INvestigate and Analyse a City - INACITY

Artur André Almeida de Macedo Oliveira

DISSERTAÇÃO APRESENTADA
AO
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
DA
UNIVERSIDADE DE SÃO PAULO
PARA
OBTENÇÃO DO GRAU DE MESTRE
EM
CIÊNCIA DA COMPUTAÇÃO

Área de Concentração: Ciência da Computação

Orientador: Prof. Dr. Roberto Hirata Junior

São Paulo, maio de 2018

INvestigate and Analyse a City - INACITY

Esta versão da dissertação contém as correções e alterações sugeridas pela Comissão Julgadora durante a defesa da versão original do trabalho, realizada em 23/04/2018. Uma cópia da versão original está disponível no Instituto de Matemática e Estatística da Universidade de São Paulo.

Comissão Julgadora:

- Prof. Dr. Roberto Hirata Junior (orientador) - IME-USP
- Prof. Dr. Alfredo Goldman Vel Lejbman - IME-USP
- Prof. Dr. Marcos Silveira Buckeridge - IB-USP

Agradecimentos

Agradeço à minha família e ao meu orientador Roberto Hirata Junior por todo o suporte ao longo destes anos sem eles este projeto nunca teria-me sido possível.

Resumo

OLIVEIRA, A. A. A. M. **INvestigate and Analyse a City - INACITY**. 2018. 136 p. Dissertação (Mestrado) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2018.

Este trabalho apresenta uma plataforma para coleta e análise de imagens urbanas, que integra Interfaces de Programação de Aplicativos "Application Programming Interfaces" (APIs) de sistemas de busca de imagens, Sistemas de Informações Geográficas (SIGs), mapas digitais e técnicas de visão computacional. Esta plataforma, INACITY, permite que usuários selecionem regiões de interesse e capturem elementos de relevância para a arquitetura urbana, como, por exemplo árvores e buracos em ruas. A implementação da plataforma foi feita de maneira a permitir que novos módulos possam ser facilmente incluídos ou substituídos possibilitando a introdução de outras APIs de mapas, SIGs e filtros de Visão Computacional. Foram realizados experimentos com as imagens obtidas através do "Google Street View" onde árvores são capturadas em áreas de bairros inteiros em questão de minutos, um ganho significativo quando comparado com o procedimento manual para levantamento deste tipo de dado. Além disso, também são apresentados resultados comparativos entre os métodos de visão computacional propostos para a detecção de árvores em imagens com outros métodos heurísticos, em um conjunto onde as árvores estão marcadas manualmente e assim as taxas de precisão e de redescoberta de cada algoritmo podem ser avaliadas e comparadas.
Palavras-chave: cidades inteligentes, visão computacional, vegetação urbana, Google Street View.

Abstract

OLIVEIRA, A. A. A. M. **INvestigate and Analyse a City - INACITY**. 2018. 136 f. Dissertação (Mestrado) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2018.

This project presents a platform that integrates Application Programming Interfaces (APIs), image retrieval systems, Geographical Information Systems (GISes), digital maps and Computer Vision techniques to collect and analyse urban images. The platform, INACITY (an acronym for INvestigate and Analyse a City), empowers users allowing them to select a region over a map and see urban features inside that region that have relevance to the urban architecture context, for instance trees. The implementation is extensible and it is designed to make it easy to add or replace new modules, for instance, to add a new API to present a map, different GISes and other Computer Vision filters.

To test the platform, a series of experiments were conducted to segment trees on some images obtained using Google Street View (GSV) platform. The amount of time to collect and analyse the images from a whole neighborhood can not be compared to the time and resources spent in field audits. Besides that, using precision and recall measures, comparative results of different methods proposed to detect greenery in images are shown for a dataset where the trees were segmented manually.

Keywords: smart cities, computer vision, urban greenery, Google Street View.

Sumário

Capítulo 1

Introdução

A motivação deste projeto reside no trabalho de Wilson e Kelling [WK82] conhecido como a Teoria das Janelas Quebradas onde os autores procuram demonstrar a correlação que existe entre desordem urbana (por ex. janelas quebradas, lixo nas ruas, muros degradados, etc.) e criminalidade. Alguns artigos, voltados à inspeção de vizinhanças urbanas, foram inspirados no trabalho de Wilson e Kelling com relação a formação de seu inventário (elementos que são averiguados durante as inspeções das vizinhanças) [CFM03, RBR+11, NPRH14]. Ainda de acordo com Wilson e Kelling [WK82] um dos meios de melhor alocar a força policial seria através de indicadores não só de violência e crimes mais severos mas também de indicadores de onde a força policial poderia evitar o surgimento e propagação destes crimes. Atualmente exitem plataformas como o "Google Street View" (GSV) [Incc] que permitem a coleta de imagens urbanas que podem ser usadas em estudos voltados à análise de indicadores urbanos relacionados à diversos fatores sociais [RBR+11].

Plataformas como o "Google Street View" (GSV) [Incc] disponibilizam meios para a visualização de imagens no nível do solo sobre todos os continentes do planeta, além desta existem outras plataformas como a do "Mapillary" [Map] que é formada através de contribuição coletiva e conta com mais de 170 milhões de imagens também distribuídas sobre os 7 continentes. Somando-se às plataformas de imageamento também existem as de mapeamento como o "Google Maps" [Inca] da empresa "Google" e o "OpenStreetMaps" [Ope17] formado por colaboração coletiva. Atualmente os serviços de imageamento no nível do solo pela empresa "Mapillary" são disponibilizados gratuitamente até um limite de 50 mil ima-

gens no total (sem renovação deste limite). Os serviços disponibilizados gratuitamente pela "Google" são um pouco menos limitados dado que permite a obtenção gratuita de até 25 mil imagens por dia (sendo renovado o limite no dia seguinte), contudo a resolução das imagens não é suficiente para análises como a de rachaduras no asfalto ou mesmo a classificação da vegetação baseada por exemplo no formato das folhas. Já um contrato não gratuito disponibiliza imagens com resoluções até 4 vezes maior. Além disso, de acordo com as restrições de uso impostas pela empresa "Google" [Incb] ao usar seus serviços o usuário não pode armazenar e disponibilizar (sem o intermédio do serviço do GSV) as imagens obtidas pelo GSV.

1.1 Objetivos

Neste projeto buscamos criar uma plataforma de software de análise e visualização de características urbanas. Além disso a plataforma deveria ser escalável de maneira a permitir que outros componentes como plataformas de imageamento, Sistemas de Informações Geográficas (SIGs), mapas e detectores de características urbanas pudessem ser facilmente acoplados à plataforma e disponibilizados como serviços através da implementação da plataforma em um servidor de computação na nuvem. Além de se projetar a plataforma também foi tido como objetivos o estudo, implementação e teste de diversos filtros de imagens para extração de características urbanas.

1.2 Contribuições

A plataforma INACITY é um primeiro passo no sentido de se explorar a integração entre plataformas de imageamento urbano, Sistemas de Informações Geográficas e Visão Computacional e suas contribuições são voltadas tanto para as áreas de Ciência da Computação, mais especificamente Visão Computacional e Sistemas de Computação, como para as áreas de Urbanismo e Cidades Inteligentes. Esta plataforma oferece aos cidadãos, administradores urbanos e demais interessados uma ferramenta capaz de facilitar a busca por características de relevância urbana e ao mesmo tempo traz aos pesquisadores e desenvolvedores uma

Interface de Desenvolvimento de Aplicações (Application Development Interface - API) na forma de um "middleware" para coleta e processamento de imagens. Além disso, através de padrões de projetos oriundos da área de Sistemas Computacionais a implementação da plataforma permite que suas funcionalidades sejam estendidas com facilidade através da inclusão de novos componentes para integração com outras plataformas de imageamento, SIGs e filtros de imagens. Neste trabalho é explorada a possibilidade de coletar imagens através da plataforma "Google Street View", processar essas imagens através de filtros de processamento de imagens e exibir os resultados ao usuário, tanto na forma das imagens filtradas como na forma de um mapa de calor, isso é, a visualização da concentração de uma dada característica urbana sobre uma região delimitada pelo usuário através da coloração de um mapa cartográfico. Os esforços com relação aos filtros de imagens foram focados na detecção de árvores e vegetação. Os resultados são então comparados com outras técnicas baseadas em heurísticas de Visão Computacional e outra contribuição é a disponibilização um novo "groudtruth" para a comparação de algoritmos detectores de vegetação em imagens. A implementação da plataforma foi hospedada no servidor de computação na nuvem Azure [Cor] e o projeto foi apresentado durante a escola de verão "São Paulo School of Advanced Science on Smart Cities" [IU].

1.3 Organização

Este trabalho está organizado da seguinte forma: no capítulo 2 é feita uma revisão bibliográfica de diversos artigos que exploram a correlação e a importância de fatores urbanos na saúde e até no comportamento de uma população, o que são cidades inteligentes, outros sistemas de apoio à estas, sendo alguns destes sistemas baseados em visão computacional e aprendizado de máquina. O capítulo 3 apresenta alguns conceitos das áreas de Sistemas de Computação e Visão Computacional. Com relação à primeira área, são revistos os principais conceitos usados no desenvolvimento da plataforma como o padrão de arquitetura "Model-View-Controller"(MVC) e as principais técnicas de morfologia matemática usadas para o desenvolvimento do filtro de vegetação são brevemente descritos. O Capítulo 4 apresenta a arquitetura de software proposta para a plataforma INACITY, diagramas de casos

de uso exemplificando seu funcionamento assim como algumas demonstrações das imagens filtradas e dos mapas de calor indicando a concentração de árvores sobre uma região urbana compreendendo diversos bairros. O Capítulo 5 apresenta as heurísticas para segmentação de imagens para a detecção de árvores que foram usadas no trabalho, assim como detalhes de suas respectivas implementações. O Capítulo 6 apresenta o conjunto de imagens e os experimentos feitos para validar as heurísticas de segmentação. Esse conjunto tem 100 imagens do Google Street View que foram escolhidas heuristicamente de um conjunto de ruas da cidade de São Paulo e as áreas de vegetação foram segmentadas manualmente. O conjunto também é uma contribuição do projeto e será disponibilizado no endereço do projeto.

Capítulo 2

Revisão Bibliográfica

Neste capítulo revisamos alguns artigos da literatura. Primeiramente trataremos dos artigos relacionados a análise de características de um bairro. Em seguida, revisamos artigos relacionados a cidades inteligentes e sistemas de apoio a elas. Terminamos o capítulo revisando artigos da área de análise de imagens relacionadas com o contexto urbano.

2.1 Auditoria de bairros

No artigo Janelas Quebradas de Wilson e Kelling [WK82] é descrito um programa de 1970 do estado de Nova Jersey nos Estados Unidos chamado "Safe and Clean Neighborhoods Program" onde uma parte da força policial seria alocada para patrulhamentos a pé. Os autores explicam que apesar das taxas de criminalidade não terem diminuído, ainda assim a sensação de segurança percebida pelos cidadãos era maior nas regiões onde policiais faziam as patrulhas a pé do que nas demais regiões. De acordo com os autores, as patrulhas a pé facilitaram mais o contato com os cidadãos do que as feitas com veículos. Este maior contato reforçou a confiança dos policiais nos cidadãos regulares da região onde foram alocados e vice-versa. A conclusão dos autores é de que evitar a deterioração de uma vizinhança é tão importante (se não mais) quanto combater a criminalidade e para isto é essencial que a polícia tenha meios para conseguir saber onde sua presença é mais necessária uma vez que seus recursos são limitados.

No estudo de Rundle, A. et al. (2011) [RBR+11] são comparadas auditorias feitas à bairros

presencialmente, ou remotamente, através da plataforma "Google Street View" (GSV). Este estudo compara os resultados obtidos por auditores que averiguaram diversas características correlacionadas com fatores da área da saúde, sócio-econômicos e até sociais, e concluem que a auditoria remota tem resultados comparáveis com as auditorias realizadas presencialmente. Vários estudos foram feitos a fim de validar a correlação entre características observadas pelos auditores em bairros de diversas cidades e outras características, como no artigo de Cohen et al. (2003) [CFM03] onde é demonstrado que algumas causas de mortalidade prematura, ou seja morte antes dos 65 anos, como homicídio e doença cardiovascular estão relacionadas ao estado socio-econômico baixo de uma região. No artigo de Agyemang et al. (2007) [AvHWV+07] é demonstrado que as respostas marcadas entre razoável e ruim a questionários de saúde auto-avaliada, isso é, uma avaliação realizada na forma de um questionário respondido pelo cidadão avaliado, estão associadas a fatores de estresse psicossociais[1] encontrados em alguns bairros e vizinhanças. Em alguns outros artigos são associadas características observadas em diferentes regiões da cidade com dados associados a caminhadas e outras formas de atividade física [HRE+05], obesidade [BHD+07, GRS06, EMB05, Gra08, SCE+07], limitações da parte inferior do corpo [BK02, SAW+06], sintomas de depressão, ansiedade, e alterações de conduta [AS96, Kim08, LC03], asma [CB04, NR06] e crime e violência [CFM03, Sko90, SR99].

Ainda de acordo com Rundle, A. et al. (2011) [RBR+11] o estudo de ambientes de bairros traz algumas dificuldades relacionadas ao esforço empregado, custos, segurança da equipe empregada na pesquisa, e também com relação a variabilidade das respostas obtidas por cada membro da equipe, no caso de pesquisas feitas presencialmente. Um outro tipo de recurso disponível aos pesquisadores para se realizar este tipo de estudo são dados administrativos, obtidos a partir de orgãos do governo ou outras organizações sociais. Porém este tipo de levantamento normalmente está vinculado estritamente aos interesses da entidade que o fez no momento da aquisição dos dados, intermitência na disponibilidade dos dados, diferentes metodologias para se adquirir os dados entre diferentes jurisdições. Finalmente, alguns levantamentos, apesar de conterem uma grande gama de informações, não são completos e

[1] Fatores de estresse psicossociais são definidos no artigo como sendo: a presença de vizinhos incômodos, uso indevido de drogas, jovens frequentemente vagando pela região, lixo nas ruas, sensação de insegurança e insatisfação com o espaço verde

carecem de informações importantes [PNL+09].

No estudo de Rundle, A. et al. (2011) [RBR+11], a validação da plataforma GSV como um meio de se fazer a auditoria de uma vizinhança remotamente foi feita através da comparação de vizinhanças auditadas num estudo anterior de 38 quarteirões em Nova York e dos resultados obtidos pela auditoria destes mesmos locais através da observação das imagens obtidas pelo GSV. Os autores concluem que para estudos que se baseiam em qualidades temporais (por ex. ruído acústico, velocidade de travessia de faixas de pedestres e etc) o uso do GSV é inviável. Além disso, outras características com grande variabilidade temporal que indicam fatores de desordem social como, por exemplo, lixo nas ruas, possuem um baixo índice de correlação entre o estudo presencial e o feito remotamente. Contudo, para itens que tendem a ser imóveis e que não mudam frequentemente sua aparência com o tempo, foi encontrada uma alta correlação entre os dois tipos de auditoria (presencial e remota). Sendo assim, estes itens que mudam menos com o tempo são mais adequados para serem averiguados nas imagens. Neste sentido a auditoria de itens como árvores e vegetação similar através de uma plataforma como o GSV é preferível a uma auditoria feita presencialmente. Essa vantagem se deve ao fato de que vistorias presenciais são custosas em termos de tempo e recursos em função de custos de deslocamento.

2.2 Cidades inteligentes

De acordo com Annalisa Cocchia (2014) [Coc14] o conceito de Cidade Inteligente ("Smart City") possui diversos significados. Além disso diversos artigos relacionados a este tema abordam conceitos similares, porém, com nomenclaturas diferentes, dentre estes se encontram os termos: Cidade Inteligente ("Smart Cityöu "Inteligent City"), Cidade Digital ("Digital City"), Cidade do Conhecimento ("Knowledge City") e diversos outros. Cada um destes termos é relacionado aos contextos: tecnológico (mais especificamente de tecnologias de comunicação e informação), social, cultural, ou mesmo institucional ou governamental. De acordo com Annalisa Cocchia (2014) [Coc14] tornar uma cidade inteligente consiste em melhorar algum aspecto envolvendo algum dos contextos citados. Em diversos dos artigos citados no estudo de Cocchia, é explicitada a importância de se monitorar, detectar e comu-

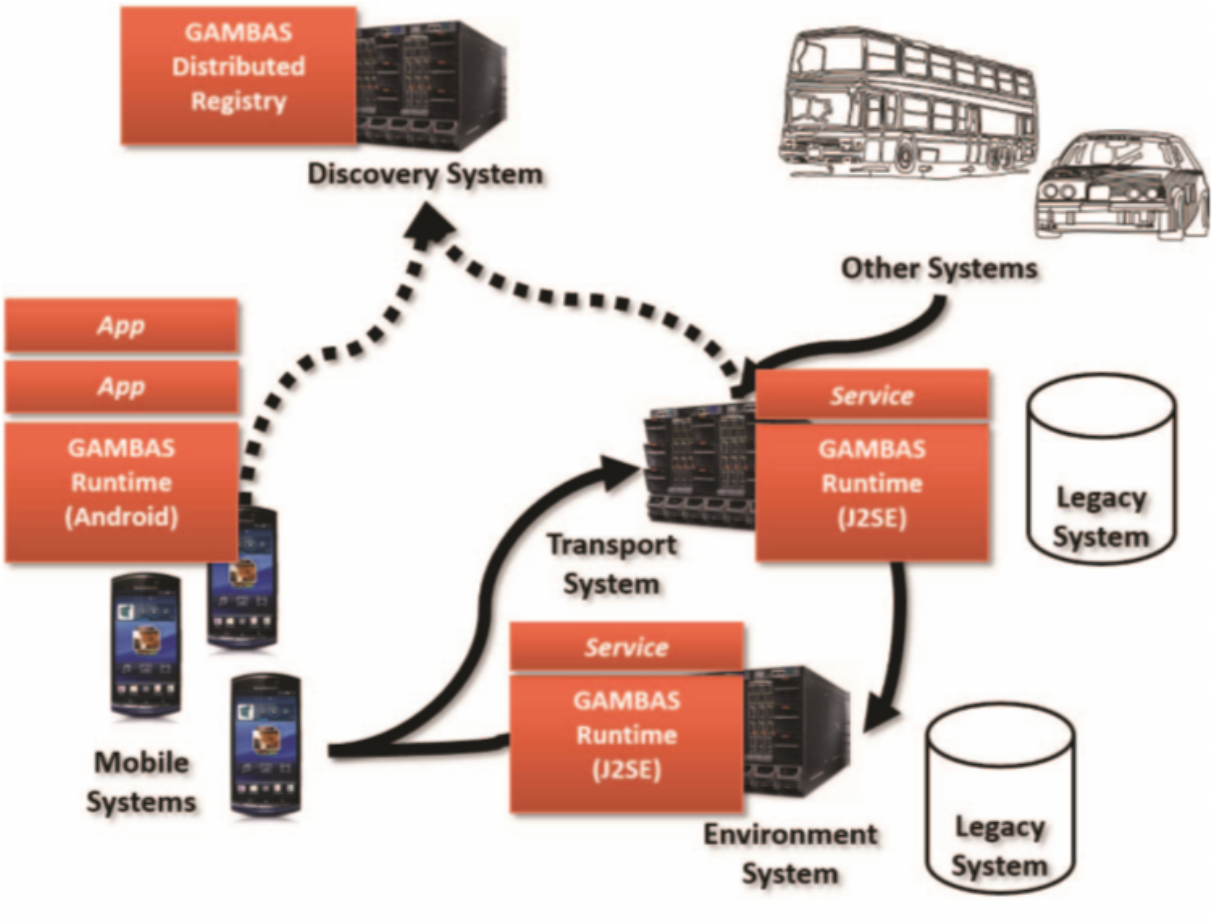
nicar características urbanas como infra-estrutura, água e energia, entre outros, para tornar decisões voltadas à melhoria da cidade mais eficazes.

2.3 Sistemas de apoio à cidades inteligentes

Baseando-se no artigo de Cocchia [Coc14], é razoável definir um sistema de apoio a cidades inteligentes como um sistema que auxilia uma cidade a atingir a definição de inteligente. Alguns destes sistemas de apoio à cidades inteligentes são "middlewares". De acordo com Bernstein, Philip A. (1996) [Ber96] o termo "middleware" representa um componente de software que se situa acima da camada do sistema operacional e de rede e abaixo da camada de aplicações específicas da indústria. É esperado de um "middleware" que nele esteja implementada uma interface de programação e protocolos padrões de maneira a possibilitar que diferentes sistemas possam interoperar, isso é, sejam capazes de acessar programas e enviar dados entre um e outro. Interoperabilidade somente é possível quando dois sistemas usam o mesmo protocolo, isto é, usam o mesmo formato e sequência de mensagens. Essa interoperabilidade permite que um sistema possa ser utilizado completamente por uma grande parte de outros sistemas e aplicações populares. O projeto GAMBAS de Apolinarski et al. (2014) [AIP14] é uma proposta de "middleware" que agrega dados de diversos dispositivos móveis (por ex. celulares Android, iOS, etc) através de aplicativos desenvolvidos com base em um Kit de Desenvolvimento de Sistemas "System Development Kit" (SDK) criado e disponibilizado também por Apolinarski et al. (2014) [AIP14]. A figura 2.1 extraída de Apolinarski et al. (2014) [AIP14] ilustra a arquitetura geral do SDK onde alguns dispositivos podem atuar como fornecedores e ou consumidores de dados e outros tem a finalidade de atuarem como parte da infraestrutura registrando e catalogando dispositivos disponíveis (sistema de descobrimento) ou ainda como servidores de armazenamento de dados. O SDK desenvolvido pelos autores permite que desenvolvedores criem aplicativos para plataformas móveis e usufruam dos dados coletados por outras aplicações que também façam uso do mesmo SDK por meio do "middleware" GAMBAS.

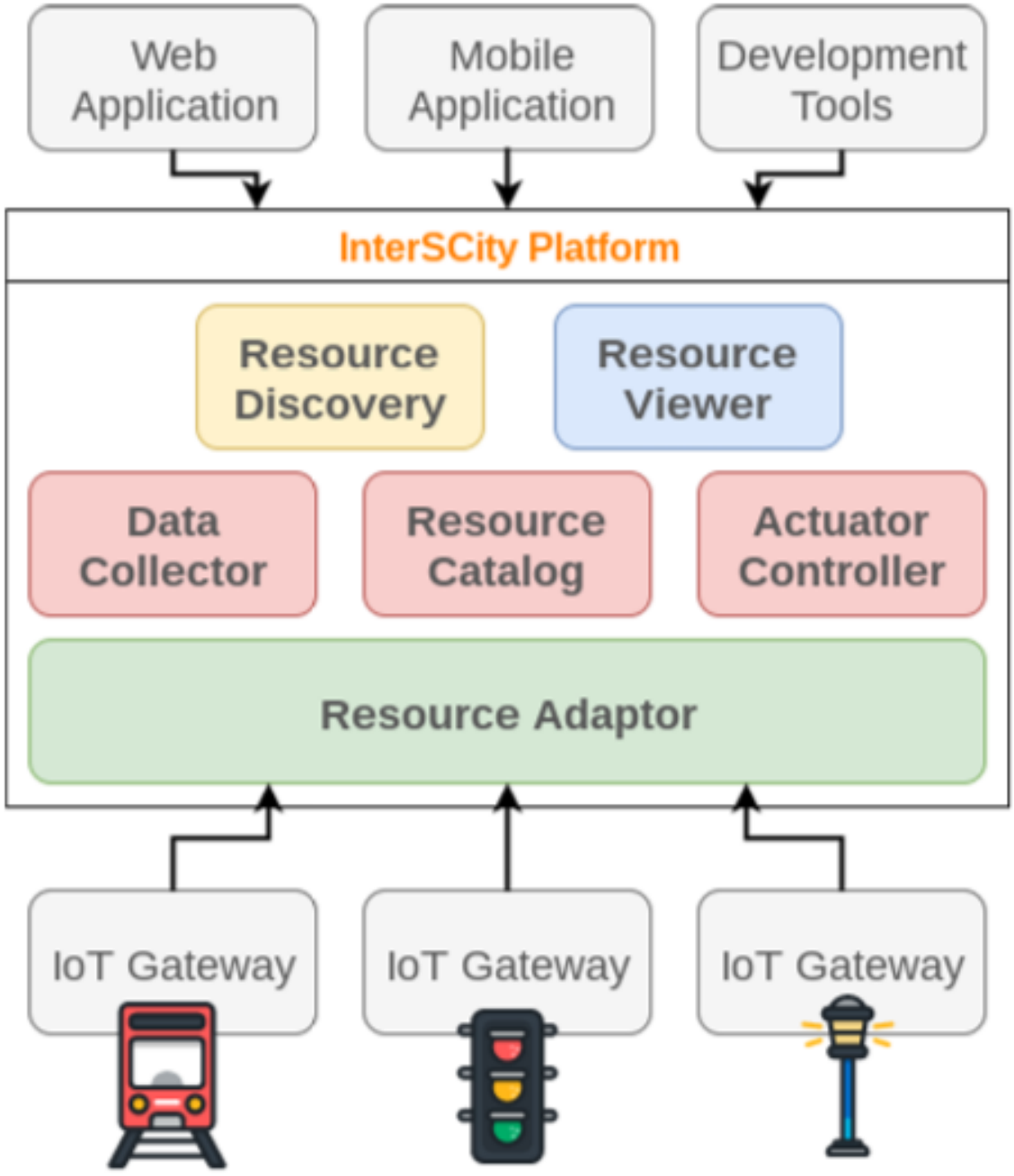
Esposte et al. (2017) [EKCL17] propõe no contexto do projeto InterSCity, descrito por Batista et al. (2016) [BGH+16], uma plataforma baseada em microserviços que permite a

Figura 2.1: *Arquitetura geral do projeto GAMBAS extraída de Apolinarski et al. (2014) [AIP14]. Nesta figura o sistema de descobrimento ("Discovery System") é responsável por manter e disponibilizar um registro dos dispositivos que usam o SDK. Estes dispositivos podem atuar como fornecedores (por ex. sensores) ou consumidores (por ex. aplicativos de mapas) de dados.*



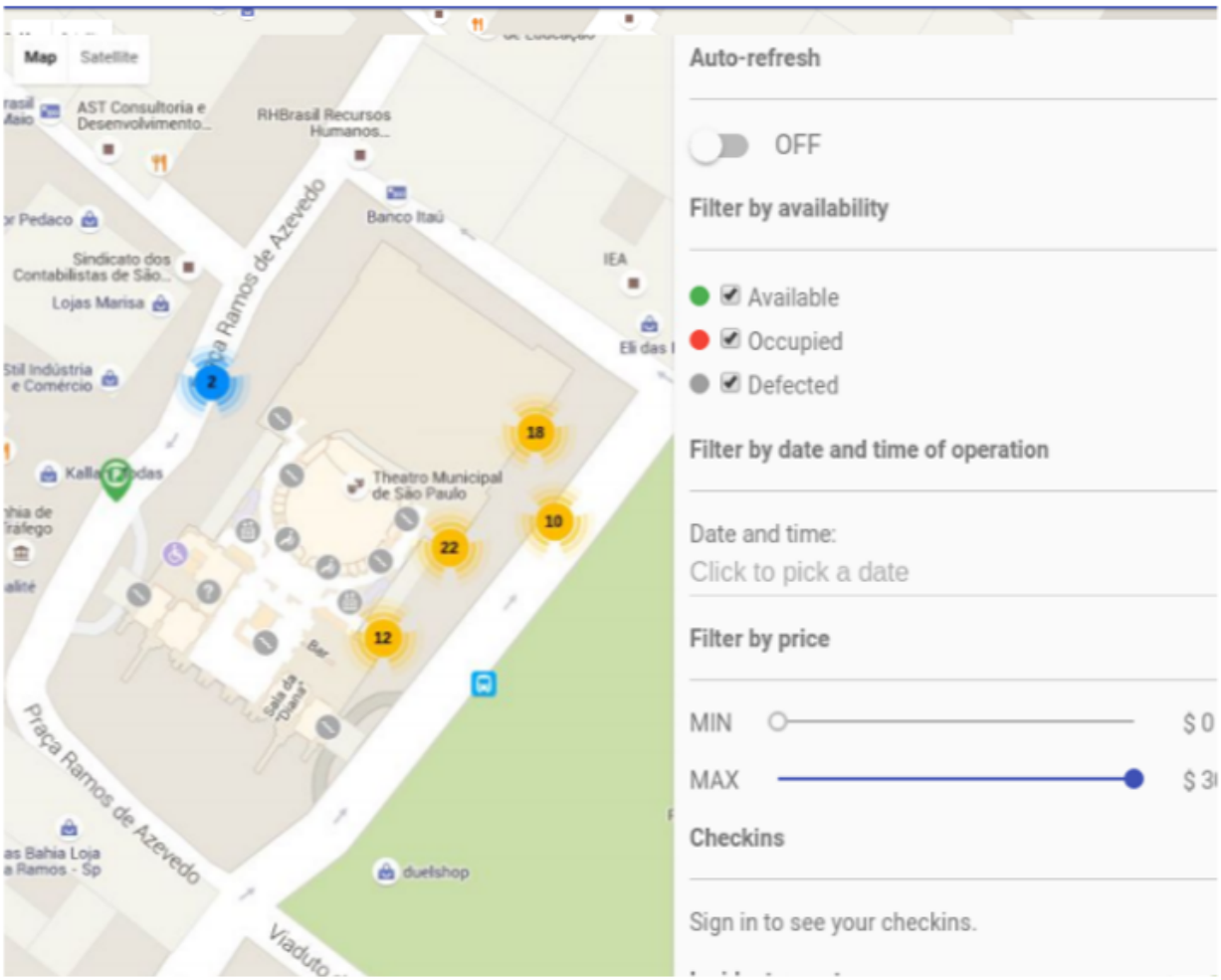
instalação dinâmica de dispositivos de internet das coisas, sendo estes tanto sensores como atuadores. Na figura 2.2 é ilustrada a arquitetura da plataforma InterSCity. Através desta plataforma, aplicações podem consultar quais atuadores e sensores estão conectados à plataforma e verificar informações disponibilizadas por estes dispositivos. Essa centralização dos dispositivos auxilia na criação de aplicações voltadas à manutenção e ao gerenciamento de cidades inteligentes. No artigo de Esposte et al. (2017) [EKCL17] é demonstrada uma aplicação que faz uso da plataforma InterSCity, destinada ao monitoramento de vagas para estacionamento chamada "Smart Parking App" ilustrada na figura 2.3. Esta aplicação usa sensores conectados via um "gateway IoT" que tem as responsabilidades de manter um registro dos dispositivos conectados e informar à plataforma InterSCity quando uma vaga se torna disponível.

Figura 2.2: *Arquitetura geral da plataforma InterSCity extraída de Esposte et al. (2017) [EKCL17]. Através da plataforma InterSCity aplicações como sites, aplicativos móveis e até outras ferramentas conseguem acessar, através do catalogo de dispositivos de internet das coisas fornecido pelo InterSCity, dados oriundos destes dispositivos sem necessariamente ter uma conexão direta com tais dispositivos.*



Kyriazis, et al. (2013) [KVW+13] propõem duas aplicações também baseadas em dispositivos de internet das coisas a fim de melhorar o uso de recursos na cidade, tornando-a assim uma cidade inteligente. A primeira aplicação descreve a implantação de um sistema de gerenciamento de energia e calor através de sensores devidamente posicionados. De acordo

Figura 2.3: *"Smart parking app" extraída de Esposte et al. (2017) [EKCL17]. Através de um "gateway IoT"os dispositivos IoT usados para monitorar o estado das vagas de estacionamento enviam este estado para o servidor do InterSCity e a aplicação então exibe aos usuários o estado de cada vaga monitorada.*

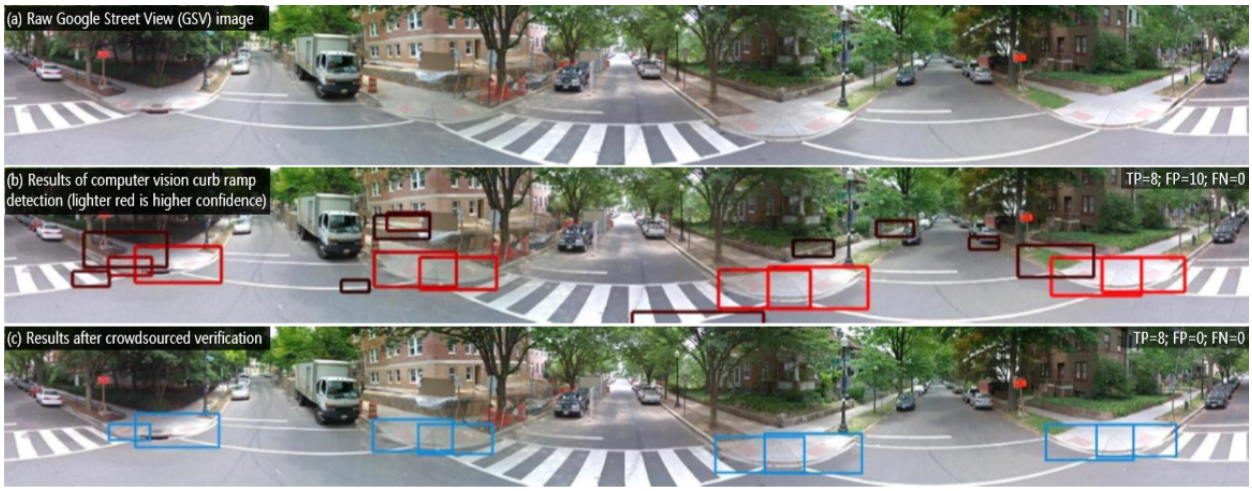


com os autores a visualização e gerenciamento do consumo de energia e produção de calor permite que ambos sejam diminuídos reduzindo assim custos e aumentando a eficiência do consumo de recursos. A segunda aplicação proposta por Kyriazis, et al. (2013) [KVW+13] consiste num sistema que tem por finalidade reduzir a poluição aérea através do controle de trafego, orientando motoristas com informações de rotas e velocidades. Essas informações são derivadas com base em sensores para elementos do tráfego urbano (semáforos, câmeras de trânsito, sensores de velocidade dos veículos, luzes de postes, etc) e a posição dos veículos através de GPS.

O projeto Tohme de Hara et al. (2014) [HLF13] para detecção de rampas de acesso é um exemplo de um sistema de suporte aos sistemas inteligentes que ajuda na auditoria de

uma determinada característica urbana. Ele é baseado num sistema de contribuição coletiva onde usuários, através de uma interface elaborada para exibir imagens do GSV, selecionam regiões nas imagens apresentadas e fazem anotações sobre essas regiões indicando a presença ou falta de rampas de acesso nas calçadas, além desse procedimento baseado na contribuição de usuários, o projeto Tohme também utiliza um segundo procedimento baseado em visão computacional para filtrar a imagem e destacar as rampas de acesso. A precisão deste segundo procedimento é prevista através de classificadores supervisionados, ou mais especificamente, de uma "Support Vector Machine" (SVM) [HDO+98]. Quando a SVM classifica os resultados obtidos dos filtros de imagem como sendo de baixa confiança, a imagem filtrada é então redirecionada a um usuário para ser corrigida manualmente se necessário. De acordo com o autor a precisão obtida manualmente é alta o suficiente para justificar essa estratégia. Através desta estratégia, o sistema é capaz de reduzir em 13% o tempo que seria dedicado à este tipo de auditoria por seres humanos, com uma precisão quase equivalente. A figura 2.4 ilustra o procedimento executado para a detecção de rampas de acesso. Na primeira linha são demonstradas imagens de curvas de acesso obtidas pelo "Google Street View". Na segunda linha os quadrados indicam rampas de acesso detectadas automaticamente, quanto mais claro o quadrado maior a confiança associada à rampa detectada. A terceira linha demonstra os resultados após a verificação por contribuição coletiva. Como dito pelos autores, o método automatizado de detecção de rampas de acesso não contempla a detecção da falta das mesmas, o que é uma oportunidade de pesquisa que pode trazer ferramentas úteis à administradores e arquitetos responsáveis pela instalação deste tipo de estrutura para acesso e mobilidade urbana. Além disso, o projeto Tohme não exibe uma interface de programação que o permita ser acoplado à outros sistemas, limitando assim seu escopo de uso, ou sua usabilidade, dentro de sistemas mais complexos que tenham por finalidade averiguar outras características urbanas além de rampas de acesso.

