LOCALIZAÇÃO GEOGRÁFICA DE INCÊNDIOS POR MEIO DE IMAGENS

15 de Outubro de 2019

Rafael Henrique Vareto
Universidade Federal de Minas Gerais
Departamento de Ciência da Computação

Introdução

O desafio proposto se refere, basicamente, ao problema de correspondência de pontos entre imagens que, embora caracterizam uma mesma cena, são obtidas de pontos de vista diferentes. Mais especificamente, o problema da correspondência refere-se ao problema de determinar quais partes de uma imagem correspondem a quais partes de outra imagem, onde as diferenças são devidas ao movimento da câmera, ao decorrer do tempo e ao movimento dos objetos capturados nas fotos.

O Problema

Dadas duas ou mais imagens da mesma cena 3D, tiradas de diferentes pontos de vista, o problema da correspondência refere-se à tarefa de encontrar um conjunto de pontos em uma imagem que pode ser identificado como os mesmos pontos em outra imagem. Para fazer isso, os pontos ou recursos em uma imagem são combinados com os pontos ou recursos correspondentes em outra imagem. As imagens podem ser tiradas de um ponto de vista diferente, em momentos diferentes ou com objetos na cena em movimento geral em relação às câmeras.

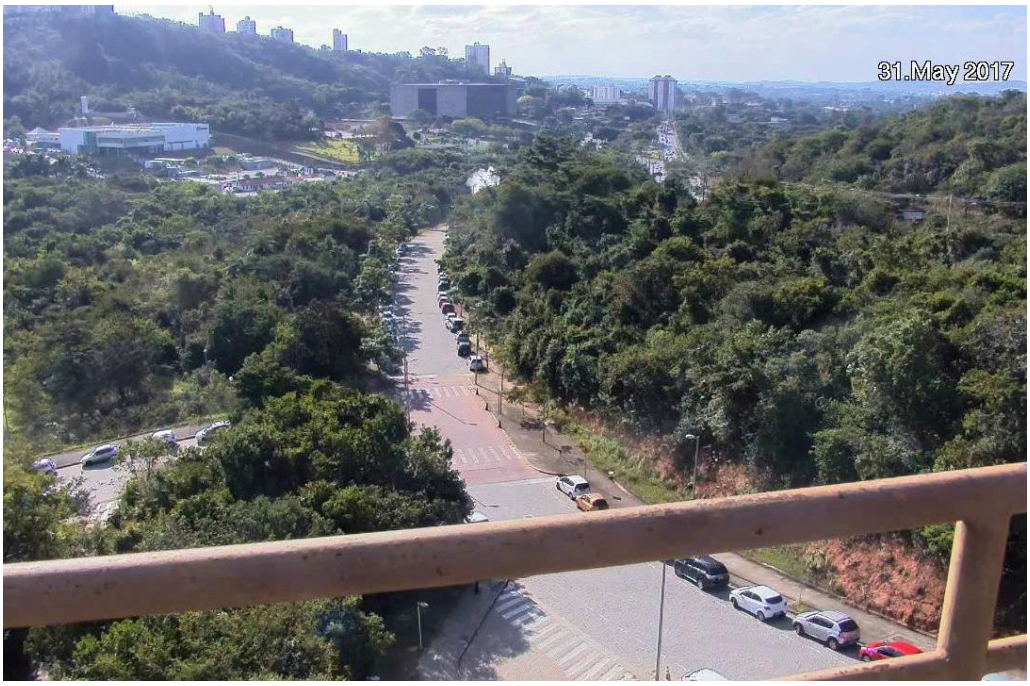

Figura 1: Fotografia tirada do prédio da BHTec, na região da Pampulha em Belo Horizonte, MG.

Assim, com as posições geográficas[1] da câmera $p = (-19.885087, -43.974836)$, tendo seu ponto de vista demonstrado na Figura 1 de resolução 1087×717, os dois objetivos do desafio são:

1. Encontrar a posição geográfica do pixel $e = (412, 598)$ e avaliar o erro sabendo que a posição geográfica correspondente são as coordenadas $p_e = (-19.877751, -43.978796)$.

