Universidade de São Paulo

Instituto de Matemática e Estatística

MAC0499 - Trabalho de Formatura Supervisionado

API de navegação no Google Street View e análise de imagens da paisagem urbana

Autores:

Rodrigo Alves Lima

Wallace Faveron de Almeida

São Paulo, SP

2013

Rodrigo Alves Lima

Wallace Faveron de Almeida

API de navegação no Google Street View e análise de imagens da paisagem urbana

Orientador:

Prof. Dr. Roberto Hirata Junior

São Paulo, SP

2013

If everything seems under control, you're just not going fast enough.

Mario Andretti

Sumário

1 Introdução **5**

1.1 Contextualização 5

1.2 Projeto Arquigrafia 6

1.3 Teoria das Janelas Quebradas 7

1.4 Objetivos 8

1.5 Organização do Trabalho 9

2 Tecnologias **10**

2.1 APIs do Google 10

2.1.1 Google Maps JavaScript API 11

2.1.2 Google Directions API 12

2.1.3 Google Geocoding API 13

2.1.4 Google Street View Image API 15

2.2 Hyperlapse.js 16

2.3 Python 17

2.4 Django 17

2.5 Python Imaging Library 18

2.6 OpenCV 18

3 Resultados **19**

3.1 Crawler do banco de dados do projeto Arquigrafia 19

3.2 Interface web 20

3.2.1 Detalhes de funcionamento e implementação 22

3.3 API de navegação no sistema Google Street View 25

3.4 Processamento de imagens do sistema Google Street View . . 28

3.4.1 Reconhecimento de objetos 29

3.4.1.1 Algoritmo SURF 29

3.4.1.2 Implementação 29
3.4.2 Detecção de cores dominantes 30
3.4.2.1 Clusterização *k-means* 31
3.4.2.2 Implementação 32
3.5 Continuidade do desenvolvimento 33

4 Conclusão **34**

5 Referências **35**

6 Parte Subjetiva **37**
6.1 Rodrigo 37
6.1.1 Desafios e frustrações 37
6.1.2 Disciplinas cursadas mais relevantes 38
6.1.3 Trabalhos futuros 38
6.1.4 Considerações finais 39
6.2 Wallace 40
6.2.1 Desafios e frustrações 40
6.2.2 Disciplinas cursadas mais relevantes 41
6.2.3 Trabalhos futuros 42
6.2.4 Considerações finais 43

1 Introdução

1.1 Contextualização

No projeto Arquigrafia, sistema web para compartilhamento de imagens de arquitetura, estruturas e edifícios são catalogados manualmente, o que produz registros incompletos ou inconsistentes. Esse sistema é uma interface para um extenso banco de dados com imagens, informações e classificações de estruturas arquitetônicas [1].

Pesquisas indicam que as características da paisagem urbana influenciam na saúde, no comportamento e no bem-estar das pessoas. A Teoria das Janelas Quebradas, segundo a qual a desordem e o descuido da paisagem urbana são fatores de elevação dos índices de criminalidade, redefiniu políticas de segurança pública [2]. O caso mais emblemático que ilustra essa teoria é a política de Tolerância Zero, símbolo da administração do ex-prefeito de Nova Iorque Rudolph Giuliani, focada na prevenção e na promoção das condições sociais de segurança, que acabou por reduzir os índices criminais da cidade.

Tanto a catalogação de edifícios do projeto Arquigrafia quanto a análise de características da paisagem urbana podem ser feitas com o auxílio do Google Street View, um serviço de visualização panorâmica de vias.

A localização de um edifício no Google Street View define suas coordenadas geográficas de latitude e longitude, o que permite seu posicionamento mais preciso em um mapa. A análise do banco de dados do projeto Arquigrafia revelou que 85% de um total de 2832 endereços de edifícios são, ao menos, parcialmente completos e podem ser localizados corretamente pela

API de geocodificação do Google.

Um estudo realizado em 2009, publicado no *American Journal of Preventive Medicine*, comparou as informações coletadas nas ruas com as obtidas no Google Street View para um relatório de 143 itens sobre a paisagem urbana. Mais da metade dos itens do relatório apresentaram concordância acima de 80%, sendo que apenas itens relacionados a detecção de características de pequeno tamanho ou que possuem variação temporal tiveram baixo índice de concordância. A conclusão do estudo foi, portanto, que o Google Street View pode ser utilizado para coletar informações da paisagem urbana [3].

1.2 Projeto Arquigrafia

O Arquigrafia é uma uma rede social, apoiada por FAPESP e Pró-reitoria de Pesquisa da USP, para a construção colaborativa de um acervo digital de imagens de arquitetura [1].

O projeto reúne uma equipe multidisciplinar de pesquisadores da FAU-USP, IME-USP e ECA-USP, tendo como objetivo principal a contribuição para o estudo, a docência, a pesquisa e a difusão da cultura arquitetônica e urbanística, ao promover interações colaborativas entre pessoas e instituições.

A equipe do projeto trabalha na integração gradual dos cerca de trinta e sete mil slides do acervo da biblioteca da FAU-USP sobre a arquitetura brasileira, catalogando individualmente as imagens. O banco de dados já possui aproximadamente duas mil e oitocentas imagens.

O sistema é construído a partir dos componentes do arcabouço Groupware Workbench, está disponível no endereço `http://www.arquigrafia.org.br` e seu código-fonte é aberto e pode ser usado para a construção de outras redes sociais baseadas no compartilhamento de conteúdo.