Figura 2.4: *Rampas de acesso detectadas e classificadas. Imagem extraída de Hara et al. (2014) [HLF13]. Na linha do topo estão algumas imagens conforme obtidas pela plataforma GSV, na linha central retângulos mais claros indicam rampas de acesso detectadas pelo Tohme com maior taxa de confiança e na linha de baixo os retângulo indicam rampas de acesso verificadas por usuários através de contribuição coletiva.*



2.4 Análise de imagens urbanas

As árvores e a vegetação são importantes para o meio urbano por fatores como dissipação de ilhas de calor, melhoria da qualidade do ar, economia de energia, paisagismo e até mesmo com relação à melhorias para a saúde como estudado em uma série de artigos [KGM+15, Lot99, LQN+08, MNH+97, NHBG14, TA78]. Além disso pelo fato de árvores serem características urbanas que variam relativamente pouco com o tempo, elas são objetos de estudo válidos através do GSV [RBR+11].

A detecção automatizada de árvores em imagens urbanas pode ser feita através de tecnologias do tipo "Light Detection and Ranging" (LIDAR), que resulta numa imagem 3D representada como nuvens de pontos, como em Monnier et al. (2012) [MVS12] e Buck et al. (2017) [BLMPN17]. No artigo de Monnier et al. [MVS12], os autores buscam um meio de identificar as árvores individualmente no contexto urbano. Para tal, são aplicados descritores geométricos e métodos probabilísticos sobre os dados obtidos, através de sensores LIDAR, para a classificação de fachadas de edifícios, postes e árvores. Os autores afirmam que 80% das árvores presentes são detectadas corretamente, isso é, a taxa de redescoberta é de 80%. De todos os objetos marcados pelo algoritmo como sendo troncos de árvore apenas 12% são falsos positivos, ou seja, a taxa de precisão do algoritmo é de 88%. A figura 2.5 ilustra

o processo proposto por Monnier et al. (2012) [MVS12]. Nesta figura são demonstrados os descritores geométricos usados que dão origem aos mapas de acumulação. Estes por sua vez serão usados para classificar as diferentes partes da imagem como sendo troncos de árvores, postes, fachadas e folhagem.

Já no artigo de Buck et al. [BLMPN17], os autores apresentam uma técnica para se realizar a contagem de árvores em florestas. O maior obstáculo encontrado, de acordo com os autores, é a oclusão de algumas árvores causada por outras. Este problema foi solucionado usando-se múltiplos pontos de vista. Os autores afirmam ter atingido desta forma uma contagem perfeita das árvores. Contudo, isso foi possível pois neste cenário podia-se escolher o tipo de sensor e o seu posicionamento. Quando não existe essa flexibilidade, é muito difícil atingir medidas perfeitas. A figura 2.6 ilustra o cenário onde as árvores devem ser contadas. Nesta imagem estão dispostas as câmeras do tipo LIDAR, alvos esféricos usados como referenciais durante múltiplas varreduras realizadas e as árvores que serão contadas. A figura 2.7 mostra um dos alvos esféricos usados para auxiliar no alinhamento de múltiplas varreduras.

No artigo de Ardila et al. (2011) [ATBS11] são usadas imagens de resolução muito alta ("very high resolution images") de satélite e são exploradas exploradas bandas multi-espectrais e pancromáticas através de técnicas de fusão de imagens. Em especial, é proposto um novo método baseado em mapeamento de super resolução ("super resolution mapping") para detectar e contar árvores individualmente. De acordo com os autores, o método possui uma precisão melhor que a obtida por classificação por máxima verossimilhança em imagens multi-espectrais e em imagens obtidas pela fusão por filtro passa alta de imagens multi-espectrais e pancromáticas. Os testes foram feitos em apenas duas áreas obtendo uma precisão de 57% e 77% em cada área respectivamente.

No artigo de Yang et al. (2009) [YZMG09], é apresentado um índice chamado "Green View Index" (GVI) que é correlacionado com a cobertura de árvores, vegetação e gramados (pontos verdes) na cidade de Berkeley. Este índice é formado a partir da razão entre o número de pontos de uma imagem classificados como pontos verdes e o total de pontos da imagem. Os autores tomaram 2252 fotografias ao longo de 563 cruzamentos (4 imagens por cruzamento) da cidade de Berkeley durante seu estudo . A classificação foi feita usando-se

Figura 2.5: *Processo de reconhecimento de árvores extraído de Monnier et al. (2012) [MVS12]. Através dos descritores geométricos da primeira linha todos os pontos obtidos por LIDAR são descritos e formam os mapas de acumulação, e por último cada região destes mapas são classificadas como troncos, postes ou colunas, fachadas de edifícios ou muros e folhagem.*

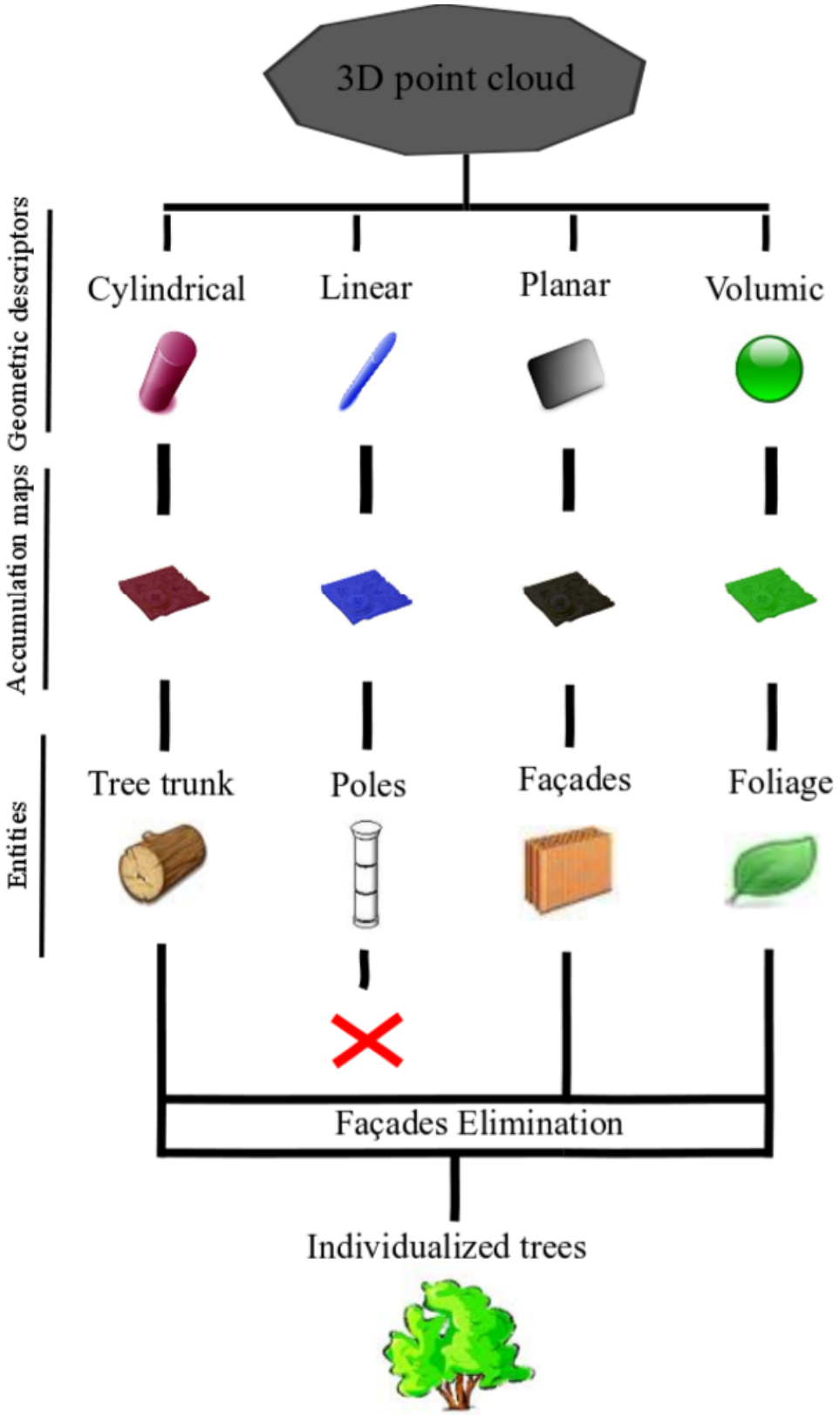
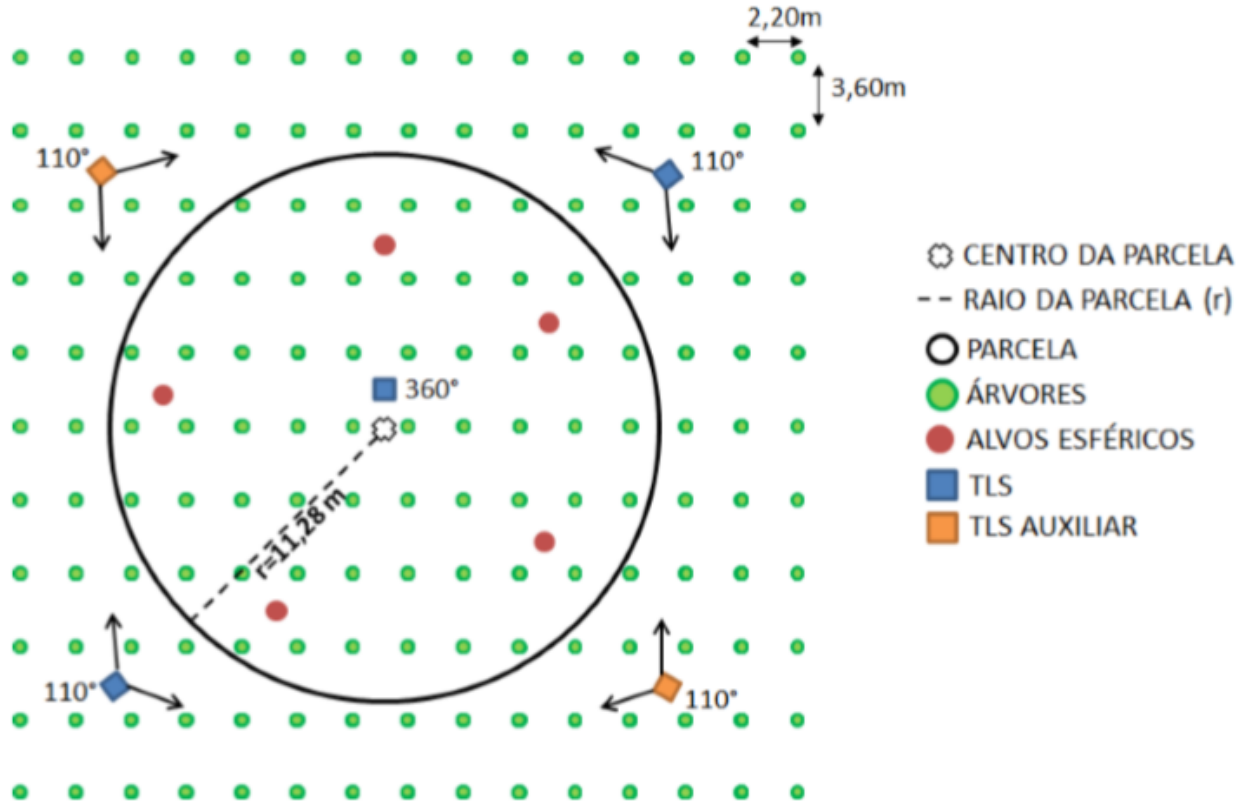


Figura 2.6: *Distribuição de árvores, câmeras e alvos de Buck et al. [BLMPN17]. Usando diversas câmeras e alvos esféricos os autores conseguem superar o problema da oclusão das árvores e obter uma contagem perfeita.*



duas características de textura e uma característica espectral incluindo o valor médio dos canais de cor vermelho, azul e uma modificação do canal verde obtida dividindo a média do canal verde pela soma das médias dos canais verde e vermelho. As características de textura foram formadas através das medidas de contraste e entropia da matriz de co-ocorrência de níveis de cinza (GLCM). A figura 2.8 ilustra a folhagem de uma árvore (marcada com a cor verde) após a imagem ter sido segmentada com o procedimento proposto por Yang et al. (2009) [YZMG09].

De acordo com Li et al. (2015) [LZL+15] o processo, descrito no artigo de Yang [YZMG09], apesar de demonstrar que a correlação entre a vegetação percebida por um pedestre e o índice proposto era alta, o processo é demorado e maçante dado que foi feito manualmente desde a aquisição das imagens até a validação e segmentação dos pontos verdes. Com isso a aplicabilidade do índice se restringe à regiões pequenas. No artigo de Li et al. (2015) [LZL+15] é proposta uma modificação do GVI em imagens obtidas pelo "Google Street View" (GSV). Essa modificação consiste num processo que toma as diferenças da subtração dos canais azul

Figura 2.7: *Exemplo de um alvo esférico usado para realizar o alinhamento das imagens durante o processo de múltiplas varreduras realizadas em diversos pontos de vista. Imagem extraída de Buck et al. [BLMPN17]*

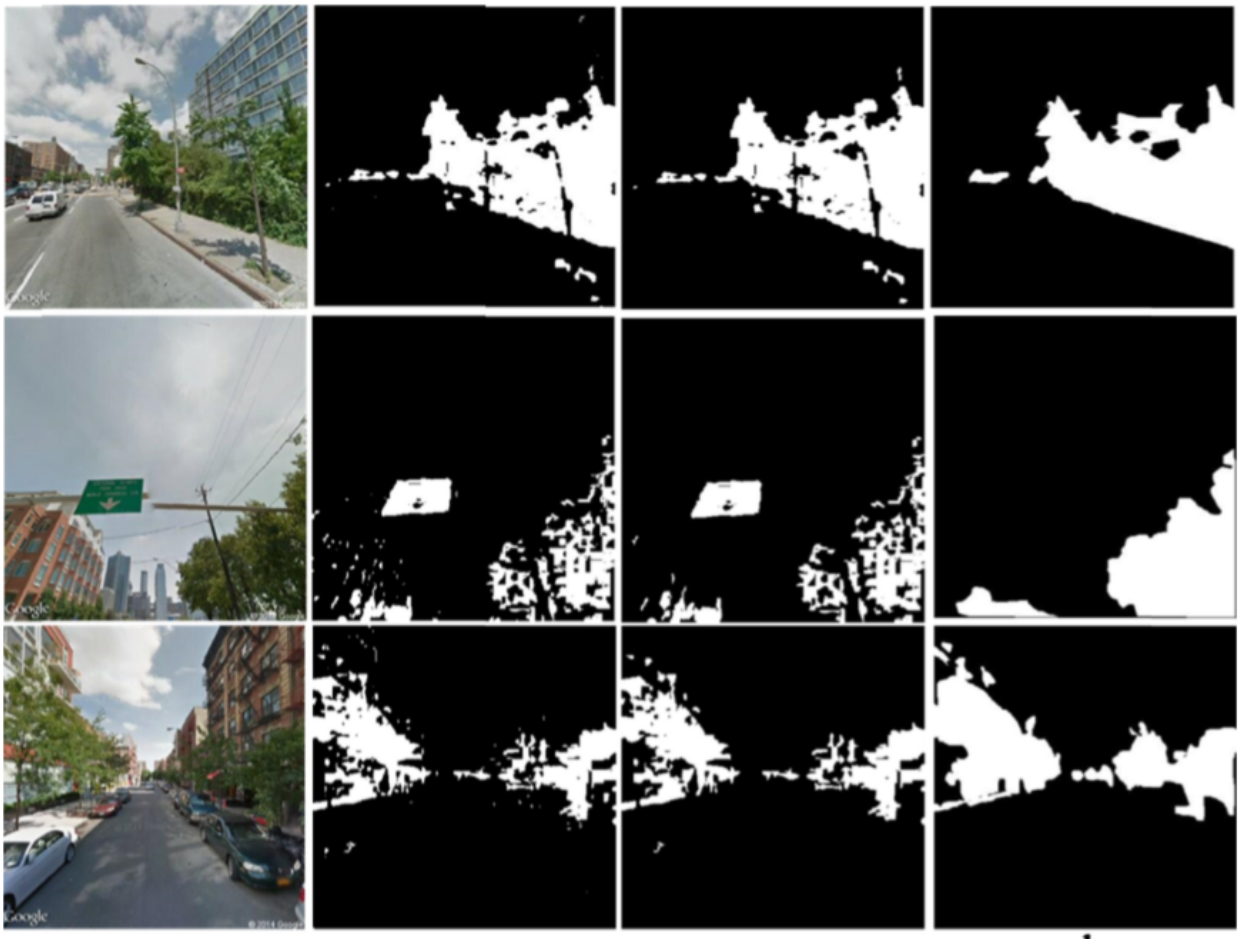
e vermelho do canal verde, e as multiplica. Essa imagem resultante é segmentada usando-se um limiar arbitrário. Um ponto da imagem original é classificado como verde se está nessa imagem segmentada e se a intensidade do resíduo da subtração do canal vermelho do canal verde é positivo. A figura 2.10 ilustra o procedimento proposto por Li et al. (2015) [LZL+15] e a figura 2.9 demonstra três casos em que o algoritmo foi usado; na primeira coluna são exibidas diferentes imagens urbanas, na segunda coluna são exibidos os resultados obtidos pelo procedimento proposto, na terceira as imagens após uma filtragem não descrita no artigo (a única referência à filtragem é a do livro de Jayaraman et al. (2011) [JST11]). A quarta coluna exibe uma segmentação manual realizada pelos autores usada como "groundtruth". Um limitante deste método é que o processo de obtenção do limiar adequado não é descrito. Além disso, informações de textura não são exploradas, o que leva a um potencial aumento de falsos positivos (por exemplo um muro pintado de verde).

Em um outro artigo de Li, Ratti e Seiferling (2017) [LRS17] os autores propõem uma maneira de se estimar o Fator de Visão do Céu ("Sky View Factor- SVF) a partir de imagens do GSV. O SVF é uma medida, adimensional e definida entre 0 e 1, do grau de visibilidade do

Figura 2.8: *Imagem de Yang et al. (2009) [YZMG09] onde a folhagem da árvore é marcada, em verde claro, pelo algoritmo de segmentação proposto pelos autores.*

céu usado em estudos de fluxos de energia em coberturas urbanas [Oke81, CT04, HGUM11]. O algoritmo proposto por Li, Ratti e Seiferling (2017) [LRS17] para estimar o SVF consiste primeiro na conversão da imagem (obtida pelo GSV) de coordenadas cilíndricas (veja a imagem 2.11 para um exemplo de panorama do GSV) para coordenadas azimutais (veja a imagem 2.12 para um exemplo de projeção azimutal da figura 2.11). Esta nova imagem em coordenadas azimutais é chamada pelos autores de imagem de olho de peixe (em referência a imagens obtidas com câmeras usando lentes com este mesmo nome). Após a projeção os autores convertem a imagem do espaço RGB para níveis de cinza usando para isso o brilho de cada ponto. O brilho de cada ponto é definido como $Brilho = \frac{R+G+B}{3}$ onde R, G e B correspondem ao valor dos canais vermelho, verde e azul de um ponto da imagem. Após esta conversão cada ponto da imagem é classificado como sendo do céu ou não através do algoritmo de limiarização de Otsu et al. (1992) [KOA92] que basicamente encontra um nível

Figura 2.9: *Imagem extraída de Li et al. (2015) [LZL+15] ilustrando o procedimento para segmentação de imagens proposto pelos autores. A primeira coluna ilustra imagens RGB conforme foram obtidas, a segunda coluna os resultados da segmentação. A terceira os resultados de uma filtragem não descrita pelos autores e a última a imagem de referência formada manualmente ("groundtruth").*



ótimo de cinza que minimiza a variância intraclasse e maximiza a variância entre classes (neste caso as classes correspondem aos pontos que pertencem ou não ao céu). Para calcular o fator de visão do céu (SVF) os autores utilizam a equação proposta por Johnson e Watson (1984) [JW84] ilustrada na equação 2.1, que é explicada a seguir:

$$SVF_P = \frac{1}{2\pi} \sin\left(\frac{\pi}{2n}\right) \sum_{i=1}^{n} \sin\left(\frac{\pi(2i-1)}{2n}\right) \alpha_i \tag{2.1}$$

No artigo de Johnson e Watson (1984) [JW84] os autores dividem a imagem de projeção azimutal (obtida no artigo com uma câmera com lentes do tipo olho de peixe) em n anéis concêntricos e de mesma largura (centralizados no centro da imagem de projeção azimutal). Na equação 2.1 o valor de SVF_P corresponde ao índice de céu visível do anel de índice i

de um conjunto de n anéis. O valor de n é arbitrário, porém no artigo de Li et al. (2017) [LRS17] é usado o valor $n = 37$. O valor de α_i não é explicitamente descrito no artigo de [LRS17], porém, no artigo de Hammerle et al. (2011) [HGUM11] ele é definido como na equação 2.2:

$$\alpha_i = \frac{T_{total} - T_{obs}}{\pi * r_2^2 - \pi * r_1^2} \tag{2.2}$$

Onde:

- T_{total} é a área total do anel de índice i;
- T_{obs} é a área obstruída (por ex. pontos marcados em preto na figura 2.13) no mesmo anel; e
- r_1 e r_2 são os raios interno e externo respectivamente do anel de índice i.

De acordo com Seiferling et al. (2017) [SNRP17] o mapeamento urbano de árvores e as técnicas existentes possuem problemas relacionados com o custo deste mapeamento, tempo e esforço exigidos. Além disso poucas abordagens consideram o ponto de vista no nível do solo, ou seja, a maneira como um pedestre percebe as árvores e a vegetação urbana no geral. No artigo de Seiferling et al. (2017) [SNRP17] é apresentada uma aplicação do estado da arte de visão computacional sobre o problema de detecção de árvores em imagens obtidas pelo GSV. O processo empregado para detecção das árvores começa com a segmentação da imagem utilizando o método proposto por Hoiem et al. (2005) [HEH05], que basicamente classifica diferentes regiões da imagens como pertencendo a diferentes classes geométricas (solo, superfícies planas verticais, superfícies não planas porosas e sólidas), essa segmentação forma "super-pixels" que são definidos como um conjunto de pontos da imagem com o mesmo rótulo. Cada "super-pixel" é uma região que respeita as bordas dos objetos, ou seja, as bordas de objetos presentes na imagem prevalecem como limites entre diferentes "super-pixels". Essa primeira etapa de rotulação é então refinada fazendo-se o agrupamento de "super-pixels" através de técnicas de aprendizagem estatística. Em seguida estas regiões de "super-pixels" agrupados são classificados com base em funções de verossimilhança aprendidas através de treinamento com um conjunto de imagens. Com base nas regiões classificadas como

árvores nas imagens, os autores apresentam uma métrica para quantizar a área de cobertura de árvores de uma cidade. Esta métrica é validada através da correlação com o mapa de cobertura de árvores desenvolvido por MacFaden et al. (2012) [MODR+12] e concluem que a métrica é uma boa medida para o quanto a vegetação é percebida do ponto de vista dos pedestres. A figura 2.14 mostra a concentração de árvores na cidade de Boston do projeto Treepedia [Lab17]. Esse mapa foi gerado a partir do processamento de imagens de Boston, obtidas pelo "Google Street View", através do método proposto por Seiferling et al. (2017) [SNRP17].

A combinação de informações que possuem diferentes momentos de aquisição se mostra um desafio e a formulação de sistemas de visão computacional para classificar e localizar objetos em uma combinação de imagens urbanas no nível do solo e aéreas é uma oportunidade de acordo com Wegner et al. (2016) [WBH+16]. Neste artigo os autores adaptam técnicas do estado da arte baseadas em redes neurais convolucionais "Convolutional Neural Networks" (CNN) para classificação e detecção de objetos. São apresentados dois módulos: um responsável por marcar sobre um mapa a localização de objetos de uma dada categoria, chamado de "det2geo", e outro responsável por classificar objetos numa dada localização, chamado "geo2cat". O módulo "det2get" foi testado e validado usando árvores como categoria de objetos à serem detectados. Os resultados são comparados com um conjunto de dados anotados que informações obtidas de imagens aéreas, de imagens no nível do solo e também de um catálogo que inclui a localização e a espécie de aproximadamente 80 mil árvores da cidade de Pasadena na Califórnia. Embora os autores não mencionem a fonte do conjunto de dados utilizado, existe um conjunto público de dados de Pasadena de aproximadamente 71 mil árvores disponível na Internet [oP14]. O resultados do módulo "det2geo" e o conjunto de dados anotados (localização e classificação de árvores) são constituídos apenas de coordenadas geográficas pontuais (por ex. latitude e longitude de cada árvore), sendo assim é necessário se fazer uma projeção baseada na distância da câmera com relação ao objeto de interesse para se obter nas imagens uma região de interesse que delimite os objetos da categoria escolhida. A figura 2.15 ilustra a sequência de etapas desde a seleção de uma região, até a exibição de árvores encontradas. Nesta figura do lado esquerdo é exibida a região de interesse, na parte central de cima são exibidas as imagens de satélite e no nível

da rua onde as características de interesse (árvores) são detectadas e suas coordenadas (do ponto de vista aéreo e terrestre) são combinadas em um sistema de coordenadas geográficas em comum. As características de interesse combinadas são então são projetadas novamente nas imagens de cada ponto de vista permitindo assim a computação de uma pontuação de detecção com cada alinhamento conhecido entre cada ponto de vista. A figura da direita demonstra o resultado da combinação da pontuação obtida através dos diferentes pontos de vista, um mapa de dados semântico (por ex. posição de ruas) e heurísticas espaciais (por ex. distância mínima entre duas árvores).

No artigo de Naik et al. (2014) [NPRH14] é proposto o projeto "Streetscore". Esse projeto foi criado para se prever, através do uso de "Support Vector Regression" (SVR) [SS04], qual o nível de segurança de uma região da cidade como percebido por uma pessoa através da imagem (ou imagens) da região numa escala de 0 à 10. Para definir qual descritor de imagens usar, eles realizam um estudo comparativo com diversos descritores de imagens. Com base nos vetores de características obtidos e usando SVR, eles criam mapas de percepção (do nível de segurança previsto) para diversas cidades nos Estados Unidos usando imagens do GSV com uma densidade de 200 imagens por milha quadrada (aproximadamente $2.59km^2$). A principal fonte de dados de treinamento provém de contribuições colaborativas oriundas do estudo de Salesses et al. (2013) [SSH13] onde são apresentados à diversos usuários pares de imagens e os usuários então escolhem qual lhes parece mais segura. Os autores do "StreetScore" indicam que diferenças de arquitetura entre cidades do mundo e características raras encontradas nas cidades (por ex. viadutos, muros pichados, etc) levam a previsões ruins por parte do "StreetScore". Porém afirmam que através do uso de mais dados obtidos por contribuição coletiva o projeto pode ser generalizado para estes outros casos e também para outras cidades ao redor do mundo. A figura 2.16 ilustra uma parte da região de Nova York demarcada pelo projeto StreetScore. Os pontos em verde escuro são classificados como regiões aparentemente mais seguras, as cores verde claro, amarelo, laranja e por fim vermelho indicam, em ordem decrescente, graus de segurança percebida menores que o verde escuro. Os mapas de percepção gerados pelos autores podem ser visualizados online [NPRH].

Figura 2.10: *Procedimento de segmentação proposto por Li et al. (2015) [LZL+15]. Com base no produto das diferenças entre o valor do canal verde e dos canais vermelho e azul de um dado ponto da imagem o ponto é classificado como pertencendo ou não à vegetação se o valor do produto for maior ou não, respectivamente, que um dado limiar arbitrário.*

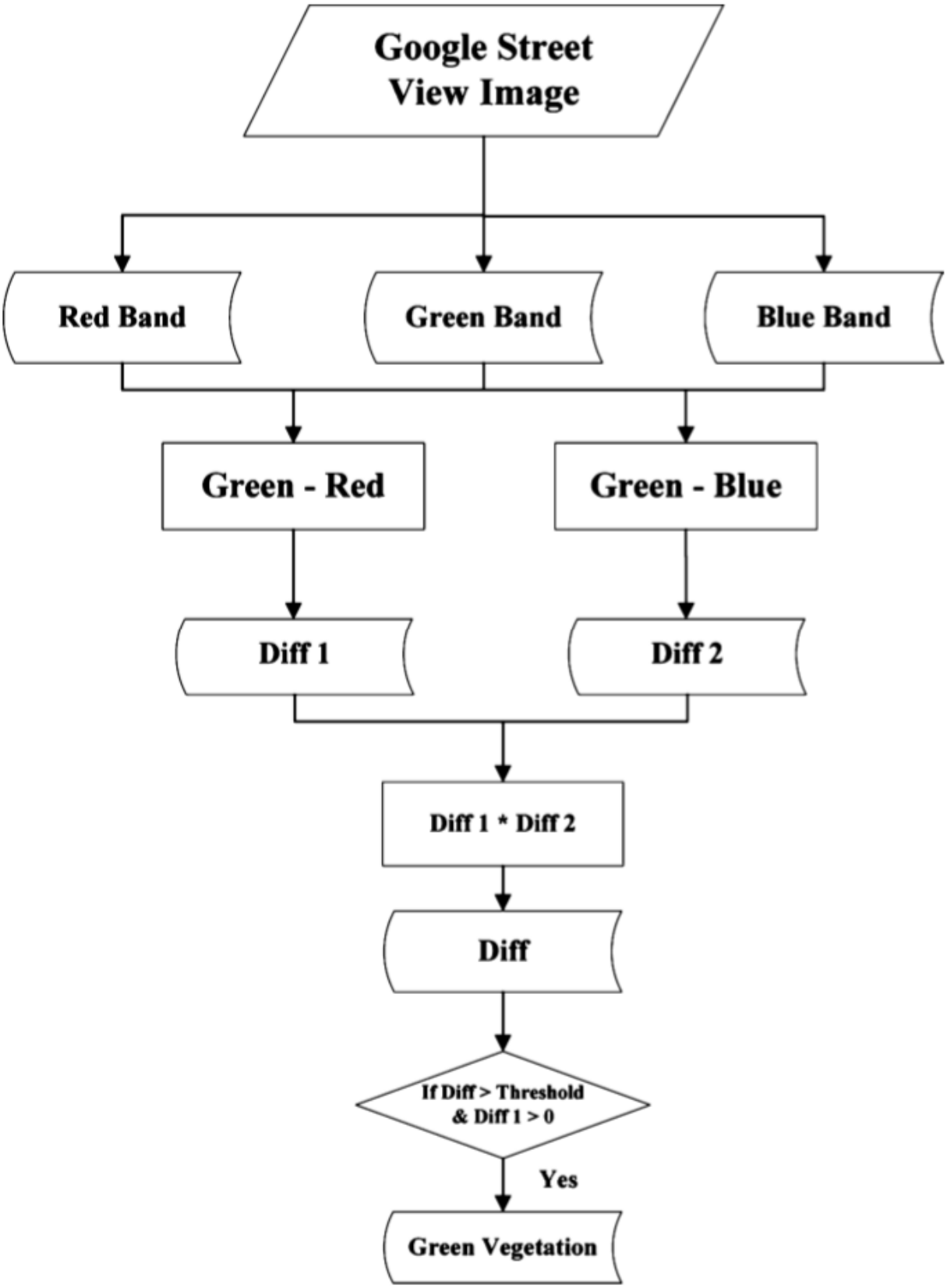


Figura 2.11: *Imagem panoramica obtida pelo GSV extraída de Li et al. (2017) [LRS17].*

Figura 2.12: *Projeção azimutal da Fig. 2.11.*

Figura 2.13: *Resultado da classificação dos pontos da Fig. 2.12 baseada na limiarização do nível de brilho de cada ponto da imagem usando-se o método de Otsu et al. (1992) [KOA92]*

.

Figura 2.14: *Mapa de concentração de árvores em Boston extraído do projeto Treepedia [Lab17]. O sistema também contém a visualização de 26 cidades até o momento de elaboração deste trabalho.*

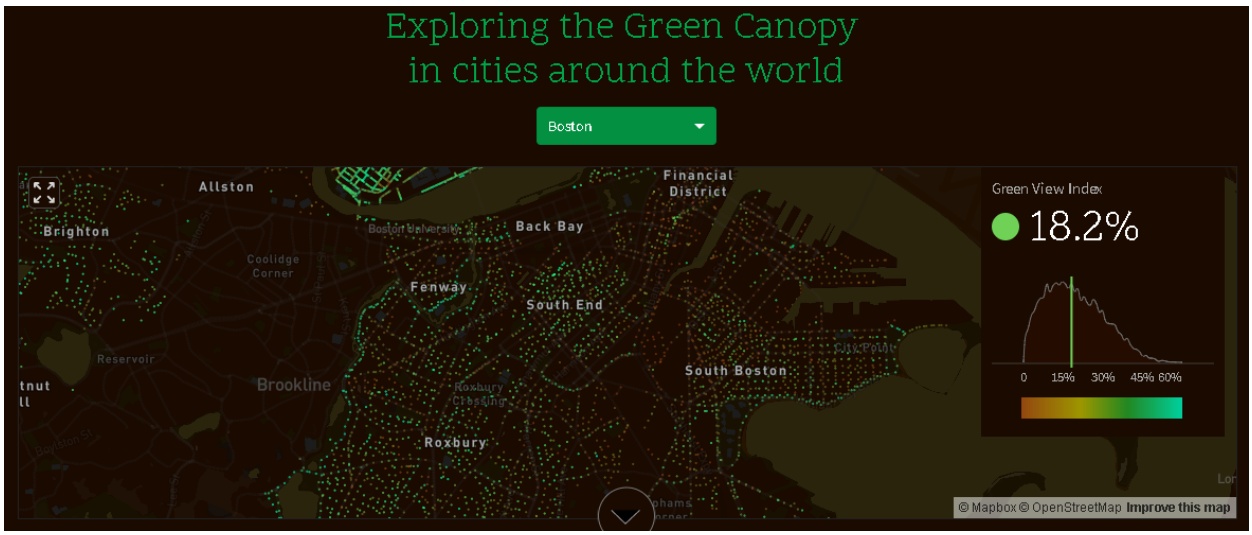


Figura 2.15: *Processo para detecção de árvores extraído de [WBH+16]. A região de interesse ("Input Region") é analisada através de imagens aéreas e no nível do solo e combinando-se as duas visualizações a posição geográfica de cada árvore é estimada.*

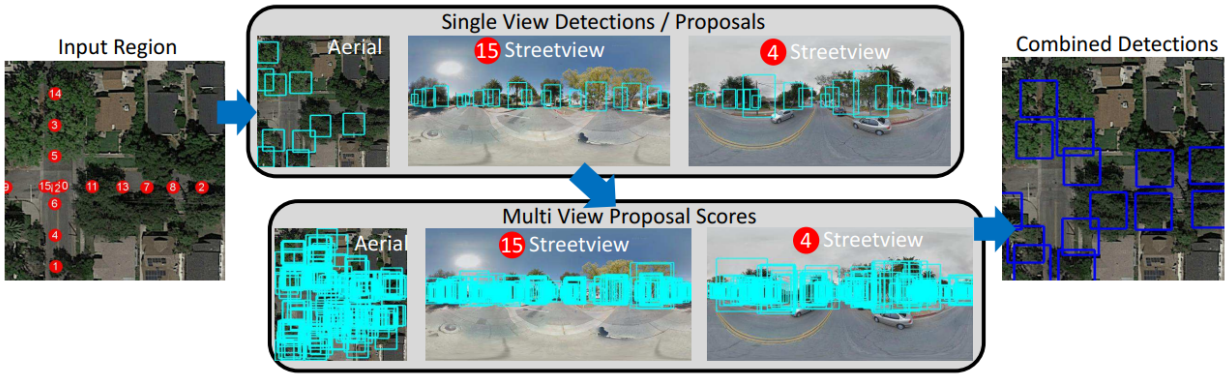
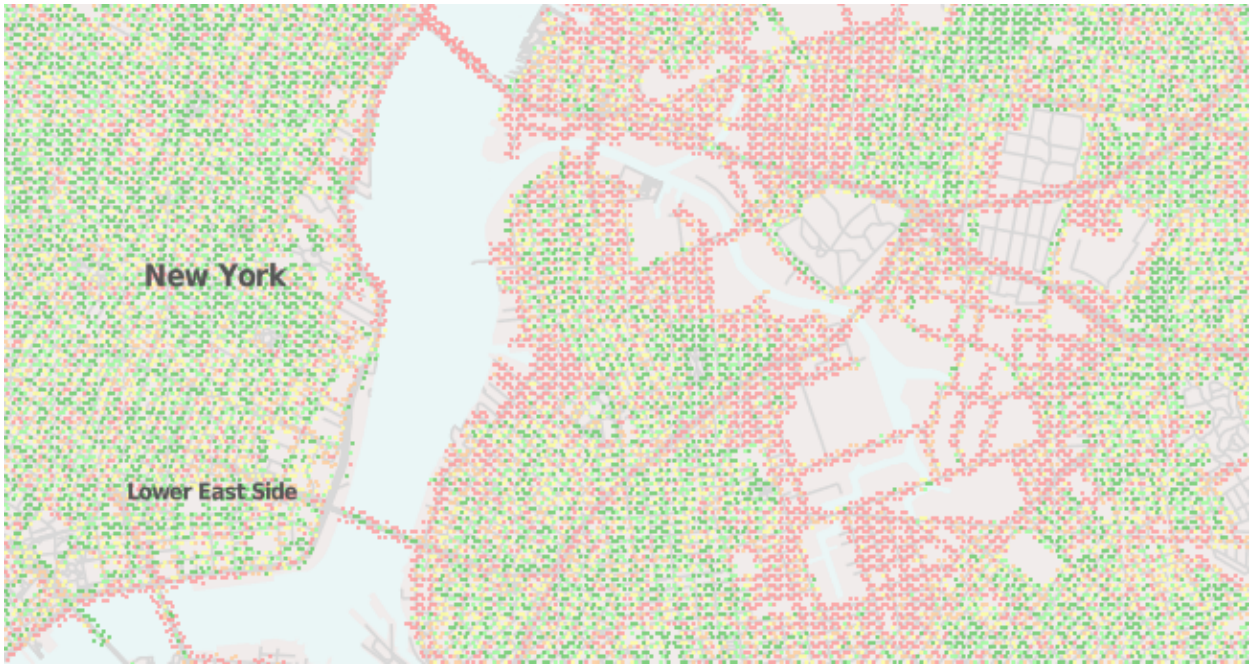


Figura 2.16: *Mapa de percepção de segurança extraído do site [NPRH]. Cores mais próximas ao verde possuem pontuações de segurança maiores, amarelas possuem pontuações menores que as verdes e as regiões marcadas com cores mais próximas ao vermelho possuem as menores pontuações.*



Capítulo 3

Fundamentos

Neste capítulo serão apresentados alguns dos conceitos principais usados na elaboração da arquitetura do sistema e também alguns dos conceitos matemáticos usados na elaboração do algoritmo de detecção de árvores em imagens. A bibliografia básica para a primeira parte foi o livro "Design patterns: Elements of reusable object-oriented software" [VHJG95]. Para a parte de análise de imagens, foram usados vários livros e artigos referenciados no texto, mas o principal foi o livro "Morphological image analysis: principles and applications" de Soille, P. (2013) [Soi13].