2. Encontrar a posição geográfica do pixel $f = (510, 523)$, o qual apresenta indícios de incêndio.

Os únicos dados disponíveis para encontrar a correspondência entre os pixeis da Figura 1 e as coordenadas[2] são:

- Pixel $a = (552, 238)$ com coordenadas $p_a = (-19.884362, -43.975112)$.
- Pixel $b = (220, 568)$ com coordenadas $p_b = (-19.880996, -43.978154)$.
- Pixel $c = (660, 613)$ com coordenadas $p_c = (-19.875391, -43.977202)$.

Vale a pena ressaltar que não foi utilizado o pixel $e = (412, 598)$ e coordenadas $p_e = (-19.877751, -43.978796)$ para o desafio 1, pois isto aumentaria drasticamente a qualidade dos nossos resultados visto que queremos calcular o erro sem o conhecimento prévio da coordenada p_e. Por outro lado, adicionamos o pixel e e sua coordenada correspondente p_e para o desafio 2 com o intuito de utilizar a maior quantidade possível de informação para encontrar o foco do possível incêndio.

Literatura

A estimação de posição é o resultado direto da reconstrução de cenas, um dos ramos da visão computacional. A grande maioria dos métodos existentes na literatura alcançam bom desempenho com imagens tiradas aproximadamente com o mesmo ponto de vista. Não é o caso deste desafio, visto que coordenadas são melhor representadas por visões aéreas e espaciais.

Uma das formas mais eficientes de encontrar a correspondência de pontos entre imagens é através da matriz de homografia [1]. Uma homografia é uma transformação que mapeia os pontos em uma imagem para os pontos correspondentes na outra imagem.

[1] Posições geográficas se referem à latitude e longitude.
[2] Coordenadas são verificadas pelo Google Maps: `https://goo.gl/maps/gCP1HhLgY4CpPeLK9`

Para calcular uma homografia entre duas imagens, é necessário conhecer pelo menos 4 correspondências de pontos entre as duas imagens. Se você tiver mais de 4 pontos correspondentes, é ainda melhor. Essas correspondências de pontos entre imagens podem encontradas automaticamente através de algoritmos de detectam *keypoints* como SIFT [2] ou SURF [3].

Também é possível encontrar a correspondência de pontos entre imagens é pela geometria epipolar [4]. A geometria epipolar é um campo da visão computacional que fornece formalismo matemático e técnicas de reconstrução de cenas a partir das correspondências dos pontos de cena da imagem, sem exigir conhecimento dos parâmetros internos ou da posição relativa das câmeras. Ela se baseia na relação entre duas imagens resultantes da projeção de uma cena através de duas câmeras em uma matriz chamada *matriz fundamental* [5], de tamanho 3×3. Esta matriz fundamental $F_{3\times3}$ encapsula a geometria intrínseca de tal forma que o ponto x no mundo é representado como x_1 na primeira visualização e x_2 na segunda [6] de tal forma que esses pontos satisfazem a relação $x_2^T F x_1 = 0$.

Outra forma amplamente utilizada para encontrar a relação entre duas variáveis contínuas é a regressão linear. Aqui, procura-se encontrar um relacionamento estatístico ao invés de uma relação determinística [7] que tende a ser utilizado quando uma variável não pode ser expressa com precisão pela outra. Em métodos de regressão a ideia central é obter uma linha que melhor se ajuste aos dados. A melhor linha de ajuste é aquela para a qual o erro total de previsão (todos os pontos de dados) é o menor possível. Erro é a distância entre o ponto e a linha de regressão. Para o problema de correspondência de pontos e coordenadas, é interessante utilizar métodos de regressão que suportam múltiplas variáveis dependentes para uma mesma entrada [8]. Possíveis soluções para o problema baseado em métodos de regressão seriam *Random Forests* [9] ou Redes Neurais [10].