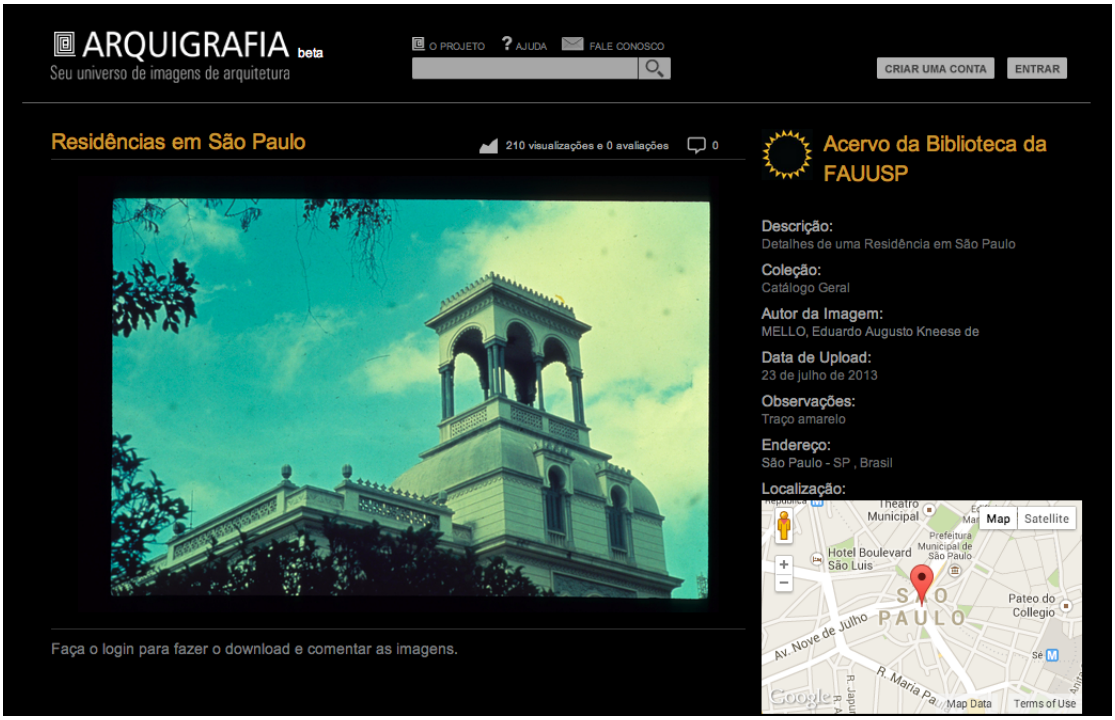

Figura 1: Página inicial do sistema Arquigrafia.

1.3 Teoria das Janelas Quebradas

A Teoria das Janelas Quebradas, publicada em 1982 por James Wilson e George Kelling, defende que a deterioração da paisagem urbana é entendida pela sociedade como ausência dos poderes públicos, o que enfraquece os controles impostos pela comunidade, aumenta a insegurança coletiva e

favorece a prática de crimes [2]. A presença de lixo nas ruas, pichações nas paredes, janelas quebradas e outros indícios de desordem urbana expressam indiferença dos poderes públicos em relação a pequenos delitos e isso acaba por incentivar a prática de delitos mais graves.

A política de limpeza do metrô de Nova Iorque em meados da década de 1980, influenciada por essa teoria, reduziu os índices de criminalidade em suas dependências e posteriormente influenciou a política de Tolerância Zero, implantada pelo prefeito Rudolph Giuliani em toda a cidade, obtendo enorme sucesso. Essa política de segurança se fundamentou na ideia de que, para controlar a ocorrência de crimes violentos, era preciso coibir os pequenos delitos e assim restaurar as noções de ordem, autoridade e segurança.

1.4 Objetivos

Inicialmente desenvolveu-se uma interface web em HTML5 e JavaScript para obter as imagens de rotas, definidas em um mapa pelo usuário, utilizando as APIs do Google.

A fim de permitir a aplicação de técnicas de processamento de imagens e de visão computacional nas visualizações panorâmicas das vias do serviço Google Street View para o reconhecimento de objetos e a extração de informações paisagísticas, iniciou-se o desenvolvimento de uma API para navegação no Google Street View e de algoritmos de processamento de imagens para analisar a paisagem das rotas de interesse.

Dois algoritmos de processamento de imagens foram utilizados, como casos de uso da API de navegação: reconhecimento de objetos, que permite

encontrar um objeto específico em imagens, e detecção de cores dominantes, que permite uma análise paisagística da região da rota.

1.5 Organização do Trabalho

Este trabalho divide-se em duas partes: objetiva e subjetiva.

A parte objetiva apresenta 5 seções, incluindo a introdução, nas quais são apresentados os conceitos abordados, as tecnologias, os resultados do trabalho e suas referências. Os resultados apresentados são: interface web, API de navegação e seus casos de uso, exemplificando a aplicação de técnicas de processamento de imagens. Ao final, estão as conclusões e as referências utilizadas.

A parte subjetiva relata os desafios e frustrações pessoais dos alunos, além de suas opiniões sobre as disciplinas cursadas mais relevantes e as possibilidades de trabalhos futuros.

2 Tecnologias

2.1 APIs do Google

As principais tecnologias utilizadas no desenvolvimento do trabalho foram as APIs de mapas, vistas panorâmicas de vias, geocodificação e cálculo de rotas do Google; a biblioteca Hyperlapse.js, que cria animações utilizando a técnica de fotografia hyper-lapse com imagens obtidas pela API de vistas panorâmicas de vias do Google; a linguagem de programação Python; o arcabouço para desenvolvimento de aplicações web Django; a biblioteca para processamento e manipulação de imagens PIL e a biblioteca de visão computacional OpenCV.

Nos últimos anos, o Google rapidamente expandiu sua oferta de serviços: o gerenciador de e-mails GMail, o navegador web Google Chrome, os aplicativos do Google Drive, o sistema operacional para dispositivos móveis Android e o tradutor Google Translate são apenas alguns exemplos de produtos e serviços que a empresa oferece a seus usuários, além de seu mecanismo de buscas.

Há ainda os serviços de geocodificação, localização em mapas, cálculo de rotas e visualização panorâmica de vias oferecidos pelos sistemas Google Maps e Google Street View [4]. Simplificadamente, esses serviços web representam uma evolução do tradicional guia de ruas, um livro impresso com as vias de cidades específicas. Além de oferecer esses serviços aos usuários, o Google disponibiliza APIs aos desenvolvedores, que podem fazer uso desses recursos em seus próprios sistemas. A integração dessas APIs permite a conversão de endereços em coordenadas geográficas de latitude e longitude,

o cálculo da rota entre dois locais de interesse e a visualização de imagens panorâmicas ao longo de uma rota.

A utilização das APIs possui limite de requisições por conta de usuário, mas o limite pode ser expandido pela inscrição em planos pagos voltados à empresas. Cada conta de usuário que utiliza as APIs do Google possui uma chave que deve ser passada como parâmetro às requisições.

2.1.1 Google Maps JavaScript API

Google Maps JavaScript API oferece visualização e interação com mapas através da linguagem de programação JavaScript. Essa API permite aos desenvolvedores a incorporação de um mapa personalizado em um documento HTML [5].

