Universidade Estadual de Campinas
Instituto de Computação

Thiago Resek Fabri dos Anjos

Inferring Geographical Location of Images

Inferência de Localização Geográfica de Imagens

CAMPINAS
2022

Thiago Resek Fabri dos Anjos

Inferring Geographical Location of Images

Inferência de Localização Geográfica de Imagens

Dissertação apresentada ao Instituto de Computação da Universidade Estadual de Campinas como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Dissertation presented to the Institute of Computing of the University of Campinas in partial fulfillment of the requirements for the degree of Master in Computer Science.

Supervisor/Orientador: Prof. Dr. Anderson de Rezende Rocha

Este exemplar corresponde à versão final da Dissertação defendida por Thiago Resek Fabri dos Anjos e orientada pelo Prof. Dr. Anderson de Rezende Rocha.

CAMPINAS
2022

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Ana Regina Machado - CRB 8/5467

R311i Resek, Thiago, 1987-
Inferring geographical location of images / Thiago Resek Fabri dos Anjos. – Campinas, SP : [s.n.], 2022.

Orientador: Anderson de Rezende Rocha.
Dissertação (mestrado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Computação forense. 2. Análise forense de imagens digitais. 3. Reconhecimento de padrões. 4. Casamento de padrões (Computação). I. Rocha, Anderson de Rezende, 1980-. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Inferência de localização geográfica de imagens
Palavras-chave em inglês:
Forensic computing
Digital image forensics
Pattern recognition
Pattern matching (Computer science)
Área de concentração: Ciência da Computação
Titulação: Mestre em Ciência da Computação
Banca examinadora:
Anderson de Rezende Rocha [Orientador]
João Paulo Papa
Sandra Eliza Fontes de Avila
Data de defesa: 14-06-2022
Programa de Pós-Graduação: Ciência da Computação

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0000-0002-4247-180X
- Currículo Lattes do autor: http://lattes.cnpq.br/7495527555948883

Universidade Estadual de Campinas
Instituto de Computação

Thiago Resek Fabri dos Anjos

Inferring Geographical Location of Images

Inferência de Localização Geográfica de Imagens

Campinas, 14 de junho de 2022

Dedication

Dedicated to my parents, Mônica and Paulo.

Vivendo, se aprende; mas o que se aprende,
mais, é só a fazer outras maiores perguntas.
(João Guimarães Rosa)

Acknowledgements

First of all, I would like to thank my advisor, Prof. Anderson Rocha, for his constant guidance and support in the development of this work, and for sharing his great passion for Science.

I would also like to thank the Institute of Computing of the University of Campinas (IC/UNICAMP) for providing the means to develop this work, and all the faculty, staff and students for the support and exchange of knowledge. In particular, a special thanks goes to all members of the Recod.ai lab for teaching me so much and for helping me whenever I needed. I am also deeply thankful to Prof. Simone Milani and Sebastiano Verde for the contribution on the early stages of this work.

A big thanks also goes to my co-workers at Motorola, specially Ana Florencio and Guilherme Graciosa, for always encouraging me to pursue my academic goals. Finally, I would like to thank my family and friends for supporting me and keeping me motivated during the period of this work. Specially, I would like to thank: my friend Anderson, for all the great tips, the encouraging talks and the useful comments; my partner Andressa for helping me see things from outside my usual perspective and for being so supportive and understanding. Lastly, I would like to thank my mother, my greatest supporter, who unfortunately could not be here to witness the final stage of this work.