3.1 Conceitos básicos do sistema

Nesta seção apresentaremos o padrão arquitetural "Model-View-Controllers" (MVC), a arquitetura conhecida como "Representational State Transfer"(REST) usada para formular a Interface de Programação de Aplicativos (API) disponibilizada pelo servidor e os padrões de projeto de arquitetura de software conhecidos como "Mediator", "Strategy" e "Observer" [VHJG95]. Assume-se o conhecimento de "Unified Model Language" (UML), que será usada em alguns diagramas ao longo do texto.

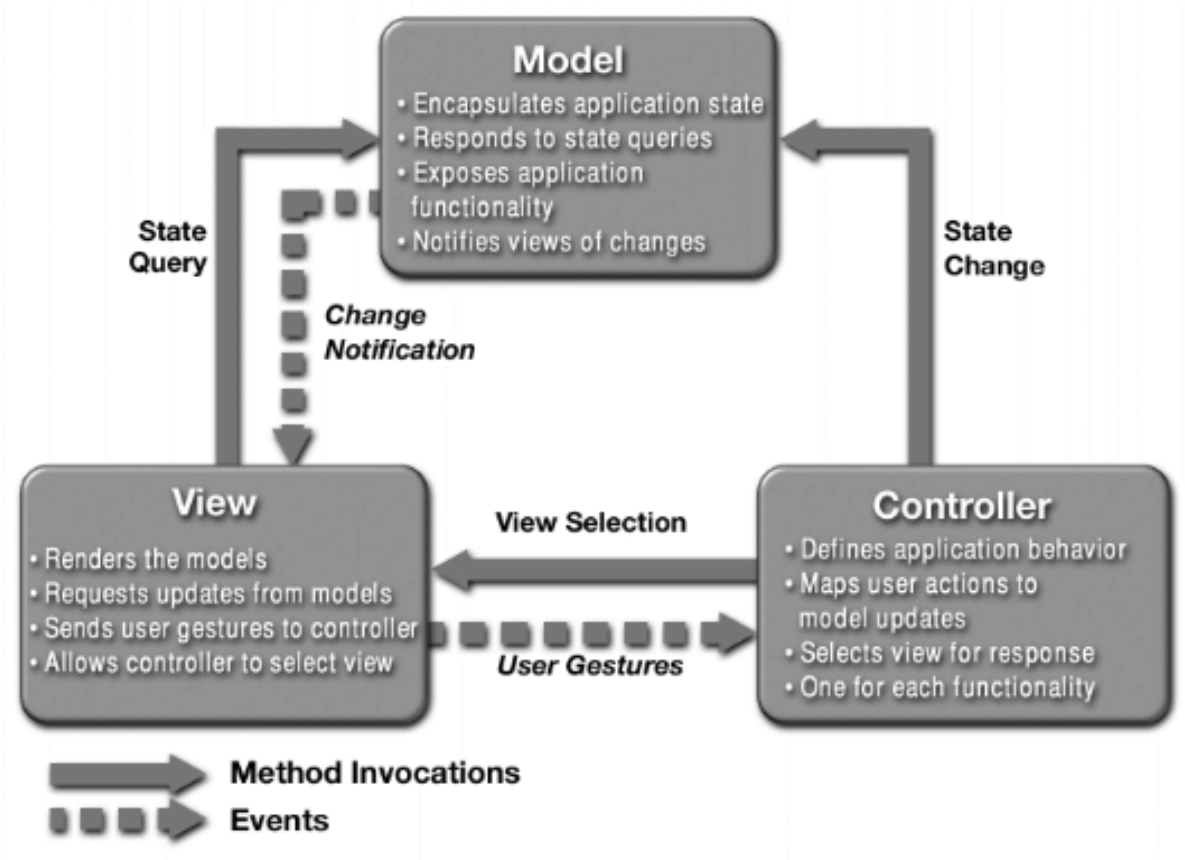
3.1.1 Padrões de arquitetura: "Model-View-Controller"e "Representational State Transfer"

De acordo com Gamma, E. et al. [VHJG95] o padrão MVC permite o desacoplamento entre a camada de dados ("Model"), de apresentação ("View") e de controle ("Controller"). A camada de dados representa o estado da aplicação, ou seja, todas as variáveis que são usadas para manter atualizada a camada de apresentação que por sua vez tem a função de interface do usuário. A comunicação entre a camadas de dados e a camada de apresentação é feita normalmente através de um mecanismo de inscrição e notificação (descrito adiante na seção 3.1.4). A camada de dados mantém uma referência à camada de apresentação e sempre que alguma alteração ocorre ela notifica a camada de apresentação que por sua vez exibe ao usuário o conteúdo, consistentemente atualizado, da camada de dados. Este tipo de mecanismo (inscrição e notificação) é descrito pelo padrão de projeto chamado "observer" e ele será melhor detalhado mais abaixo na subseção 3.1.4. A figura 3.1 exemplifica o relacionamento entre as diferentes partes do modelo MVC.

A camada de controle permite que a lógica de interação do usuário (por ex. pressionar um botão) seja desacoplada do resto do sistema. Desta forma um mesmo componente pode ter comportamentos diferentes mesmo em tempo de execução. Com isso a resposta do sistema às diferentes interações do usuário pode mudar sem que as camadas de apresentação e de dados sejam alteradas. A interação entre as camadas de controle ("controller") e de apresentação ("view") pode ser feita através de uma especialização da classe (subclasse) que representa a camada de controle na camada de apresentação, ou seja a "controller" é definida como sendo uma classe abstrata enquanto que uma subclasse desta "controller" irá ser uma especialização dela e irá encapsular o comportamento de uma "view" em um determinado momento, dependendo do estado da "view" o comportamento pode mudar e essa mudança pode ser realizada mudando-se a subclasse da "controller" utilizada pela "view", o que é descrito pelo padrão de projeto "strategy" descrito logo abaixo na subseção 3.1.3.

Apesar do padrão MVC ter sido originalmente desenvolvido para aplicações locais ele também é bastante utilizado no desenvolvimento de aplicativos de internet, porém, de acordo com de acordo com Leff, A. e Rayfield, J. (2001) [LR01] a interpretação de como as respon-

Figura 3.1: *Diagrama MVC [Mica]. Partindo-se da camada de visualização, primeiro um usuário (não representado neste diagrama) inicia algum tipo de interação, esta interação é notificada como um evento para a camada de controle que pode então definir qual o comportamento adequado para aquela interação e também mudar o estado da camada de modelo. A camada de modelo se for alterada notifica a camada de visualização sobre esta alteração e a camada de alteração agora pode consultar os novos dados da camada de modelo e exibi-los para o usuário.*



sabilidades de cada componente devem ser dividas entre cliente e servidor são variadas.

Segundo a definição de Fielding, R. e Taylor, R. (2008) [FT00], uma aplicação seguindo as definições da arquitetura REST deve:

- ser baseada na separação de conceitos entre servidor e cliente;
- não manter um estado, ou seja, cada requisição ao servidor deve ser independente de outras requisições feitas anteriormente;
- marcar todo conteúdo explicitamente como localmente armazenável ou não-localmente armazenável de forma que o cliente deve apenas manter em cache o que for explicitamente marcado pelo servidor como localmente armazenável;

- possuir uma interface (de programação) unificada;
- possuir camadas bem definidas (como no padrão de arquitetura MVC) de forma que cada componente inserido em uma determinada camada somente possa interagir com os componentes inseridos nas camadas adjacentes;
- opcionalmente permitir que códigos customizados sejam obtidos pelo cliente sob demanda.

3.1.2 Padrão de projeto: "Mediator"

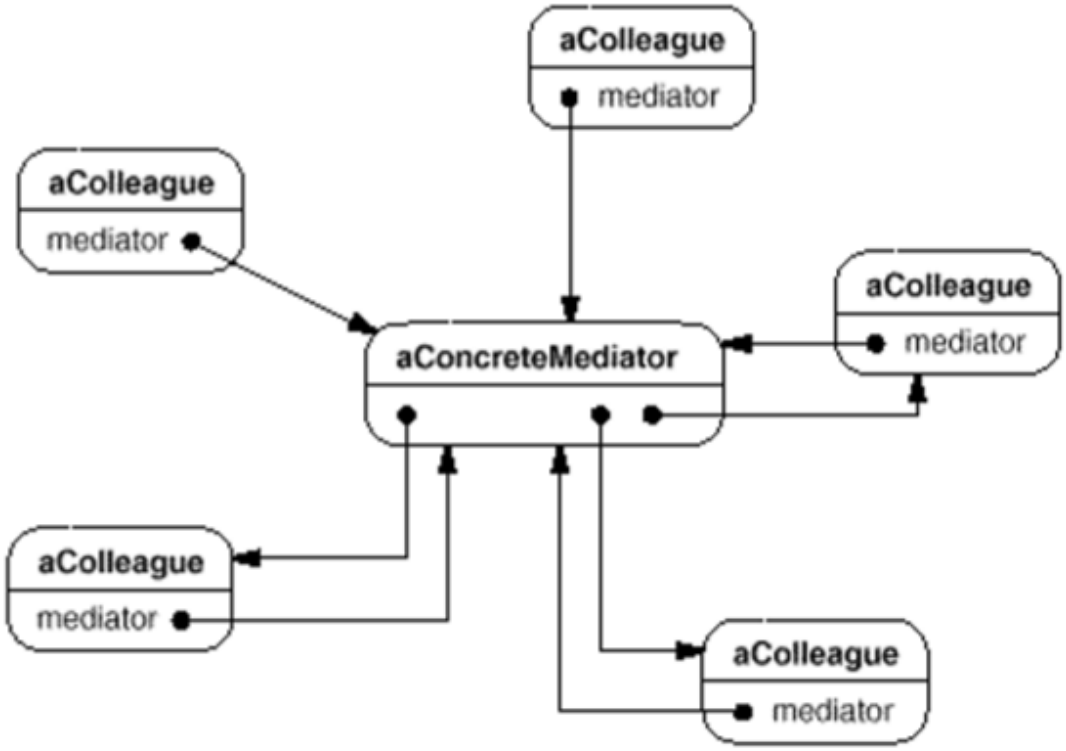
De acordo com Gamma et al. (1995) [VHJG95] o padrão "mediator" tem como responsabilidade reduzir o nível de acoplamento de um sistema coordenando a comunicação entre diferentes componentes do sistema. Os componentes ao invés de serem ligados diretamente, aumentando assim o acoplamento do sistema, são ligados ao "mediator" que será responsável por coordenar a comunicação entre eles e também manter um registro de quais componentes estão presentes em cada momento da execução da aplicação. Baseadas no conceito do "mediator", exitem as classes chamadas de "managers" que permitem o desacoplamento entre os componentes de controle (da camada "controller" do MVC) e outros componentes como, por exemplo, os filtros de imagens ou mesmo os componentes para busca de imagens.

Ainda de acordo com Gamma et al. (1995) [VHJG95] uma das possíveis desvantagens do padrão "mediator" é que, pelo fato dele ser responsável pela interação de diversos componentes, conforme esta interação se torna mais complexa a sua implementação também se torna mais complexa, dificultando sua manutenção e compreensão. A figura 3.2 extraída de Gamma et al. (1995) [VHJG95] exemplifica o funcionamento do padrão "mediator".

3.1.3 Padrão de projeto: "Strategy"

De acordo com Gamma et al. (1995) [VHJG95] o padrão "strategy" é usado quando se é necessária a mudança de um algoritmo em tempo de execução de forma transparente, isso é, sem a intervenção direta no código. Isso significa que dada uma família de algoritmos é possível a substituição de um algoritmo por outro da mesma família sem que seja necessária alguma alteração no código do projeto. A maneira indicada para se obter esse tipo de

Figura 3.2: *Cada objeto "aColleague" possui uma referência para o objeto "aConcreteMediator" que neste caso é o "mediator". Sendo o "mediator" ele é responsável por coordenar a comunicação entre os diferentes objetos "aColleagues" e manter uma lista de todos os objetos "aColleague" presentes. Cada objeto "aColleague" implementa a interface definida pelo "mediator". Gamma et al. (1995) [VHJG95]*



característica é através de uma interface comum à todas as classes que implementam algum membro da família de algoritmos.

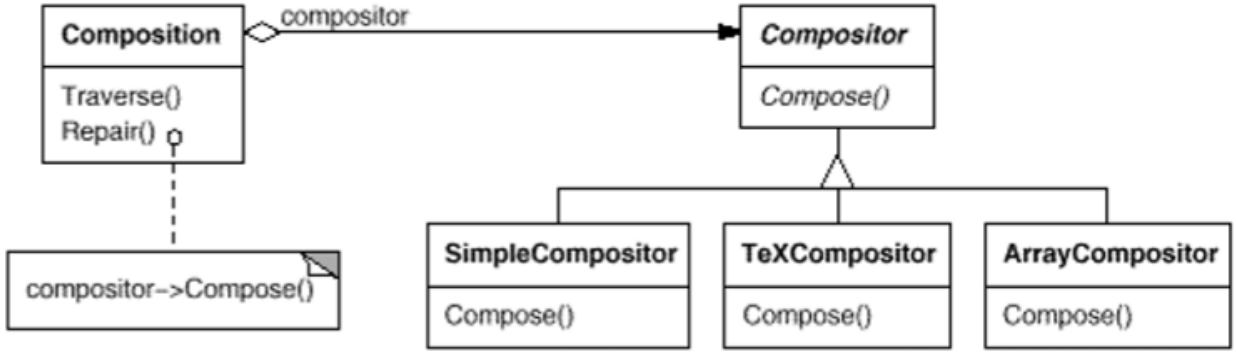
Na figura 3.3, extraída de Gamma et al. (1995) [VHJG95], é exemplifcada uma aplicação do padrão[1] "strategy" para um compositor de texto com comportamentos diferentes para diferentes algoritmos para quebra de linha.

3.1.4 Padrão de projeto: "Observer"

De acordo com Gamma et al. (1995) [VHJG95] o padrão "observer" é usado para permitir que diversos componentes possam ser notificados a respeito de alterações em outros componentes sem que nem os componentes que recebem as notificações nem os que tiveram uma alteração em seu estado tenham referências explicitas de um para o outro. Para isso é usada

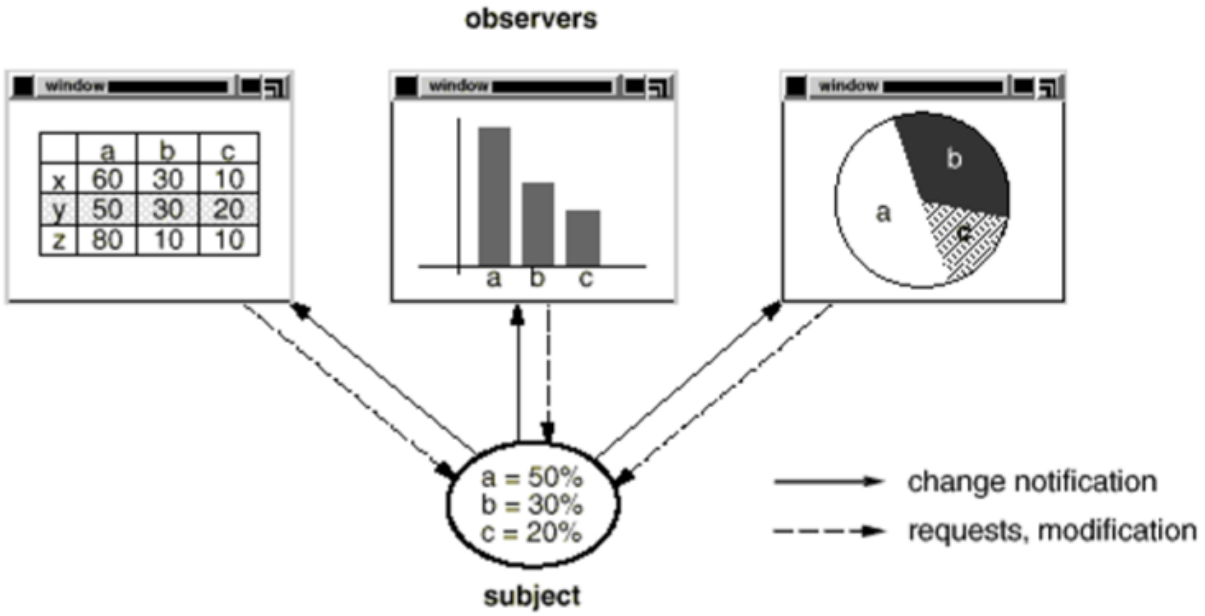
[1] O termo "padrão de projeto" algumas vezes será substituído simplesmente pelo termo "padrão".

Figura 3.3: *Diagrama "strategy" de Gamma et al. (1995) [VHJG95]. O objeto "Composition" que é responsável por manter e atualizar quebras de linha exibidas num visualizador de texto. A fim de não encapsular dentro dele todos os diferentes algoritmos usados para se realizar quebras de linha, cada algoritmo é implementado individualmente em uma subclasse de "Compositor", que é uma classe abstrata. O objeto "Composition" mantém então uma referência à classe abstrata "Compositor", que é responsável por reformatar o texto. Neste caso, a instância de "Composition" especifica qual algoritmo deve ser usado, instanciando a subclasse desejada de "Compositor".*



uma interface comum a todos os objetos que deverão receber tais notificações, chamados de observadores ("observers"). O objeto que notifica os outros a respeito de alterações em seu estado, chamado "sujeito" ("subject"), irá conter uma lista de objetos que implementam a interface para notificação. Uma vez que ocorra o evento em que o estado do "sujeito" é alterado, todas os observadores são notificados através de uma chamada declarada na interface de notificação e implementada em cada "observador". Com esse mecanismo é possível se mudar dinamicamente o número de "observadores" e também permite que o "sujeito" seja indiferente a quem são os "observadores", a única exigência é que estes "observadores" implementem a interface de notificação. A figura 3.4 extraída de Gamma et al. (1995) [VHJG95] ilustra um caso em que diferentes objetos gráficos, que neste caso atuam como "observadores", são responsáveis por apresentar ao usuário diferentes visualizações de um mesmo conjunto de dados mantido por outro objeto que atua como "sujeito". O "sujeito" é responsável por notificar todos os "observadores" de alterações ocorridas em seu estado. Para isto é necessário que os "observadores" primeiramente se inscrevam na lista de referências, mantida pelo "sujeito", para serem notificados.

Figura 3.4: *Diagrama "observer" de Gamma et al. (1995) [VHJG95]. O sujeito ("subject"), representado nesta figura como um conjunto de variáveis mantém um registro de cada observador, representados nesta figura como uma planilha, um gráfico de barras e um de pizza, sempre que uma alteração ocorre em uma das variáveis do sujeito ele notifica todos os observadores registrados e estes então podem atualizar suas visualizações de maneira a se manterem consistentes com os dados mais atualizados.*



3.2 Conceitos do detector de árvores

Nesta seção serão explicados brevemente os conceitos matemáticos dos métodos de processamento de imagens usados para extração de vegetação. Estes métodos são baseados em transformações entre diferentes espaços de cor e algumas técnicas de análise de texturas e estruturas presentes nas imagens. Além disso, alguns conceitos sobre morfologia matemática também serão abordados.

3.2.1 Espaços de cor

De acordo com Joblove, G. e Greeberg, D. (1978) [JG78], espaços de cor representam uma faixa de cores através de um sistema de coordenadas tridimensional inspirado no sistema visual humano, que contém sensores de diferentes faixas de comprimentos de ondas eletromagnéticas visíveis.

O conjunto composto pelos comprimentos de onda responsáveis pelas cores vermelha, verde e azul pode produzir o maior número de cores através da combinação de diferentes

proporções de cada comprimento de onda. Sendo assim um grande número de dispositivos gráficos tenta reproduzir, com a maior semelhança possível, essas três cores e combinando-as produzem diferentes cores. O conjunto de cores formadas pela combinação das cores vermelha, verde e azul é conhecido como espaço de cor RGB ("red", "green" e "blue"). Esse espaço de cores é comumente encontrado normalizado e nesse caso denota-se por nRGB (normalized RGB). A equação 3.1 generaliza a transformação de cada canal (R, ou G, ou B), denotado na equação por k.

$$k = \begin{cases} \frac{k}{R+G+B}, & \text{se } R+G+B > 0 \\ 0, & \text{se } R+G+B = 0 \end{cases} \tag{3.1}$$

Onde:

- R, G e B são as intensidade (números inteiros positivos) dos canais de cor vermelho, verde e azul respectivamente da imagem
- $k \in \{R, G, B\}$

Um outro espaço de cor bastante utilizado é o HSV, de "Hue-Saturation-Value". De acordo com Vadivel et al. (2005) [VSM05] o HSV separa o componente de intensidade, "Value", dos componentes de crominância de cor: matiz, "Hue", e saturação, "Saturation". Além disso, ele é útil para segmentação de imagens de acordo com Vadivel et al. (2005) [VSM05].

De acordo com Cavallaro et al. (2005) [CSE05], alguns espaços de cor como o HSV e o RGB possuem a propriedade de serem fotometricamente invariantes. Isso significa que os componentes de cromaticidade de cada cor mudam pouco com variações na intensidade da iluminação ou no caso da ocorrência de sombras sobre superfícies iluminadas. A normalização do espaço de cor RGB corresponde a uma projeção do vetor de cor no plano descrito pela equação $r + g + b = 1$, sendo r, g e b os componentes vermelho, verde e azul do vetor de cor no espaço de cor RGB. Através de uma análise empírica, os autores no artigo de Cavallaro et al. (2005) [CSE05] identificam três regras: a primeira é que uma sombra escurece cada componente de cor do ponto onde ela é projetada, ou seja, reduzem os valores de cada componente de cor do ponto no espaço de cor RGB ou diminuem o valor do brilho ("Value")

no espaço de cor HSV; a segunda regra observada é que estes pontos mantém sua ordem sob o efeito de sombras, isto é, num espaço de cor como o RGB a ordenação por intensidade de cada canal é mantida após o efeito da sombra (se $B \leq G \leq R$ originalmente, após o efeito da sombra a ordenação se mantém B" $\leq G$" $\leq R$"); a terceira regra observada é que as características fotométricas invariantes (por ex. matiz e saturação no espaço de cor HSV) se mantém inalteradas sob o efeito de sombras. É importante notar que essas regras expressam que a normalização do espaço de cor RGB o tornam mais robusto à variações de iluminação, mas não invariante à mudanças muito bruscas na iluminação (por ex. um ponto que se torna escuro o suficiente vai ter suas propriedades fotométricas invariantes alteradas).

A conversão do espaço de cor RGB para o HSV é feita de acordo com o algoritmo 1,

extraído do livro de Agoston, K (2005) [AA05].

Algoritmo 1: CONVERSÃO DE RGB PARA HSV

Entrada: $r, g, b \in [0, 1]$ um vetor no espaço de cor "RGB"

Saída: $h \in [0, 360], s, v \in [0, 1]$ um vetor no espaço de cor "HSV"

início
  Seja $V = \max r, g, b$
  Seja $X = \min r, g, b$
se $V = 0$ **então**
  Seja $S = 0$
senão
  $S = \frac{V-X}{V}$
se $S = 0$ **então**
  $h = 0$ caso de uma cor sem saturação
senão
  Seja $d = V - X$
  se $r = V$ **então**
    $h = \frac{g-b}{d}$
  senão se $g = V$ **então**
    $h = 2 + \frac{b-r}{d}$
  senão se $b = V$ **então**
    $h = 4 + \frac{r-g}{d}$
$h = h * 60$
se $h < 0$ **então**
  $h = h + 360$
retorna $[h, s, v]$

3.2.2 Morfologia matemática

As técnicas de morfologia matemática são normalmente usadas para filtragem de imagens e remoção de pequenos artefatos.

Nesta subseção alguns operadores e operações morfológicas usadas no texto serão apresentadas. Uma descrição mais detalhada do assunto pode ser encontrada no livro "Morphological image analysis: principles and applications" de Soille, P. (2013) [Soi13].

Os operadores morfológicas usados no filtro de árvores foram: dilatação, erosão, abertura e fechamento morfológico, e também o operador cartola-dual [Soi13]. Os operadores morfológicos podem ser escritos como combinações de dilatação, erosão e operações de interseção e complemento. Os operadores são normalmente parametrizados por um subconjunto do domínio chamado de elemento estruturante.

Considerando uma imagem binária, X, definida como um subconjunto do plano, $X \subset Z \times Z$, com a origem denotada por $\mathbf{o}$ e um subconjunto $B \subset Z \times Z$, que é chamado de elemento estruturante, as operações de erosão e dilatação morfológica são definidas pelas equações 3.2 e 3.3, respectivamente:

$$\varepsilon_B(X) = \{x \in Z \times Z \mid B_x \subseteq X\} \tag{3.2}$$

$$\delta_B(X) = \{x \in Z \times Z \mid \breve{B}_x \cap X \neq \emptyset\} \tag{3.3}$$

Onde B_x é o conjunto B transladado pelo vetor x e $\breve{B}$ é a reflexão (ou transposição) do conjunto B:

- $B_x = \{x + b \mid b \in B\}$
- $\breve{B} = \{-b \mid b \in B\}$

A figura 3.5 ilustra a erosão do conjunto de pontos X pelo elemento estruturante B. Neste caso X contém dois objetos desconexos e o menor deles é completamente erodido por B uma vez que B não se encaixa em lugar nenhum dentro da região deste objeto menor.

A figura 3.6 ilustra um caso onde o conjunto X, desta vez, é dilatado pelo elemento estruturante B. Neste caso os diferentes elementos de X são unidos, uma vez que a interseção entre o elemento estruturante B e X nunca é vazia quando B é posicionado no espaço entre os dois elementos de X.

Com base nos operadores de dilatação e erosão é possível definir outros operadores como

Figura 3.5: *Erosão morfológica extraída de Soille, P. (2013) [Soi13]*

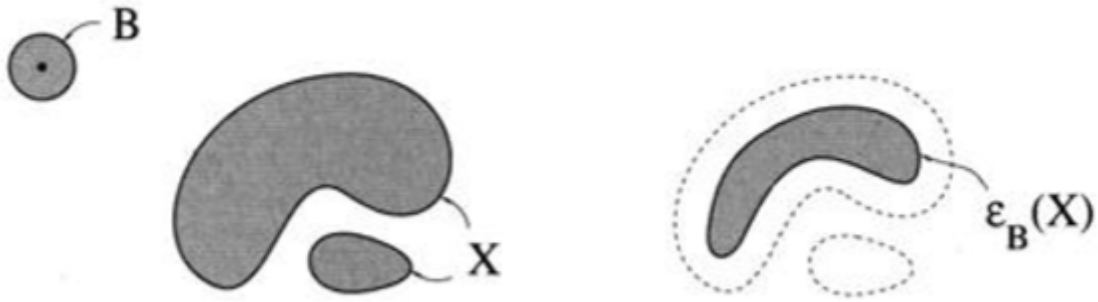
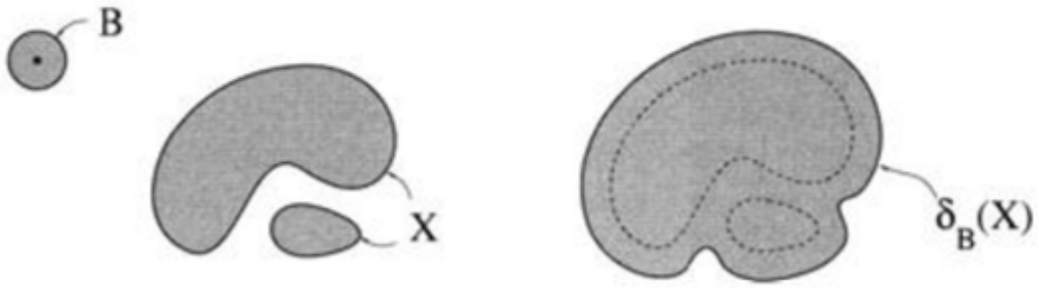


Figura 3.6: *Dilatação morfológica extraída de Soille, P. (2013) [Soi13]*



a abertura e o fechamento morfológico. Os operadores de abertura e fechamento são tipicamente usados para filtrar imagens de maneira a preservar outras estruturas presentes nelas (por ex. remover ruído sem diminuir ou aumentar outros elementos presentes). O operador de abertura é definido pelo de erosão seguido pelo de dilatação sobre a mesma imagem usando-se o mesmo elemento estruturante em ambos. Após a aplicação desse operador, elementos menores que o elemento estruturante são removidos. O operador de abertura morfológica aplicado sobre uma imagem binária X com um elemento estruturante B é descrito pela equação 3.4. A figura 3.7 ilustra o operador de abertura morfológica de X com um elemento estruturante B. Neste caso todos os elementos em X menores que B foram erodidos, porém as regiões não erodidas completamente não tiveram suas área reduzidas graças ao operador de dilatação que sucede o de erosão.

$$\gamma_B(X) = \delta_B(\varepsilon_B(X)) \tag{3.4}$$

O fechamento morfológico tem a mesma finalidade da abertura, isso é, eliminar algumas estruturas e preservar as demais. A diferença entre elas é que, agora, preservam-se as estru-

Figura 3.7: *Abertura morfológica extraída de Soille, P. (2013) [Soi13]*

turas do fundo que estão inseridas nos objetos. A formulação é dual da abertura morfológica e é descrita pela equação 3.5. A figura 3.8 demonstra a aplicação da operação de fechamento morfológico sobre um conjunto X usando um elemento estruturante B. Neste caso elementos do complemento da imagem menores que o elemento estruturante serão erodidos, o que implica que na imagem original essas regiões serão preenchidas, porém as outras regiões que não pertencem ao complemento da imagem se manterão inalteradas graças à operação de erosão que segue a de dilatação.

$$\phi_B(X) = \varepsilon_B(\delta_B(X)) \tag{3.5}$$

Figura 3.8: *Fechamento morfológico extraído de Soille, P. (2013) [Soi13]*

De acordo com Soille, P. (2013) [Soi13] o gradiente morfológico de uma imagem pode ser obtido de diversas formas, isso porque existe mais de uma maneira de se obter o equivalente discreto do gradiente contínuo. O gradiente de Beucher é obtido pela diferença entre a dilatação e a erosão morfológica de f por um elemento estruturante B e pode ser representado pela equação 3.6 de acordo com Soille, P. (2013) [Soi13]:

$$\nabla_B(f) = \delta_B(f) - \varepsilon_B(f) \tag{3.6}$$

O operador cartola de acordo com Soille (2013) [Soi13] suprime regiões homogêneas

e destaca regiões menores que o elemento estruturante, no caso do operador cartola-dual essas regiões destacadas devem ser escuras (por ex. ter valores inferiores aos pontos em seus arredores) e menores que o elemento estruturante. Esta operação é definida pela subtração do fechamento da imagem pela imagem original e é descrita pela equação 3.7:

$$BTH_B(f) = \phi_B(f) - f \tag{3.7}$$

Onde:

- BTH significa "Black Top-Hat" que é a definição usada por Soille, P. (2013) [Soi13] para descrever a operação cartola-dual.
- $\phi_B(f)$ é a operação de fechamento sobre a imagem f pelo elemento estruturante B.

Todos estes operadores podem ser definidos para imagens em níveis de cinza[2], denotada como sendo, de acordo com Soille, P. (2013) [Soi13]:

$$f : \mathcal{D}_f \subset \mathbb{Z}^n \to \{0, 1, \ldots, t_{max}\}$$

Onde t_{max} é o valor máximo que um elemento de $\mathcal{D}_f$ pode assumir. Neste caso f é um mapeamento do subconjunto $\mathcal{D}_f$ para um dos valores do conjunto $\{0, 1, \ldots, t_{max}\}$. Sendo assim cada ponto do domínio imagem $\mathcal{D}_f$ será associado por f a um valor que representará um tom de cinza.

[2] A dilatação ou erosão no caso das imagens em níveis de cinza toma o máximo ou mínimo respectivamente em cada ponto de $\mathcal{D}_f$ dentro da região do elemento estruturante.

Capítulo 4

Inacity

O resultado principal desta proposta é o projeto INACITY (INvestigate and Analyse a CITY), que compreende tanto uma interface para sistemas de informações geográficas quanto uma plataforma para coleta e análise de imagens por meio de técnicas de visão computacional. Além disso ela também foi criada tendo em vista a possibilidade de ser usada como uma Interface de Programação de Aplicações (API) através da exposição dos serviços de filtragem e coleta de imagens seguindo o design de arquitetura REST.

A fim de disponibilizar para um usuário final os serviços contidos na API, foi desenvolvida uma interface de usuário que a consuma. Neste capítulo serão descritas os detalhes técnicos para o consumo da API, a infraestrutura a nível de hardware implementada, as arquiteturas de software do lado do servidor ("back-end") e do lado cliente ("front-end") e os casos de uso da plataforma do ponto de vista do usuário do "front-end".

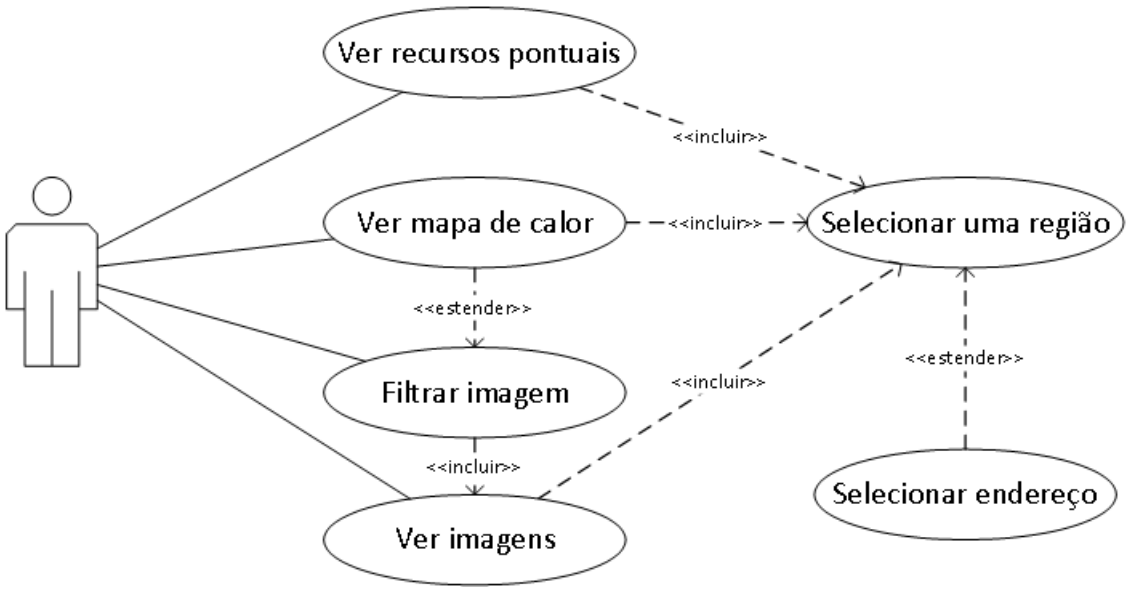
4.1 Casos de uso

Neste seção serão apresentados os principais casos de uso da plataforma: a exibição de objetos geográficos pontuais (por ex. pontos de ônibus) na subseção 4.1.1; a coleta de imagens de ruas e regiões na subseção 4.1.2 e a visualização de imagens nas quais características urbanas (por ex. árvores) são filtradas e suas concentrações são exibidas no mapa digital na forma de um mapa de calor na subseção 4.1.3. Um mapa de calor é uma coloração do mapa digital em que cada cor indica uma densidade maior ou menor de uma dada característica na

região colorida. A coloração adotada varia entre o vermelho para indicar menores densidades e verde para maiores densidades [WF09]. Note que a formação do mapa de calor é uma consequência da filtragem de imagens e portanto não é um caso de uso. Na figura 4.1 é apresentado o diagrama de casos de uso do "front-end" onde:

- Ver recursos pontuais: permite que o usuário localize dentro de uma região onde estão localizados objetos estáticos (por ex. pontos de ônibus, farmácias, escolas, etc.). Este caso de uso será detalhado na subseção 4.1.1.
- Ver mapa de calor: se refere à possibilidade de visualizar sobre um mapa a concentração de uma característica urbana (por ex. árvores) sobre uma região na forma de um mapa de calor. Este caso de uso será detalhado na subseção 4.1.3.
- Filtrar imagem: se refere à possibilidade do usuário, durante a visualização das imagens de uma rua ou região, selecionar um filtro de imagens (por ex. árvores) e visualizar as imagens filtradas e também a concentração da característica filtrada (por ex. árvores) sobre o mapa. Este caso de uso será detalhado na subseção 4.1.3.
- Ver imagens: se refere à possibilidade de visualizar as imagens presentes na região selecionada ou da rua selecionada pelo usuário. Este caso de uso será detalhado na subseção 4.1.2.