Os trecho de código a seguir exemplifica a integração de um mapa a um documento HTML.

```
<html>
    <head>
        <script type="text/javascript">
            /* Função de inicialização do mapa */
            function init() {
                var start_point = new google.maps.LatLng(-23.55853,-46.731440000000006);
                var mapOpt = {
                    mapTypeId: google.maps.MapTypeId.ROADMAP,
                    center: start_point,
                    zoom: 15
                };
                map = new google.maps.Map(document.getElementById("map"), mapOpt);
            }
            /* jQuery */
            $(document).ready(init);
        </script>
    </head>
```

```
    <body>
        <div id="map">
        </div>
    </body>
</html>
```

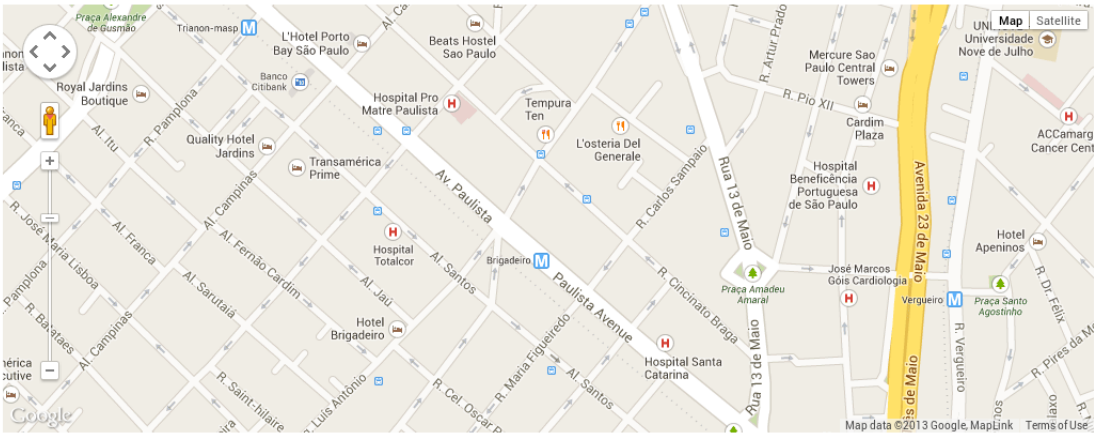

Figura 2: Exemplo de mapa da Google Maps JavaScript API.

2.1.2 Google Directions API

A Google Directions API calcula, por requisições HTTP ou HTTPS, rotas entre duas localidades. As localidades envolvidas podem estar em formato de endereço ou em coordenadas geográficas de latitude e longitude [6].

Os parâmetros obrigatórios para a consulta são:

- **origin:** Especifica o endereço ou coordenadas geográficas de latitude e longitude de origem da rota a ser calculada.
- **destination:** Especifica o endereço ou coordenadas geográficas de latitude e longitude de destino da rota a ser calculada.

- **sensor:** Especifica se a requisição vem de um dispositivo com um sensor de localização GPS.

  Alguns parâmetros opcionais podem personalizar o cálculo da rota:

- **mode:** Especifica o meio de transporte, permitindo o cálculo de rotas para automóveis, pedestres e ciclistas.

- **waypoints:** Especifica um vetor de localidades intermediárias que a rota a ser calculada deverá conter.

- **alternatives:** Especifica se o serviço pode calcular rotas alternativas e assim devolver mais de uma rota em sua resposta.

- **avoid:** Especifica se a rota a ser calcula deve evitar pedágios ou rodovias.

- **language:** Especifica o idioma em que a rota deve ser devolvida.

- **units:** Especifica o sistema de unidades em que a rota deve ser calculada.

A requisição HTTP a seguir calcula a rota entre as cidades de Toronto e Montreal e devolve o resultado em formato JSON:

```
http://maps.googleapis.com/maps/api/directions/json?origin=Toronto&destination=
Montreal&sensor=false
```

2.1.3 Google Geocoding API

A Google Geocoding API calcula, por requisições HTTP ou HTTPS, as coordenadas geográficas de latitude e longitude para um endereço fornecido em formato tradicional: logradouro, número, cidade e estado [7]. Esse

serviço é denominado geocodificação.

A API também converte coordenadas geográficas de latitude e longitude para um endereço em formato tradicional, para ser lido por humanos. Esse serviço é denominado geocodificação reversa.

Os parâmetros obrigatórios para a consulta são:

- **address** ou **latlng:** Especifica o endereço ou coordenadas geográficas de latitude e longitude a ser convertida.
- **sensor:** Especifica se a requisição vem de um dispositivo com um sensor de localização GPS.
  Alguns parâmetros opcionais podem personalizar o cálculo da rota:
- **bounds:** Especifica coordenadas geográficas de latitude e longitude de uma região que influencia o resultado, caso a localidade a ser consultada possa ter mais de um resultado possível, como cidades que possuem o mesmo nome.
- **language:** Especifica o idioma em que a rota deve ser devolvida.

A requisição HTTP a seguir calcula as coordenadas geográficas de latitude e longitude da Rua do Matão, 1010, São Paulo, SP e as devolve em formato JSON:

```
http://maps.googleapis.com/maps/api/geocode/json?address=Rua+do+
Matao,+1010,+Sao+Paulo,+SP&sensor=false
```

2.1.4 Google Street View Image API

A Google Street View Image API devolve, por requisições HTTP ou HTTPS, uma imagem estática de uma vista panorâmica da localidade consultada [8].

Os parâmetros obrigatórios para a consulta são:

- **size:** Especifica o tamanho da imagem em pixels.
- **location:** Especifica a localidade com um endereço ou coordenadas geográficas de latitude e longitude.
- **sensor:** Especifica se a requisição vem de um dispositivo com um sensor de localização GPS.

  Alguns parâmetros opcionais podem personalizar o cálculo da rota:
- **heading:** Especifica o ângulo de rotação em graus da câmera em relação ao ponto cardeal norte.
- **fov:** Especifica o campo de visão da imagem em graus. Valores menores representam maior nível de zoom.
- **pitch:** Especifica o ângulo de inclinação em graus da câmera em relação ao veículo que a suporta. Valores positivos inclinam a câmera para cima; valores negativos inclinam a câmera para baixo.

A requisição HTTP a seguir devolve a imagem da Avenida Paulista, 900, São Paulo, SP, com 600 pixels de largura e 300 pixels de altura; rotação da câmera de 90 graus e inclinação da câmera de 30 graus.

`http://maps.googleapis.com/maps/api/streetview?size=600x300&location=Avenida+Paulista,+900,+Sao+Paulo,+SP&heading=90&pitch=30&sensor=false`

Figura 3: Imagem da Avenida Paulista, 900, São Paulo obtida pela Google Street View Image API.

2.2 Hyperlapse.js

Hyperlapse.js é uma biblioteca que faz uso da técnica homônima de fotografia aplicada a imagens obtidas pela API Google Street View Image [9]. Seu código-fonte foi desenvolvido na linguagem de programação Javascript pela +Labs, uma divisão da empresa TeeHanLax, sob a licença de software MIT.

A biblioteca combina as funcionalidades de outras duas bibliotecas: Tree.js, que disponibiliza uma interface simplificada para o desenho de cenas em duas ou três dimensões em documentos HTML, e GSVPano.js, que facilita requisições à API Google Street View Image e a integração dos resultados

obtidos para a renderização de imagens panorâmicas.