Uma das principais limitações da geometria epipolar gira em torno da quantidade de pontos mínimos necessários para obter uma boa correspondência entre duas imagens. Em meu conhecimento, estes métodos precisam, obrigatoriamente, de pelo menos sete ou oito pontos em cada imagem para estabecer o relacionamento entre elas[3]. Visto que o desafio divulgou somente três pixels (a, b e c) e suas três respectivas coordenadas geográficas (p_a, p_b e p_c), decidimos descartar esta abordagem e focar em estratégias de homografa e métodos de regressão que suportem múltiplas variáveis dependentes. Embora métodos de regressão não estabeleçam uma quantidade mínima de amostras, é recomendado utilizar uma quantidade considerável para uma boa generalização do método.

[3] `https://web.stanford.edu/class/cs231a/course_notes/03-epipolar-geometry.pdf`

Estratégia 1: Random Forests

Random Forests [11] é um algoritmo de aprendizado supervisionado capaz de executar tarefas de regressão e classificação com o uso de várias árvores de decisão. É uma técnica baseada em Bagging [12] que aprende diversos métodos de regressão (modelos) em paralelo sem que haja qualquer interação entre eles. Cada modelo representa uma característica diferente do conjunto de dados pois subconjuntos dos dados são amostrados de forma aleatória e permite uma melhor compreensão da variância e do viés do conjunto de dados. Ele opera construindo uma infinidade de árvores de decisão no momento do treinamento e retorna a previsão média (regressão) das árvores individuais. Ao final, ele combina a predição realizada por cada modelo e faz a agregação através de votos ou simplesmente a média de todos eles. Uma das principais vantagens do *Random Forests* é o fato de ser bastante preciso, fornecendo estimativas de quais variáveis são importantes para a regressão.

Código Fonte

```
from geopy.distance import geodesic
from scipy.spatial import distance
from sklearn.ensemble import RandomForestRegressor

trainX = [
    [552, 238], # a (red)
    [220, 568], # b (red)
    [660, 613], # c (red)

    # [548, 598], # Usiminas (green)
    # [489, 595], # Usiminas (blue)
    # [611,  18], # BHTec prox (blue)
    # [146, 224], # BHTec descida (blue)
]

trainY = [
    [-19.884362, -43.975112], # a (red)
    [-19.880996, -43.978154], # b (red)
    [-19.875391, -43.977202], # c (red)

    # [-19.877094, -43.978037], # Usiminas (green)
    # [-19.877851, -43.978243], # Usiminas (blue)
    # [-19.884660, -43.974987], # BHTec prox (blue)
    # [-19.884490, -43.975472], # BHTec descida (blue)
]

validX = [[412,598]]
validY = [[-19.877751,-43.978796]]
```

```
probeF = [[510,523]]

regr = RandomForestRegressor(max_depth=20, random_state=3, n_estimators=200)
regr.fit(trainX, trainY)

predY = regr.predict(validX)
euc_error = distance.euclidean(validY, predY)
met_error = geodesic(validY, predY).meters

print(validY, predY, euc_error, met_error)
```

Listing 1: Código-fonte com Random Forests implementado para a obtenção de valores aproximados para as coordenadas.

Com a implementação de *Random Forests* disponível na biblioteca *Scikit Learn*[4], configuramos os seguintes parâmetros empiricamente: `max_depth` define a profundidade máxima da árvore. `n_estimators` determina o número de árvores na floresta. `random_state` é a semente usada pelo gerador de números aleatórios. A saída do código fonte contido em **Listing 1** imprime a coordenada esperada, a coordenada predita, a distância euclidiana das coordenadas, e a distância em metros das coordenadas.

Estratégia 2: Homografia

Uma homografia é uma transformação em perspectiva de um plano, ou seja, uma reprojeção de um plano de uma câmera para uma visualização diferente, sujeita a alterações na translação (posição) e rotação (orientação) da câmera [13]. As transformações de perspectiva mapeiam pontos 3D em planos de imagem 2D usando a matriz de transformação que incorpora as características da câmera: distância focal, centro óptico e os parâmetros extrínsecos (rotação, translação). Para reprojetar um ponto de uma câmera para outra, teríamos que projetar inversamente o ponto x da imagem no sistema de coordenadas do mundo e, em seguida, reprojetar no novo plano de imagem com a câmera traduzida e girada, para obter a homografia.