Figura 4.1: *Diagrama de casos de uso*



4.1.1 Interface para sistemas de informações geográficas (SIGs)

O uso mais simples da plataforma consiste na possibilidade de visualizar sobre um mapa os objetos estáticos oriundos de SIGs (por ex. "OpenStreetMaps") vinculados à plataforma. A figura 4.2 ilustra um diagrama de sequência para a busca de uma dada característica urbana (por ex. pontos de ônibus) numa região selecionada; o usuário seleciona a região de interesse e inicia a requisição por pontos de ônibus dentro desta região, a requisição é enviada para o servidor através de uma chamada ao "controller" chamado "MapMiner", que retorna ao "front-end" uma lista dos pontos de ônibus dentro da região selecionada e o "front-end" exibe-os ao usuário como pode ser visto na figura 4.3; cada ponto vermelho corresponde à um ponto de ônibus, em algumas ruas os pontos de ônibus aparecem em pares, isso de deve ao fato que nestas ruas há um ponto de ônibus para cada sentido da rua e eles estão aproximadamente um de frente para o outro. Neste exemplo o SIG utilizado foi o "GeoSampa" [dSP].

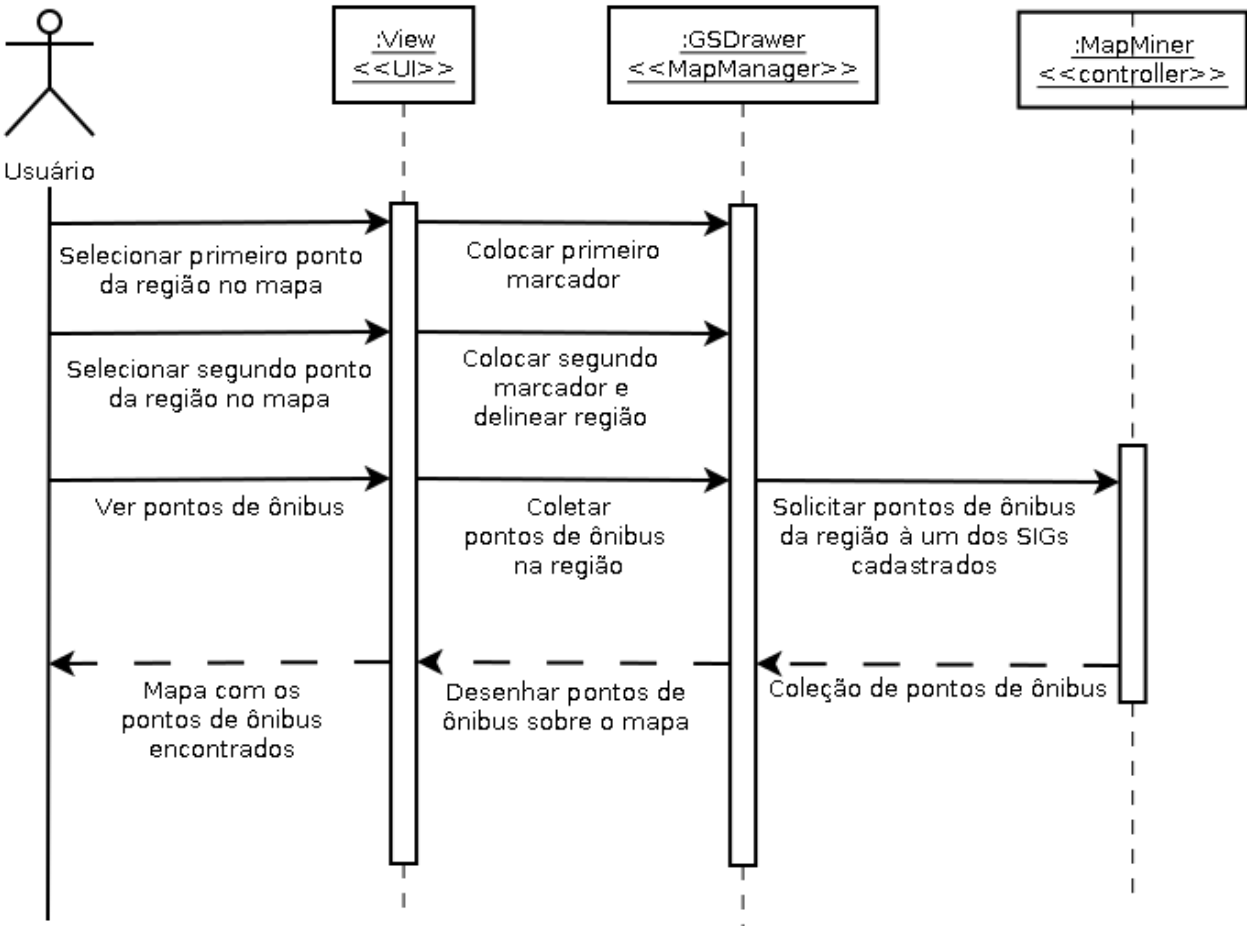
4.1.2 Visualização de uma rua ou região

A função de visualização permite que o usuário inspecione as imagens de uma determinada rua ou até mesmo de uma região inteira. Desta forma o sistema irá coletar imagens, através dos provedores de imagens vinculados a ele como o "Google Street View", e exibi-las ao lado do mapa, além de também indicar no mapa o ponto de onde a imagem em exibição foi tomada como exibido na figura 4.5. A figura 4.4 ilustra o diagrama de sequência para a visualização das imagens de uma região selecionada pelo usuário.

4.1.3 Detecção e visualização de uma característica urbana

Esta opção permite ao usuário selecionar uma característica urbana (por ex. árvores) para ser filtrada nas imagens coletadas através de técnicas de visão computacional. Além disso também é possível visualizar na forma de um mapa de calor a densidade desta característica ao longo de uma rua ou região sobre o mapa. A fim de melhorar o desempenho computacional da plataforma os resultados relacionados à concentração da característica urbana sobre uma região ou rua são armazenados num banco de dados sem as imagens, desta forma também é

Figura 4.2: *Diagrama de sequência de visualização de uma característica urbana. O usuário através da interação com a interface exibida pelo "front-end" requisita pontos de ônibus dentro de uma região de interesse, a requisição é enviada a um "endpoint" no "controller" chamado "MapMiner" e este retorna ao "front-end" uma lista de pontos de ônibus para que ele exiba ao usuário.*



possível a visualização do mapa de calor sem realizar a filtragem das imagens novamente. Na figura 4.6 a cor vermelha indica pontos de maior densidade, e a cor verde representa regiões com menor concentração de árvores.

Figura 4.3: *Mapa de uma região de São Paulo com pontos de ônibus marcados*

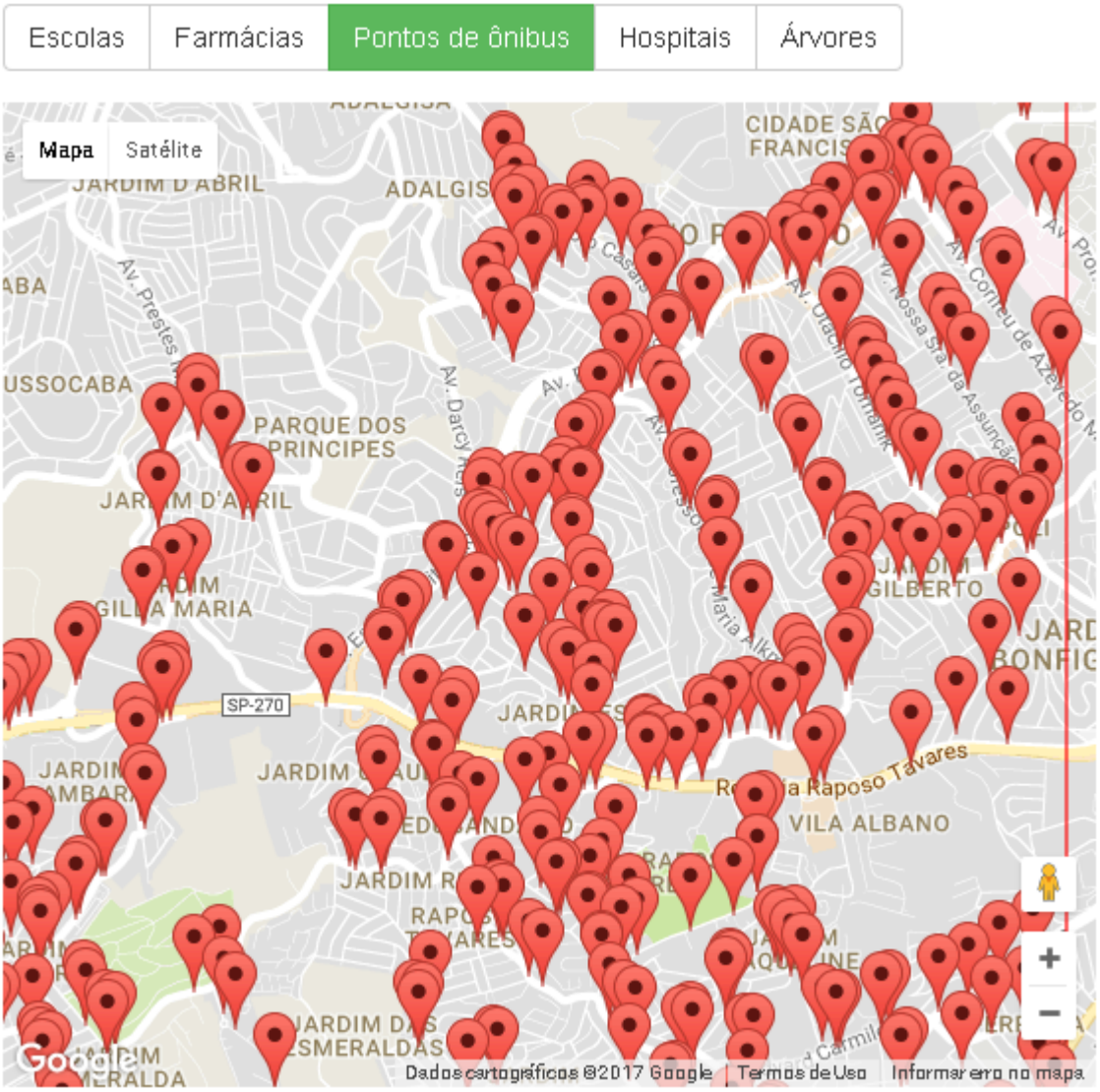


Figura 4.4: *Diagrama de sequência para a visualização de imagens de uma região*

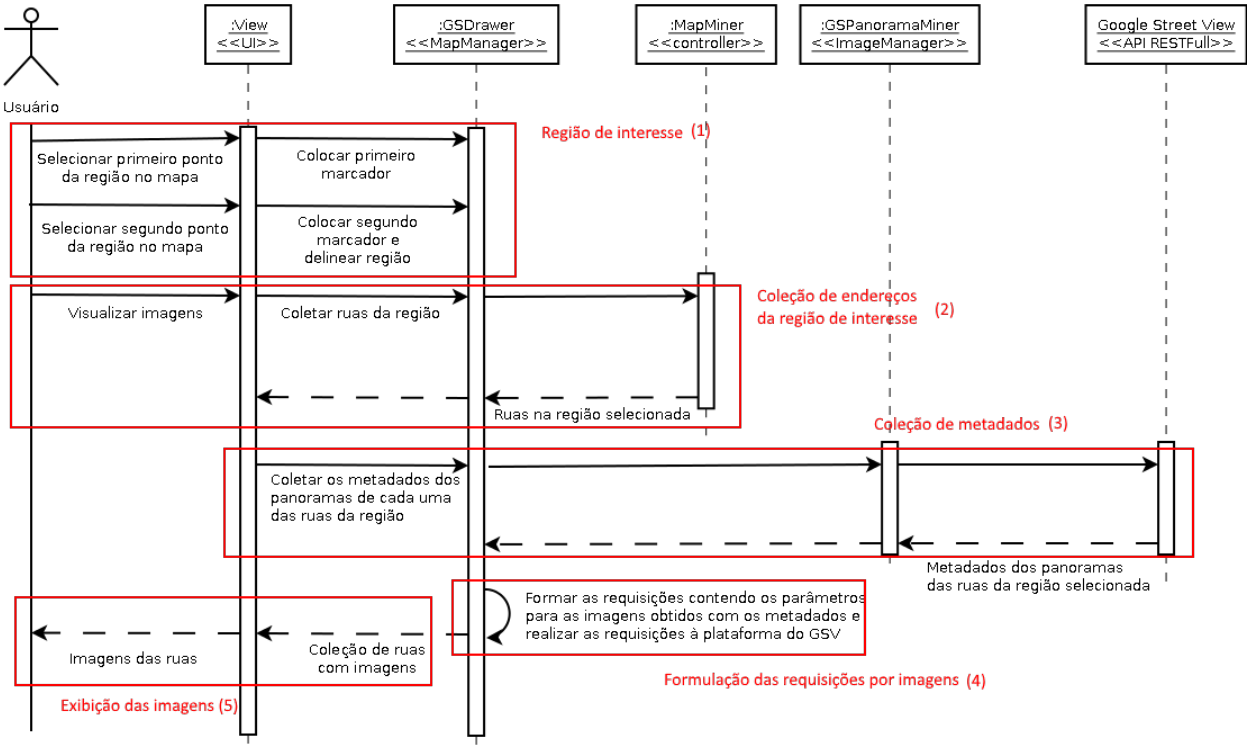


Figura 4.5: *Mapa com rua selecionada e suas respectivas imagens*

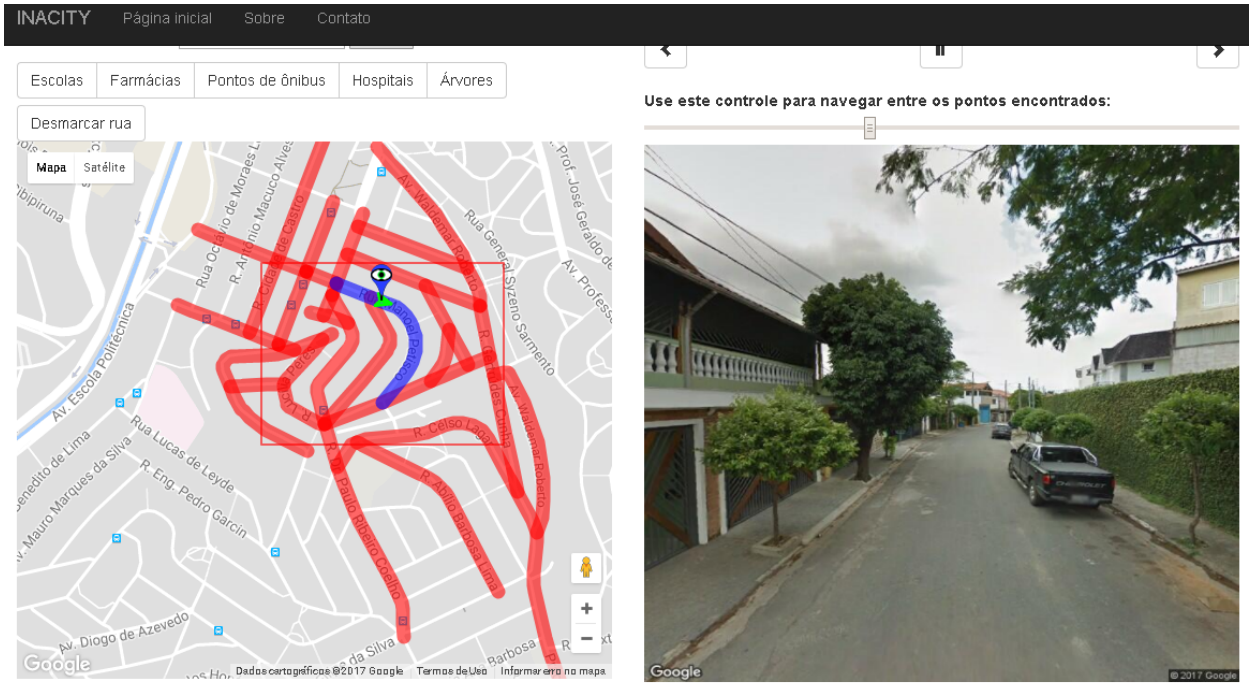
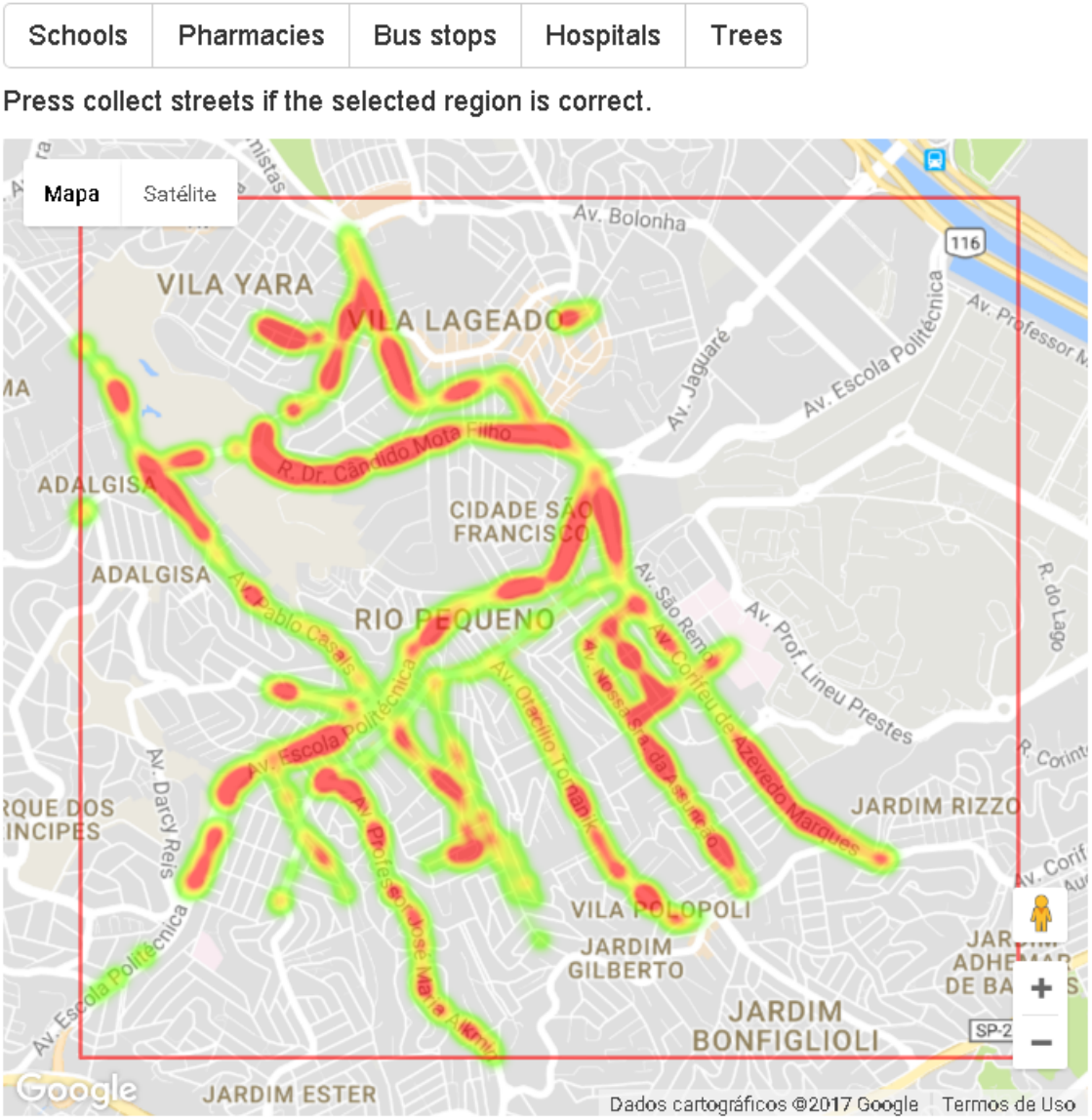


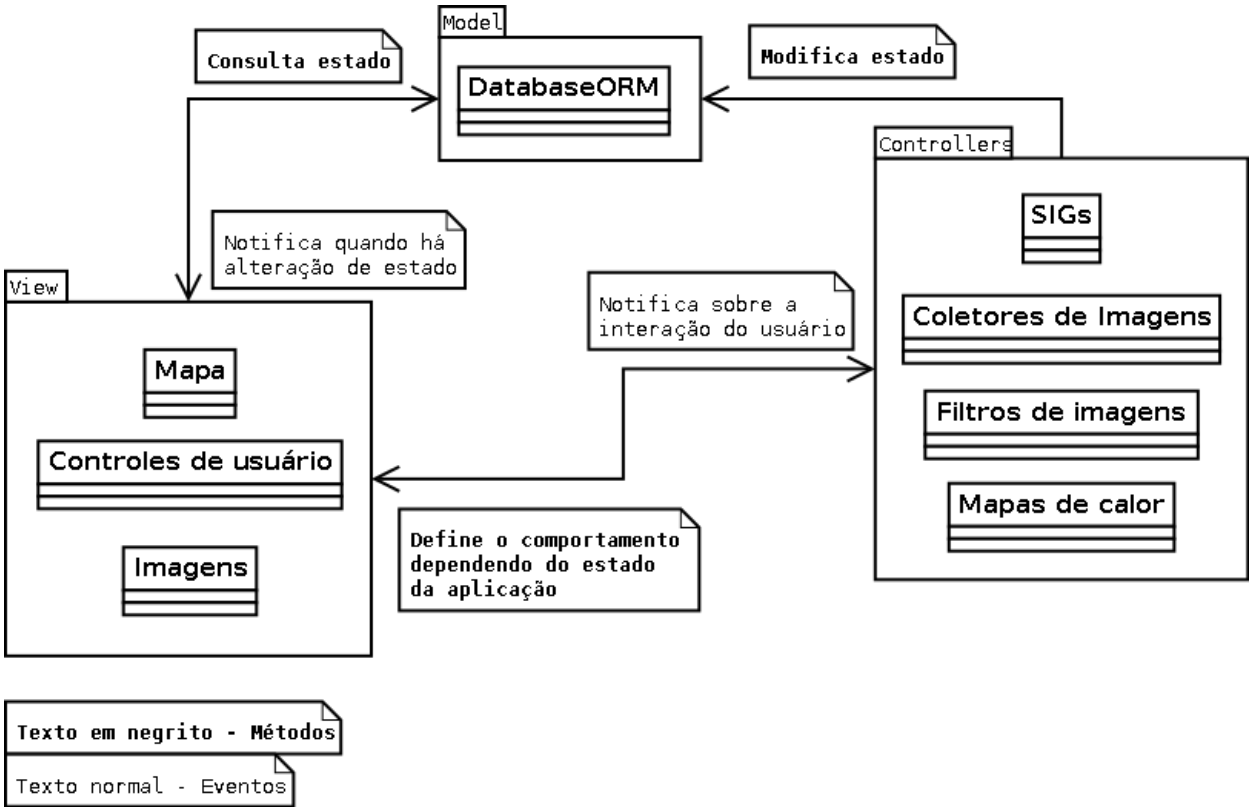
Figura 4.6: *Mapa de calor para árvores*



4.2 Proposta de arquitetura do servidor

A composição da arquitetura em alto nível compreende classes para integrar serviços de diferentes provedores e para processar requisições e imagens. As classes de integração irão servir como uma camada para abstrair os serviços oferecidos por sistemas de informação geográfica[1](por ex. "OpenStreetMap") e provedores de imagens (por ex. "Google Street View"). As classes usadas para processar as requisições e as imagens são chamadas de "controllers", pois pertencem à camada de Controle do modelo de arquitetura MVC adotado no projeto. Além disso estas classes também são responsáveis por intermediar o acesso ao banco de dados ("model") pela "view". A figura 4.7 ilustra a composição das classes da arquitetura do projeto INACITY organizadas de acordo com o modelo MVC.

Figura 4.7: *Através da interação do usuário com a "view" requisições podem ser feitas à "Controller" que por sua vez mantém o controle sobre o estado da "Model". A "Model" em conjunto com a "Controller" retornam à "view" dados que serão usados para determinar o estado e o comportamento da "view"*



Os componentes de integração contidos no lado do servidor ("back-end")[2] são categori-

[1]Sistemas de informação geográfica serão referidos algumas vezes como SIGs ao longo do texto

[2]De agora em diante, usaremos apenas a palavra "back-end" quando nos referirmos a arquitetura do lado do servidor.

zados como geográficos, "controllers", gerenciadores ("managers") e comparadores.

A categoria de componentes geográficos compreende abstrações do sistema relacionadas à características geográficas usadas para facilitar a comunicação entre a API do INACITY e a de SIGs. Os componentes geográficos são:

- "Ponto" (geográfico): Um objeto que representa uma coordenada geográfica (latitude e longitude);
- "Instalação" é um objeto que representa qualquer característica urbana com uma posição fixa (por ex. escolas, pontos de ônibus, etc.);
- "Ponto de calor": Um objeto que representa um valor escalar associado à um objeto do tipo "Ponto" (coordenadas geográficas);
- "Limite" (de uma região): Um conjunto de quatro pontos usado para delimitar uma região de interesse;
- "Rua": Um objeto que contém as informações de uma rua ou avenida da maneira como estas são armazenadas em um SIG específico (por ex. "OpenStreetMap");
- "Região": Um objeto que possui um "Limite" (de uma região) e contém uma coleção de todas as ruas dessa região;

Um objeto da categoria de "controllers" contém os pontos de acesso ("endpoints") que fornecem os serviços usados pelo "front-end" e podem ser usados como interfaces para programação de aplicativos por outras plataformas. Para uma descrição das chamadas implementadas no INACITY consulte o capítulo 8 de apêndices. Os "controllers" usados na API do INACITY são:

- "DBHeatMap": Responsável pelo armazenamento no banco de dados local de informações pertinentes aos mapas de calor;
- "ImageFilter": Responsável pelas chamadas que recebem como entradas uma imagem e uma característica urbana e devolvem a mesma imagem com a característica selecionada destacada para visualização e também informações para a formação do mapa de calor;

- "ImageMiner": Responsável pela coleta das imagens pertencentes à região ou rua selecionada;
- "MapMiner": Responsável pela coleta de "Instalações" nos sistemas de informações geográficas cadastrados.

A categoria de gerenciadores ("managers") compreende objetos usados internamente pelo servidor e são responsáveis por organizar coleções de provedores, esses provedores são classificados como "Models" na arquitetura MVC.

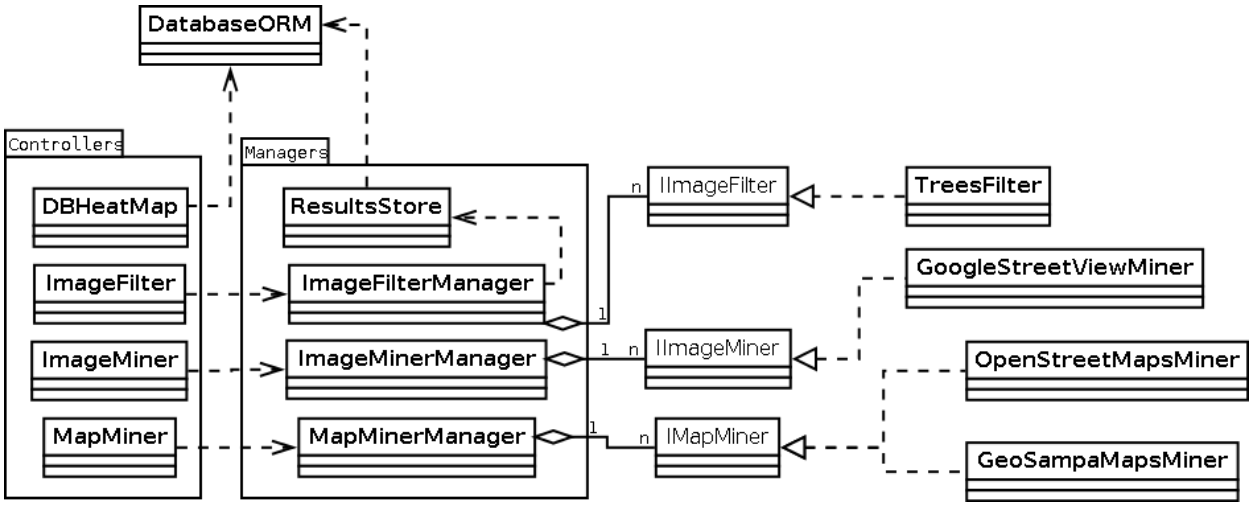
- "ResultsStore": Fornece meios para a consulta de informações de mapas de calor presentes no banco de dados local. Este "manager" é usado por questões de otimização e substituí a visualização ocasionada pela visualização de imagens através de algum filtro;
- "ImageFilter": Este objeto agrega e disponibiliza ao sistema a coleção de filtros registrados no sistema;
- "ImageMiner": Este objeto disponibiliza ao servidor todos os provedores de imagens registrados;
- "MapMiner": Este objeto disponibiliza ao servidor uma interface para acesso aos sistemas de informações geográficas cadastrados.

A última categoria de objetos é a de comparadores. Estes são responsáveis por resolver ambiguidades e inconsistências geradas durante a obtenção de recursos de um mesmo tipo através de um ou mais SIGs. O INACITY conta com os seguintes comparadores:

- Comparador de nós "OSM": Usado para comparar objetos do tipo "nó" do "OpenStreetMap" e excluir nós repetidos originados durante consultas em regiões;
- Comparador de pontos: Usado para averiguar se dois pontos geográficos correspondem à mesma coordenada geográfica e remover pontos equivalentes;
- Comparador de ruas: Usado para identificar e agrupar referências à uma mesma rua dentro de uma mesma coleção de ruas, ao invés de diversas coleções separadas.

Na figura 4.8 cada "controller" contém um conjunto de "endpoints" que fornecem serviços para o "front-end" ou, no caso de aplicações externas, fornece uma interface de programação de aplicativos. Cada "manager" é responsável por agrupar e fornecer diferentes recursos com funções similares (por ex. diferentes filtros de imagens). Os pontos usados para gerar mapas de calor são armazenados num banco de dados para serem fornecidos mais rapidamente.

Figura 4.8: *Diagrama do lado do servidor*



4.3 Proposta de arquitetura do lado do cliente

A arquitetura do lado do cliente compreende basicamente os componentes que formam a interface de usuário e os componentes usados para comunicação com o "back-end".

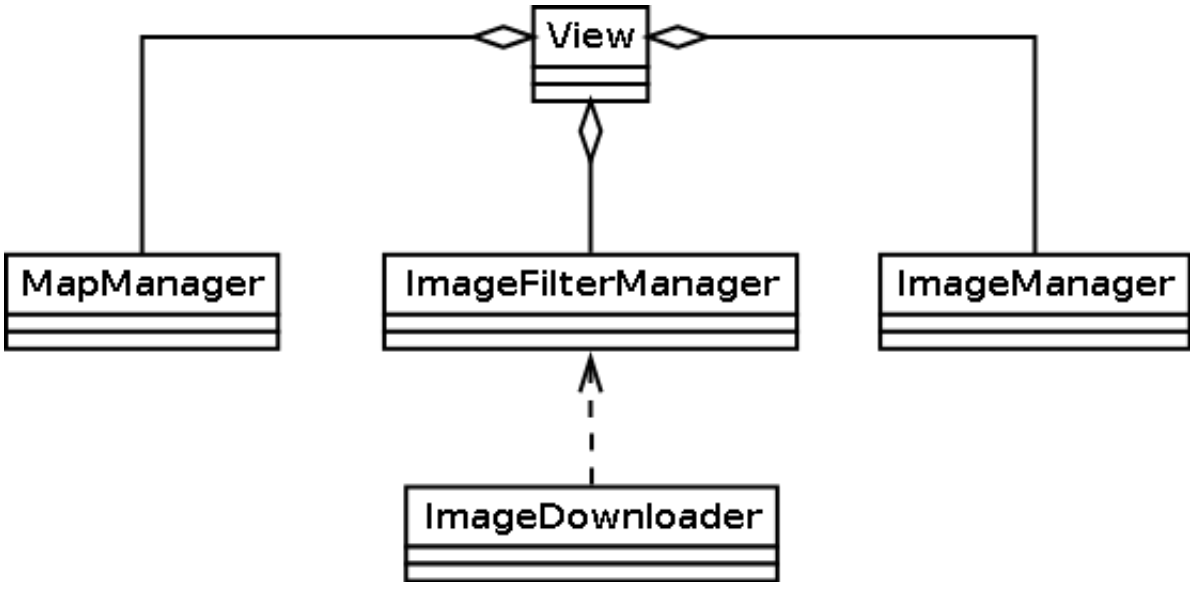
Assim como no lado do servidor o lado do cliente também faz uso de objetos categorizados como "managers", ou seja, objetos responsáveis por organizar e disponibilizar para outros objetos internos do "front-end" recursos diferentes que pertencem à camada "model" de acordo com o modelo MVC.

Na categoria de gerenciadores do "front-end" estão situados o gerenciador de mapas, o de panoramas e o de imagens. O gerenciador de mapas é responsável por exibir o mapa geográfico e pelo controle de marcações sobre o mapa. Estas marcações podem ser pontos de calor, pontos de ônibus, farmácias, escolas, etc. O gerenciador de panoramas é responsável por organizar objetos chamados de "Panoramas" que são oriundos do "Google Street View" (GSV). Estes objetos incluem informações a respeito de imagens associadas a uma localização

geográfica específica e também as imagens associadas a esta localização. O gerenciador de panoramas irá consultar a disponibilidade de um panorama em uma dada localização, se o panorama estiver disponível o gerenciador irá solicitar diretamente ao GSV as imagens associadas a ele e as informações associadas a este panorama como direção frontal do veículo da "Google" que coleta as imagens.

Na figura 4.9 o componente "View" representa a interface de usuário propriamente dita, "MapManager" o gerenciador de mapas, "ImageManager" o gerenciador de imagens, "PanoramaManager" o gerenciador de panoramas e "ImageDownloader" o componente auxiliar usado para permitir ao usuário o download das imagens sendo exibidas pelo "ImageManager".

Figura 4.9: *Diagrama de componentes da interface de usuário*



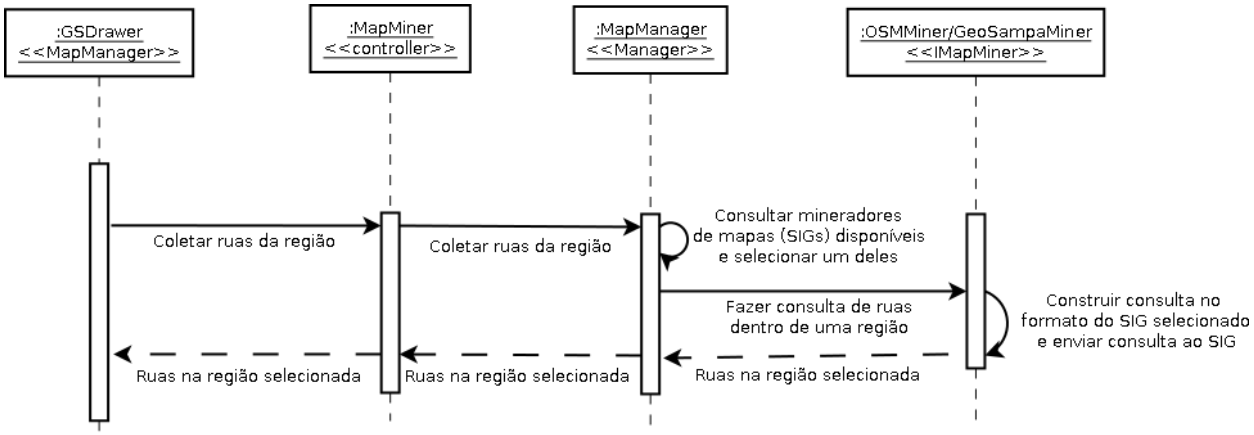
4.4 Heurísticas

A fim de implementar a plataforma proposta e realizar a integração de outras plataformas com esta foram necessárias diversas soluções para resolver os seguintes problemas:

- Como coletar as ruas de uma região;
- como coletar as imagens de uma rua ou uma região; e
- como direcionar a câmera corretamente.