2.3 Python

Python é uma linguagem de programação de alto nível, interpretada, de tipagem forte e dinâmica, multi-paradigma e multi-plataforma criada por Guido van Rossum [10].

Suas principais características, decorrentes de decisões de design que enfatizam a simplicidade feitas por seu criador, são a concisão e a legibilidade. O código escrito em Python, portanto, possui grande expressividade.

Entre outros fatores, essas características tornaram a linguagem Python muito popular e aumentaram substancialmente a quantidade de bibliotecas disponíveis.

2.4 Django

O arcabouço para desenvolvimento de aplicações web Django, escrito na linguagem de programação Python, simplifica a criação de sistemas conhecidos como CRUD (*Create, Read, Update, Delete*), que são sistemas para manipulações simples de banco de dados [11].

Seus componentes promovem o reúso de código, facilitam a criação de interfaces administrativas para sistemas de banco de dados e automatizam tarefas.

Sua arquitetura segue o princípio DRY (*Don't repeat yourself*) e suas aplicações seguem uma variação do padrão de design MVC (*Model-View-*

Controller). Nesse padrão de design, a lógica do negócio, a camada de apresentação e a modelagem do banco de dados são separadas, o que facilita o desenvolvimento, a validação e, principalmente, a manutenção do sistema.

2.5 Python Imaging Library

A Python Imaging Library, ou simplesmente PIL, é uma biblioteca multi-plataforma da linguagem de programação Python para processamento e manipulação de imagens [12].

Algumas de suas funcionalidades são: manipulação de máscaras e transparências; aplicação de filtros, como suavização e detecção de bordas; e ajustes de brilho e contraste.

2.6 OpenCV

OpenCV é uma biblioteca multiplataforma, escrita nas linguagens de programação C e C++, para o desenvolvimento de sistemas de visão computacional [13]. Pode ser utilizada, através de adaptadores, em diversas linguagens de programação, como Python, Java, C# e Visual Basic.

A biblioteca possui módulos de processamento de vídeos e imagens, estrutura de dados, álgebra linear e mais de 350 algoritmos de visão computacional, como filtros de imagem, calibração de câmera, reconhecimento de objetos e análise estrutural.

3 Resultados

3.1 Crawler do banco de dados do projeto Arquigrafia

Desenvolveu-se um *web crawling*, programa de computador que navega pela web de forma metódica e automática com o objetivo de extrair informações, para o projeto Arquigrafia.

Esse programa, *crawler.py*, foi escrito na linguagem de programação Python e faz o download de todos os documentos relativos a imagens do banco de dados, extrai suas informações utilizando expressões regulares e as escreve em um arquivo em formato CSV (*comma-separated values*).

Outro programa, *populate_bd.py*, escrito na linguagem de programação Python, lê esse arquivo CSV com todas as informações do projeto Arquigrafia estruturadas e popula um banco de dados SQLite de uma aplicação web desenvolvida com o arcabouço Django.

A aplicação web possui uma interface de administração que permite a edição de informações e categorização das imagens do projeto Arquigrafia. Para o desenvolvimento do trabalho, duas categorizações foram feitas: se o edifício ou estrutura da imagem pode ser visto a partir de uma via pública e, portanto, pode ser identificada em imagens do Google Street View; se o endereço do banco de dados está, ao menos, parcialmente completo e, portanto, pode ser geocodificado pelo API do Google.

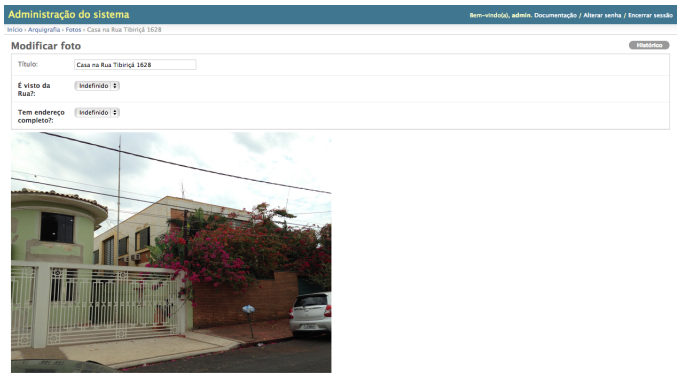

Figura 4: Classificação de estrutura do projeto Arquigrafia.

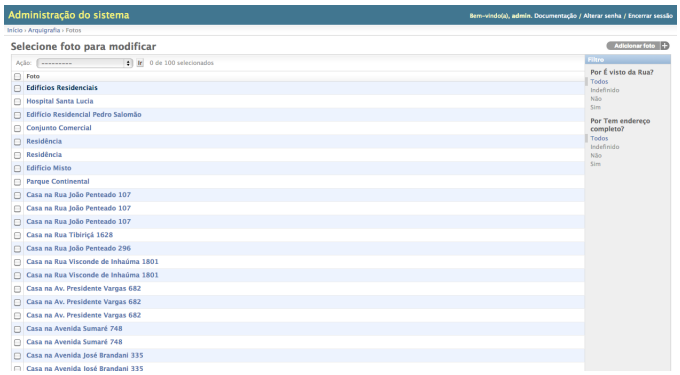

Figura 5: Lista de imagens obtidas do projeto Arquigrafia.

3.2 Interface web

Uma requisição à API do Google Street View Image devolve uma imagem que pode ser incorporada a um documento HTML. A estratégia utilizada inicialmente para obter as imagens de um trajeto foi realizar requisições sequenciais à API, utilizando o programa de linha de comando *curl*, aumentando o número aproximado do endereço utilizado, ou variando o valor das coordenadas geográficas de latitude e longitude. O resultado, no entanto, foi impreciso, além de não proporcionar uma interface intuitiva para utilização.

Havia, portanto, a necessidade de se desenvolver um sistema mais ela-

borado e intuitivo que exibisse imagens sequenciais de um trajeto definido. Iniciou-se, então, o desenvolvimento de uma interface web que atendesse a esses requisitos.

A interface web é um sistema desenvolvido em HTML5, CSS e JavaScript que proporciona ao usuário uma interface amigável, exibindo imagens sequenciais do sistema Google Street View em um mesmo quadro, o que produz o efeito de navegação. É possível, ainda, alterar alguns parâmetros de visualização durante a exibição das imagens de um trajeto, tais como a rotação e a inclinação da câmera, o campo de visão e o tempo de exibição de cada imagem.

Para utilizar a interface, o usuário informa o endereço a ser localizado. A API de mapas do Google busca o endereço e o exibe no mapa. O usuário deve, então, posicionar um marcador no início do trajeto e outro no final do trajeto, clicar no botão "Percorrer" e, possivelmente, alterar os parâmetros de navegação.

Figura 6: Interface web para manipulação das APIs de mapas do Google.