Código Fonte

```
from geopy.distance import geodesic
from scipy.spatial import distance
from sklearn.ensemble import RandomForestRegressor

import cv2 as cv
import numpy as np
```

[4] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

```
7
8  pixelsPhoto = np.array([
9      [552, 238], # a (red)
10     [220, 568], # b (red)
11     [660, 613], # c (red)
12
13     # [247, 522], # Aglomerado1 (green)
14     # [548, 598], # Usiminas (green)
15     # [489, 595], # Usiminas (blue)
16     # [611,  18], # BHTec prox (blue)
17     # [459, 396], # BHTec med (blue)
18     # [468, 484], # BHTec longe (blue)
19     # [146, 224], # BHTec descida (blue)
20 ])
21
22 coords = np.array([
23     [-19.884362, -43.975112], # a (red)
24     [-19.880996, -43.978154], # b (red)
25     [-19.875391, -43.977202], # c (red)
26
27     # [-19.881952, -43.977276], # Aglomerado1 (green)
28     # [-19.877094, -43.978037], # Usiminas (green)
29     # [-19.877851, -43.978243], # Usiminas (blue)
30     # [-19.884660, -43.974987], # BHTec prox (blue)
31     # [-19.883896, -43.975428], # BHTec med (blue)
32     # [-19.882713, -43.975915], # BHTec longe (blue)
33     # [-19.884490, -43.975472], # BHTec descida (blue)
34 ])
35
36 homo, status = cv.findHomography(pixelsPhoto, coords)
37 Ephoto = [[412, 598, 1]]
38 Ecoord = [[-19.877751, -43.978796]]
39
40 result = np.dot(homo, np.transpose(Ephoto))
41 normed =  (1/result[2]) * result
42
43 euc_ds = np.sqrt((normed[0] - Ecoord[0][0])**2 + (normed[1] - Ecoord[0][1])**2)
44 met_error = geodesic(Ecoord, normed).meters
45 print(Ecoord, np.transpose(normed[0:2]), euc_ds[0], met_error)
```

Listing 2: Código-fonte com Homografia implementado para a obtenção de valores aproximados para as coordenadas.

Com a implementação da Matriz de Homografia disponível na biblioteca *OpenCV*[5], fornecemos os pontos referente aos pixeis da foto e suas respectivas coordenadas geográficas (latitude e longitude) a fim de relacionar a transformação entre dois possíveis planos. Semelhantemente à estratégia anterior, a saída do código fonte contido em **Listing 2** retorna a coordenada esperada, a coordenada predita, a distância euclidiana das coordenadas, e a distância em metros das coordenadas.

[5] `https://docs.opencv.org/master/d9/dab/tutorial_homography.html`

Experimentos

Com preocupações práticas como memória e tempo do processamento de lado, é difícil imaginar nenhuma situação onde ter dados de treinamento mais representativos leve a um resultado pior. Assim, aumentar o tamanho do conjunto de dados tende a reduzir correlações espúrias, diminuir a variância dos dados e melhorar o desempenho de métodos de regressão. A possibilidade de discrepâncias faz parte do que torna importante ter uma grande quantidade de amostras. Em outras palavras, aumentar o tamanho da amostra comumente evita estatísticas enganosas, principalmente se houver alguma irregularidade (*outliers*) em algumas amostras. O tamanho da amostra está diretamente relacionado à margem de erro de uma estatística ou à precisão com que ela pode ser calculada. Por esta razão, **decidimos fazer experimentos adicionais inserindo novos pontos ao desafio**. De fato, realizamos uma avaliação partindo dos três pontos fornecidos inicialmente pelo desafio e depois vamos adicionando gradativamente novos pontos. A Figure 2 exibe os pontos originais em vermelho e os pontos adicionais em azul.

