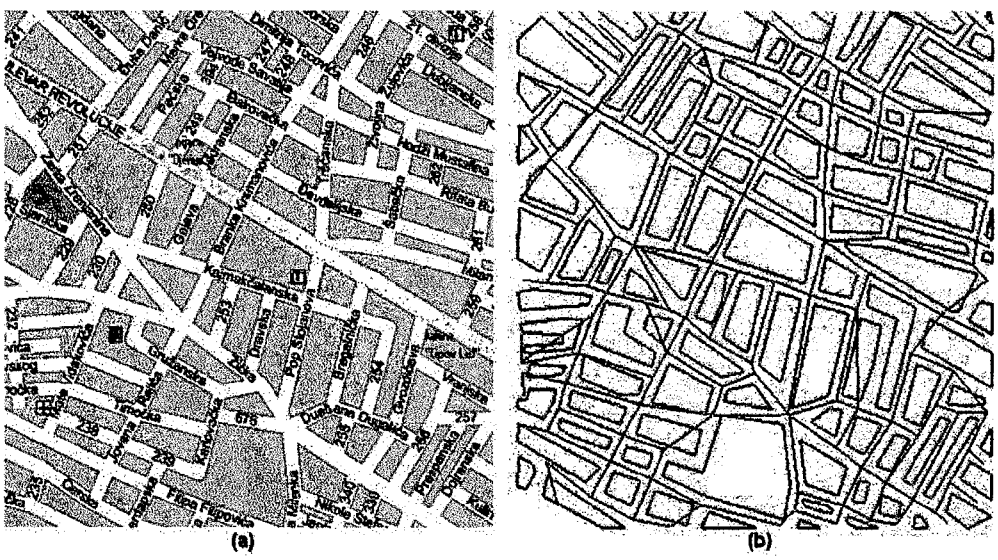


Figura C.7: (a)Mapa testmap01 e em (b) o resultado obtido.

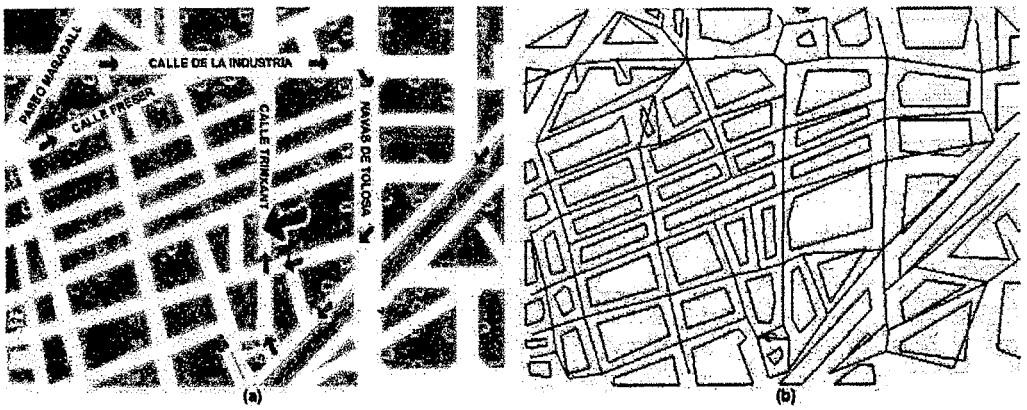


Figura C.8: (a)Mapa testmap02 e em (b) o resultado obtido.

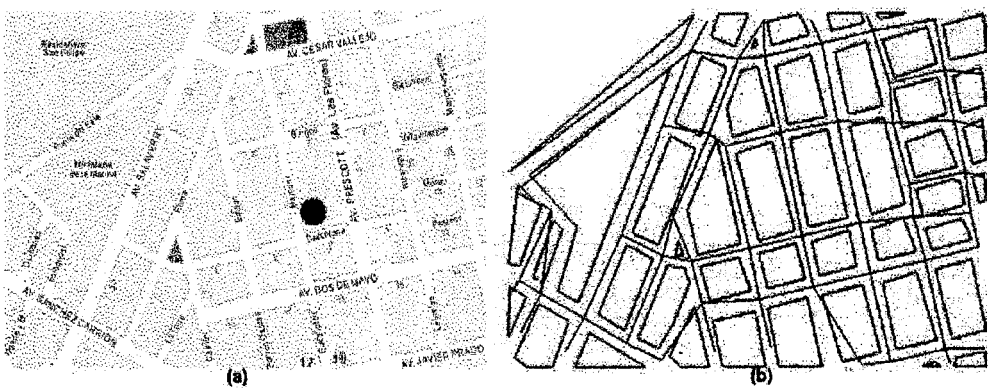


Figura C.9: (a)Mapa testmap03 e em (b) o resultado obtido.