3.2.1 Detalhes de funcionamento e implementação

Os principais arquivos de código-fonte JavaScript, responsáveis pela manipulação das APIs do Google são o *interface.js* e o *maps_ init.js*. Logo após o documento ser carregado pelo navegador, o objeto JavaScript que representa o mapa é criado, utilizando-se a Google Maps JavaScript API.

Os marcadores do mapa são representados por objetos do tipo *Marker*. O trecho de código a seguir constrói o objeto definindo as propriedades *position*, vinculada a uma coordenada geográfica, *draggable*, indicando se é possível mover o marcador pelo mapa, e *map*, que representa o objeto mapa no qual o marcador será exibido.

```
var start_point  = new google.maps.LatLng(-23.55853,-46.731440000000006);
start_pin = new google.maps.Marker({
    position: start_point,
    draggable: true,
    map: map
});
```

Assim que um dos marcadores é posicionado, é necessário obter as informações de sua localização para o cálculo da rota. Para isso, registra-se no mapa um *eventListener* para o posicionamento dos marcadores.

```
directions_service = new google.maps.DirectionsService();
directions_renderer = new google.maps.DirectionsRenderer(
     {
       draggable:false,
       markerOptions:{visible: false}
     });
directions_renderer.setMap(map);
directions_renderer.setOptions({preserveViewport:true});
```

```
google.maps.event.addListener (start_pin, 'dragend', function (event) {
    snapToRoad(start_pin.getPosition(), function(result) {
        start_pin.setPosition(result);
        start_point = result;
    });
});
```

Antes de qualquer interação com o usuário, é necessário construir o objeto de exibição de imagens da biblioteca Hyperlapse.js e cadastrar seus *eventListeners*. O objeto *pano* representa o elemento do documento HTML que receberá, como filho na hierarquia de elementos do documento, um elemento *<canvas>*, no qual serão exibidas as imagens do trajeto.

```
var pano = document.getElementById('pano');
  hyperlapse = new Hyperlapse(pano, {
          fov: 80,
          millis: 250,
          width: 400,
          height: 300,
          zoom: 2,
          use_lookat: false,
          distance_between_points: 5,
          max_points: 100,
          elevation: 0
      });
```

Após a solicitação de localização de um endereço pelo usuário, é realizada uma chamada a função *findAddress*, que tenta geocodificar o endereço. Em caso de resposta válida, o mapa exibe a região nas proximidades do endereço encontrado e os marcadores são posicionados.

```
geocoder = new google.maps.Geocoder();
function findAddress(address) {
        geocoder.geocode( { 'address': address }, function(results, status) {
            if (status == google.maps.GeocoderStatus.OK) {
```

```
                map.setCenter(results[0].geometry.location);
                drop_pins(); /* Posiciona marcadores */
            } else {
                /* Erro! */
            }
        });
}
```

Assim que os marcadores de início e fim do trajeto são posicionados, calculam-se os pontos intermediários, que são adicionados ao mapa. As informações referentes a cada ponto do trajeto serão, então, utilizadas para obter sua imagem. A função *generate*, que realiza uma chamada à função *hyperlapse.generate* implementa essa ação.

```
function generate(){
    console.log( "Gerando rota..." );

    directions_renderer.setDirections({routes: []});

    var marker;
    while(_route_markers.length > 0) {
        marker = _route_markers.pop();
        marker.setMap(null);
    }

    request = {
        origin: start_point,
        destination: end_point,
        travelMode: google.maps.DirectionsTravelMode.DRIVING
    };

    directions_service.route(request, function(response, status) {
        if (status == google.maps.DirectionsStatus.OK) {
            hyperlapse.generate({route: response});
        } else {
            /* Erro! */
        }
    })
```

```
}
```

Ao término da geração da rota pela API, a chamada à função *hyperlapse.play* inicia a exibição das imagens do trajeto no elemento *<canvas>*.

```
hyperlapse.onLoadComplete = function(e) {
    hyperlapse.position.x = -90;
            hyperlapse.play();
   };
```

3.3 API de navegação no sistema Google Street View

A API de navegação no sistema Google Street View, escrita na linguagem de programação Python, encapsula e estende funções de geocodificação, cálculo de rotas e obtenção de imagens das APIs: Google Geocoding, Google Directions e Google Street View Image.

As principais classes do sistema são:

- **Point:** representa um ponto na terra definida por suas coordenadas geográficas de latitude e longitude.
  Seus principais métodos estáticos são:
  - for_address(cls, address): recebe um endereço a ser geocodificado e devolve um objeto *Point* ou *None*, se a API de geocodificação do Google devolveu zero ou mais de um resultado.
  - deg_to_rad(deg): converte um ângulo em graus para radianos.
  - rad_to_deg(rad): converte um ângulo em radianos para graus.

  Seus principais métodos são:

  - latitude(self): devolve a latitude em graus.
  - latitude_in_rad(self): devolve a latitude em radianos.
  - longitude(self): devolve a longitude em graus.
  - longitude_in_rad(self): devolve a longitude em radianos.
  - bearing_to(self, point): devolve o ângulo em graus, relativo ao ponto cardeal norte, até outro ponto.

- **RoutePoint:** representa um ponto em uma rota, especificado pelas suas coordenadas geográficas de latitude e longitude e o ângulo em graus, relativo à frente do veículo equipado com a câmera do sistema Google Street View, até o próximo ponto da rota.
  Seu principal método é:

  - google_street_view_image(self, heading=0, pitch=0, width=640, height=640): devolve um objeto *StreetViewImage* que representa a imagem do ponto no sistema Google Street View, com as opções especificadas de inclinação e rotação da câmera e tamanho da imagem.

- **Route:** representa uma rota definida por uma sequência de pontos.
  Seu principal método estático é:

  - between(cls, start_point, end_point, max_distance_between_points=20): devolve um objeto *Route* de uma rota entre dois pontos, com pontos vizinhos na rota não tendo uma distância maior do que a especificada, ou *None*, se a API de cálculo de rotas do Google não devolver um resultado. A API de cálculo de rotas do Google devolve apenas pontos de mudança de direção na trajetória, portanto são obtidos pontos intermediários entre esses com

base em uma aproximação: são calculados pontos intermediários nas retas entre pares de pontos consecutivos devolvidos pela API de cálculo de rotas do Google a cada distância especificada pelo usuário.

Seu principal método é:

- google_street_view_images(self, heading=0, pitch=0, width=640, height=640): devolve uma lista de objetos *StreetViewImage*, que representam as imagens dos pontos da rota no sistema Google Street View, com as opções especificadas de inclinação e rotação da câmera e tamanho da imagem.

Essas classes permitem um alto nível de abstração no código que, aliada à expressividade da linguagem de programação Python, resulta em códigos simples e legíveis.