Resumo

Ser capaz de associar uma imagem com a localização geográfica onde ela foi obtida, sem o uso de informação adicional além da própria imagem, é um problema de interesse para várias áreas como jornalismo e ciência forense digital. Atualmente, dada a disponibilidade praticamente global de imagens de satélite com informações de localização, inferir a localização geográfica de uma imagem pode ser reduzido a um problema de casamento de imagens de diferentes pontos de vista (imagens de solo e imagens de satélite). Neste trabalho, apresentamos e estudamos os primeiros passos para o desenvolvimento de um algoritmo não-supervisionado para o problema de casamento de imagens de solo e de satélite. Exploramos e estudamos as relações de adjacências entre pontos de referência visíveis em ambas as imagens, e como tais relações podem se manter mesmo com a mudança de ponto de vista. Apresentamos um algoritmo baseado em comparação de grafos que visa localizar o ponto de vista de uma imagem a nível de solo (um panorama em 360 graus) dentro de uma imagem aérea mais ampla da mesma região. Tal algoritmo recebe como entrada um conjunto de pontos de referência, inicialmente extraídos de forma manual, e funciona por meio do casamento de grafos de pontos de referência de ambos os pontos de vista, de acordo com um modelo de probabilidade especificamente desenvolvido para o problema. Em sequência, foi desenvolvido um processo automático para extração de tais pontos de referência, baseado em segmentação semântica de imagens, processo que incluiu a obtenção de um novo conjunto de dados e o treinamento de uma rede neural específicos para o problema de segmentação de imagens de satélite. Isto permite a criação de um procedimento totalmente automatizado para a localização de imagens baseada em casamento de grafos. Por fim, apresentamos um estudo detalhado do funcionamento do algoritmo proposto, inclusive a adaptação de alguns parâmetros, e também possíveis extensões como, por exemplo, o uso de aprendizagem profunda (*deep learning*) para pós-processamentos nos grafos, mitigando possíveis erros gerados em etapas anteriores.

Abstract

Associating an image to its geographical location, without the use of data besides the image itself, is a significant concern in journalism and digital forensics. Given the availability of geo-tagged satellite imagery for most of the Earth's surface, retrieving the location of a generic picture can be addressed as a cross-view image matching between aerial and ground views. In this work, we propose initial steps toward a fully-unsupervised algorithm for ground to aerial image matching, exploiting and focusing on the view-invariant adjacency relationships between landmarks appearing in both views. We introduce a graph-based strategy that, given a set of (initially manually annotated) landmarks, localizes the viewpoint of a ground-level 360-degree image within a broad aerial view of the same area, by matching the respective landmark graphs according to a specifically designed likelihood model. We further develop a process to automatically extract landmarks, based on semantic segmentation, including the collection of a new dataset and the training of a convolutional neural network for aerial image semantic segmentation. We develop a fully-automated pipeline for localization based on graph matching. We also present an in-depth study of how this algorithm works, in terms of parameter-tuning, and potential extensions, such as applying a Deep Learning approach to post-process the generated graph and mitigate mistakes from previous steps, among others.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

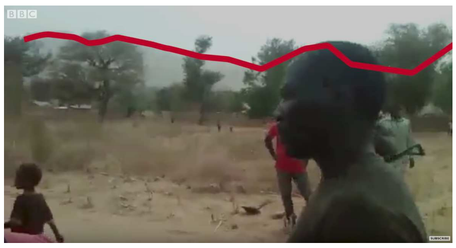
In July 2018, a video depicting the brutal shooting of women and children at the hands of a group of men in military uniforms shocked the world. Tracking down the precise location where the event took place showed to be a daunting task. After a thorough forensic investigation [8], BBC Africa was able to pinpoint a precise area in the Far North Region of Cameroon. Analysts exploited the presence of few macro-scale clues – such as a distinctive mountain range visible in the background – to narrow down the region of interest. Figure 1.1 depicts this procedure and some of the matching elements found by the investigators.

BBC's investigation required a laborious process of manual search through satellite imagery, as well as a prior knowledge of the approximate area to be inspected (reporters received a tip that the event happened around the northern Cameroonian border). Such event highlighted the need for automated tools aiding this process.

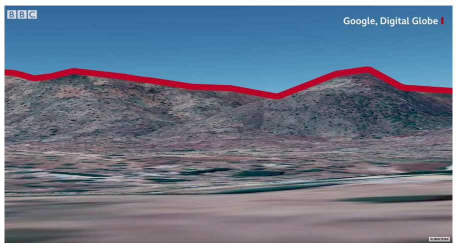
Retrieving the geographic location from where a picture was taken, with no GPS-like information or metadata available, is a challenging task even for humans. It has potential applications in areas such as forensics (as the example provided above), fake-news analysis (it could be used to verify if a given image was taken in a different place than where it claims), autonomous cars (to help localize a self-driving car if GPS connection is poor or unavailable), among others. Consistent effort has been addressed to the problem during the last decade, starting from reducing the number of potential candidate locations, by calculating probability distributions over the Earth's surface [28]. As seen for BBC's case, macro-scale features such as mountainous terrains and elevation were also investigated [4]. A more accurate localization was later achieved through ground-to-ground matching with geo-tagged images, using standard keypoint-matching techniques [82] and higher-level city descriptors [87]. These approaches are limited by the availability of geo-referenced images, which are typically abundant for urban or touristic areas only. Such limitations and the broad availability of satellite imagery caused the research focus to shift from ground-to-ground to ground-to-aerial (or *cross-view*) image matching.