Coletar as ruas de uma região é necessário para que seja possível disponibilizar ao usuário a opção de se inspecionar uma rua individualmente e também para poder coletar informações sobre cada rua em uma outra plataforma. Sem esta heurística somente o mapa fornecido da forma como ele é disponibilizado poderia ser visto. A figura 4.10 ilustra o diagrama de sequência usado para coletar as ruas de uma região. Primeiramente o usuário interage com o sistema e seleciona uma região de interesse, os limites dessa região são codificados na linguagem que será interpretada pelo sistema de informação geográfica (SIG) que retornará quais ruas existem dentro destes limites. Na implementação atual do INACITY os únicos SIGs em uso são: o "OpenStreetMaps"e o "GeoSampa". A seleção entre qual SIG usar é feita pela classe "MapMinerManager" ilustrada na figura 4.8; caso o usuário selecione pontos de ônibus dentro da região de São Paulo o "GeoSampa"será usado, caso contrário será usado o "OpenStreetMaps".

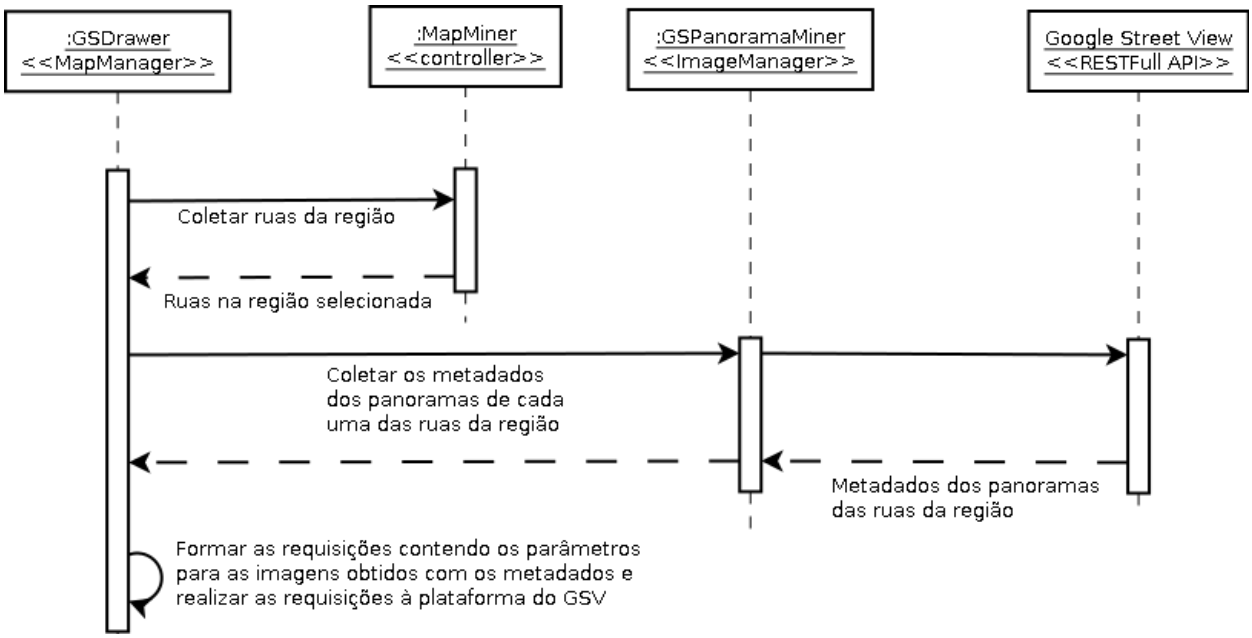
Figura 4.10: *Diagrama de sequência para a coleta de ruas de uma região. Este diagrama foca no processo de coleta de ruas feito pelo servidor sem considerar a interação do usuário e portanto o ator (usuário) e a classe de interface "View"foram omitidas (veja a figura 4.4 para um exemplo que leva o usuário e a classe de interface em consideração).*



Na implementação atual é usada a plataforma "Google Street View" (GSV) para a obtenção de imagens urbanas. Uma requisição à plataforma GSV deve conter a localização (expressa pela latitude e longitude) da imagem desejada, e opcionalmente pode conter as angulações horizontal ("heading") e vertical ("pitch") da câmera e por fim um raio que define até qual distância a câmera pode estar da localização informada para caso a imagem não esteja disponível naquela localização exata. É com base nestes parâmetros que as imagens são requisitadas ao GSV. A localização das imagens é obtida com base nos dados

obtidos pelos SIGs durante a fase de coleta de características ou ruas urbanas. As angulações da câmera são obtidas diretamente pelo GSV também, pois além de permitir a requisição por imagens o GSV também disponibiliza o acesso a objetos chamados "panoramas", que fornecem informações a respeito das imagens disponíveis em um local. Estas informações incluem a direção frontal do veículo ("heading"), com relação ao norte verdadeiro, assim como a inclinação do veículo ("pitch"), com relação ao horizonte. Estas duas direções (frontal e inclinação do veículo) são usadas como parâmetros de "heading"e "pitch"na requisição por imagens ao GSV. O parâmetro raio foi definido arbitrariamente como 10 metros, implicando que a tolerância máxima de distância entre uma localização e a câmera pode ser de 10 metros, se dentro deste raio não houver nenhum panorama disponível então a requisição é descartada.

Figura 4.11: *Diagrama de sequência para a coleta de imagens de uma região. Este diagrama foca no processo de coleta de imagens feito pelo "front-end". O processo para obtenção de imagens de uma rua é similar, a diferença é que uma das ruas coletadas deve ser selecionada antes da requisição por imagens. A parte de interação do usuário foi omitida para simplificar o diagrama (o diagrama da Fig. 4.4 ilustra a interação do usuário durante o processo de requisição por imagens). Note que como a plataforma de imageamento usada é a do GSV as requisições são feitas diretamente à plataforma do GSV ao invés de serem intermediadas pelo "back-end" do INACITY.*



Capítulo 5

Heurísticas de segmentação de imagens

Neste capítulo serão apresentadas as implementações de cinco heurísticas de segmentação de imagens usadas para detectar árvores nas imagens obtidas através do "Google Street View".

Há uma grande variedade de técnicas e sensores empregados na detecção de árvores e vegetação no geral e apresentados em diversos artigos [YZMG09, ATBS11, MVS12, LZL+15, WBH+16, BLMPN17, LRS17, SNRP17]. Em termos de linha de visada a vegetação é imageada por satélites, drones e no nível do solo (veja as figuras 5.1 e 5.2 para exemplos). Em termos de sensores, são usados sensores passivos multiespectrais, ou câmeras RGB, ou até sensores ativos como LiDAR ("Light Detection and Ranging"). Além disso, também há grande variabilidade de tamanho e resolução dos sensores.

De acordo com Wegner et al. (2016) [WBH+16] foi observado, empiricamente, que o uso de múltiplos pontos de vista para o reconhecimento de objetos 3D melhora a taxa de precisão

Figura 5.1: *Exemplo de imagem aérea [U.S]*

Figura 5.2: *Image de satélite [Ind]*

média de 42% para 71% e também a de reconhecimento de espécies de árvores de 70% para 80%.

No artigo de Parmehr et al. (2016) [PAF16] é proposto um método para se mapear a cobertura da copa das árvores (veja a figura 5.3 para um exemplo) através da fusão de imagens de satélite e LiDAR aéreo. Os autores demonstram que a fusão dos dois tipos de imagem superam limitações encontradas no uso dos métodos individualmente para o mapeamento de coberturas de copas de árvores. Os principais problemas no uso do LiDAR para este tipo de mapeamento, de acordo com os autores, acontece no contexto urbano em função da dificuldade em alguns casos de se distinguir entre árvores e edifícios quando estes estão muito próximos. Sendo assim as imagens multiespectrais de satélite (imagens formadas pelos canais RGB e infravermelho no artigo de Parmehr et al. (2016) [PAF16]) melhoram a qualidade do mapeamento complementando as informações obtidas pelo LiDAR aéreo.

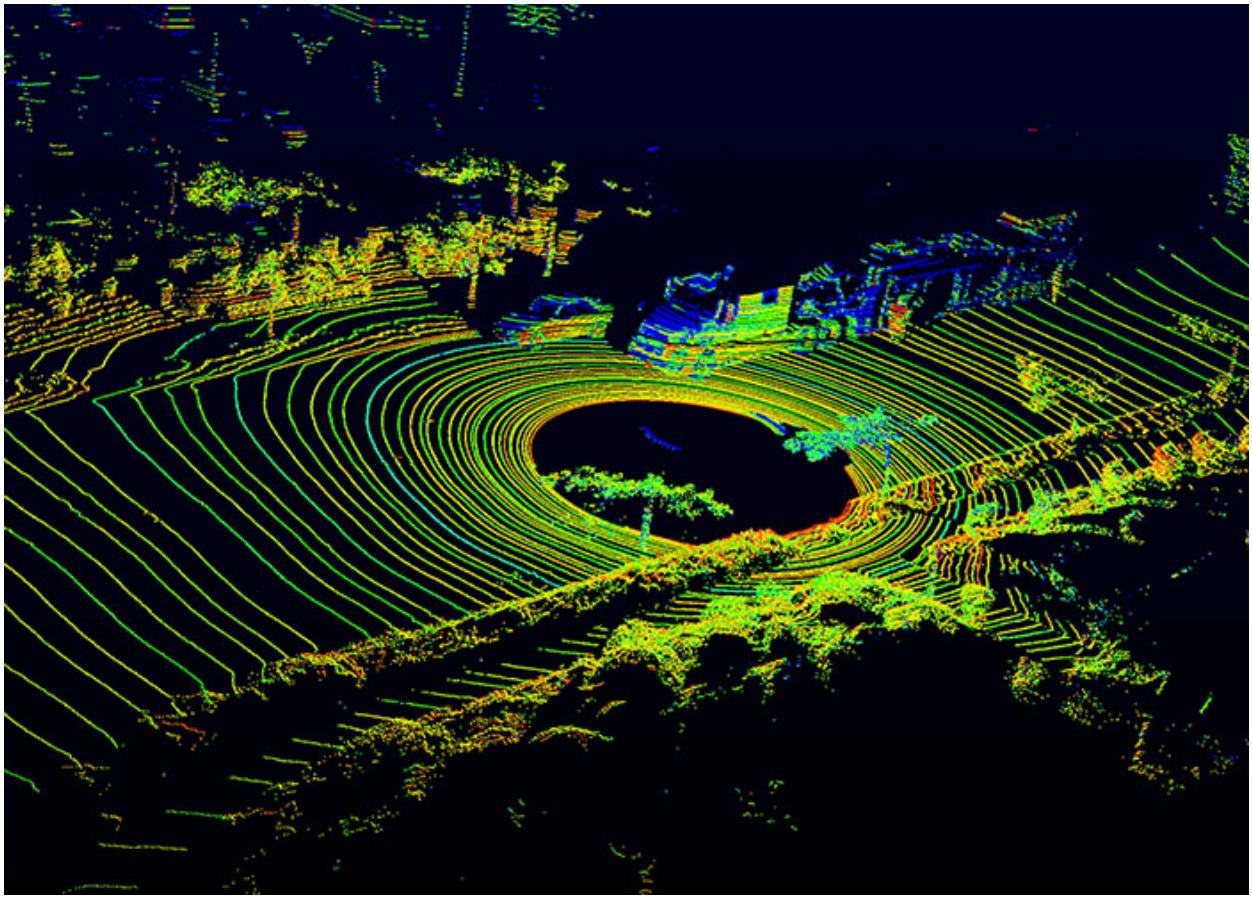
A plataforma do "Google Street View" (GSV) permite a obtenção de imagens coloridas (RGB) no nível do solo, que são obtidas através de um conjunto de câmeras RGB e LiDAR. A figura 5.4 ilustra o veículo usado para obtenção das imagens do GSV e a figura 5.5 ilustra uma imagem obtida com o uso dos sensores LiDAR também presentes no veículo do GSV. além de permitir que a busca por imagens seja feita através de geolocalização (latitude e longitude).

Para este tipo de imagem (RGB), a fim de detectar árvores e demais tipos de vegetação similar (por ex. vegetação com cor predominantemente esverdeada) foram aplicadas técnicas de segmentação baseada em diferentes espaços de cor e em textura. Na seção 5.1 são descritos os diferentes filtros de árvores e vegetação experimentados.

Figura 5.3: *Cobertura de árvores no cemitério de Crow Hill em Indiana nos Estados Unidos. Imagem extraída de [KIB].*

Figura 5.4: *Veículo usado para coletar as imagens da plataforma GSV. No topo do carro ficam as câmeras RGB e sensores LiDAR usados. [Ars]*

Figura 5.5: *Imagem obtida pelos sensores LiDAR. Cada anel é criado por uma laser rotatório individual. Esta imagem contém um total de 64 lasers. [Ars]*



5.1 Segmentação da imagem

A plataforma INACITY proporciona ao pesquisador um ambiente flexível para que ele experimente com heurísticas de segmentação e extração de características de imagens, ou de metadados. No entanto, para proporcionar ao usuário final uma experiência de maior qualidade, comparamos cinco heurísticas de segmentação através de experimentos controlados usando um conjunto de dados (dataset) de imagens com as áreas verdes anotadas.

Quatro das heurísticas testadas são baseadas no método proposto por Li et al (2015) [LZL+15]:

- mt-li - método original como proposto no artigo
- mt-li1 - variação do método original (seção 5.1.1)
- mt-li2 - variação levando em conta o espaço de cor HSL (seção 5.1.2)
- mt-li3 - variação usando apenas informações espectrais (seção 5.1.3)

Uma última heurística foi baseada no método descritopor Brancati et al. (2017) [BDPFG17] que baseia-se em propriedades geométricas dos pixels das imagens no espaço YCbCr e que chamaremos no texto de mt-br.

5.1.1 Filtro com textura e segmentação por cor

Inspirado no trabalho de Li et al. (explicado no capítulo 2) apresentamos uma primeira variação que chamamos de "mt-li1". Ela se baseia em duas máscaras, uma gerada a partir da aplicação do operador cartola-dual sobre a versão em tons de cinza da imagem original e outra formada pela limiarização da imagem no espaço de cor HSV. Primeiramente a imagem em RGB é normalizada e depois é convertida para HSV. A normalização da imagem segue a equação 3.1 do capítulo 3 (veja a figura 5.12 para um exemplo de imagem RGB normalizada). De acordo com Sanin et al. (2012) [SSL12] essa normalização do espaço RGB é usada para se realizar a detecção de sombras em imagens coloridas. No caso do detector de árvores a normalização é usada para reduzir o efeito da iluminação e facilitar a limiarização feita nos canais de matiz e saturação do espaço HSV como exemplificado na figura 5.13.

Figura 5.6: *Imagem contendo árvores e vegetação, a área em vermelho é ampliada na Fig. 5.7.*

Figura 5.7: *Seção ampliada da Fig. 5.6 ilustrando uma região onde a folhagem forma uma textura heterogênea em função das sombras projetadas sobre a folhagem por si mesma.*

A folhagem da vegetação normalmente projeta sombras sobre outras partes da folhagem, como nas figuras 5.6 e 5.7. Esta observação motivou o uso do operador cartola-dual (vide equação 3.7) para extração de regiões heterogêneas na imagem. A figura 5.10 ilustra o efeito do operador cartola-dual sobre a imagem da figura 5.6. A máscara de regiões heterogêneas em conjunto com a filtragem por cor ajuda a distinguir entre vegetação e outros objetos verdes como muros, carros e placas (vide as figuras 5.8 e 5.9).

A segunda máscara é gerada a partir da imagem no espaço de cor HSV onde são selecionados intervalos de matiz ("hue") e de saturação ("saturation") correspondentes aos da folhagem das árvores e da vegetação. Estes intervalos foram obtidos através da maximização das métricas de precisão e redescoberta usando-se um conjunto de imagens anotadas ("groundtruth"), os valores usados assim como um detalhamento do processo de aprendizagem dos parâmetros do filtro serão expostos no capítulo 8 de apêndices.

As duas máscaras são então combinadas através de intersecção, em outras palavras, como as duas máscaras correspondem a duas matrizes binárias para tomar a intersecção delas basta multiplica-las ponto a ponto. Os pontos da nova matriz que possuem valor 1 representam pontos marcados como pertencentes à vegetação, os de valor 0 são pontos marcados pelo filtro como não pertencentes à vegetação. Após essa filtragem é realizada um procedimento

Figura 5.8: *Imagem urbana tomada no nível do solo extraída de Li et al. (2015) [LZL+15].*

Figura 5.9: *segmentação pelo método 'mt-li', que não leva em conta informações de textura da imagem, o que levou à classificação errônea da placa de trânsito como sendo vegetação. Imagem extraída de Li et al. (2015) [LZL+15].*

Figura 5.10: *Operador cartola-dual com elemento estruturante retangular de 15 por 9 pixels aplicado sobre a Fig. 5.6 em níveis de cinza.*

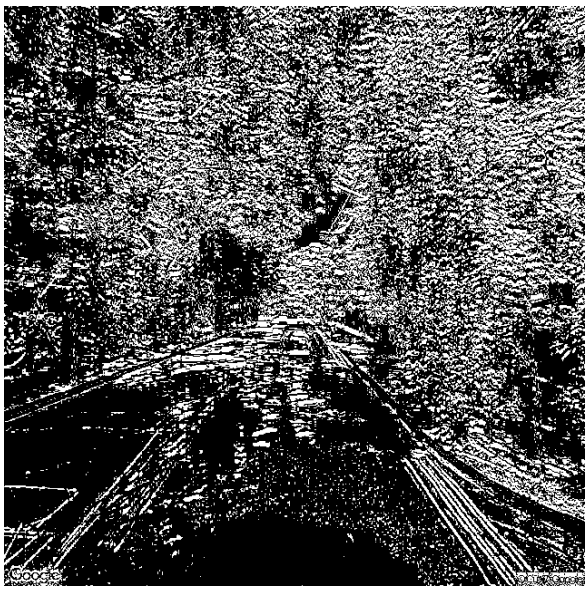


Figura 5.11: *Limiarização da Fig. 5.10 pelo produto $T * \mu(I_G)$ onde T é um parâmetro arbitrário definido no intervalo $[0, 1]$, I_G é a imagem convertida em níveis de cinza e $\mu(I_G)$ é o nível médio de cinza da imagem I_G.*

Figura 5.12: *Versão da Fig. 5.6 após a normalização do espaço de cor RGB.*

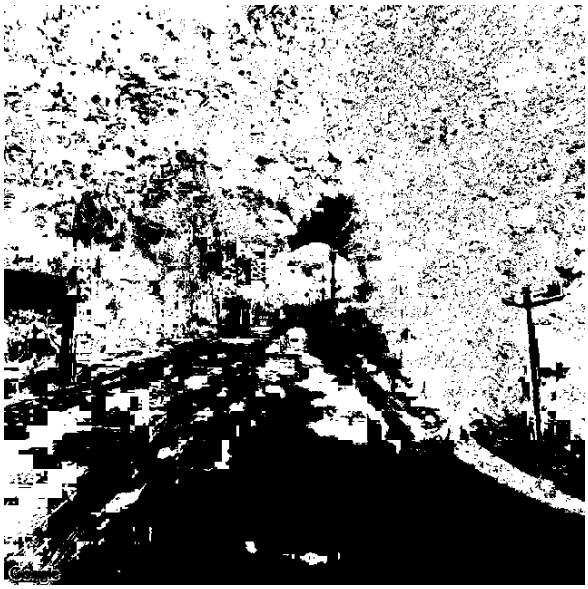


Figura 5.13: *Máscara binária formada pela limiarização (no espaço HSV) da Fig. 5.12.*

de filtragem de ruído do tipo sal e pimenta ("salt and pepper noise") através das operadores morfológicos de abertura e fechamento assim como é descrito por Soille, P. (2013) [Soi13]. Esta última filtragem ajuda a eliminar regiões pequenas, que na maior parte dos casos são falsos positivos e negativos. A figura 5.14 exibe os resultados do método "mt-li1" antes da aplicação dos operadores de abertura e fechamento morfológicos, a figura 5.15 exibe os resultados após. O algoritmo 2 apresenta os passos deste processo de filtragem de vegetação. Nele as linhas 3 até 7 são responsáveis pela máscara de textura, as linhas 8 até 11 são responsáveis pela obtenção da segunda máscara.

Figura 5.14: *Resultado da aplicação do método "mt-li1" sobre a Fig. 5.6 sem a filtragem pelos operadores morfológicos de abertura e fechamento.*

Figura 5.15: *Resultado da aplicação do método "mt-li1" sobre a Fig. 5.6 após a aplicação dos operadores de abertura e fechamento.*

Algoritmo 2: FILTRO DE VEGETAÇÃO PARA IMAGENS COLORIDAS

Entrada: I uma imagem no espaço de cor RGB

Saída: M a matriz binária demarcando a vegetação

início

Seja $Norm_I$ a imagem I após ser normalizada.

Seja $Gray$ a imagem $Norm_I$ convertida em uma imagem em níveis de cinza.

Seja STR_1 um elemento estruturante.

Seja T um limiar com valor entre 0 e 1.

Seja $\mu(Gray)$ o valor médio dos níveis de cinza da imagem $Gray$.

Seja M_1 a primeira máscara definida como:

$$M_1 = (BTH(Gray, STR_1)) > T * \mu(Gray).$$

Seja HSV_I a imagem $Norm_I$ após ser convertida para o espaço de cor HSV.

Sejam h_1, h_2, s_1 e s_2 intervalos para matiz e saturação respectivamente.

Seja M_2 a segunda máscara definida como:

$$M_2 = h_1 \leq HSV_I \leq h_2 \cap s_1 \leq HSV_I \leq s_2.$$

Seja M_C a matriz binária definida como: $M = M_1 \cap M_2$.

Sejam STR_CLOSE e STR_OPEN dois elementos estruturantes.

Seja M a matriz binária final definida como:

$$M = (M_C \bullet STR\_CLOSE) \circ STR\_OPEN$$

fim

retorna M

Observações sobre o algoritmo 2:

- A notação "$BTH(M, STR)$" é usada para representar o operador cartola-dual aplicado sobre a matriz M com o elemento estruturante STR.
- A notação $M \bullet STR$ é usada para indicar a operação de fechamento morfológico de uma matriz M por um elemento estruturante STR.
- A notação $M \bullet STR$ é usada para indicar a operação de abertura morfológica de uma

matriz M por um elemento estruturante STR.

- O símbolo "$>$" indica operador de limiarização onde valores abaixo ou iguais ao valor do limiar (por ex. T) devem ser mapeados para o valor "0" e valores acima do limiar devem ser mapeados para o valor "1".

- A notação "$x_1 \leq I \leq x_2$" indica que todos os valores da matriz "I" que estiverem fora do intervalo fechado "$[x_1, x_2]$" serão mapeados para o valor "0" enquanto que os valores dentro do intervalo serão mapeados para o valor "1".

- A notação "$M_1 \cap M_2$" representa neste caso uma multiplicação ponto a ponto entre as matrizes binárias "M_1" e "M_2".

5.1.2 "mt-li2": O espaço de cor HSL

Após análise exploratória com a ferramenta de visualização de cores "3D Color Inspector" de Barthel, U. K. [Bar], notamos que um aumento na saturação de cada ponto da imagem no espaço de cor HSL aumentava a distância entre as cores pertencentes à vegetação das demais cores. Isso motivou uma outra variação do método "mt-li1" que será chamada de "mt-li2" no texto a seguir. Nele faz-se as mesmas operações que em "mt-li1", exceto que antes de gerar a máscara baseada na segmentação do espaço de cor HSV a imagem primeiro é convertida para o espaço de cor HSL e o canal de saturação é multiplicado por um fator θ. A figura 5.16 apresenta o aplicativo "3D Color Inspector" de Barthel, U. K. [Bar] e uma imagem urbana. Do lado direito da figura está a visualização das cores da imagem no espaço de cor HSL e na figura 5.17 temos a mesma imagem após o canal de saturação ser multiplicado por 4. Nas figuras 5.18 e 5.19 é possível visualizar o espaço de cor pelo topo. Como esperado as cores com saturação maior que zero se distanciam do centro do cilindro que representa o espaço de cor HSL. Este fator (θ) pode ser escolhido arbitrariamente, ou estimado através de exemplos, como será discutido no capítulo 6 de resultados experimentais.

Figura 5.16: *3D Color Inspector - Imagem original*

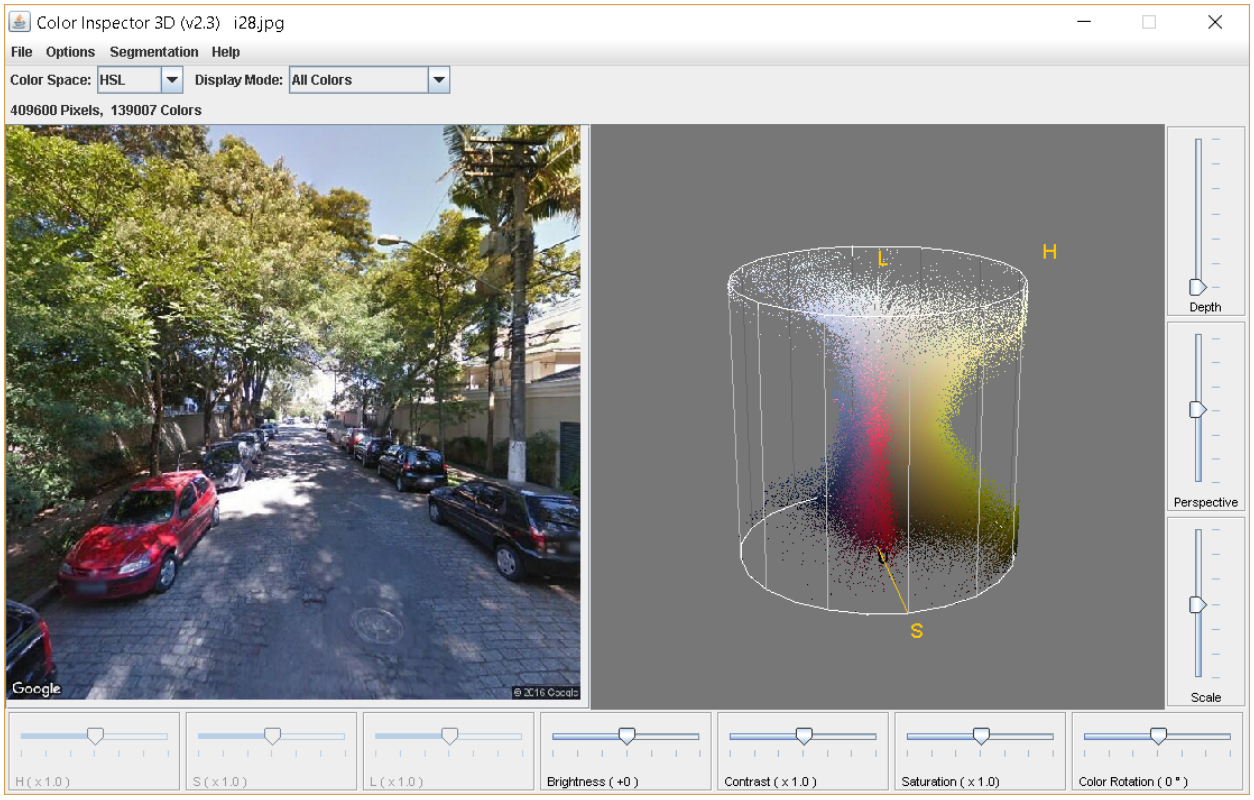


Figura 5.17: *3D Color Inspector - Imagem com saturação ampliada*

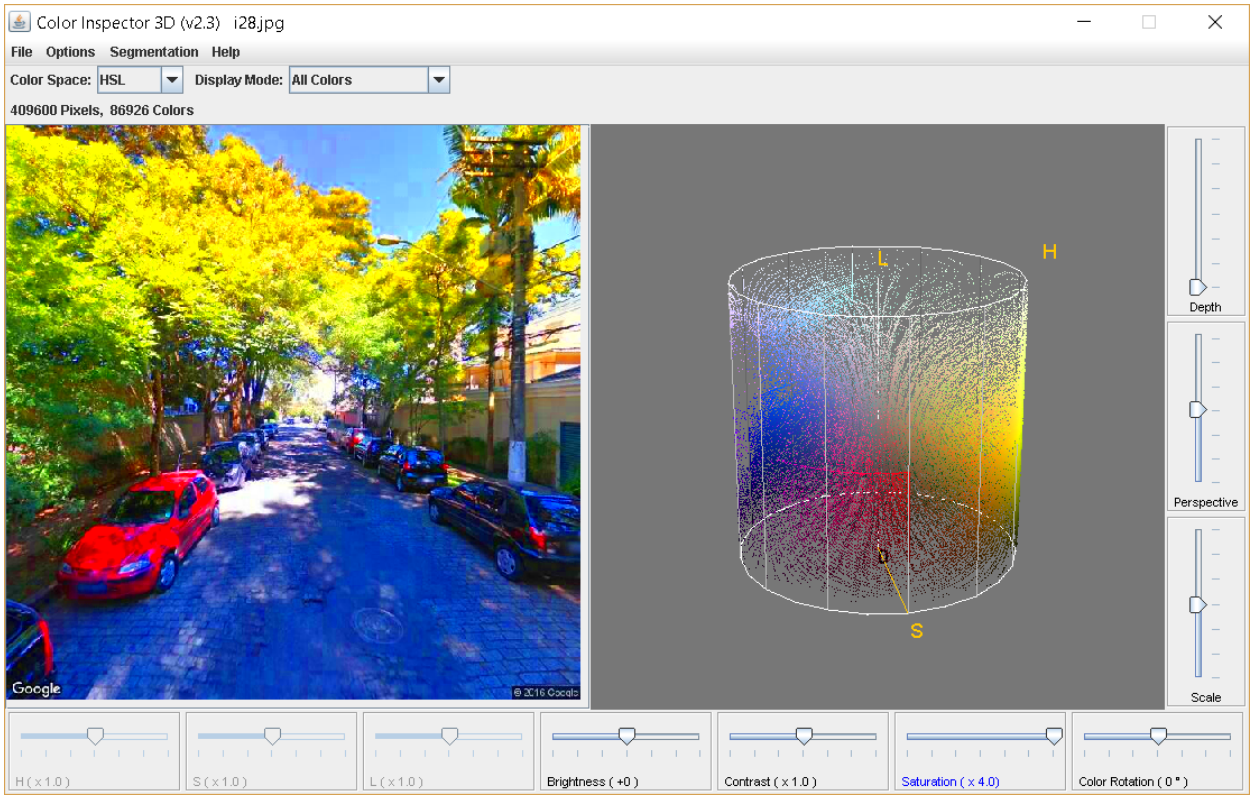
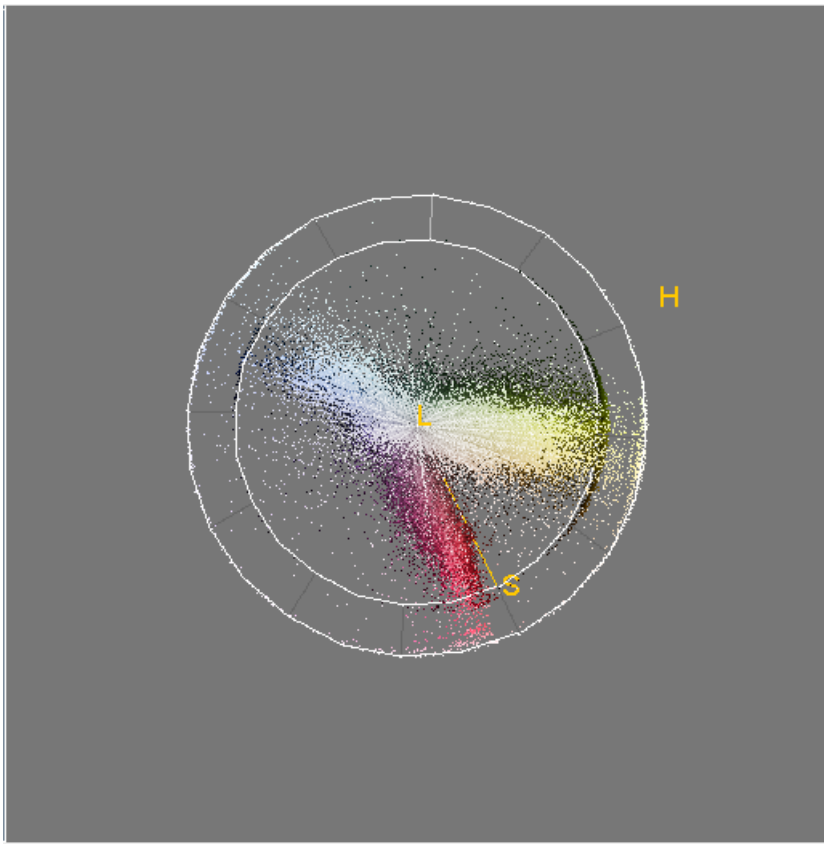


Figura 5.18: *3D Color Inspector - Visualização das cores pelo topo da imagem original*



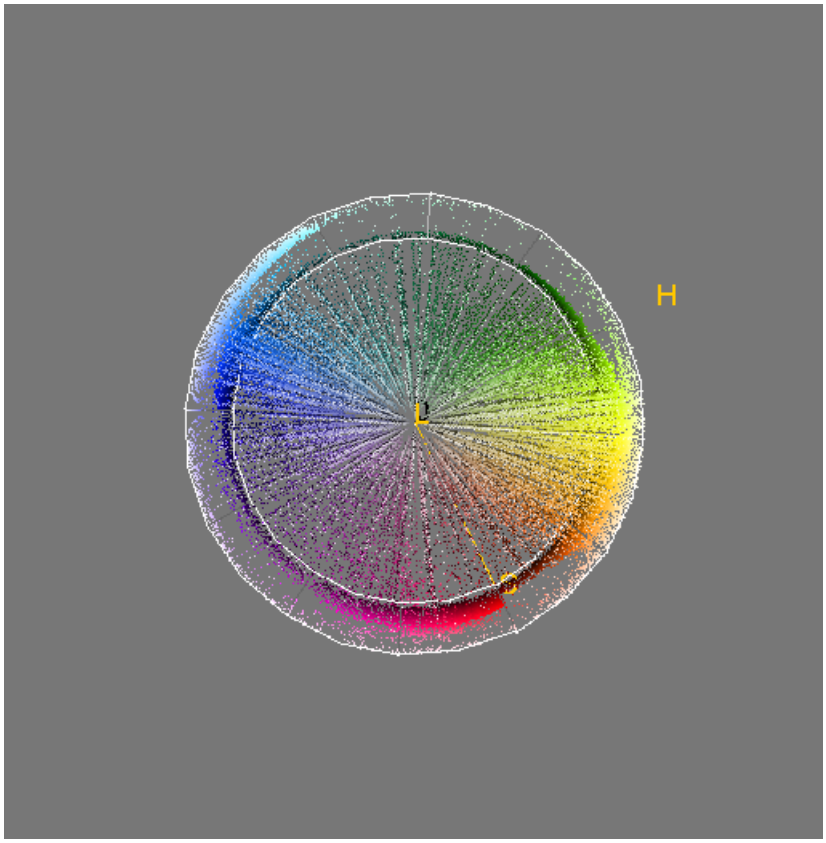
5.1.3 "mt-li-espectral": Filtros espectrais

Uma terceira heurística foi criada a fim de se testar a eficácia da filtragem baseada apenas em características espectrais, isso é, sem se levar em conta informações de textura que será chamada de "mt-li-espectral". O método "mt-li-espectral"realiza segmentações nos espaços de cor RGB, HSV e RGB normalizado. Cada segmentação dá origem a uma matriz binária. Estas matrizes são então combinadas através da operação de intersecção (multiplicação ponto a ponto).