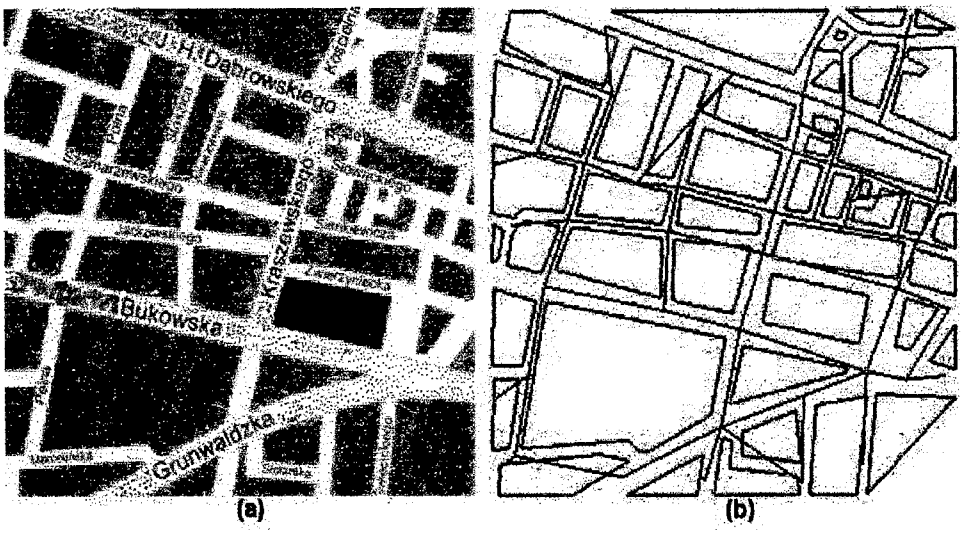


Figura C.10: (a)Mapa testmap04 e em (b) o resultado obtido.

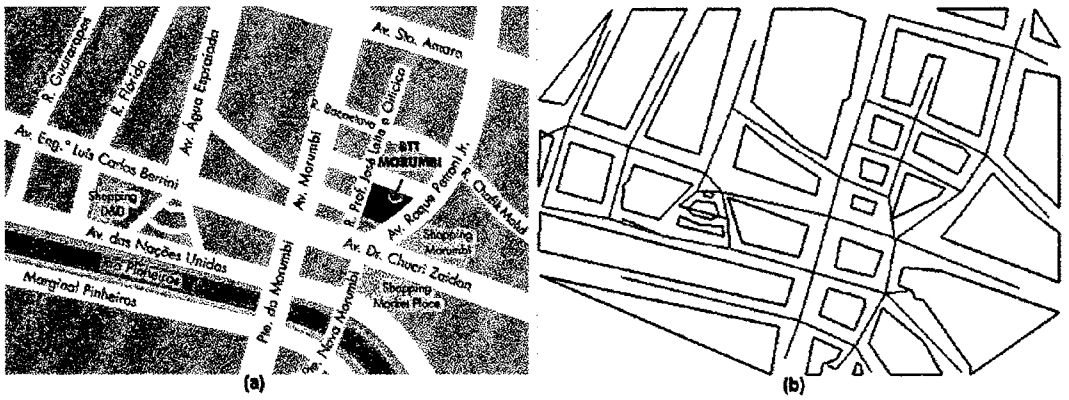


Figura C.11: (a)Mapa testmap05 e em (b) o resultado obtido.