Ground-to-aerial scenarios are characterized by a strong perspective change, making unfeasible to perform keypoint matching via traditional feature descriptors (e.g., SIFT [42]). Cross-view matching was first tackled in urban scenarios, using oblique bird's-eye view (BEV) images to extract and match building facades [7]. The approach was further integrated with pose estimation and 3D modeling [55] and extended to non-urban areas [39] by introducing hand-crafted features [19].

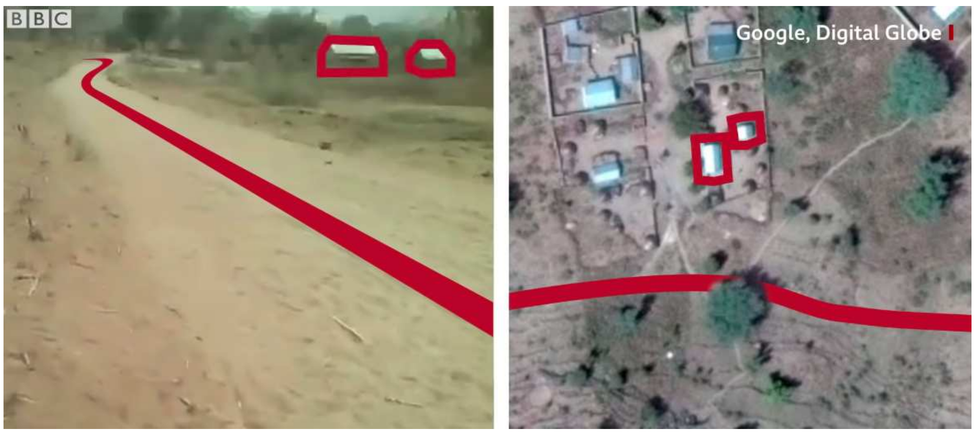
More recently, the success of deep learning (DL) in computer vision has influenced the field of cross-view matching. A widely adopted approach employs a Siamese network architecture to learn a common representation for ground and aerial images from pixel data [40, 78, 30, 41, 65, 69, 58, 76, 88, 59, 60]. Generative adversarial networks (GANs) were employed to synthesize ground-level views from overhead imagery and bridge the gap between the two domains [23, 50, 70]. In general, state-of-the-art DL solutions frame the problem as an image-to-image matching task (i.e., verifying the correspondence between the ground image and the associated aerial one) and are designed in a supervised formulation, requiring annotated data as training examples.



(a) A frame from the original video showing a distinctive mountain range, highlighted in red.



(b) A matching mountain range in north Cameroon, found through Google Earth.



(c) Matching elements between the ground images from the video (to the left) and the satellite images (right).

Figure 1.1: BBC was able to pinpoint the location where the video took place by finding a matching mountain range in Google Earth. Frames extracted from the video [8].

Differently from current data-driven efforts, here we propose an unsupervised approach that leverages the view-independent adjacency properties of visible landmarks to create comparable graph structures. We observed how the proximity relationships between nearby objects or *landmarks* (e.g., the two buildings in Figure 1.1), are maintained in both images. Our hypothesis is that, despite actual distances being disrupted by the perspective change, the mutual adjacency relationships among nearby objects, or landmarks, are preserved across the two views. Although the proximity of two buildings is surely not sufficient to uniquely localize a ground image, our claim is that the localization becomes progressively more accurate as the proximity graph grows richer (i.e., by including buildings, roads, vegetation and so on). The main research question we are interested in is if the adjacency properties of visible landmarks, modelled as graph structures, can be compared and used to leverage information about the geographic localization of the input images.

In this work, we outline the first steps in the direction of a comprehensive ground-to-aerial image matching system, by investigating the viewpoint localization of a 360° ground image within a broader aerial view of the same area. We describe a pipeline, containing methods designed to extract salient points from an image, to generate landmark graphs from such salient points, to extract candidate locations from the aerial image and to calculate a probability distribution over them, according to a specifically adopted likelihood model.