5.1.4 "mt-br": Detector de pele modificado

O método "mt-br" é baseado no trabalho de Brancati et al. (2017) [BDPFG17]. Naquele artigo os autores propõem um método robusto a mudanças de iluminação para detecção de pele humana em imagens. Para tal, converte-se a imagem para o espaço de cor YCbCr e através de um processo de aprendizagem derivam os parâmetros de dois trapézios simétricos. Com base nestes, assim como em algumas heurísticas, cada ponto da imagem é classificado

Figura 5.19: *3D Color Inspector - Visualização das cores pelo topo da imagem com saturação ampliada*



como sendo ou não pele. Segundo esta ideia de buscar algum tipo de geometria formado no espaço de cor YCbCr cada ponto da imagem é projetado nos espaços formados pelas eixos YCb e YCr. Observou-se que pontos pertencentes a uma mesma classe de objetos tendem a formar simetrias com relação às suas projeções nos planos YCb e YCr. No caso das árvores percebeu-se que estes pontos formam objetos elípticos, onde uma elipse se forma no plano YCb e outra no YCr. Sendo assim para capturar os pontos pertencentes a estas elipses foram estimados os parâmetros de duas superfícies gaussianas, uma para cada elipse. Um ponto da imagem é classificado como sendo parte da vegetação se suas projeções nos planos YCb e YCr pertencem simultaneamente às respectivas elipses presentes em cada espaço. Foram usadas funções gaussianas para modelar as elipses e em função disso foi usada a desigualdade de Chebchev [AFC05] aplicada a vetores, como formulado por Chen, X. (2007) [Che07], para calcular qual o limiar, isto é, a distância a partir da média da gaussiana, no qual o volume da gaussiana corresponde a uma probabilidade maior que um fator arbitrário $\frac{n}{\epsilon}$ onde n é o número de dimensões do vetor e ϵ é um limiar arbitrário. A formulação da desigualdade de

Chebchev derivada por Chen, X. (2007) [Che07] é exibida na equação 5.1.

$$Pr\{(\vec{p} - E[\vec{p}])^\top \Sigma^{-1} (\vec{p} - E[\vec{p}]) > \epsilon\} \leq \frac{n}{\epsilon}, \forall \epsilon > 0 \tag{5.1}$$

Onde:

- $\vec{p} \in$ YCb ou $\vec{p} \in$ YCr representa um vetor do plano YCb ou YCr;
- Σ^{-1} é a matriz de precisão ou o inverso da matriz de covariância;
- $E[\vec{p}] = (E[\mathrm{Y}], E[\mathrm{Cb}])$ ou $E[\vec{p}] = (E[\mathrm{Y}], E[\mathrm{Cr}])$ representa a média aritmética das posições de todos os vetores projetados ou no plano YCb ou no YCr;
- n é o número de dimensões do vetor (neste caso $n = 2$);
- ϵ é um limiar arbitrário maior que zero.

Com base nesta equação a cada ponto da imagem é atribuído um valor binário (0 ou 1) que depende da distância do ponto com relação à média estimada para cada gaussiana, uma definida no espaço YCb e outra no espaço YCr. Através da atribuição de um valor à variável ϵ é possível se definir uma função de limiarização que classifica os vetores $\vec{p}$ em duas classes, uma em que a função de probabilidade da equação 5.1 possua valores maiores que a fração $\frac{n}{\epsilon}$ e outra em que não. A equação 5.2 descreve a função de distância de um vetor com relação à média de uma gaussiana baseada na equação 5.1.

$$D_\mu^\Sigma(\vec{p}) = (\vec{p} - \mu)^\top \Sigma^{-1} (\vec{p} - \mu) \tag{5.2}$$

Onde:

- Σ é a matriz de covariância formada a partir de todos os vetores $\vec{p}$ do plano YCb ou YCr;
- $\mu = E[\vec{p}]$ é a média formada a partir de todos os vetores $\vec{p}$ do plano YCb ou YCr.

A heurística elaborada para se realizar a segmentação da imagem se baseia na ideia de razões entre distâncias (conforme a equação 5.2) dos pontos da imagem e a média das médias dos vetores formados nos planos YCb e YCr. A equação 5.3 exprime a razão entre

as distâncias de um ponto da imagem original projetado nos planos YCb e YCr e a média das médias formadas por todos os pontos da imagem projetados nestes planos.

$$R(\vec{p}) = R(\vec{p_{Cb}}, \vec{p_{Cr}}) = \frac{\|\vec{p_{Cb}} - \vec{G_\mu}\|}{\|\vec{p_{Cr}} - \vec{G_\mu}\|} \tag{5.3}$$

Onde:

- $\vec{p_{Cb}}$ e $\vec{p_{Cr}}$ correspondem às projeções de um ponto da imagem original do espaço RGB nos planos YCb e YCr respectivamente. Para simplificar a notação a função $R(\vec{p_{Cb}}, \vec{p_{Cr}})$ será escrita simplesmente como $R(\vec{p})$ daqui em diante;
- $\vec{G_\mu} = \frac{E[\vec{p_{Cb}}]+E[\vec{p_{Cr}}]}{2}$ representa a média das médias dos vetores nos espaços YCb e YCr da classe de interesse (árvores neste caso) e será chamado de grande média no texto que segue.

Para um ponto da imagem original ser classificado como um ponto pertencente à classe de interesse ele deve estar a uma distância inferior ao limiar $\frac{n}{\epsilon}$ tanto no plano YCb quanto no YCr com relação as médias das respectivas funções gaussianas em cada plano e a razão das distâncias das projeções com relação à grande média deve ser compatível com o intervalo delimitado pelas razões mínima e máxima da classe de interesse (neste caso árvores). A equação 5.4 ilustra o processo de segmentação baseado nas equações 5.2 e 5.3.

$$f(\vec{p}) = \begin{cases} 1, & \text{se } D(\vec{p}) \leq \frac{n}{\epsilon} \wedge \min R(\vec{p}) \leq R(\vec{p}) \leq \max R(\vec{p}) \\ 0, & \text{em caso contrário} \end{cases} \tag{5.4}$$

Onde $\min R(\vec{p})$ e $\max R(\vec{p})$ representam as razões mínima e máxima toleradas da classe de interesse (árvores neste caso). Note que o vetor $\vec{p}$ representa um ponto no plano YCb ou YCr e portanto a notação $D(\vec{p}) \leq \frac{n}{\epsilon}$ corresponde à duas comparações. Contudo a função de razão $R(\vec{p})$ relaciona as duas projeções e é usada de maneira equivalente à $R(\vec{p_{Cb}}, \vec{p_{Cr}})$.

As variáveis ϵ, $\min R(\vec{p})$, $\max R(\vec{p})$ e as demais usadas nos outros métodos foram determinadas através do processo exemplificado no capítulo 8 de apêndices na subseção 8.3.2 de otimização de parâmetros (vide os algoritmos 8.1 e 8.2).

Capítulo 6

Resultados experimentais

Neste capítulo serão apresentados alguns resultados obtidos através do uso da plataforma, algumas limitações e resultados comparativos das heurísticas para segmentação de imagens realizados com base na aplicação dos métodos sobre um conjunto de imagens anotadas manualmente ("groundtruth").

6.1 Conjunto de dados

Para validar os algoritmos de detecção de árvores em imagens foi criado um conjunto contendo 100 imagens anotadas, que será chamado de "groundtruth". As imagens do "groundtruth"são todas imagens urbanas coletadas pela plataforma INACITY; para cada uma delas foi gerada uma outra imagem em que os pixels das áreas com vegetação são rotulados com o valor $(255, 0, 0)$ e os pixels das áreas que não contém vegetação foram rotulados com o valor $(255, 0, 255)$, de acordo com o espaço de cor RGB, como exemplificado pelas figuras 6.1 e 6.2. Estes dois valores não correspondem à cor de nenhum outro pixel em nenhuma das outras imagens, sendo assim qualquer pixel com outro valor é ignorado, ou seja, não é usado para a comparação dos algoritmos.

6.1.1 Construção do groundtruth

A produção do "groundtruth" foi feita de maneira manual e conservadora, isso significa que apenas regiões que não apresentavam ambiguidade com relação ao seu conteúdo foram

Figura 6.1: *Imagem original*

Figura 6.2: *Imagem anotada. Áreas em vermelho correspondem à vegetação, áreas em rosa não e os demais pontos são ignorados.*

marcadas, pontos da imagem ambíguos, como por exemplo os pontos na extremidade da folhagem de uma árvore que possui atrás de si outros objetos que não são vegetação (como por ex. a figura 6.3) e portanto podem ser interpretados de maneiras divergentes, não foram marcados, ou seja, são pontos simplesmente ignorados que não influenciam na contagem de verdadeiros positivos e verdadeiros negativos na avaliação das métricas de precisão e redescoberta. O "groundtruth" conta com 100 imagens e é hospedado e disponibilizado no repositório do "GitHub" [M.17].

Na figura 6.2 é ilustrado um exemplo de imagem anotada onde as regiões marcadas em vermelho representam áreas que contém folhagem de árvores e vegetação, as regiões em rosa são áreas que não contém nenhum tipo de vegetação e as demais áreas não marcadas não são utilizadas para comparação, isso é, elas são ignoradas durante o cálculo das taxas de precisão e redescoberta.

Em algumas imagens o tamanho das áreas que pertencem ou não à vegetação podem ser diferentes (ex. imagens com pouca vegetação); o tamanho de cada área é normalizado para que as taxas de precisão e redescoberta não sejam enviesadas nem pela quantidade de verdadeiros positivos (áreas anotadas como sendo vegetação), nem pela de verdadeiros negativos (áreas anotadas como não contendo vegetação).

Figura 6.3: *A região no interior do círculo laranja possuí pontos do céu que são muito parecidos com os da folhagem das árvores e vice-versa, este tipo de região ambígua não foi marcada.*

Tabela 6.1: *Relação de endereços e número de imagens coletadas para uso no "groundtruth"*

Endereço	Total de imagens
"5th Avenue" (Nova York)	3
Av. José Joaquim Seabra	5
Av. Quarto Centenário	15
R. Catanumi	2
R. Comendador Elias Zarzur	4
R. Heitor Penteado	13
R. Jorge Ward	13
R. Leyla Haddad	20
R. Pascoal Vita	25
Total	100

6.1.2 Localizações

As imagens presentes no "groundtruth" foram coletadas na região da cidade de São Paulo com exceção de 3 da "5th Avenue" de Nova York. A figura 6.4 ilustra uma região da cidade de São Paulo com as ruas usadas no "groundtruth" marcadas. As ruas foram selecionadas de maneira arbitrária. As figuras 6.5 e 6.6 exibem exemplos de imagens urbanas presentes no "groundtruth".

Figura 6.4: *Mapa com as ruas usadas para formar o "groundtruth". As marcas em azul representam as ruas selecionadas com seus nomes adjacentes nas caixas pretas e as marcas em vermelho são ruas próximas às ruas usadas no "groundtruth". A "5th Avenue" de Nova York não está presente no mapa.*

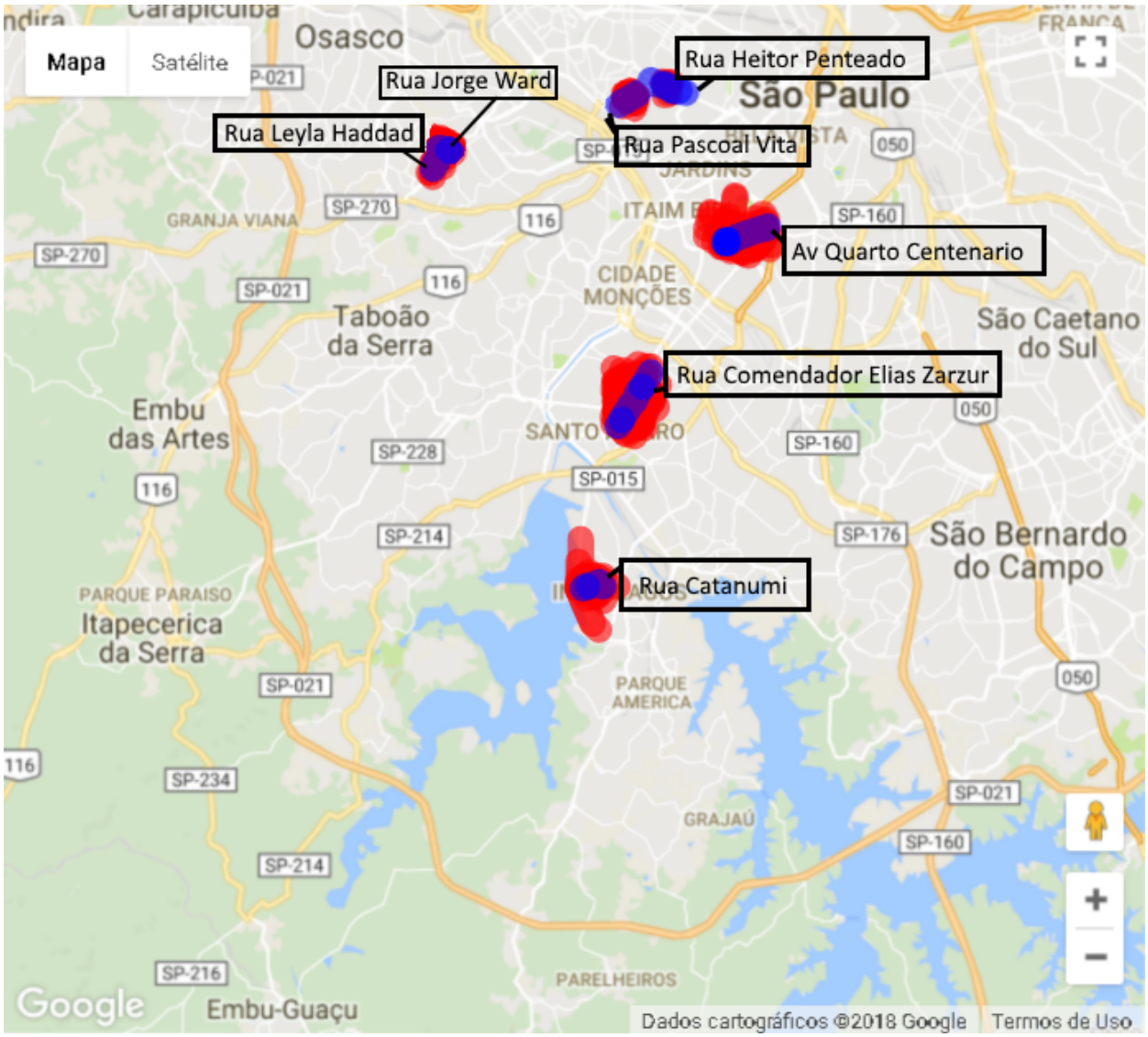


Figura 6.5: *Imagem contendo a maior área de vegetação maracada no "groundtruth".)*

Figura 6.6: *Imagem contendo a menor área de vegetação maracada no "groundtruth".*

6.2 Limitações da plataforma

Nesta seção serão abordadas limitações percebidas através da coleta de imagens na plataforma do "Google Street View" (GSV) e da integração entre o GSV e a plataforma INACITY.

6.2.1 Taxa de amostragem insuficiente

A primeira limitação notada com relação à cobertura do GSV foi que algumas regiões mais próximas da periferia, algumas alamedas muito estreitas e outras regiões sem um motivo claro, não possuem imagens disponibilizadas pelo GSV.

Algumas vezes as imagens de algumas ruas são escarças o que acarreta em distâncias muito grandes entre duas imagens consecutivas em uma mesma rua e com isso a taxa de amostragem acaba sendo menor do que a desejada.

Uma possível solução seria mesclar as bases de imagens do GSV com outras como o "Mapillary" ou até mesmo bases obtidas por colaboração coletiva, porém não é claro como fazer essa mistura entre as duas bases pois a qualidade das imagens, posições das câmeras com relação ao veículo e outros parâmetros variam entre as bases.

6.2.2 Direção da câmera

Algumas vezes a angulação da câmera não parece estar alinhada com a direção da rua, isso é, a câmera parece estar apontando num ângulo não paralelo ao eixo principal da rua. Esse tipo de problema ocorre em momentos em que o veículo do GSV, responsável pela aquisição de imagens, não está alinhado paralelamente com a rua, ou seja, está numa curva ou fazendo uma conversão entre faixas.

Experimentou-se a subtração de vetores entre dois nós[1] consecutivos de uma mesma rua (obtidos neste caso através do "OpenStreetMap") como um meio para se obter a direção paralela ao eixo da rua, porém por vezes estas coordenadas não estão localizadas sobre o eixo da rua, e sendo assim há variações indesejáveis com relação à angulação da câmera. Empiricamente foi observado que usar a direção frontal do veículo, obtida através dos metadados da imagem retornados pela API do GSV, traz resultados, com relação à angulação,

[1] Pontos localizados por latitude e longitude

mais consistentes do que os obtidos pela subtração de vetores. No caso das árvores é usada a direção frontal de veículo, porém para outros tipos de detecção, como por exemplo muros pichados, outras direções como as laterais do veículo podem ser mais úteis. Para se alinhar a câmera como as laterais do veículo basta se somar e subtrair 90 graus à angulação frontal do veículo. Porém, assim como a direção frontal do veículo nem sempre está alinhada com o eixo da rua, em alguns casos essa soma não resultará numa direção que seja perpendicular à rua. É importante dizer que, na maior parte dos casos, observou-se que o veículo está alinhado com a rua em que trafega, ocorrendo em menor frequência casos em que a câmera está desalinhada. Embora pudesse-se implementar a correção automatica do alinhamento da câmera, que implicaria primeiramente em se detectar quando ela não está alinhada corretamente e estimar o angulo da câmera com relação ao eixo principal da rua para essa correção, isso perderia o sentido se o sistema tivesse acesso às fotos capturadas originalmente pelos carros do GSV (por causa da forma como as câmeras estão estruturadas, veja Fig. 5.4).

6.2.3 Heterogeneidade temporal

As imagens obtidas pelo GSV são as mais recentes disponibilizadas, no entanto algumas regiões são atualizadas mais frequentemente do que outras. Isso resulta por vezes em mudanças drásticas na cena durante a transição de um nó, cuja imagem mais recente, data de uma época diferente da imagem mais recente do próximo nó. Foi observado que em alguns casos a diferenças entre as imagens mais recentes de dois nós relativamente próximos, com apenas alguns metros de distância entre eles, era de alguns anos. Em algumas situações observando-se o mapa pode-se ver a localização de alguns objetos fixos como pontos de ônibus, farmácias, escolas entre outros, porém quando as imagens são analisadas não estes mesmos objetos não estão presentes. Este tipo de erro pode acontecer em situações em que há divergências entre as datas das anotações no mapa e a tomada de fotos. As figuras 6.7 e 6.8 ilustram fotos da mesma região tomadas a partir de posições relativamente próximas (cerca de 17 metros). A figura 6.7 data de Julho de 2013 enquanto que a figura 6.8 é de Maio de 2011. Ambas as imagens são as mais recentes disponibilizadas pelo GSV e mesmo assim existe uma diferença de 2 anos o que implica numa mudança drástica entre as cenas (ex. o surgimento do posto de gasolina).

Figura 6.7: *Imagem de Julho de 2013 extraída do GSV [Incc]. Nesta imagem é possível ver o posto de gasolina que ainda não estava presente na Fig. 6.8 que mostra o mesmo lugar em Maio de 2011.*

Figura 6.8: *Imagem de Maio de 2011 extraída do GSV [Incc]. Esta imagem exibe o mesmo local da Fig. 6.7, porém 2 anos antes. Aqui por exemplo não existe a presença do posto de gasolina visível naquela figura.*

6.3 Comparação

A implementação do método proposto por Li et al. (2015) [LZL+15], como citado no capítulo 2 de revisão bibliográfica, consiste de uma modificação no "Índice de Visualização Verde"proposto por Yang (2009) et al. [YZMG09]. Recordando, essa modificação consiste numa subtração convenientemente arranjada dos canais de cor de uma imagem definida no espaço de cor RGB seguida por uma limiarização com base num valor de limiar que não é explicitamente definido no artigo de Li et al (2015) [LZL+15]. Para definir qual o melhor valor para limiarização daquele método o mesmo procedimento usado para otimizar os parâmetros do filtro de árvores, proposto neste trabalho, foi usado.

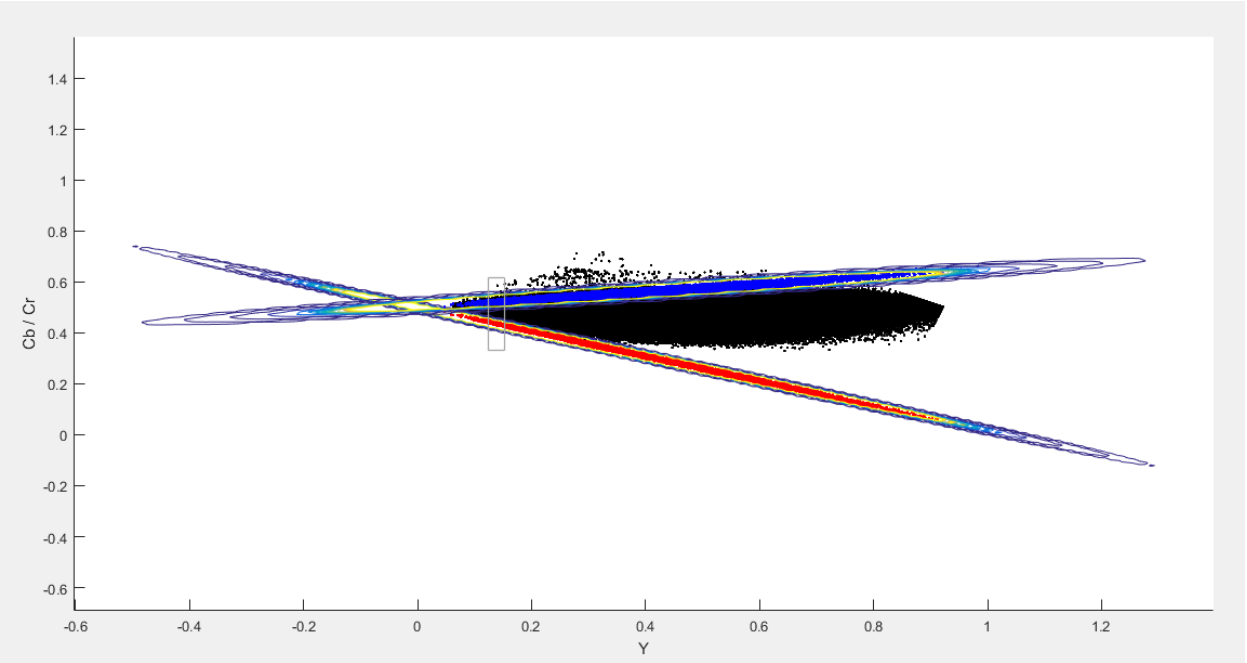
O filtro para detecção de pele humana proposto por Brancati et al. (2017) [BDPFG17] também foi considerado. Recordando, lá é proposta uma segmentação baseada no espaço de cor "YCbCr"através de métodos estatísticos e propriedade geométricas observadas em pontos da imagem que pertencem à pele humana. O método "mt-br"é uma modificação, do detector de pele humana, feita para detectar a vegetação e foi baseada na observação de uma outra configuração geométrica estabelecida entre os pontos da imagem pertencentes à vegetação no espaço de cor "YCbCr". Na figura 6.9 é ilustrada a projeção dos pontos de todas as imagens do "groundtruth" nos espaços YCr e YCb, os pontos em azul e vermelho são pontos dos espaços YCb e YCr respectivamente marcados como árvores nas imagens do "groundtruth" os demais pontos em preto são pontos classificados como não sendo árvores nas imagens do "groundtruth". Cada superfície gaussiana foi estimada de maneira a representar a densidade da função de probabilidade bidimensional da distribuição dos pontos correspondentes às árvores nos espaços YCb e YCr.

Também foi explorada uma classe de filtros de vegetação baseada na ideia de que a vegetação e a folhagem das árvores é esverdeada, assim como explorado por Li et al (2015) [LZL+15].

6.3.1 Medidas usadas

Baseando-se na competição de Visão Computacional conhecida como "THE PASCAL Visual Object Classes Challenge" [EEVG+15] foram usadas as medidas de precisão, redes-

Figura 6.9: *Pontos do "groundtruth" no espaços YCr e YCb sobrepostos. Os pontos em azul e vermelho são pontos pertencentes às árvores nos espaços YCb e YCr respectivamente. Os demais pontos são pontos que não estão marcados como árvores no "groundtruth". As elipses demarcam curvas de nível de duas superfícies gaussianas com parâmetros estimados com base nas médias e desvios padrões das posições dos pontos das árvores nos espaços YCb e YCr.*



coberta e "F1" para se comparar as 5 heurísticas descritas no capítulo 5.

Todas as heurísticas para segmentação de imagens, estudadas neste trabalho, são baseadas na classificação de pixels. O "groundtruth"conta com aproximadamente 3.8 milhões (3831147) de pixels marcados como sendo vegetação e com aproximadamente 25.4 milhões (25401768) de pixels marcados como não sendo vegetação. A quantidade de pixels positivos (pixels marcadas como vegetação) é aproximadamente 6.68 vezes menor que a de casos negativos (pixels marcados como não sendo de vegetação). Esta diferença entre a quantidade de amostras de cada classe implica, de acordo com D. M. Powers (2011) [Pow11], , que erros de classificação estão mais propensos a ocorrer em função de uma baixa taxa de especificidade do que de uma baixa taxa de redescoberta, para uma dada heurística. Assim, outros estudos são necessários a fi m de se estimar o desempenho dos métodos em imagens arbitrárias, onde há variações entre as estações do ano, clima da região, horário em que as imagens são tomadas, entre outros, que afetam a coloração das árvores e que podem limitar o uso do dataset utilizado neste trabalho.

A taxa de precisão de um algoritmo é avaliada com base na quantidade de pontos marca-

dos pelo algoritmo corretamente como vegetação de acordo com o "groundtruth". A equação 6.1 descreve o cálculo de precisão:

$$\text{precisão} = \begin{cases} 0, & \text{se } TP + FP = 0 \\ \frac{TP}{(TP+FP)}, & \text{em caso contrário} \end{cases} \tag{6.1}$$

Onde TP e FP correspondem ao número de pontos da imagem classificados corretamente e erroneamente como vegetação, respectivamente.

A taxa de redescoberta relaciona a quantidade de pontos de uma imagem que pertencem à vegetação com o número de pontos de vegetação marcados como tal por um dado algoritmo seguindo a definição da equação 6.2:

$$redescoberta = \frac{TP}{GTP} \tag{6.2}$$

Onde GTP corresponde ao número total de pontos classificados manualmente no "groundtruth" como vegetação.

O critério usado para se decidir qual dos algoritmos testados possui o melhor desempenho foi a medida estatística conhecida como "F1 Score". De acordo com Manning et al. (2008) [MRS] esta medida pode ser considerada como uma média ponderada entre as taxas de precisão e de redescoberta e seu valor máximo é 1, no caso em que as taxas de precisão e redescoberta são ambas iguais à 1 (todos os pontos da vegetação presentes no "groundtruth" foram corretamente marcados) enquanto que o mínimo é 0, no caso em que uma ou ambas as taxas são iguais à 0. O caso genérico desta medida também é conhecido pelos nomes "F-score" ou "F-measure" e sua fórmula geral para algum número real positivo β é ilustrado na equação 6.3:

$$f_\beta = (1 + \beta^2) * \frac{P * R}{(\beta^2 * P) + R} \tag{6.3}$$

Onde P e R são as taxas de precisão e redescoberta, do resultado de um filtro quando comparado com o "groundtruth", respectivamente. Não havendo motivos para justificar um peso maior para a medida de precisão sobre a de redescoberta foi adotado o valor $\beta = 1$,

Tabela 6.2: *Média das medidas de redescoberta, precisão e "F1 Score" para os métodos aplicados sobre o "groundtruth"*

Método	Redescoberta	Precisão	"F1 Score"
mt-li	0.5462	0.8706	0.6083
mt-br	0.7016	0.3908	0.4576
mt-li1	0.8675	0.6012	0.6300
mt-li2	**0.8814**	0.8196	0.8133
mt-li-espectral	0.7861	**0.9019**	**0.8163**

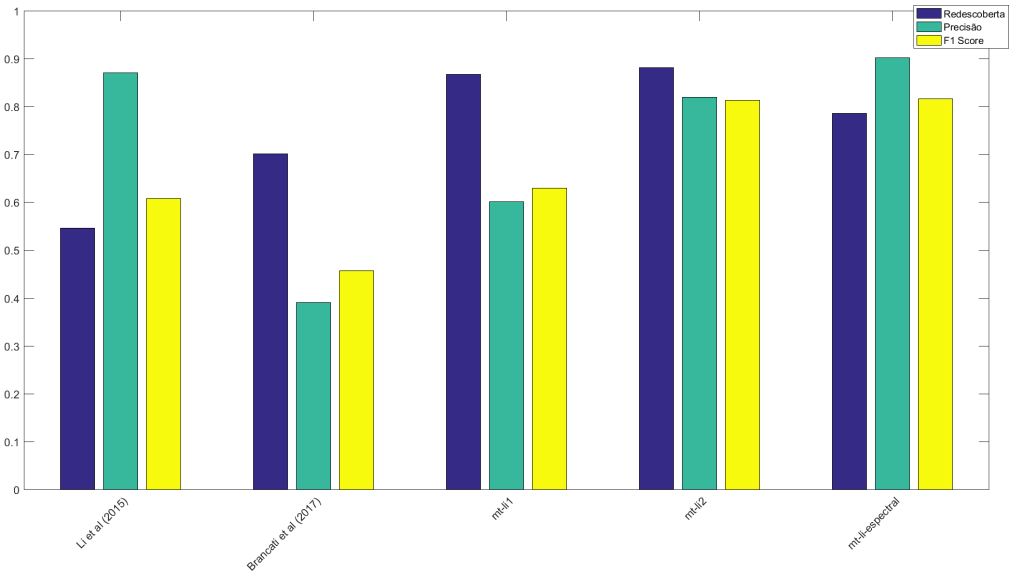
levando à equação 6.4 da medida "F1 Score":

$$F_1Score = 2 * \frac{P * R}{P + R} \tag{6.4}$$

6.3.2 Resultados

Na figura 6.10 é exibido um gráfico de barras contendo a comparação entre a precisão, taxa de redescoberta e a combinação das duas na medida estatística "F1 Score" dos cinco métodos testados para filtragem e detecção de árvores. A medida "F1 Score" é definida como na equação 6.4. A tabela 6.2 contém as mesmas métricas exibidas no gráfico de barras da figura 6.10.

Figura 6.10: *Comparação entre os 5 métodos para detecção de vegetação. Cada barra toma a média da métrica do método testado sobre todas as imagens do "groundtruth".*



Na figura 6.11 são exibidos exemplos da segmentação de algumas imagens urbanas por

Tabela 6.3: *Média da medida "F1 Score" e o Erro Padrão da Média da medida "F1 Score" para os métodos aplicados sobre o "groundtruth"*

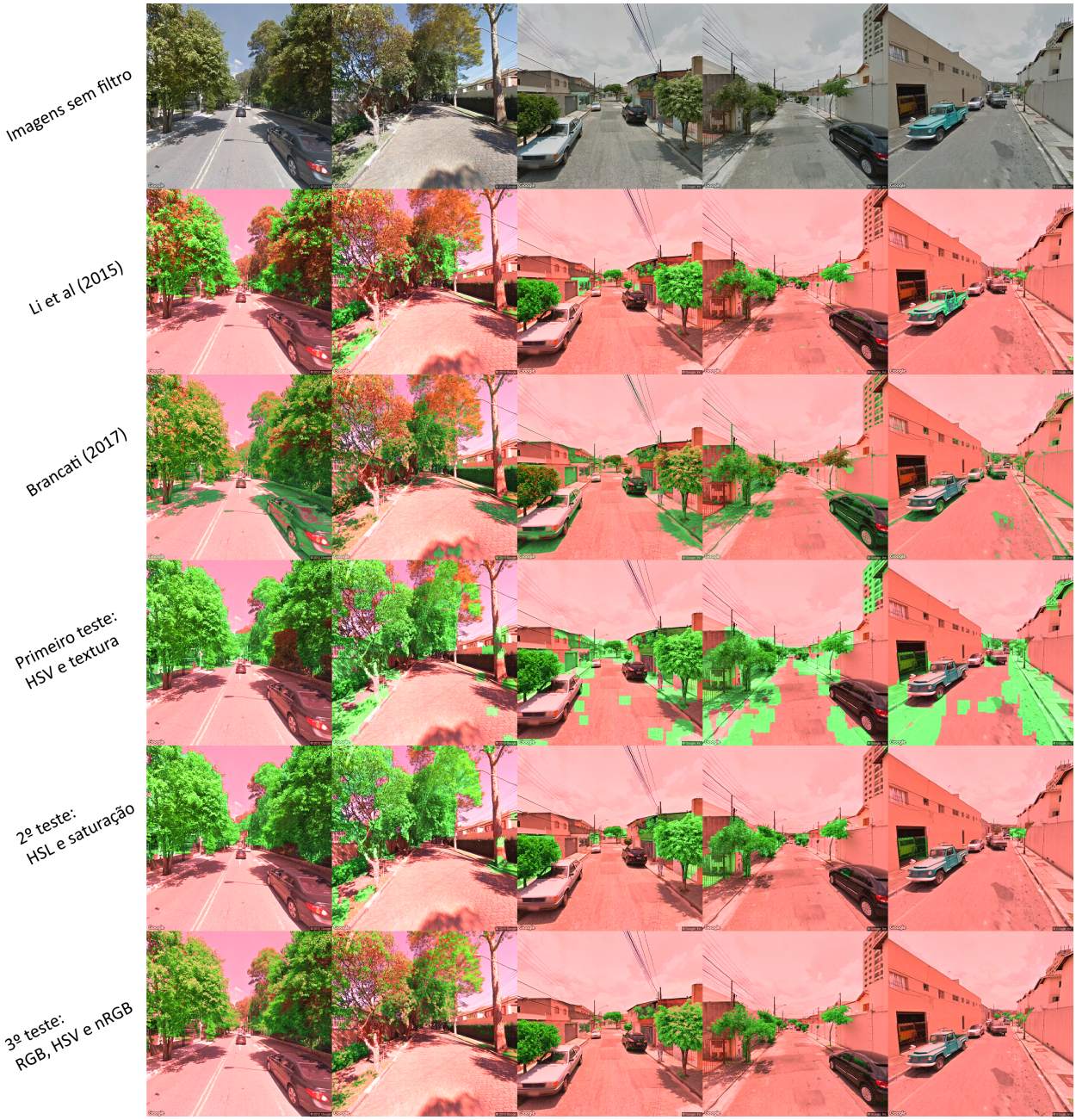
Método	"F1 Score"	Erro Padrão da Média
mt-li	0.6083	0.0237
mt-br	0.4576	0.0247
mt-li1	0.6300	0.0288
mt-li2	0.8133	0.0217
mt-li-espectral	**0.8163**	**0.0147**

cada um dos filtros experimentados, na primeira linha são exibidos os resultados da filtragem pelo filtro proposto por Li et al (2015) [LZL+15], na segunda a adaptação do método proposto por Brancati (2017) [BDPFG17] para detecção de pele, na terceira o filtro baseado na segmentação através de um intervalo no espaço HSV e textura, na quarta sua modificação levando em conta o aumento de saturação no espaço de cor HSL e por fim na quinta a segmentação considerando somente diferentes intervalos em diferentes espaços de cor. Os três últimos filtros são descritos no capítulo 5 de heurísticas de segmentação de imagens nas subseções 5.1.1, 5.1.2 e 5.1.3 respectivamente.

Nas figuras 6.12, 6.13 e 6.14 são exibidos diagramas de caixas ("boxplots") representando as taxas de precisão, redescoberta e da medida estatística "F1 Score" de cada método, aplicado às imagens do "groudtruth", respectivamente. Os círculos em azul indicam discrepâncias ("outliers"). As medidas das taxas de precisão e de redescoberta, foram tomadas com base nas equações 6.1 e 6.2 respectivamente. Observamos nestes diagramas que há vários "outliers", principalmente abaixo da mediana de cada caixa. Isto pode indicar que o método, ou os parâmetros usados, ou ambos, não é estável. O estudo da robustez está fora do escopo do trabalho, mas poderia ser feito através de uma busca em grade ("grid search") da melhor parametrização para se segmentar a imagem de forma estável.

A última análise gráfica exibida na figura 6.15 demonstra a média da medida "F1 Score" de cada método sobre todas as imagens. As barras pretas indicam a soma e subtração da média da medida "F1 Score" ao erro padrão da média estimada de cada método aplicado a cada uma das imagens do "groundtruth". A equação 6.5 exibe o cálculo da estimativa do erro padrão da média (de acordo com Ahn e Fessler (2003) [AF03]).