O exemplo de código a seguir exemplifica a obtenção dos endereços das imagens de uma rota entre um ponto de origem e um ponto de destino.

```
origin_point = Point.for_address(origin)
destination_point = Point.for_address(destination)
route = Route.between(origin_point, destination_point,
maximum_distance_between_points)

for image in route.google_street_view_images(heading, pitch):
    # image.url(API_KEY)
```

As imagens a seguir ilustram uma sequência de imagens de pontos consecutivos em uma rota na Avenida Paulista, localizada na cidade de São Paulo.

Figura 7: Imagens consecutivas da Avenida Paulista, São Paulo obtidas pela Google Street View Image API.

3.4 Processamento de imagens do sistema Google Street View

Associadas à API de navegação no sistema Google Street View, algoritmos de visão computacional e técnicas de processamento de imagens permitem a exploração automática da paisagem urbana.

Algoritmos de visão computacional, por exemplo, podem reconhecer um objeto específico ou um padrão em imagens de uma rota, enquanto técnicas de processamento de imagens podem extrair informações relevantes para a análise da paisagem, como as suas cores dominantes.

Desenvolveu-se um módulo de processamento de imagens associado à API de navegação no sistema Google Street View com os seguintes algoritmos, como exemplos de casos de uso.

3.4.1 Reconhecimento de objetos

Algoritmos de reconhecimento de objetos em imagens tem a finalidade de detectar a presença de um objeto específico, definido em uma imagem, em imagens genéricas. Dessa forma, é possível, por exemplo, localizar uma estrutura ou edifício de uma imagem do banco de dados do projeto Arquigrafia no sistema Google Street View e assim obter suas coordenadas geográficas de latitude e longitude.

3.4.1.1 Algoritmo SURF

O algoritmo utilizado para o reconhecimento de objetos foi o SURF (*Speeded up robust features*), que é composto por um detector de pontos característicos e um descritor de pontos [14]. Uma das principais propriedades do algoritmo é a extração de pontos característicos que são invariantes com relação a rotação e escala, calculados a partir de uma matriz Hessiana baseada na imagem do objeto.

O reconhecimento de um objeto, resumidamente, consiste na busca de seus pontos característicos em uma imagem. Para cada ponto característico detectado, o algoritmo SURF utiliza o descritor de pontos para o cálculo de um identificador baseado na textura ao redor do ponto.

3.4.1.2 Implementação

Utilizou-se a implementação do algoritmo SURF da biblioteca de visão computacional OpenCV. O adaptador para a linguagem de programação Python, pyopencv, permitiu a integração à API de navegação no sistema

Figura 8: Pontos característicos encontrados pelo algoritmo SURF em prédios do banco de dados do projeto Arquigrafia.

Google Street View.

A expressividade e legibilidade do código de reconhecimento de um objeto em uma rota calculada pela API de navegação pode ser vista no exemplo a seguir.

```
origin_point = Point.for_address(origin)
destination_point = Point.for_address(destination)
route = Route.between(origin_point, destination_point,
maximum_distance_between_points)

for image in route.google_street_view_images(heading, pitch):
# image.surf(API_KEY, object_filename)
```

3.4.2 Detecção de cores dominantes

A detecção das cores dominantes de uma rota permite a análise paisagística de uma região. A presença de uma sequência de cores dominantes verdes, por exemplo, pode significar a presença de um parque ou região ar-

borizada, enquanto a presença de uma sequência de cores dominantes cinzas pode significar a presença de uma região industrial.

Os exemplos a seguir ilustram a análise paisagística possível pela detecção das cores dominantes da paisagem. A Avenida Paulista, em São Paulo, é caracterizada pela presença marcantes de tons de cinza, com destaque para o ponto vermelho do Museu de Arte de São Paulo Assis Chateaubriand, o MASP; a Avenida Doutor Arnaldo, em São Paulo, é caracterizada pelas áreas verdes da Faculdade de Medicina da Universidade de São Paulo; a Quinta Avenida, em Nova Iorque, é caracterizada por uma extensa área verde do *Central Park* e tons de cinza típicos de uma região comercial.

Figura 9: Cores dominantes em margem da Avenida Paulista, São Paulo.

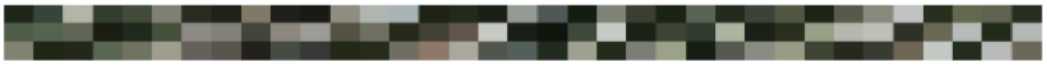

Figura 10: Cores dominantes em margem da Avenida Doutor Arnaldo, São Paulo.

Figura 11: Cores dominantes em margem da Quinta Avenida, Nova Iorque.

3.4.2.1 Clusterização *k-means*

O algoritmo utilizado para a detecção das cores dominantes foi o de clusterização *k-means*, que agrupa os pixels pela semelhança de suas cores

em grupos [15]. Os centros dos *clusters*, grupos de semelhança, resultantes são as cores que prevalecem nas imagens.

Inicialmente, um número de pixels aleatórios da imagem igual ao número de cores dominantes desejadas é selecionado: são os *clusters* iniciais. Em uma iteração do algoritmo, cada pixel é inserido no *cluster* cuja distância euclidiana entre a cor do pixel e a do centro do *cluster* é a menor; após todos os pixels serem analisados, a o centro, que é a média das cores, de cada *cluster* é recalculada. Quando a distância euclidiana entre as cores dos centros e os novos centros de todos os *clusters* é menor do que um valor específico, as iterações são encerradas.

3.4.2.2 Implementação

Utilizou-se a biblioteca de processamento de imagens PIL (Python Imaging Library) para a obtenção das cores dos pixels e implementou-se o algoritmo de clusterização *k-means*.

A expressividade e legibilidade do código de detecção das cores dominantes em uma rota calculada pela API de navegação pode ser vista no exemplo a seguir.

```
origin_point = Point.for_address(origin)
destination_point = Point.for_address(destination)
route = Route.between(origin_point, destination_point,
maximum_distance_between_points)

for image in route.google_street_view_images(heading, pitch):
# colors = image.dominant_colors(API_KEY, num_of_colors)
```

3.5 Continuidade do desenvolvimento

A fim de aumentar suas funcionalidades, pretende-se continuar o desenvolvimento da API de navegação no sistema Google Street View e implementar outros algoritmos de visão computacional e processamento de imagens.

A próxima funcionalidade a ser implementada na API de navegação no sistema Google Street View é a exploração de todas as vias dentro de um polígono passado como parâmetro. Dessa forma, um polígono definido pelo usuário devolveria uma lista de objetos *Route*, permitindo a aplicação das técnicas de processamento de imagens e algoritmos de visão computacional em um bairro, cidade, estado ou país. Pelo limite de requisições imposto, tal funcionalidade só poderia ser aplicada a polígonos de grande área com a utilização de uma conta de serviços do Google com um maior limite de requisições.