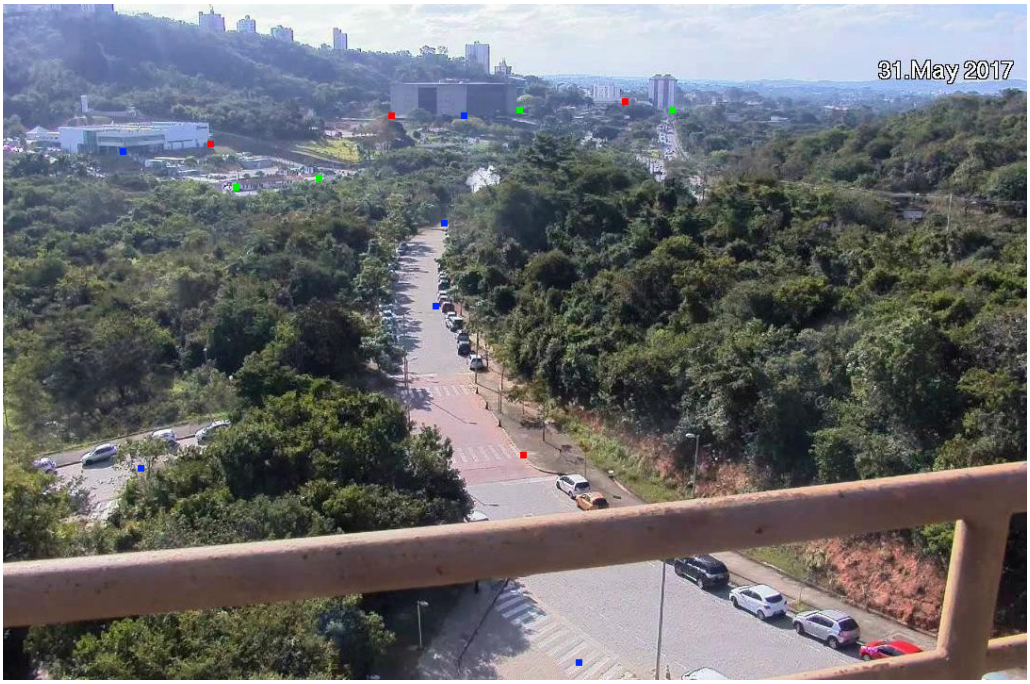

Figura 2: Fotografia tirada do prédio da BHTec, na região da Pampulha em Belo Horizonte, e contendo os pontos demarcados. Em vermelho estão os pontos fornecidos pelos organizadores do desafio e em verde e azul se encontram os pontos adicionais.

Pontos	**Random Forest**		**Homografia**	
Métrica	Euclidiana	Metros	Euclidiana	Metros
3	0.002084	**223.938**	0.015892	1758.382
4	0.000987	**103.495**	0.003088	341.900
5	0.000729	**76.676**	0.000842	89.979
6	0.000709	**74.449**	0.000742	79.881
7	0.000693	72.674	0.000549	**59.022**
8	0.000725	76.039	0.000288	**30.623**
9	0.000777	81.791	0.000177	**18.660**
10	0.000833	87.523	0.000141	**15.009**

Tabela 1: Resultados (erro) obtidos em cada uma das estratégias para o desafio 1

Na Tabela 1 apresentamos os erros obtidos com as duas estratégias. Apresentamos tanto a distância euclidiana quanto a distância em metros de cada uma das coordenadas reais para as coordenatas preditas pelo modelo. Quando é empregado somente três pontos (pixeis a, b e c) e suas respectivas posições geográficas (coordenadas p_a, p_b, p_c) é alcançada uma distância de aproximadamente 223 e 1758 metros empregando *Random Forests* e Homografia, respectivamente. Perceba que ao adicionar mais pontos e coordenadas, a acurácia de ambos os métodos cresce consideravelmente, resultando numa menor taxa de erro. Como mencionado anteriormente, esse comportamento é esperado visto que adicionamos amostras e, consequentemente, aumentamos a capacidade de generalização das duas abordagens. Assim, com pontos adicionais, alcançamos a distância de cerca de 72 e 15 metros empregando *Random Forests* e Homografia, respectivamente.