Figura 6.11: *Imagens urbanas e filtragens de árvores*



$$\sigma_{\bar{x}} = \frac{\sigma_x}{\sqrt{n}} \tag{6.5}$$

Onde:

- σ_x é o desvio padrão da média da medida "F1 Score" de cada método sobre cada imagem do "groundtruth";
- n é o número de imagens no "groundtruth" que neste caso é 100.

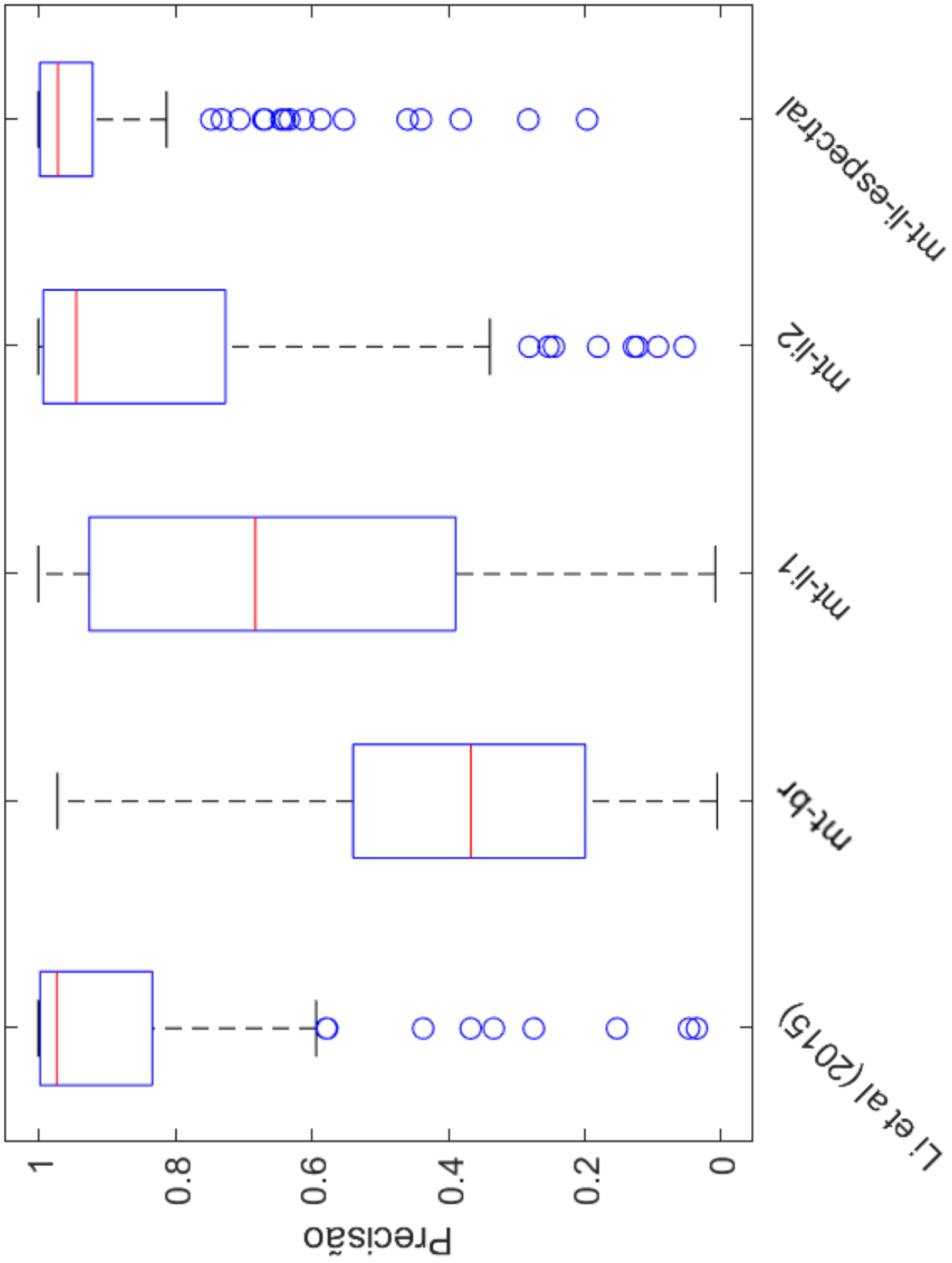


Figura 6.12: *Diagrama de caixa da taxa de precisão de cada método.*

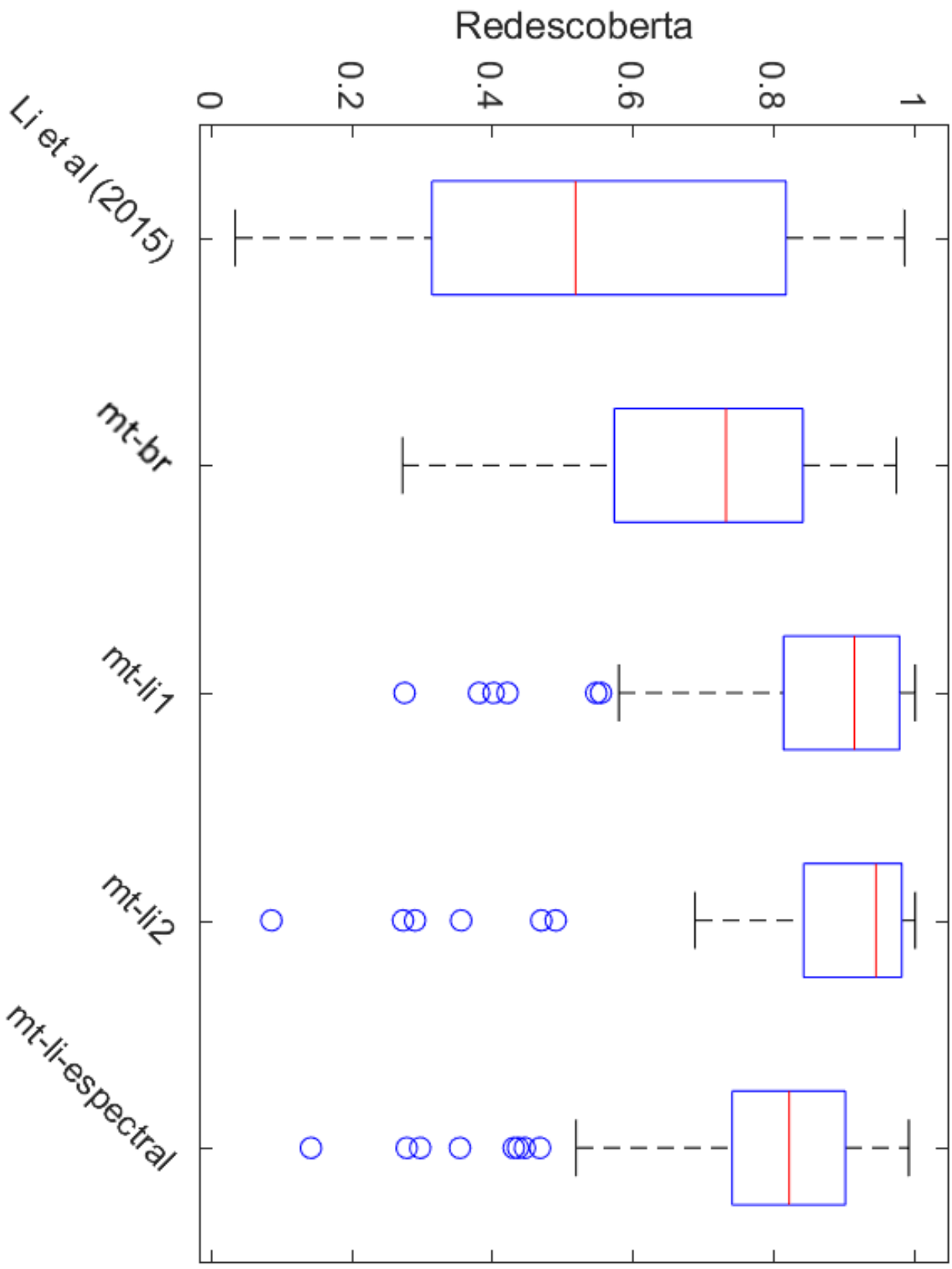


Figura 6.13: *Diagrama de caixa da taxa de redescoberta de cada método sobre cada imagem.*

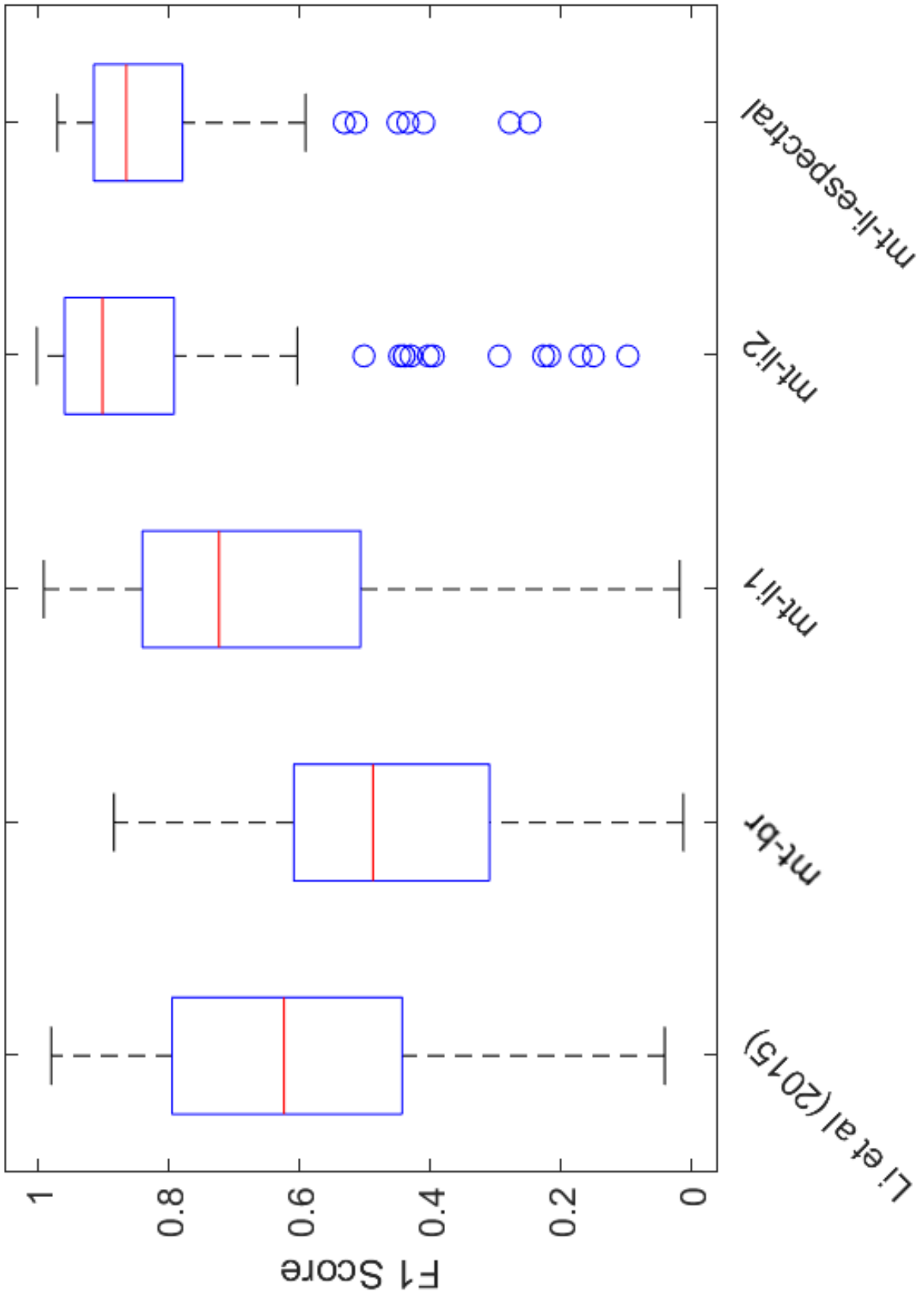
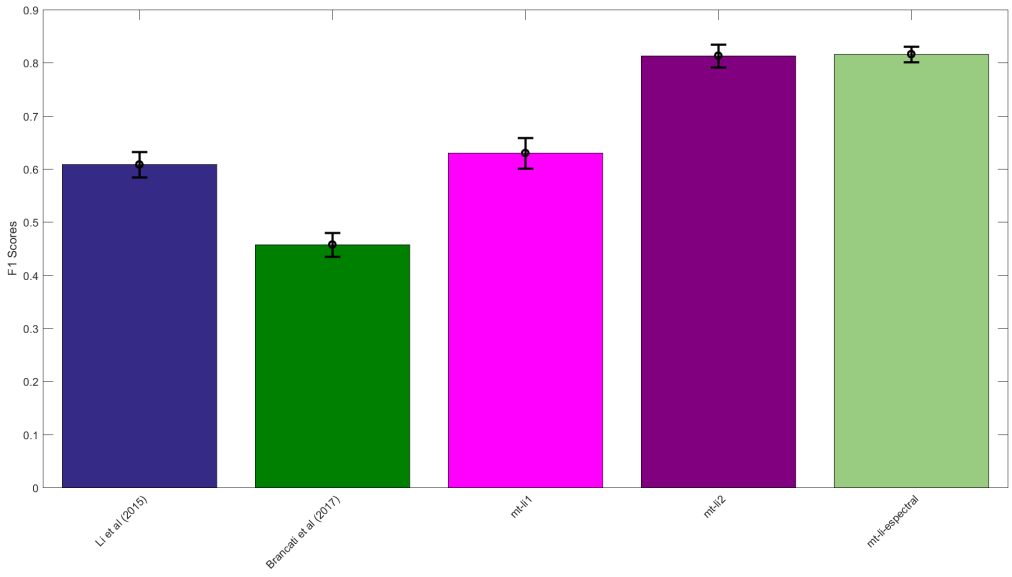


Figura 6.14: *Diagrama de caixa da medida "F1 Score" de cada método sobre cada imagem.*

Figura 6.15: *Gráfico de médias da medida "F1 Score". As linhas pretas sobre cada barra indicam o erro padrão médio (desvio padrão da média) de cada método.*



Capítulo 7

Conclusões

Neste trabalho foi apresentada a arquitetura do projeto INACITY, uma plataforma voltada para a coleta e processamento de imagens que permite a visualização das imagens urbanas de uma região, das imagens processadas e um visualizador para SIGs, ou seja, de pontos geográficos que correspondem à características urbanas como árvores, pontos de ônibus, escolas, etc. Esta plataforma foi projetada para ser escalável para a integração de outras plataformas de imageamento, SIGs e para a implementação de outros filtros de imagens. Ela oferece um filtro para detecção de árvores e vegetação que pode ser integrado à outras plataformas através de chamadas REST, isso é, outras plataformas podem fazer uso do filtro implementado no INACITY. Além disso a plataforma pode ser integrada com filtros de imagens externos, disponibilizados através de serviços via chamadas REST (vide o capítulo 8 de apêndices para ver detalhes de como integrar a plataforma com filtros externos).

7.1 Contribuições do trabalho

A plataforma foi pensada de maneira a permitir que cidadãos possam usufruir dos serviços de coleta e processamento de imagens e também da visualização cartográfica de elementos extraídos de diferentes sistemas de informações geográficas. Além de permitir o uso pelo usuário final a plataforma também serve como base para estudantes e pesquisadores, principalmente de áreas de ciência da computação, arquitetura, urbanismo e áreas correlatas, trazendo a estes uma ferramenta para formação de conjuntos de dados. Uma outra contribui-

ção derivada deste projeto é o conjunto de imagens anotadas manualmente que é chamado ao longo do texto de "groundtruth" e serve como base para comparação de algoritmos que fazem a detecção de árvores em imagens. Por fim, a última contribuição deste trabalho é a descrição de alto nível da arquitetura do projeto, esta pode servir como base para futuras plataformas de apoio à cidades inteligentes que também são baseadas no conceito de coleta e processamento de imagens.

7.2 Sugestões para Pesquisas Futuras

Atualmente existem recursos como a visualização em $3D$ do "Google Maps" [Incc] como na figura 7.1. No início da elaboração deste projeto não havia nenhum recurso similar a este, isto é, uma modelagem tridimensional da vista de satélite da cidade. Este tipo de recurso pode enriquecer a análise dos dados como é feito no artigo de Wegner et al. (2016) [WBH+16] em que são combinadas informações de imagens de satélite e no nível da rua para se inferir a localização geográfica de objetos na cidade. Contudo, observa-se que mesmo com novas plataformas e tecnologias como a do "Google Maps" com modelos tridimensionais da cidade, contudo as limitações do "Google Street View" descritas no capítulo 6 ainda persistem. Isso indica que ainda será necessário algum tempo até que todas estas soluções sejam integradas.

As áreas de visão computacional e a integração de diferentes sistemas de informação geográfica (SIG) trazem grandes desafios interessantes que no contexto deste projeto explora-se somente uma pequena parte. O maior desafio explorado na implementação deste projeto foi a integração daqueles elementos, porém ainda sim há espaço para melhorias em cada um dos tópicos individualmente.

7.2.1 Visão computacional

Com relação à área de Visão Computacional pode-se propor a implementação de filtros capazes de detectar outras características urbanas. Porém, mais do que detectar características em imagens, uma preocupação que deve ser levada em conta é o desempenho computacional deste filtro, isso é, quanto tempo ele leva para processar cada imagem dado que uma mesma consulta pode conter em alguns casos mais de uma centena de imagens.

Figura 7.1: *Imagem de exemplo da visualização em 3D do "Google Maps"*

Além disso a fusão de imagens oriundas de diferentes sensores como no caso do artigo de Wegner et al. (2016) [WBH+16] também se classifica como um desafio cuja solução traria resultados extremamente proveitosos, isso é, o uso de imagens aéreas e de satélite em conjunto com as já obtidas no nível do solo enriquece a precisão com que os objetos são detectados além de permitir um detalhamento ainda melhor em sua observação.

7.2.2 Integração de serviços

Atualmente o projeto conta com dados geográficos de três SIGs diferentes, estes são o "Google Maps", "OpenStreetMap" e o "GeoSampa". Em função de erros de precisão de GPS, em alguns casos, o mesmo elemento (como uma rua) é posicionada em locais diferentes em cada SIG, isso leva a problemas na integração de diferentes SIGs. Utilizar diferentes SIGs é interessante porque cada um deles pode conter informações não presentes nos demais, como é o caso da localização de pontos de ônibus na cidade de São Paulo descrita melhor pelo "GeoSampa" do que pelo "Google Maps" ou pelo "OpenStreetMap". Uma sugestão de pesquisa com relação aos SIGs é a elaboração de algoritmos focados na resolução dos problemas de alinhamento entre eles.

Por fim, as limitações descritas no capítulo 6 de resultados experimentais (por exemplo

a direção da câmera) são tópicos de pesquisa interessantes para a melhoria da plataforma principalmente no que se refere à integração de diferentes fontes de dados.

Capítulo 8

Apêndices

Neste capítulo serão apresentados aspectos da implementação da plataforma, assim como alguns exemplos de limitações, casos de uso, trechos de código e como integrar outras APIs com o INACITY.

8.1 Manual de uso

A principal uso da plataforma é permitir aos usuários encontrar características urbanas, mas também existem as possibilidades de uso voltados à pesquisa científica. As principais etapas para o uso da plataforma como uma ferramenta de coleta e processamento de imagens são:

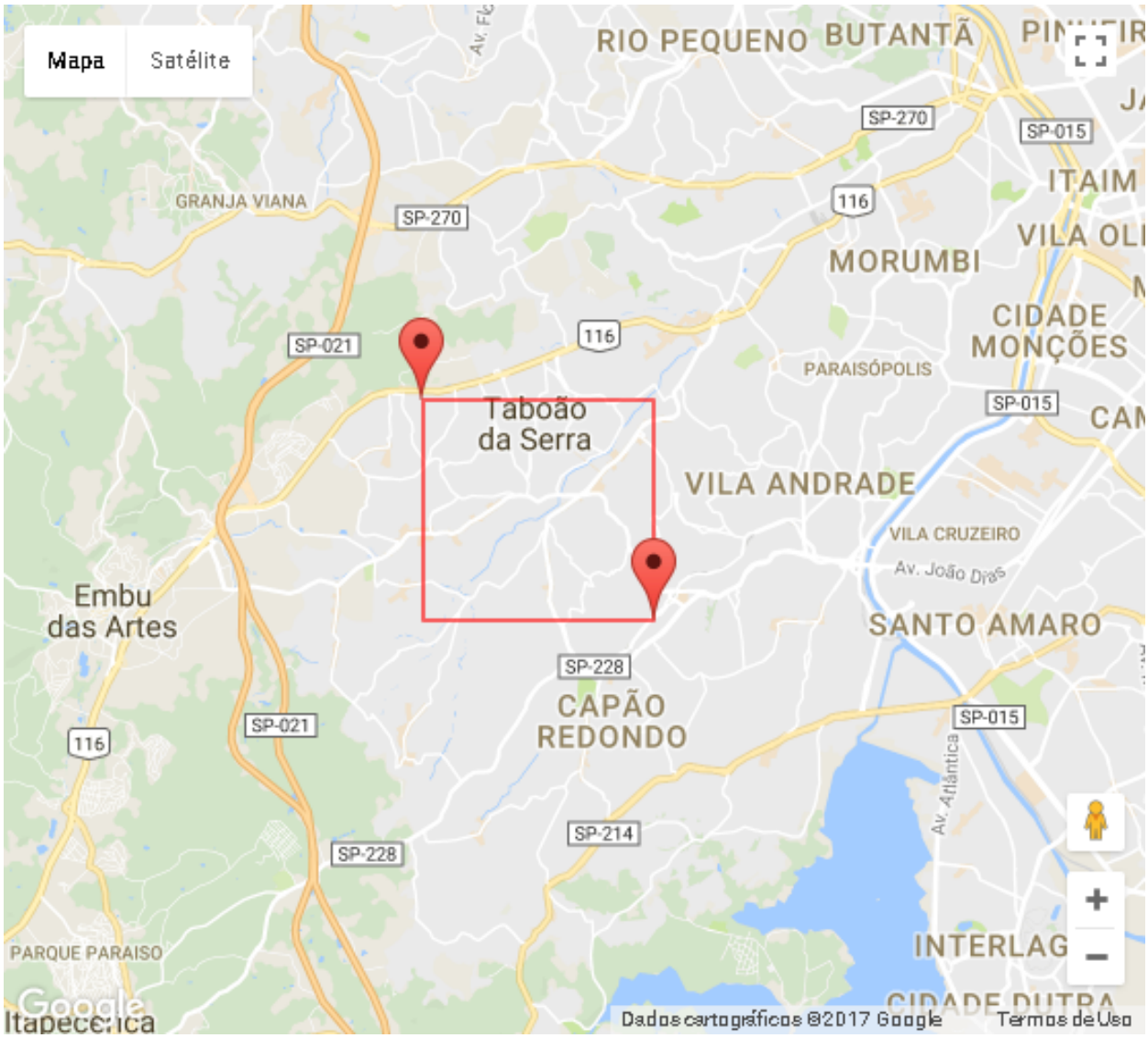
1. Selecionar uma região
2. Escolher uma característica pontual
3. Selecionar ruas
4. Visualizar imagens das ruas e filtradas
5. Usar outros filtros

8.1.1 Selecionar uma região

Selecionar uma região sobre o mapa é a primeiro passo para qualquer uma das outras etapas é a seleção da região de interesse. Para selecionar uma região basta clicar uma vez para

definir o lado superior esquerdo ou direito e outra para o lado inferior direito ou esquerdo respectivamente da região de interesse. Essa seleção será demarcada por linhas vermelhas como na figura 8.1.

Figura 8.1: *Seleção de uma região de interesse no mapa do "Google Maps". Os pinos vermelhos indicam os cantos superior esquerdo e inferior esquerdo do retângulo que delimita a região de interesse.*



8.1.2 Escolher uma característica pontual

Uma característica pontual é qualquer característica que possui coordenadas geográficas (ex. latitude e longitude) como escolas, farmácias ou até mesmo pontos de ônibus. Neste projeto estas características são representadas por nós obtidos através do "OpenStreetMaps".

A plataforma INACITY permite a busca de escolas, farmácias, pontos de ônibus e hospitais registrados no "OpenStreetMaps". Para encontrar tais características primeiro selecione uma região sobre o mapa (vide subseção 8.1.1) e então "clique" sobre o botão que representa a característica desejada como na figura 8.2.

Figura 8.2: *Pontos de ônibus extraídos do "OpenStreetMap" e exibidos no mapa do "Google Maps". Cada pino vermelho indica a presença de um ponto de ônibus. Pares de pinos vermelhos muito próximos normalmente indicam pontos de ônibus em cada sentido de uma rua; aproximadamente um de frente ao outro.*

8.1.3 Selecionar e inspecionar ruas e regiões

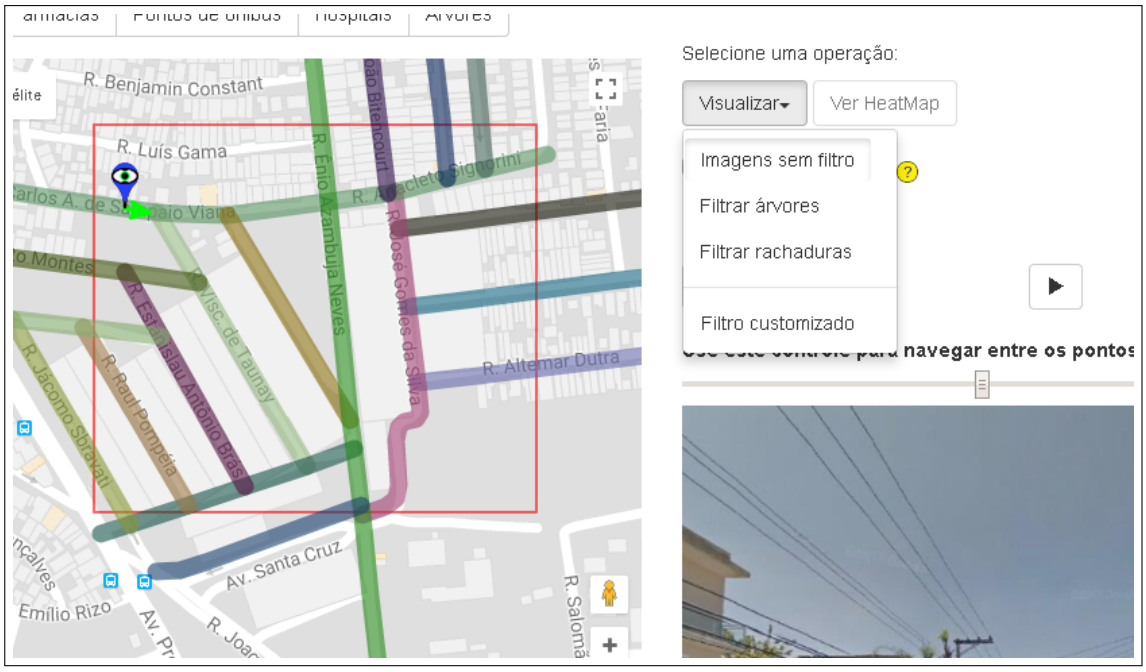
Uma vez selecionada a região é possível se escolher uma rua para ser analisada. Neste caso basta pressionar o botão "Coletar ruas" e em seguida clicar sobre uma das ruas que agora são marcadas por linhas azuis como na figura 8.3. Também é possível a inspeção de uma região inteira, neste caso basta selecionar a região, não selecionar nenhuma rua e pressionar o botão de "Visualizar imagens" como na figura 8.4. Se uma rua for selecionada é possível realizar a inspeção visual desta através do botão de "Visualizar imagens". Desta vez as imagens serão somente da rua selecionada ao invés da região inteira selecionada.

Figura 8.3: *Selecionando uma rua para inspeção. A linha azul indica uma rua selecionada para inspeção. As linhas em vermelho são as demais ruas presentes na região de interesse não selecionadas.*

Uma vez que a inspeção visual foi iniciada é possível usar os filtros de imagem disponíveis na plataforma para detectar características urbanas nas imagens. Para isto basta clicar sobre algum dos botões de filtro abaixo do botão de "Imagens sem filtro". Isso resultará na filtragem e exibição da sequência de imagens coletadas como na figura 8.5

Ao mesmo tempo é exibido no mapa cartográfico, através do "Google Maps"), em forma de um mapa de calor, a densidade das características buscadas em cada coordenada em que uma imagem foi obtida. A figura 8.6 mostra um exemplo de mapa de calor para árvores

Figura 8.4: *Selecionando uma região para inspeção. Cada linha colorida indica uma rua que pode ser selecionada para inspeção. O retângulo vermelho com linhas mais finas indica a região de interesse selecionada.*



buscadas em uma região.

Figura 8.5: *Filtragem de árvores em uma sequência de imagens. Regiões da imagem esverdeadas indicam pontos marcados pelo filtro de vegetação como sendo parte da vegetação, regiões avermelhadas foram classificadas pelo filtro como não sendo parte da vegetação.*

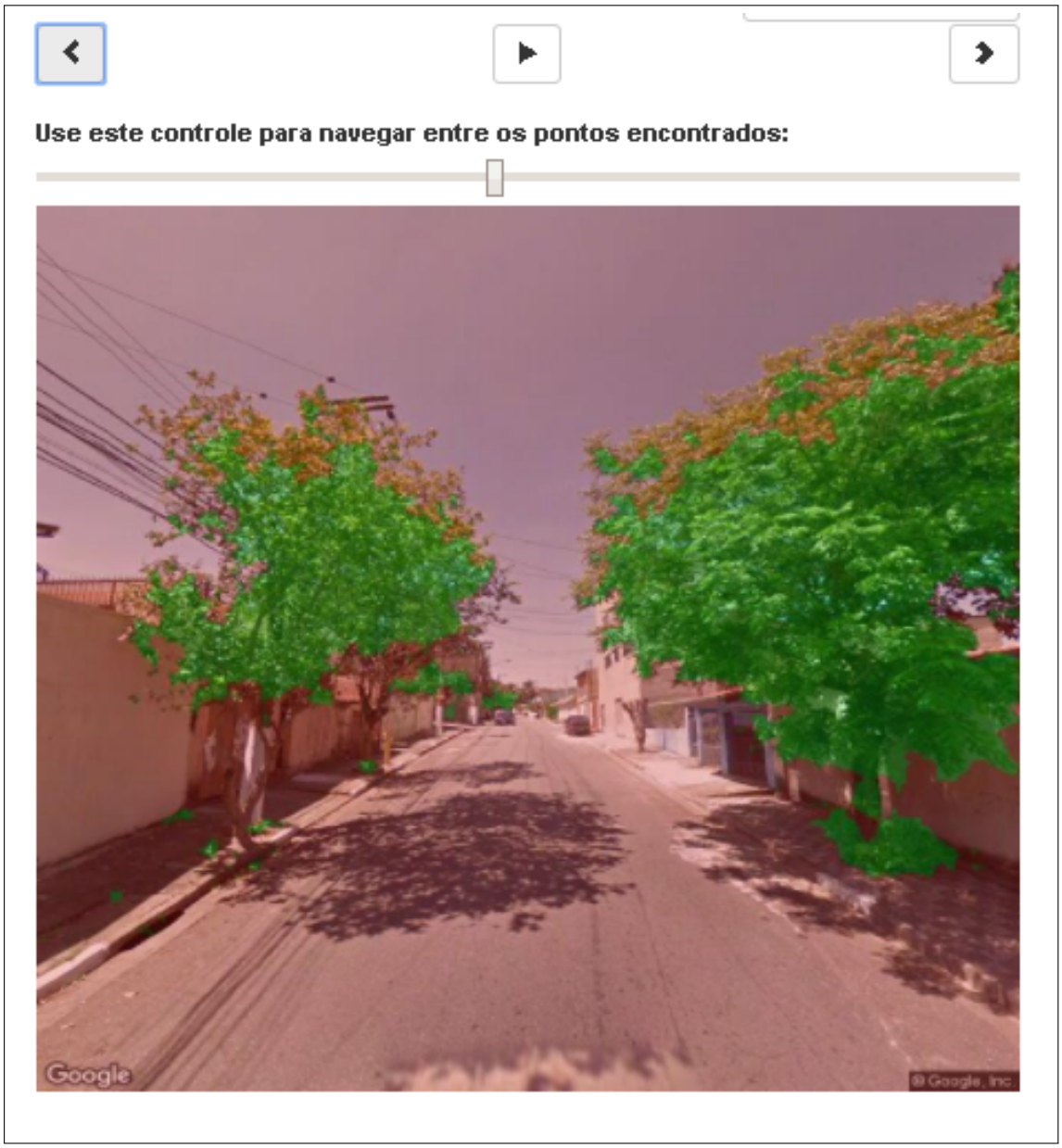
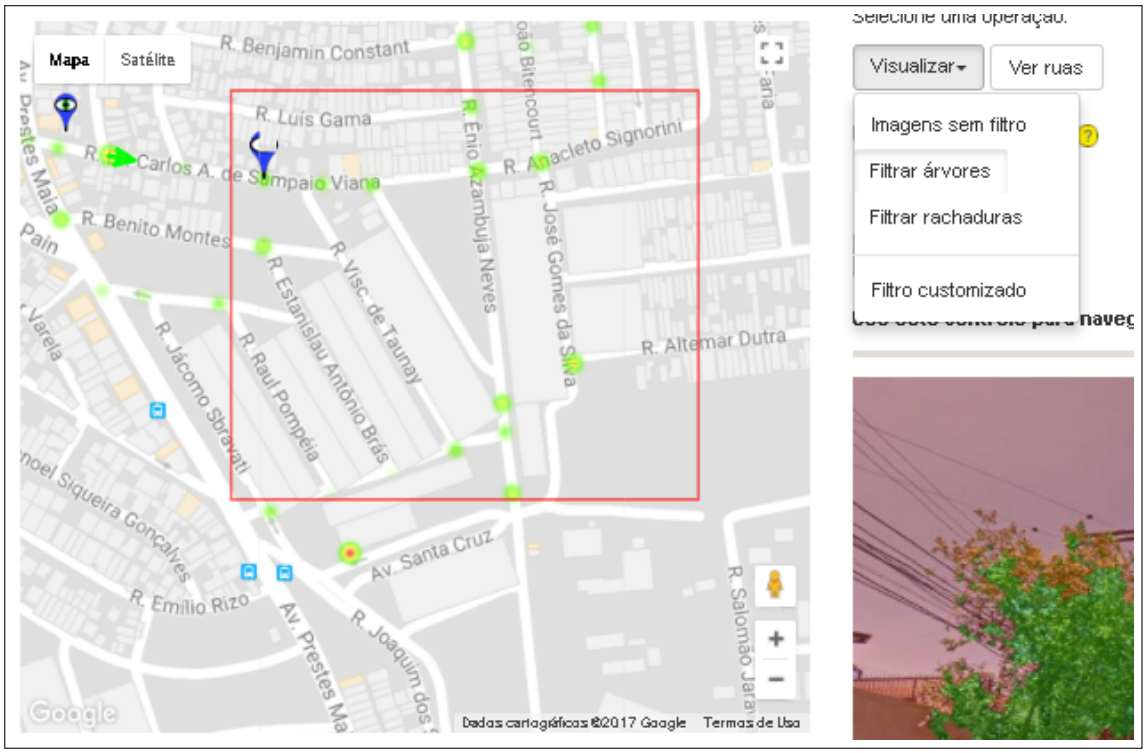


Figura 8.6: *Mapa de calor representando a densidade de árvores detectadas. Cada ponto esverdeado ou avermelhado indica um ponto onde foi tomada uma imagem. Pontos avermelhados indicam uma concentração maior de vegetação detectada, enquanto que os esverdeados indicam uma concentração menor.*



8.2 Integração com filtros externos

A integração da plataforma INACITY com outras plataformas é possível de duas formas, ou usando o "frontend" do INACITY para exibir os resultados do processamento das imagens obtidas pelo INACITY e processadas por um serviço externo ou usando-se o "backend" do INACITY para processar as imagens obtidas de outras formas. Para acoplar um serviço externo de processamento de imagens ao "frontend" basta escolher a opção de visualização "filtro customizado" após já ter obtido as imagens sem filtros. Ao escolher a opção de "filtro customizado" será necessário inserir o endereço do serviço de processamento de imagens, os detalhes da interface de programação esperada são descritos na subseção 8.2.1 de definições para trocas de dados.

Uma outra opção para trabalhar com as imagens coletadas, tanto as originais (sem nenhum processamento) quanto as filtradas, é através da obtenção das imagens por "download". Para descarregar localmente as imagens obtidas pelo INACITY basta pressionar o botão "Descarregar imagens" uma vez que as imagens já tenham sido coletadas e estejam sendo visualizadas na plataforma.