O próximo algoritmo de visão computacional a ser implementado é o de detecção de padrões em imagens. Dessa forma, o usuário pode selecionar um conjunto de imagens de treinamento, como, por exemplo, buracos em ruas, para então identificá-los nas imagens das vias. Esse algoritmo está disponível na biblioteca OpenCV e utiliza uma sequência de classificadores para a detecção dos padrões.

4 Conclusão

Apresentamos aqui algumas tecnologias que estudamos e utilizamos no desenvolvimento do trabalho, além dos principais resultados obtidos: a interface web para manipulação das APIs de mapas do Google, a API de navegação no sistema Google Street View escrita na linguagem de programação Python e dois casos de uso que exemplificam a análise da paisagem urbana, bem como uma sucinta apresentação dos algoritmos SURF e clusterização *k-means*. Buscamos ainda apresentar os resultados com imagens, que são fundamentais para a compreensão das técnicas de processamento de imagens e algoritmos de visão computacional, e visualização da interface web desenvolvida em HTML5, CSS e JavaScript.

Esse trabalho nos levou ao estudo de diversos artigos e algoritmos e ao aprendizado de novas tecnologias. Os resultados obtidos compravaram que as APIs de mapas do Google, associadas a técnicas de processamento de imagens, podem ser utilizadas para a extração automática de informações sobre a paisagem urbana. A API de navegação no sistema do Google Street View desenvolvida é o cerne para essa exploração e será disponibilizada sob uma licença de software livre.

5 Referências

Referências

[1] Arquigrafia. Disponível em: <http://www.arquigrafia.org.br>. Acesso em: 01/11/2013.

[2] James Q. Wilson, George L. Kelling. BROKEN WINDOWS: The police and neighborhood safety. Disponível em: <http://www.manhattan-institute.org/pdf/_atlantic_monthly-broken_windows.pdf>. Acesso em: 01/11/2013.

[3] RUNDLE, Andrew G. et al. Using Google Street View to Audit Neighborhood Environments. *American Journal of Preventive Medicine*. Janeiro de 2011.

[4] Google Maps API. Disponível em: <https://developers.google.com/maps/documentation>. Acesso em: 01/11/2013.

[5] Google Maps JavaScript API Disponível em: <https://developers.google.com/maps/documentation/javascript/tutorial>. Acesso em: 01/11/2013.

[6] Google Directions API Disponível em: <https://developers.google.com/maps/documentation/directions>. Acesso em: 01/11/2013.

[7] Google Geocoding API Disponível em: <https://developers.google.com/maps/documentation/geocoding>. Acesso em: 01/11/2013.

[8] Google Street View Image API. Disponível em: <https://developers.google.com/maps/documentation/streetview>. Acesso em: 01/11/2013.

[9] Hyperlapse. Disponível em: <http://hyperlapse.tllabs.io>. Acesso em: 01/11/2013.

[10] About Python. Disponível em: <http://www.python.org/about>. Acesso em: 01/11/2013.

[11] Django. Disponível em: <https://www.djangoproject.com>. Acesso em: 01/11/2013.

[12] Python Imaging Library. Disponível em: <http://www.pythonware.com/products/pil>. Acesso em: 01/11/2013.

[13] OpenCV. Disponível em: <http://www.opencv.org>. Acesso em: 01/11/2013.

[14] OpenCV - Feature Detection and Description. Disponível em: <http://docs.opencv.org/modules/nonfree/doc/feature_detection.html>. Acesso em: 01/11/2013.

[15] Using python and k-means to find the dominant colors in images. Disponível em: <http://charlesleifer.com/blog/using-python-and-k-means-to-find-the-dominant-colors-in-images>. Acesso em: 01/11/2013.

6 Parte Subjetiva

6.1 Rodrigo

6.1.1 Desafios e frustrações

O primeiro desafio do trabalho foi a escolha do tema. Com a ajuda do professor Roberto Hirata Junior, escolhemos uma aplicação prática de visão computacional, tendo como base o projeto Arquigrafia, desenvolvido pelo Instituto de Matemática e Estatística da Universidade de São Paulo, em conjuntos com outras unidades.

A partir daí, estudamos muitos algoritmos e tecnologias de visão computacional e nos propomos a analisar o impacto da aplicação de nosso trabalho no projeto Arquigrafia. Da dificuldade em medir esse impacto, surgiu um dos resultados do trabalho: um software de web crawling para o website.

Encontramos inúmeras dificuldades ao longo do trabalho, relacionadas às tecnologias utilizadas e aos resultados que não nos satisfaziam. Entretanto, as sugestões de nosso orientador deram uma nova direção ao trabalho: reunimos o que funcionava, abandonamos o que não tinha viabilidade e desenvolvemos o que, talvez, seja o principal resultado do trabalho: a API de navegação no sistema Google Street View.

Meu principal desafio ao longo do ano foi gerenciar minha disponibilidade de tempo. A conciliação do desenvolvimento do trabalho de conclusão de curso com as disciplinas cursadas na graduação, atividades na atlética e trabalhos profissionais se mostrou difícil, mas, ao final, posso dizer que o desafio foi superado.

6.1.2 Disciplinas cursadas mais relevantes

Destaco as disciplinas que foram úteis para o desenvolvimento do trabalho e para meu desenvolvimento pessoal e profissional:

- **MAC0110 - Introdução à Computação:** apesar de já ter experiência em programação, essa disciplina foi de grande importância para o aprendizado de técnicas de documentação de código que foram utilizadas ao longo de todo o curso.
- **MAC0122 - Princípios de Desenvolvimento de Algoritmos e MAC0323 - Estruturas de Dados:** sem dúvidas, são o cursos mais importantes do Bacharelado em Ciência da Computação, por todo o conteúdo teórico de algoritmos.
- **PCS2590 - Criação e administração de empresas de computação:** essa disciplina, cursada na Escola Politécnica, me deu uma visão de como estruturar e viabilizar uma empresa inovadora de computação para a solução de problemas reais.

6.1.3 Trabalhos futuros

Dando continuidade ao desenvolvimento dos resultados do trabalho, vamos disponibilizar o código-fonte da API de navegação no sistema Google Street View sob uma licença de software livre. Após testar e consolidar ainda mais a base de código atual, vamos adicionar novas funcionalidades já descritas na seção de resultados.

Além de incrementar os softwares desenvolvidos, pretendemos enviar para a equipe do projeto Arquigrafia uma base de dados de coordenadas geográficas de latitude e longitude de todas as suas estruturas localizadas com o algoritmo de reconhecimento de objetos.