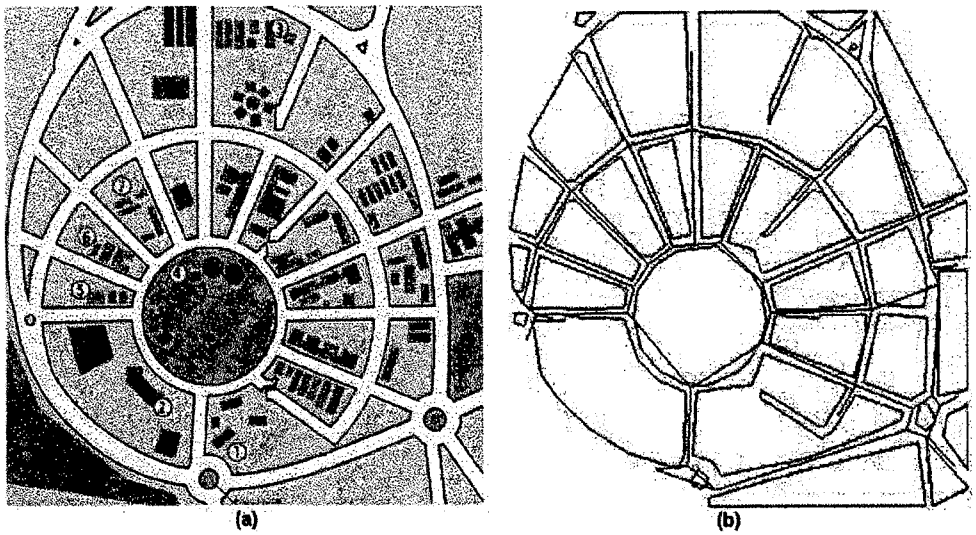


Figura C.12: (a)Mapa testmap06 e em (b) o resultado obtido.

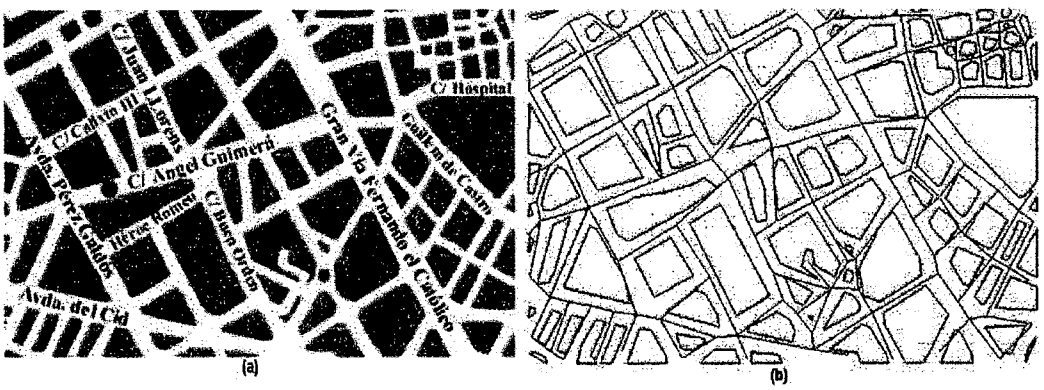


Figura C.13: (a)Mapa testmap07 e em (b) o resultado obtido.

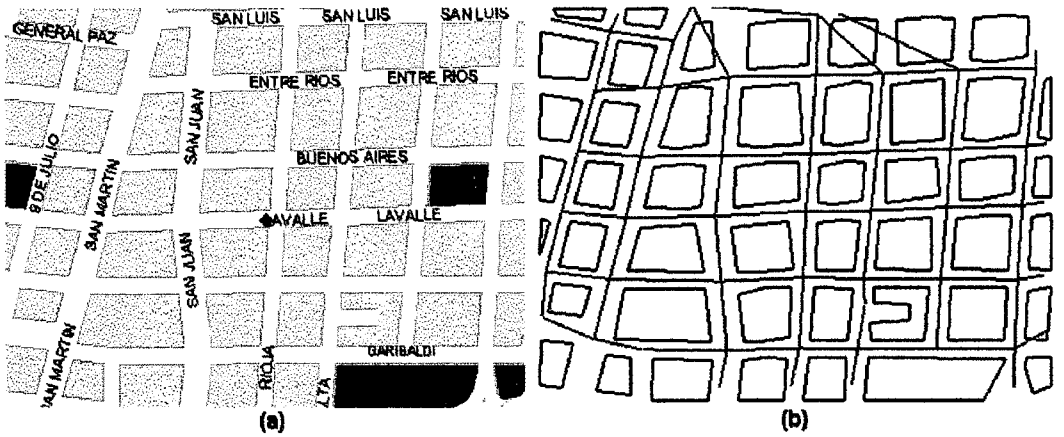


Figura C.14: (a)Mapa testmap08 e em (b) o resultado obtido.