8.2.1 Definições para trocas de dados

Para permitir o uso do "frontend" e do "backend" por serviços externos foram desenvolvidos quatro objetos para transferência de dados (DTOs) chamados:

- "FilterResultDTO": Contém a versão da imagem original após o processamento e também informações para uso do mapa de calor e para ordenação das imagens no "frontend";
- "Panorama DTO": Um panorama é uma coleção de imagens, este objeto é usado para organizar imagens obtidas em uma mesma localização, porém com angulações diferentes;
- "Picture DTO": Este objeto compreende a imagem original e informações relacionadas à ela como localização, URL, versões derivadas de diferentes filtros, etc.;

- "Point DTO": Este objeto encapsula as coordenadas geográficas (latitude e longitude) e será usado como classe de conveniência pelos demais objetos.

A definição dos DTOs na linguagem "javascript" [Micb] é definida como:

```
/*
* frontAngle - Corresponde ao ângulo frontal do veículo do GSV;
* pitch - Corresponde à inclinação vertical do veículo do GSV;
* pano - Corresponde ao identificador do panorama;
* Pictures - Este vetor contém todas as imagens obtidas a partir do
panorama identificado pela propriedade "pano".
*/
function PanoramaDTO()
{

this.frontAngle = 0.0;
this.pitch = 0.0;
this.pano = null;

//Vetor de PictureDTO
this.Pictures = [];
}



/*
* imageID - Identificador usado para manter a ordenação das imagens
filtradas;
* base64image - Imagem codificada usando-se base64;
* location - Objeto do tipo PointDTO indicando a localização da
imagem filtrada;
* type - Indica o tipo de característica urbana filtrada
(ex. Árvores);
* isCaracteristicPresent - Valor booleano indicando se a
característica filtrada está presente na imagem;
```

```
* density - Valor entre 0 e 1 indicando a porcentagem da imagem
correspondente à característica filtrada;
* processedArea - Corresponde ao número total de pontos da imagem
classificados como parte da característica filtrada.
*/

function FilterResultDTO()
{
this.imageID = -1;
this.base64image = "";

//Instância de um objeto PointDTO
this.location = {};

this.type = null;
this.isCaracteristicPresent = null;
this.density = null;
this.processedArea = null;
}

/*
* imageID - Identificador da imagem usado para manter a ordenação
das imagens;
* panoID - Identificador do panorama ao qual esta imagem pertence;
* heading - Angulação, relativa à frente do veículo do GSV,
usada para tomar essa imagem;
* base64image - Imagem codificada em base64;
* imageURI - Endereço da imagem de acordo com o GSV;
* location - Instância de PontoDTO indicando a localização
geográfica de onde a imagem foi tomada;
```

```
* filterResults - Vetor que contém diferentes versões obtidas
por diferentes filtragens da imagem.
*/

function PictureDTO()
{
this.imageID = -1;
this.panoID = -1;
this.heading = -1;
this.base64image = "";
this.imageURI = "";

//Instância de um objeto PointDTO
this.location = {};

//Vetor de FilterResultDTO
this.filterResults = [];
}

/*
* ID - Identificador usado somente no banco de dados para
armazenar pontos de calor;
* lat - Latitude geográfica;
* lng - Longitude geográfica.
*/

function PointDTO()
{
this.ID = -1;
this.lat = -1;
```

```
this.lng = -1;
}
```

O endereço do serviço externo, informado pelo usuário, que irá processar a imagem (pressionando-se o botão "filtro customizado"), deve ser de um ponto de entrada ("end-point") HTTP que aceita o verbo "post". A plataforma INACITY irá enviar a este "end-point", através do corpo de uma chamada "post", um vetor (codificado usando-se JSON [Int13]) de objetos "PictureDTO". Note que não será enviada a imagem em si e sim uma referência a ela que pode ser obtida no lado do servidor através da propriedade "imageURI". Espera-se do servidor como retorno um vetor de objetos do tipo "FilterResultDTO".

8.3 Parâmetros e "groundtruth"

O conjunto de imagens de árvores marcadas ("groundtruth") foi criado para permitir a comparação entre os diferentes algoritmos usados para se detectar e marcar árvores em imagens. A partir deste conjunto de imagens foi possível não apenas gerar métricas de comparação como as taxas de precisão e de redescoberta de cada método, mas também estimar parâmetros ótimos para cada um dos métodos dado que todos os métodos testados são paramétricos.

8.3.1 Elaboração do "groundtruth"

O "groundtruth" foi criado usando-se imagens obtidas no "Google Street View" (GSV) e o processo de anotação foi feito através da edição manual de cada uma das imagens selecionadas para o "groundtruth". Essa edição manual consistiu da marcação de regiões que possuem árvores com a cor vermelha pura que na codificação de 8 bits no espaço de cor RGB corresponde ao vetor $255, 0, 0$. Para a marcação de regiões que não possuem árvores foi usada a cor rosa que no espaço de cor RGB, usando-se a codificação de 8 bits para cada canal, corresponde ao vetor $255, 255, 0$. Foi verificado à priori se estas duas cores não correspondiam a nenhuma cor presente em nenhuma das imagens a fim de se evitar falsos positivos ou negativos. A resolução das imagens não é alta o suficiente para que detalhes pequenos como as extremidades das folhas possam ser marcados de forma não ambígua,

sendo assim regiões que apresentaram ambiguidade com relação à se eram ou não partes da vegetação não foram marcadas, apenas as regiões claramente visíveis como sendo ou não parte da vegetação, ampliando-se a imagem nestas regiões, foram marcadas.

8.3.2 Otimização de parâmetros

Usando-se o software Matlab [MAT13], cada um dos algoritmos apresentados no capítulo 5 de heurísticas para segmentação de imagens foi implementado. Foi aplicada a função "fminsearch" sobre os parâmetros do tipo ponto flutuante de cada método, para os parâmetros do tipo inteiro foi usada a solução encontrada pelo "fminsearch" em conjunto com uma estrutura de laço para percorrer um subconjunto de valores viáveis para tais parâmetros. O algoritmo 8.1 ilustra o trecho de código do Matlab usado para otimizar os parâmetros do método "mt-li1". No algoritmo 8.1 as linhas 3 à 10 e 13 à 19 inicializam as variáveis dos métodos "mt-li1" e "mt-li2" respectivamente que serão otimizadas com o método "fminsearch". Os valores para inicialização foram tomados arbitrariamente e servem para agilizar o processo de convergência. As linhas 29 à 61 servem para se determinar qual o tamanho ótimo para os elementos estruturantes usados nas operações de abertura e fechamento morfológico dos métodos "mt-li1" e "mt-li2". As linhas 64 à 77 servem para determinar o tamanho ótimo do elemento estruturante retangular usado na operação cartola-dual do método "mt-li1" para se obter as bordas (gradiente morfológico) da imagem.

Algoritmo 8.1: *Otimização dos métodos "mt-li1" e "mt-li2"*

```
1
2           %Inicialização das varíaveis do método "mt-li1"
3           hsv_h_bottom_lim = 0.0001; %
4           hsv_h_upper_lim = 0.9374;
5           hsv_s_bottom_lim = 0.1936;
6           hsv_s_upper_lim = 0.8421;
7           hsv_str_elem_close = 6;
8           hsv_str_elem_open = 24;
9           hsv_str_elem_gradient = [15 9];
```

```
10          hsv_gradient_factor = 0.8554; %
11
12          %Inicialização das varíaveis do método "mt-li2"
13          hsl_h_bottom_lim = 0.0013; %
14          hsl_h_upper_lim = 0.5840;
15          hsl_s_bottom_lim = 0.2815;
16          hsl_s_upper_lim = 1;
17          hsl_hue_scale = 1.7133;
18          hsl_str_elem_close = 11;
19          hsl_str_elem_open = 20; %
20
21          %Os valores ótimos para as variáveis do método "mt-li1"\
             ↪  serão armazenadas na variável a1
22          a1 = fminsearch(@base_function_hsv,  [hsv_h_bottom_lim,
             ↪  hsv_h_upper_lim, hsv_s_bottom_lim, hsv_s_upper_lim,
             ↪  hsv_gradient_factor]); %
23
24          %Os valores ótimos para as variáveis do método "mt-li2"\
             ↪  serão armazenadas na variável a2
25          a2 = fminsearch(@base_function_hsl, [hsl_h_bottom_lim,
             ↪  hsl_h_upper_lim, hsl_s_bottom_lim, hsl_s_upper_lim,
             ↪  hsl_hue_scale]); %
26
27
28          %Tamanho do elemento estruturante quadrado usado para a
             ↪  operação de abertura morfológica do método "mt-li1".
29          min_i1 = 1; %
30          %min_j1 é usado para o fechamento morfológico do método
             ↪  "mt-li1".
31          min_j1 = 1;
```

```
32
33          %min_i2 é usado para a abertura morfológica do método
             ↪  "mt-li2".
34          min_i2 = 1;
35          %min_j2 é usado para o fechamento morfológico do método
             ↪  "mt-li2".
36          min_j2 = 1;
37
38          %O valor ótimo para as variáveis min_i e min_j são
             ↪  definidos com base nas métricas f1 score.
39          f1score1 = 0;
40          f1score2 = 0;
41
42          for i = 1:30
43          for j = 1:30
44          %val1 representa o valor médio da métrica f1 score obtida
             ↪  pelo método mt-li1, sobre o groundtruth, com os
             ↪  valores obtidos pela função "fminsearch"\ em conjunto
             ↪  com as operações morfológicas de abertura e
             ↪  fechamento com os elementos estruturantes quadrados
             ↪  de tamanhos i e j.
45          val1 =
             ↪  base_function_hsv([a1(1),a1(2),a1(3),a1(4),a1(5),i,
             ↪  j]);
46
47          if val1 < f1score1
48          f1score1 = val1
49          min_i1 = i
50          min_j1 = j
51          end
```

```
52
53          val2 =
            ↪  base_function_hsl([a2(1),a2(2),a2(3),a2(4),a2(5),i,
            ↪  j]);
54
55          if val2 < f1score2
56          f1score2 = val2
57          min_i2 = i
58          min_j2 = j
59          end
60          end
61          end %
62
63
64          min_i3 = 1; %
65          min_j3 = 1;
66          f1score3 = 0;
67          for i = 1:30
68          for j = 1:30
69          val3 =
            ↪  base_function_hsv([a1(1),a1(2),a1(3),a1(4),a1(5),min_i1,
            ↪  min_j1, i, j]);
70
71          if val3 < f1score3
72          f1score3 = val3
73          min_i3 = i
74          min_j3 = j
75          end
76          end
77          end %
```

78

O algoritmo 8.1 exprime as duas ideias centrais usadas para otimizar os parâmetros para todos os métodos de detecção de árvores testados e apresentados, isso é, otimizar variáveis do tipo ponto flutuante usando-se a função "fminsearch" e variáveis inteiras de uma maneira iterativa sobre um conjunto de valores definido empiricamente.

No artigo de Brancati et al. 2017 [BDPFG17] é exibido um método para detecção de pele humana em imagens a partir das relações geométricas dos pontos das imagens no espaço de cor YCbCr. Os autores usam heurísticas baseadas na formação de dois trapézios (um no espaço YCb e outro no YCr) e a partir destas heurísticas classificam cada ponto da imagem como sendo parte de pele humana ou não. Seguindo este raciocínio foi experimentado e implementado no algoritmo 8.2 a mesma ideia de se usar relações geométricas formadas pelos pontos da imagem convertidos para o espaço YCbCr e separados nos dois espaços YCb e YCr. Conforme observado no capítulo 6 de resultados experimentais, seção 6.3 (vide figuras 6.9 e **??**) ao invés de trapézios foram usadas curvas de nível de superfícies gaussianas para agrupar os pontos das árvores.

No algoritmo 8.2 as linhas 9 à 39 são responsáveis pela conversão cada ponto da imagem original no espaço RGB para o espaço YCbCr e também pela estruturação dos pontos no espaço YCbCr em vetores dos espaços YCb e YCr. As linhas 42 à 46 estimam as médias e matrizes de covariância para os conjuntos de vetores formados a partir dos pontos da imagem nos espaços YCb e YCr.

Algoritmo 8.2: *Estimação de parâmetros das gaussianas do método "mt-br"*

1

2
```
%Seja inputSet uma lista (objeto do tipo cell) com todas
↪ as imagens do "groundtruth"
```

3

4
```
%Seja gt_positive uma lista com as máscaras binárias
↪ correspondentes aos pontos que pertencem às árvores
```

5

```
6          %Seja gt_negative uma lista com as máscaras binárias
           ↪  correspondentes aos pontos que não pertencem às
           ↪  árvores
7
8
9          ycblist = cell(1, length(inputSet)); %
10         ycrlist = cell(1, length(inputSet));
11
12
13         for i = 1:length(inputSet)
14         bw = gt_positive{i};
15
16         %Aqui a imagem é convertida do espaço RGB para o espaço
           ↪  YUV (equivalente ao espaço YCbCr)
17         yuv = rgb2yuvsimetrico(inputSet{i});
18
19         %Aqui somente os pontos pertencentes às árvores são
           ↪  mantidos, os demais são eliminados
20         yuv = yuv.*repmat(bw,[1,1,3]);
21         y = yuv(:,:,1);
22         cb = yuv(:,:,2);
23         cr = yuv(:,:,3);
24
25         y = y(:);
26         cb = cb(:);
27         cr = cr(:);
28
29         idx = find(bw(:));
30
```

```
31          %Por convêniencia os índices (coordenadas na imagem) de
             ↪ cada ponto pertencente às árvores é armazenado na
             ↪ forma de um vetor cuja primeira coordenada
             ↪ corresponde à posição Y\ do ponto e a segunda à
             ↪ posição Cb\ ou Cr
32          ycblist{i} = [y(idx), cb(idx)];
33          ycrlist{i} = [y(idx), cr(idx)];
34          end
35
36          numPoints = 0;
37
38          ycb = vertcat(ycblist{:, :});
39          ycr = vertcat(ycrlist{:, :}); %
40
41          %A seguir são tomadas as médias e covariâncias das
             ↪ posições dos vetores nos espaços YCb\ e YCr
42          mycb = mean(ycb); %
43          cycb = cov(ycb);
44
45          mycr = mean(ycr);
46          cycr = cov(ycr); %
47
48
49
50
51          %A média entre as duas médias é então tomada de maneira a
             ↪ se estabelecer um ponto de referência para se
             ↪ realizar o processo de limiarização
52          grand_mean = (mycb + mycr)/2;
53
```

```
54          %Aqui são tomadas as distâncias de cada vetor nos espaços
             ↪  YCb e YCr com relação à média tomada na etapa
             ↪  anterior (grand_mean)
55          dmcb = bsxfun(@plus,ycb, -grand_mean);
56          dmcb = sqrt(sum(dmcb.^2, 2));
57          dmcr = bsxfun(@plus,ycr, -grand_mean);
58          dmcr = sqrt(sum(dmcr.^2, 2));
59
60          %Aqui é tomada a razão entre a distância dos vetores YCb
             ↪  e YCr de um ponto da imagem com relação à variável
             ↪  "grand_mean"
61          rmcbcr = dmcb./dmcr;
62
63          %As variáveis a seguir representam a razão mínima e
             ↪  máxima respectivamente dentre o conjunto de razões de
             ↪  distâncias dos vetores YCb e YCr com relação à
             ↪  variável "grand_mean"
64          minratio = min(rmcbcr);
65          maxratio = max(rmcbcr);
66
```

Algoritmo 8.3: *Função de limiarização ("mt-br") baseada no método de Brancati et al. 2017 [BDPFG17]*

```
1
2           function [ output_args ] = filter2Dgaussian( rgb,
3           mycb, cycb, mycr, cycr, minratio, maxration, k)
4
5           [rows, cols, ~] = size(rgb);
6
7           yuv = rgb2yuvsimetrico(rgb);
```

```
8           y = yuv(:,:,1);
9           cb = yuv(:,:,2);
10          cr = yuv(:,:,3);
11
12          ycb = [y(:), cb(:)];
13          ycr = [y(:), cr(:)];
14
15          dycb = dist2Dgaussian(ycb,mycb, cycb);
16          dycr = dist2Dgaussian(ycr,mycr, cycr);
17
18          grand_mean = (mycb(2) + mycr(2))/2;
19          dmcb = abs(ycb(:, 2) - grand_mean);
20          dmcr = abs(ycr(:, 2) - grand_mean);
21
22          rmcbcr = (dmcb./dmcr)";
23
24          %Aqui é gerada a máscara binária que representa pontos da
             ↪  imagem que pertencem ou não às árvores.
25          mask = (dycb < k) & (dycr < k) & rmcbcr >= minratio &
             ↪  rmcbcr <= maxratio;
26
27          mask = reshape(mask, rows, cols);
28
29          output_args = mask;
30
31          end
```

Algoritmo 8.4: *Conversão de uma imagem no espaço RGB para o espaço YUV (equivalente ao espaço YCbCr). Extraída de ITU-R Recommendations [REC95]*

```
1
```

```
2          function [ ycbcr ] = rgb2yuvsimetrico( rgb )
3          % Esta função converte uma imagem do espaço de cor RGB
   ↪  para um espaço de cor customizado baseado no YUV (equivalente
   ↪  ao YCbCr):
4
5          r = rgb(:,:,1);
6          g = rgb(:,:,2);
7          b = rgb(:,:,3);
8          y = 0.299*r + 0.587*g + 0.114*b;
9          cb = -0.169*r + 0.331*g +.5;
10         cr = -0.418*g - 0.0813*b +.5;
11
12         ycbcr = cat(3, y, cb, cr);
13
14         end
15
```

Referências Bibliográficas

[AA05] Max K Agoston e Max K Agoston. *Computer graphics and geometric modeling*, volume 1. Springer, 2005. 36

[AF03] Sangtae Ahn e Jeffrey A Fessler. Standard errors of mean, variance, and standard deviation estimators. *EECS Department, The University of Michigan*, páginas 1–2, 2003. 85

[AFC05] Brett G Amidan, Thomas A Ferryman e Scott K Cooley. Data outlier detection using the chebyshev theorem. Em *2005 IEEE Aerospace Conference*, páginas 3814–3819. IEEE, 2005. 69

[AIP14] Wolfgang Apolinarski, Umer Iqbal e Josiane Xavier Parreira. The gambas middleware and sdk for smart city applications. Em *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2014 IEEE International Conference on*, páginas 117–122. IEEE, 2014. 8, 9

[Ars] Ars Technica. Google's Street View cars are now giant, mobile 3D scanners | Ars Technica. https://arstechnica.com/gadgets/2017/09/googles-street-view-cars-are-now-giant-mobile-3d-scanners/. Último acesso em 12/01/2017. 58, 59

[AS96] Carol S Aneshensel e Clea A Sucoff. The neighborhood context of adolescent mental health. *Journal of health and social behavior*, páginas 293–310, 1996. 6

[ATBS11] Juan P Ardila, Valentyn A Tolpekin, Wietske Bijker e Alfred Stein. Markov-random-field-based super-resolution mapping for identification of urban trees in vhr images. *ISPRS journal of photogrammetry and remote sensing*, 66(6):762–775, 2011. 14, 55

[AvHWV+07] Charles Agyemang, Carolien van Hooijdonk, Wanda Wendel-Vos, Ellen Lindeman, Karien Stronks e Mariël Droomers. The association of neighbourhood psychosocial stressors and self-rated health in amsterdam, the netherlands. *Journal of Epidemiology & Community Health*, 61(12):1042–1049, 2007. 6

[Bar] Kai Uwe Barthel. 3d color inspector/color histogram. https://imagej.nih.gov/ij/plugins/color-inspector.html. Último acesso em 21/10/2017. 66

[BDPFG17] Nadia Brancati, Giuseppe De Pietro, Maria Frucci e Luigi Gallo. Human skin detection through correlation rules between the ycb and ycr subspaces based on dynamic color clustering. *Computer Vision and Image Understanding*, 155:33–42, 2017. 60, 68, 81, 85, 112, 115

[Ber96] Philip A Bernstein. Middleware: a model for distributed system services. *Communications of the ACM*, 39(2):86–98, 1996. 8

[BGH+16] Daniel Macêdo Batista, Alfredo Goldman, Roberto Hirata, Fabio Kon, Fabio M Costa e Markus Endler. Interscity: Addressing future internet research challenges for smart cities. Em *Network of the Future (NOF), 2016 7th International Conference on the*, páginas 1–6. IEEE, 2016. 8

[BHD+07] TK Boehmer, CM Hoehner, AD Deshpande, LK Brennan Ramirez e Ross C Brownson. Perceived and observed neighborhood indicators of obesity among urban adults. *International journal of obesity*, 31(6):968, 2007. 6

[BK02] Jennifer L Balfour e George A Kaplan. Neighborhood environment and loss of physical function in older adults: evidence from the alameda county study. *American Journal of Epidemiology*, 155(6):507–515, 2002. 6

[BLMPN17] André Leonardo Bortolotto Buck, Christel Lingnau, Álvaro Muriel Lima Machado e Sylvio Péllico Netto. Tree detection in point clouds derived from terrestrial laser scanning. *Boletim de Ciências Geodésicas*, 23(1):21–38, 2017. 13, 14, 16, 17, 55

[CB04] Kathleen A Cagney e Christopher R Browning. Exploring neighborhood-level variation in asthma and other respiratory diseases. *Journal of general internal medicine*, 19(3):229–236, 2004. 6

[CFM03] Deborah A Cohen, Thomas A Farley e Karen Mason. Why is poverty unhealthy? social and physical mediators. *Social science & medicine*, 57(9):1631–1641, 2003. 1, 6

[Che07] Xinjia Chen. A new generalization of chebyshev inequality for random vectors. *arXiv preprint arXiv:0707.0805*, 2007. 69, 70

[Coc14] Annalisa Cocchia. Smart and digital city: A systematic literature review. Em *Smart city*, páginas 13–43. Springer, 2014. 7, 8

[Cor] Microsoft Corporation. O que é o azure? https://azure.microsoft.com/pt-br/overview/what-is-azure/. Último acesso em 14/3/2018. 3

[CSE05] Andrea Cavallaro, Elena Salvador e Touradj Ebrahimi. Shadow-aware object-based video processing. *IEE Proceedings-Vision, Image and Signal Processing*, 152(4):398–406, 2005. 34

[CT04] L Chapman e JE Thornes. Real-time sky-view factor calculation and approximation. *Journal of Atmospheric and Oceanic Technology*, 21(5):730–741, 2004. 18

[dSP] Prefeitura de São Paulo. Mapa digital da cidade de são paulo. http://geosampa.prefeitura.sp.gov.br/PaginasPublicas/_SBC.aspx. Último acesso em 18/10/2017. 43

[EEVG+15] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn e Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, 111(1):98–136, 2015. 81

[EKCL17] Arthur M. Del Esposte, Fabio Kon, Fabio M. Costa e Nelson Lago. Interscity: A scalable microservice-based open source platform for smart cities. Em *Proceedings of the 6th International Conference on Smart Cities and Green ICT Systems - Volume 1: SMARTGREENS,*, páginas 35–46. INSTICC, SciTePress, 2017. 8, 9, 10, 11

[EMB05] Anne Ellaway, Sally Macintyre e Xavier Bonnefoy. Graffiti, greenery, and obesity in adults: secondary analysis of european cross sectional survey. *Bmj*, 331(7517):611–612, 2005. 6

[FT00] Roy T Fielding e Richard N Taylor. *Architectural styles and the design of network-based software architectures*. University of California, Irvine Doctoral dissertation, 2000. 29

[Gra08] Irina B Grafova. Overweight children: assessing the contribution of the built environment. *Preventive medicine*, 47(3):304–308, 2008. 6

[GRS06] Thomas A Glass, Meghan D Rasmussen e Brian S Schwartz. Neighborhoods and obesity in older adults: the baltimore memory study. *American journal of preventive medicine*, 31(6):455–463, 2006. 6

[HDO+98] Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt e Bernhard Scholkopf. Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28, 1998. 12

[HEH05] Derek Hoiem, Alexei A Efros e Martial Hebert. Geometric context from a single image. Em *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 1, páginas 654–661. IEEE, 2005. 20

[HGUM11] M Hammerle, Tamás Mátyás Gál, János Unger e A Matzarakis. Introducing a script for calculating the sky view factor used for urban climate investigations. *Acta Climatologica et Chorologica*, 44:83–92, 2011. 18, 20

[HLF13] Kotaro Hara, Vicki Le e Jon Froehlich. Combining crowdsourcing and google street view to identify street-level accessibility problems. Em *Proceedings of the SIGCHI conference on human factors in computing systems*, páginas 631–640. ACM, 2013. 11, 13

[HRE+05] Christine M Hoehner, Laura K Brennan Ramirez, Michael B Elliott, Susan L Handy e Ross C Brownson. Perceived and objective environmental measures and physical activity among urban adults. *American journal of preventive medicine*, 28(2):105–116, 2005. 6

[Inca] Google Inc. Google maps. https://www.google.com/maps. Último acesso em 05/11/2017. 1

[Incb] Google Inc. Google maps apis terms of service | google maps apis | google developers. https://developers.google.com/maps/terms#section_10. Último acesso em 11/02/2018. 2

[Incc] Google Inc. Google street view. https://www.google.com/maps/streetview/. Último acesso em 05/11/2017. 1, 80, 92

[Ind] Indian Space Research Organisation. A satellite photo provided by ISRO. https://www.indiatoday.in/india/chhattisgarh/story/isro-satellite-images-aid-chhattisgarh-security-forces-track-maoists-323000-2016-05-12. Último acesso em 12/01/2017. 55

[Int13] Ecma International. 404: The json data interchange format. http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf, 2013. Último acesso em 12/11/2017. 107

[IU] CENTRO DE COMPENTÊNCIA EM SOFTWARE LIVRE IME-USP. São paulo school of advanced science on smart cities. http://interscity.org/advanced-school/. Último acesso em 14/3/2018. 3

[JG78] George H Joblove e Donald Greenberg. Color spaces for computer graphics. Em *ACM siggraph computer graphics*, volume 12, páginas 20–25. ACM, 1978. 33

[JST11] Esakkirajan S Jayaraman S e Veerakumar T. *Digital Image Processing*. Tata McGraw Hill Education, 2011. 17

[JW84] Glenn T Johnson e Ian D Watson. The determination of view-factors in urban canyons. *Journal of Climate and Applied Meteorology*, 23(2):329–335, 1984. 19

[KGM+15] Omid Kardan, Peter Gozdyra, Bratislav Misic, Faisal Moola, Lyle J Palmer, Tomáš Paus e Marc G Berman. Neighborhood greenspace and health in a large urban center. *Scientific Reports*, 5:11610, 2015. 13

[KIB] Inc. & Department of Geography in the School of Liberal Arts at Indiana University-Purdue University Indianapolis (IUPUI) Keep Indianapolis Beautiful. ddressing social and environmental needs through community tree planting - canopy cover estimates. Último acesso em 28/01/2017. 57

[Kim08] Daniel Kim. Blues from the neighborhood? neighborhood characteristics and depression. *Epidemiologic reviews*, 30(1):101–117, 2008. 6

[KOA92] Takio Kurita, Nobuyuki Otsu e N Abdelmalek. Maximum likelihood thresholding based on population mixture models. *Pattern recognition*, 25(10):1231–1240, 1992. 18, 24

[KVW+13] Dimosthenis Kyriazis, Theodora Varvarigou, Daniel White, Andrea Rossi e Joshua Cooper. Sustainable smart city iot applications: Heat and electricity management & eco-conscious cruise control for public transportation. Em *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2013 IEEE 14th International Symposium and Workshops on a*, páginas 1–5. IEEE, 2013. 10, 11

[Lab17] MIT Senseable City Lab. Treepedia :: Mit senseable city lab. http://senseable.mit.edu/treepedia, 2017. Último acesso em 25/11/2017. 21, 25

[LC03] Carl A Latkin e Aaron D Curry. Stressful neighborhoods and depression: a prospective study of the impact of neighborhood disorder. *Journal of health and social behavior*, páginas 34–44, 2003. 6

[Lot99] Andrew Lothian. Landscape and the philosophy of aesthetics: is landscape quality inherent in the landscape or in the eye of the beholder? *Landscape and urban planning*, 44(4):177–198, 1999. 13

[LQN+08] Gina Schellenbaum Lovasi, James W Quinn, Kathryn M Neckerman, Matthew S Perzanowski e Andrew Rundle. Children living in areas with more street trees have lower prevalence of asthma. *Journal of Epidemiology & Community Health*, 62(7):647–649, 2008. 13

[LR01] Avraham Leff e James T Rayfield. Web-application development using the model/view/controller design pattern. Em *Enterprise Distributed Object Computing Conference, 2001. EDOC'01. Proceedings. Fifth IEEE International*, páginas 118–127. IEEE, 2001. 28

[LRS17] Xiaojiang Li, Carlo Ratti e Ian Seiferling. Mapping urban landscapes along streets using google street view. Em *International Cartographic Conference*, páginas 341–356. Springer, 2017. 17, 18, 20, 24, 55

[LZL+15] Xiaojiang Li, Chuanrong Zhang, Weidong Li, Robert Ricard, Qingyan Meng e Weixing Zhang. Assessing street-level urban greenery using google street view and a modified green view index. *Urban Forestry & Urban Greening*, 14(3):675–685, 2015. 16, 17, 19, 23, 55, 60, 62, 81, 85

[M.17] Oliveira A. A. A. M. Groundtruth inacity. https://github.com/arturandre/INACITY_groundtruth, 2017. Último acesso em 06/12/2017. 74

[Map] Mappilary. About | mappilary. https://www.mapillary.com/about. Último acesso em 14/3/2018. 1

[MAT13] MATLAB. *R2013a)*. The MathWorks Inc., Natick, Massachusetts, 2013. 108

[Mica] Sun Microsystems. Java blueprints model-view-controller. http://www.oracle.com/technetwork/java/mvc-detailed-136062.html. Último acesso em 13/11/2017. 29

[Micb] Mozilla e individual contributors. Javascript | mdn. https://developer.mozilla.org/en-US/docs/Web/JavaScript. Último acesso em 12/11/2017. 104

[MNH+97] E Gregory McPherson, David Nowak, Gordon Heisler, Sue Grimmond, Catherine Souch, Rich Grant e Rowan Rowntree. Quantifying urban forest structure, function, and value: the chicago urban forest climate project. *Urban ecosystems*, 1(1):49–61, 1997. 13

[MODR+12] Sean W MacFaden, Jarlath PM O'Neil-Dunne, Anna R Royar, Jacqueline WT Lu e Andrew G Rundle. High-resolution tree canopy mapping for new york city using lidar and object-based image analysis. *Journal of Applied Remote Sensing*, 6(1):063567–1, 2012. 21

[MRS] Christopher D Manning, Prabhakar Raghavan e Hinrich Schutze. Introduction to information retrieval‖ cambridge university press, 2008. *Ch*, 20:405–416. 83

[MVS12] Fabrice Monnier, Bruno Vallet e Bahman Soheilian. Trees detection from laser point clouds acquired in dense urban areas by a mobile mapping system. *Proceedings of the ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences (ISPRS Annals), Melbourne, Australia*, 25:245–250, 2012. 13, 14, 15, 55

[NHBG14] David J Nowak, Satoshi Hirabayashi, Allison Bodine e Eric Greenfield. Tree and forest effects on air quality and human health in the united states. *Environmental Pollution*, 193:119–129, 2014. 13

[NPRH] Nikhil Naik, Jade Philipoom, Ramesh Raskar e César Hidalgo. Streetscore. http://streetscore.media.mit.edu/. Último acesso em 13/10/2017. 22, 26

[NPRH14] Nikhil Naik, Jade Philipoom, Ramesh Raskar e César Hidalgo. Streetscore-predicting the perceived safety of one million streetscapes. Em *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, páginas 779–785, 2014. 1, 22

[NR06] Lenna Nepomnyaschy e Nancy E Reichman. Low birthweight and asthma among young urban children. *American Journal of Public Health*, 96(9):1604–1610, 2006. 6

[Oke81] Tim R Oke. Canyon geometry and the nocturnal urban heat island: comparison of scale model and field observations. *International Journal of Climatology*, 1(3):237–254, 1981. 18

[oP14] City of Pasadena. City of pasadena, ca - open data. http://cityofpasadenaca-pasgis.opendata.arcgis.com/, 2014. Último acesso em 08/09/2017. 21

[Ope17] OpenStreetMap contributors. Planet dump retrieved from https://planet.osm.org . https://www.openstreetmap.org, 2017. 1

[PAF16] Ebadat G Parmehr, Marco Amati e Clive S Fraser. Mapping urban tree canopy cover using fused airborne lidar and satellite imagery data. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 3:181, 2016. 56

[PNL+09] Marnie Purciel, Kathryn M Neckerman, Gina S Lovasi, James W Quinn, Christopher Weiss, Michael DM Bader, Reid Ewing e Andrew Rundle. Creating and validating gis measures of urban design for health research. *Journal of environmental psychology*, 29(4):457–466, 2009. 7

[Pow11] David Martin Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. 2011. 82

[RBR+11] Andrew G Rundle, Michael DM Bader, Catherine A Richards, Kathryn M Neckerman e Julien O Teitler. Using google street view to audit neighborhood environments. *American journal of preventive medicine*, 40(1):94–100, 2011. 1, 5, 6, 7, 13

[REC95] ITUR REC. Bt. 601-5: Studio encoding parameters of digital television for standard 4: 3 and wide-screen 16: 9 aspect ratios. https://www.itu.int/rec/R-REC-BT.601-5-199510-S/en, 1995. Último acesso em 23/12/2017. 116

[SAW+06] Mario Schootman, Elena M Andresen, Fredric D Wolinsky, Theodore K Malmstrom, J Philip Miller e Douglas K Miller. Neighborhood conditions and risk of incident lower-body functional limitations among middle-aged african americans. *American Journal of Epidemiology*, 163(5):450–458, 2006. 6

[SCE+07] Mai Stafford, Steven Cummins, Anne Ellaway, Amanda Sacker, Richard D Wiggins e Sally Macintyre. Pathways to obesity: identifying local, modifiable determinants of physical activity and diet. *Social science & medicine*, 65(9):1882–1897, 2007. 6

[Sko90] Wesley G Skogan. *Disorder and decline: Crime and the spiral of decay in American neighborhoods*. Univ of California Press, 1990. 6

[SNRP17] Ian Seiferling, Nikhil Naik, Carlo Ratti e Raphäel Proulx. Green streets-quantifying and mapping urban trees with street-level imagery and computer vision. *Landscape and Urban Planning*, 165:93–101, 2017. 20, 21, 55

[Soi13] Pierre Soille. *Morphological image analysis: principles and applications*. Springer Science & Business Media, 2013. 27, 37, 38, 39, 40, 63

[SR99] Robert J Sampson e Stephen W Raudenbush. Systematic social observation of public spaces: A new look at disorder in urban neighborhoods. *American journal of sociology*, 105(3):603–651, 1999. 6

[SS04] Alex J Smola e Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222, 2004. 22

[SSH13] Philip Salesses, Katja Schechtner e César A Hidalgo. The collaborative image of the city: mapping the inequality of urban perception. *PloS one*, 8(7):e68400, 2013. 22

[SSL12] Andres Sanin, Conrad Sanderson e Brian C Lovell. Shadow detection: A survey and comparative evaluation of recent methods. *Pattern recognition*, 45(4):1684–1695, 2012. 60

[TA78] Robert L Thayer e Brian G Atwood. Plants, complexity, and pleasure in urban and suburban environments. *Journal of Nonverbal Behavior*, 3(2):67–76, 1978. 13

[U.S] U.S. Fish and Wildlife Service. Aerial view of national park forest. http://www.public-domain-image.com/full-image/nature-landscapes-public-domain-images-pictures/national-parks-reserves-public-domain-images-pictures/aerial-view-of-national-park-forest.jpg.html. Último acesso em 12/01/2017. 55

[VHJG95] John Vlissides, Richard Helm, Ralph Johnson e Erich Gamma. Design patterns: Elements of reusable object-oriented software. *Reading: Addison-Wesley*, 49(120):11, 1995. 27, 28, 30, 31, 32, 33

[VSM05] A Vadivel, Shamik Sural e Arun K Majumdar. Human color perception in the hsv space and its application in histogram generation for image retrieval. Em *Color Imaging: Processing, Hardcopy, and Applications*, páginas 598–609, 2005. 34

[WBH+16] Jan D Wegner, Steven Branson, David Hall, Konrad Schindler e Pietro Perona. Cataloging public objects using aerial and street-level images-urban trees. Em *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, páginas 6014–6023, 2016. 21, 25, 55, 92, 93

[WF09] Leland Wilkinson e Michael Friendly. The history of the cluster heat map. *The American Statistician*, 63(2):179–184, 2009. 42

[WK82] James Q Wilson e George L Kelling. Broken windows. *Critical issues in policing: Contemporary readings*, páginas 395–407, 1982. 1, 5

[YZMG09] Jun Yang, Linsen Zhao, Joe Mcbride e Peng Gong. Can you see green? assessing the visibility of urban forests in cities. *Landscape and Urban Planning*, 91(2):97–104, 2009. 14, 16, 18, 55, 81