6.1.4 Considerações finais

Finalizar esse trabalho é uma grande satisfação. Desenvolvê-lo ao lado de um grande amigo, Wallace, e sob a orientação do professor Roberto Hirata Junior é ainda mais gratificante. Todo o esforço empregado valeu a pena.

Gostaria de agradecer aos amigos do IME sem os quais os dias no Instituto não seriam tão divertidos; do Núcleo de Empreendedorismo da USP, por todos os conhecimentos adquiridos; da IME Jr, pelo desenvolvimento profissional; da Associação Atlética Acadêmica da Matemática, em especial ao time de handebol, por proporcionarem momentos inesquecíveis nas vitórias e nas derrotas, dentro e fora das quadras; da Pró-Reitoria de Pós-graduação da USP, pela primeira experiência profissional; finalmente, à minha família, sem a qual nada disso seria possível.

6.2 Wallace

6.2.1 Desafios e frustrações

Aproximadamente cinco anos se passaram desde o dia da minha matrícula no curso de Bacharelado em Ciência da Computação desse Instituto. Conquistei essa vaga ao ser aprovado em 2ª chamada no vestibular, quando este curso e os de Engenharia da Escola Politécnica, ainda pertenciam a uma mesma carreira.

Frequentei o Ensino Fundamental em escolas públicas do meu bairro. Concluído esse ciclo, obtive aprovação no "vestibulinho" para cursar o Ensino Médio em uma escola técnica estadual. Em seguida, estudei mais um ano em um cursinho pré-vestibular antes de iniciar minha graduação.

Após uma reunião no trabalho, recebi de minha namorada na época, uma mensagem de texto no celular me parabenizando. Meu nome estava na lista de aprovados. Não conseguia descrever em palavras a minha felicidade em poder estudar na USP.

Depois de todo esse tempo e do esforço empregado, é chegado o momento de concluir mais uma fase da minha vida. Com o forte objetivo de me formar, compartilhei durante esse último ano, as tarefas do trabalho de conclusão de curso com um dos grandes amigos que tive a oportunidade de fazer no período em que fui aluno de graduação.

6.2.2 Disciplinas cursadas mais relevantes

Um dos primeiros desafios do TCC foi a definição do tema. Existiam algumas possiblidades, mas era necessário escolher cuidadosamente para que pudéssemos chegar até o fim e obter sucesso. Tendo decidido o tema, entramos em contato com o professor Hirata para que ele pudesse ser o nosso orientador ao longo do trabalho.

Fizemos algumas reuniões para então iniciarmos nossas atividades. Após algumas dessas reuniões, tivemos a necessidade de alterar o foco do que estava sendo estudado e desenvolvido. Por fim, chegamos à atual abordagem e aos resultados aqui apresentados.

Algumas restrições de segurança da linguagem Javascript atrasaram e trouxeram limitações ao desenvolvimento da interface web. Além disso, eu tinha pouca experiência na linguagem Python. Foi um desafio de aprendizagem, para que eu pudesse contribuir com a implementação dos módulos que fazem parte da API de navegação.

Estavámos cientes que ao realizar um trabalho de conclusão de curso em dupla, seria necessário encontrar horários comuns aos integrantes para realizar as atividades e reuniões, o que nem sempre seria uma tarefa simples, além da exigência um pouco maior quanto a qualidade e aos resultados.

Por fim, meu maior desafio para o TCC, e também para a graduação, foi o de aprimorar a capacidade de conciliar as disciplinas do semestre, os prazos e as cobranças do ambiente profissional e as demais atividades fora da universidade. Ainda tenho muito a melhorar, mas posso dizer que, em

partes, obtive sucesso.

- **MAC0110 - Introdução à Computação:**Mesmo tendo concluído um curso técnico e já atuando na área, essa disciplina permitiu complementar algumas deficiências de aprendizado, além de me apresentar uma noção do que seria cobrado de um aluno no ensino superior.
- **MAC0122 - Princípios de Desenvolvimento de Algoritmos e MAC0323 - Estruturas de Dados:** Essas disciplinas permitiram que eu conhecesse algoritmos clássicos da área e implementação de estruturas já fornecidas por algumas linguagens de programação de alto-nível.
- **MAC0342 - Laboratório de programação eXtrema:** Proporcionou a aquisição de um conhecimento teórico e formal de técnicas conhecidas previamente no mercado, além da oportunidade de palestras com ex-alunos.
- **MAC0316 - Conceitos fundamentais de linguagens de programação:** Nessa disciplina foi possível obter os primeiros contatos com outros importantes paradigmas de programação, em particular o paradigma funcional.

6.2.3 Trabalhos futuros

Um dos objetivos do trabalho era fornecer uma API que pudesse ser um agente facilitador para utilização dos serviços do Google StreetView e para análise da paisagem urbana, não apenas para os casos de uso apresentados, mas também para atuar em conjunto com alguns classificadores.

Como possíveis expansões e abordagens futuras para esse trabalho estão uma integração mais otimizada entre os resultados da interface web para serem melhor utilizados como dados de entrada da API de navegação em Python, além da adição de funcionalidades aos módulos já existentes ou ainda a implentação de novos módulos.

6.2.4 Considerações finais

Hoje, após esse período em que fui aluno do BCC, posso afirmar que não foi uma tarefa fácil chegar até aqui, muito perto de concluir o curso. Foram muitos os desafios encontrados, obstáculos superados, noites mal dormidas, ou mesmo em claro. Mesmo assim, a impressão é que o tempo passou muito rápido.

Meus agradecimentos são destinados para aqueles que estiveram presentes em minha vida de alguma forma durante esses anos: familiares, professores, amigos e Deus.

Aos meus familiares, que sempre me incentivaram a continuar estudando e ofereceram todo o suporte que estava ao alcance deles para que isso fosse possível.

Aos professores do Instituto e ao orientador, que compartilharam parte de seu conhecimento e contribuíram para o meu aprendizado.

Aos meus amigos de infância, que me ensinaram que devemos contribuir com o mundo e com as pessoas além dos muros da universidade.

Aos amigos que fiz no curso, que me ajudaram e serviram de exemplo, além de terem proporcionado alegres momentos e ótimos bate-papos, cada um à sua maneira particular. Espero ter conseguido contribuir com eles da mesma forma.

Aos meus colegas de trabalho, sempre prestativos, com quem pude compartilhar, e também ouvir, aflições acadêmicas e profissionais.

Por fim, a Deus pela oportunidade única que recebi de traçar meu caminho de vida dessa maneira. Não consigo imaginar de outro jeito. Se por acaso tivesse que escolher mais uma vez, faria tudo novamente, para chegar mais uma vez onde estou e assim continuar crescendo como profissional e principalmente como pessoa. Meu muito obrigado a todos!