Coord.	**Desafio 1**		**Desafio 2**	
Pontos	Random Forests	Homografia	Random Forests	Homografia
3	-19.879141,-43.977242	-19.893563,-43.980389	-19.880094,-43.977404	-19.878460,-43.977367
4	-19.877645,-43.977814	-19.880839,-43.978741	-19.879404,-43.977768	-19.883478,-43.975773
5	-19.877561,-43.978091	-19.878240,-43.978110	-19.879410,-43.978081	-19.887209,-43.975063
6	-19.877613,-43.978100	-19.878265,-43.978260	-19.879324,-43.977977	-19.886716,-43.975103
7	-19.877726,-43.978102	-19.878113,-43.978382	-19.879072,-43.978070	-19.885775,-43.975281
8	-19.877798,-43.978071	-19.877894,-43.978545	-19.879075,-43.978091	-19.885516,-43.975222
9	-19.877976,-43.978052	-19.877798,-43.978624	-19.881488,-43.976608	-19.885377,-43.975218
10	-19.877918,-43.977979	-19.877810,-43.978667	-19.881690,-43.976844	-19.885181,-43.975246

Tabela 2: Coordenadas obtidas em cada uma das estratégias para os desafios 1 e 2

Na Tabela 2 apresentamos as coordenadas obtidas em ambos os desafios com as duas estratégas implementadas. Vale a pena ressaltar que para os desafios 1 e 2, não utilizamos dados referentes ao pixel e e sua respectiva coordenada p_e. Por outro lado, adicionamos pontos anotados manualmente levando em conta o ponto de vista da fotografa dada e as coordenadas obtidas pelo Google Maps. Ou seja, esses pontos podem estar sujetos a ruídos humanos e podem gerar mais entropia nos dados do que, de fato, melhorar os resultados. Isto provavelmente justifica o aumento da taxa de erro na estratégia baseada em *Random Forests* quando sete pontos/coordenadas ou mais são adicionados.

Conclusão

Em ambos os desafios propostos foi possível avaliar algumas estratégias difundidas na literatura que visam resolver o problema de correspondência entre pontos. A utilização de métodos estatísticos (*Random Forests*) de regressão e algébricos (Homografia) permitiram encontrar coordenadas geográficas relativamente aproximadas de suas localizações originais. Também constatamos que ao adicionar mais pontos e coordenadas aos métodos, mesmo que manualmente anotadas, há um ganho na acurácia dos mesmos com a diminuição da distância (erro) entre a posição genuína e a predita. Todos os resultados se encontram nas tabelas. Foram utilizadas duas bibliotecas da linguagem Python: `OpenCV 4.0.0` e `Scikit Learn 0.21.3`. Os códigos também estão em anexo no e-mail e podem ser utilizados para conferência dos resultados obtidos.

Bibliografia

[1] Elan Dubrofsky. Homography estimation. *Diplomová práce. Vancouver: Univerzita Britské Kolumbie*, 2009.

[2] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[3] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

[4] Zhengyou Zhang. Determining the epipolar geometry and its uncertainty: A review. *International journal of computer vision*, 27(2):161–195, 1998.

[5] Quang-Tuan Luong, Rachid Deriche, Olivier Faugeras, and Theodore Papadopoulo. On determining the fundamental matrix: Analysis of different methods and experimental results. 1993.

[6] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[7] George AF Seber and Alan J Lee. *Linear regression analysis*, volume 329. John Wiley & Sons, 2012.

[8] Hanen Borchani, Gherardo Varando, Concha Bielza, and Pedro Larrañaga. A survey on multi-output regression. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 5(5):216–233, 2015.

[9] Artur Jordao Lima Correia and William Robson Schwartz. Oblique random forest based on partial least squares applied to pedestrian detection. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 2931–2935. IEEE, 2016.

[10] Jingjing Liu, Shaoting Zhang, Shu Wang, and Dimitris N Metaxas. Multispectral deep neural networks for pedestrian detection. *arXiv preprint arXiv:1611.02644*, 2016.

[11] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[12] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[13] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Deep image homography estimation. *arXiv preprint arXiv:1606.03798*, 2016.