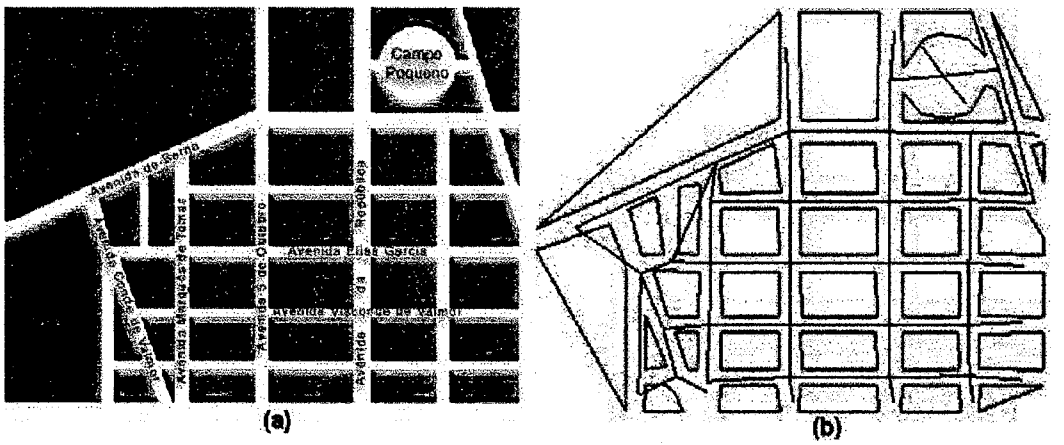


Figura C.15: (a)Mapa testmap09 e em (b) o resultado obtido.

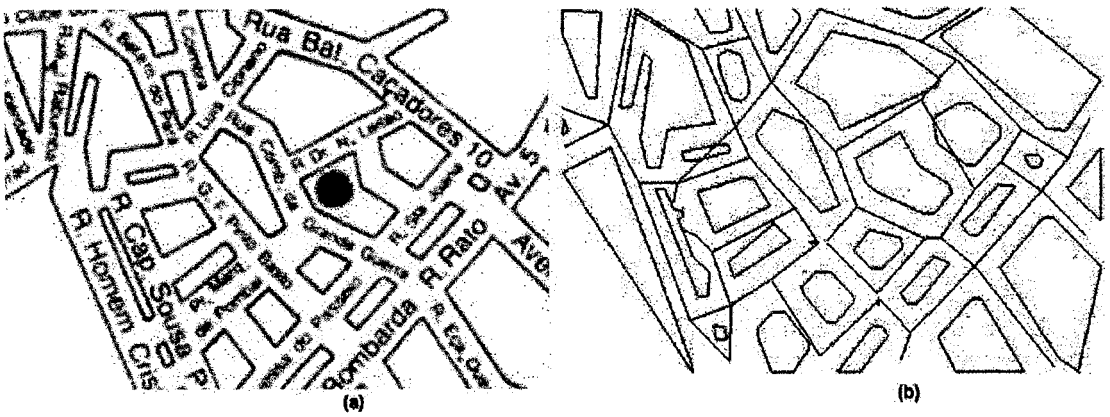


Figura C.16: (a)Mapa testmap10 e em (b) o resultado obtido.

Referências Bibliográficas

[1] Ahuja, R.K., Magnanti, T.L., Orlin, J.B. *Network flows. Theory, algorithms and applications.* Prentice-Hall, New-York, 1993.

[2] Aichholzer, Oswin, Aurenhammer, Franz, Alberts, David, Gärtner, Bernd. "A Novel Type of Skeleton for Polygons". *JUCS: Journal of Universal Computer Science*, v. 12, pp. 752–761, 1995.

[3] Amato, Nancy M. "Determining the Separation of Simple Polygons". *International Journal of Computational Geometry and Applications*, v. 4, n. 4, pp. 457–474, 1994.

[4] Baumgartner, A., Steger, C., Mayer, H., Eckstein, W. "Semantic Objects and Context for finding Roads". *In Proc. SPIE Conf. on Integrating Photogrammetric Techniques with scene Analysis and Machine Vision (SPIE3072)*, pp. 98–109, 1997.

[5] Borghys, D., Perneel, C., Nyssen, E., Acheroy, M. "Road Detection on Digitized Maps". *Proc. Int. Symp. on Pattern Recognition "In memoriam Pierre Devijver", Brussels, Belgium*, 1999.

[6] C., Heipke, A., Englisch, S., Stier, R, Kutka. "Semi-Automatic Extraction of Roads from Aerial Images". *In Proc. ISPRS Commission III Symposium: Spatial Information from Digital Photogrammetry and Computer Vision*, pp. 353–360, 1994.

[7] Caliari, Cinthia Cristina Lúcio. *Identificação de Atributos de Grafos em Imagens de Mapas Urbanos.* Master's thesis, Universidade Federal do Espírito Santo - UFES, Vitória - ES, Brasil, 2000.

[8] Cormen, Leiserson e Rivest. *Introduction to Algorithms.* MIT Press, 1990.

[9] Gonzales, Rafael, Woods, Richard. *Digital Image Processing.* Addison-Wesley, 1993.

[10] Gruen, A., Li, H. "Semi-Automatic Road Extraction by Dinamic Programming". *In Proc. ISPRS Commission III Symposium: Spatial Information from Digital Photogrammetry and Computer Vision*, pp. 324–332, 1994.

[11] H., Douglas D., K, Peucker T. "Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or its Caricature". *Canadian Catographer*, pp. 112–122, 1973.

[12] Hershberger, John, Snoeyink, Jack. "Speeding Up the Douglas-Peucker Line-Simplification Algorithm". *Proceedings of the 5th International Symposium on Spatial Data Handling*, v. 1, pp. 134–143, 1992.

[13] Hinz, S., Baumgartner, A., Steger, C., Mayer, H., Eckstein, W., Ebner, H., Radig, B. "Road Extraction in Rural and Urban Areas". *In Proc. Semantic Modeling for the Aquisition of Topographic Information from Images and Maps (SMATI)*, pp. 133–153, 1999.

[14] J., Austin, M., Brown, S., Buckle, I, Kelly. "ADAM Neural Networks for Parallel Vision". *Advanced Computer Architecture Group*, 1993.

[15] Janssen, Rik D. T., Vossepoel, Albert M. "Adaptive Vectorization of Line Drawing Images". *Computer Vision and Image Understanding*, v. 65, pp. 38–56, 1997.

[16] Kass, M., Witkin, A., Terzopoulos, D. "Snakes: Active Contour Models". *International Journal of Computer Vision*, pp. 321–331, 1988.

[17] Katartzis, A., Pizurica, V., Sahli, H. "Application of mathematical morphology and Markov random field theory to the automatic extraction of linear features in airborne images". *In Proc. International Symposium on Mathematical Morphology and its Applications to Image and Signal Processing V, California, USA*, pp. 405–414, 2000.

[18] M-F., Auclair Fortier, D., Ziou, C., Armenakis, S, Wang. *Survey of Work on Road Extraction in Aerial and Satellite Images.* Technical report, Département de mathématiques et informatique - Université de Sherbrooke and Center of Topographic Information - Geomatics Canada, Québec, Canda, J1K2R1 and Ontario, Canada, K1A0E9, 2000.

[19] Mantyla, M. *Geometric and Solid Modeling: an Introduction.* Computer Science Press, 1988.

[20] Moissinac, H., Maître, H., Bloch, I. "Urban Aerial Image Understanding Using Symbolic Data". *In Proc. Image and Signal Processing for Remote Sensing, SPIE 2315*, pp. 310–321, 1994.

[21] Moissinac, H., Maître, H., Bloch, I. "Graph Based Urban Scene Analysis Using Symbolic Data". *In Proc. Integrating Photogrammetric Techniques with Scene Analysis and Machine Vision II, SPIE 2486*, pp. 93–104, 1995.

[22] Nagy, G., Samal, A., Seth, S., Fisher, T., Guthmann, E., Kalafala, K., Li, L., Sivasubramaniam, S., Xu, Y. *Reading Street Names from Maps - Technical Challenges.* Technical report, Renssealer Polytechnic Institute - University of Nebrasca-Lincoln, New York 12180 and Lincoln, Nebrasca 68588, USA, 1997.

[23] Price, K. "Road Grid Extraction and Verification". *In International Archives of Photogrammetry and Remote Sensing*, v. 32, pp. 101–106, 1999.

[24] R., Ogniewicz, M, Ilg. "Voronoi skeletons: Theory and Applications". *In Proc. IEEE Conf. Computer Vision and Pattern Recongnition, CVPR*, pp. 63–69, 1992.

[25] R., Ruskoné, S., Airault, O, Jamet. "Road Network Interpretation: a Topological Hypothesis driven System". *In Proc. ISPRS Commission III Symposium: Spatial Information from Digital Photogrammetry and Computer Vision*, pp. 711–717, 1994.

[26] R., Ruskoné, S., Airault, O., Jamet, L, Guigues. "Vehicle Detection on Aerial Images: A Structural Approach". *In Proc. International Conference on Pattern Recognition, ICPR'96*, pp. 900–904, 1996.

[27] Shneiderman, Ben. *Designing the User Interface : Strategies for Effective Human-Computer Interaction.* Addison-Wesley, 3 ed., 2002.

[28] X., Décoret, F, Sillion. "Street Generation for City Modelling". *In Proc. First International Workshop on Architectural and Urban Ambient Environment*, 2002.