This document is further structured as follows: in Chapter 2, we analyze some of the related work found in the literature and the current state-of-the-art; in Chapter 3, we describe our proposed method for localization through graph matching, including how such graphs are generated and compared; in Chapter 4, we discuss semantic segmentation of images and how it can be employed to achieve automatic extraction of landmark points; in Chapter 5, we present and discuss the results of several experiments performed throughout this work; lastly, in Chapter 6 we provide the conclusions of our work.

Chapter 2

Related Work

2.1 Directly Related Work

The problem of estimating the geographical location of a ground query image has been approached in the literature using many different strategies, but most of them have a common point: they try to match the input image, of unknown location, against a database of images of known geographical location. If a match is found, the location of the query image can then be said to be the same location of the matching image. Existing research can generally be grouped by the type of images used for matching (ground or satellite images) and by the type of features used to describe the images (hand-crafted features or learnable features).

Ground-to-ground image matching was approached by Hays and Efros [28] in 2008 in a very influential research: they employ a set of hand-crafted features to match ground images to a large database of other ground images. Other similar works, especially in the early 2010 decade, employ local features (such as SIFT [42], *Scale Invariant Feature Transform*, in [82]) and global features (for instance, [87]). The main issue with ground-to-ground image matching is that large databases of ground images with known GPS locations, usually extracted from Google Street View or Flicker, are mostly restricted to large urban or touristic areas, making this approach less likely to work well on rural areas, for instance.

One way to overcome this limitation is to make use of satellite images, which are available practically worldwide with tools like Google Maps. This problem is known in the literature as *cross-view image matching*. Bansal *et al.* [7] was one of the first works to employ this approach: bird's eye-view images are used to extract building facades, which are then warped and described using hand-craft features and matched against the facades visible in ground images. This method has two serious drawbacks: it only works in urban areas, and it requires that bird's eye-view images are also available (since facades are not visible in a satellite image). There are other methods in the literature that propose solutions specific to certain terrain types: Baatz *et al.* [4] focus on mountainous terrains, trying to extract mountain contours from the images and match them against digital elevation models; Bansal *et al.* [6] try to extract corners and roof-line edges of buildings in the ground view and match them to elevation maps. Lin *et al.* [39] try to overcome these limitations by describing the whole ground and satellite images using a set of hand-craft features such as HOG [19] (*Histograms of Oriented Gradients*) and color histograms. Besides ground and aerial imagery, their method also uses land cover attributes (an aerial view of the area classifying each pixel in land cover types such as water, grassland, human use) as input. While cross-view matching solves the problem of data availability, it introduces another problem: since the viewpoints are extremely different, matching using hand-crafted features does not perform as well as it does in the ground-to-ground scenario.

With the advance of Deep Learning and its successful application in many computer vision problems, most of the recent work in the literature focuses on describing images using deep, learnable features. In one of the first published works on this line, Lin *et al.* [40] propose a solution that would be the inspiration for most of the subsequent work in this area: using a Siamese network with CNNs (convolutional neural networks) in each branch, as illustrated in Figure 2.1, to learn a common embedding to describe ground and aerial images (in their case, they use bird's eye-view imagery instead of satellite). This is shown to perform better than hand-crafted features and other general-purpose networks (such as AlexNet [35] trained on ImageNet [21]). Using a similar approach, Workman *et al.* [78] show that, by training different networks for different scales of satellite images, the matching can be performed at world-wide level. They also introduce CVUSA, a dataset containing over 1.5 million geo-tagged image pairs, extracted randomly from locations within the United States. Vo and Hays [74] further advance on this topic by investigating rotation invariance, new architectures, such as a Triplet Network, and a new loss function, the DBL (distance-based logistic), which is shown to increase performance. They also introduce GTCrossView, which contains approximately 900,000 pairs of images collected in 11 cities of the United States. Though widely used in early works in the area, this dataset is not common in most recent works, mainly because the images are of lower resolution than other datasets, as CVUSA, for example. Zhai *et al.* [84] study a slightly different problem, segmentation of aerial images, but their approach can be extended to cross-view matching: they extract features of the aerial image using a CNN and apply a (learned) transformation to map these features onto the ground-level perspective. They also build a new version of the CVUSA dataset, initially proposed by [78], using camera's extrinsic parameters to warp the ground-view panoramas, generating aligned pairs of images (35,532 pairs for training and 8,884 for testing)[1]. Hu *et al.* [30] present CVM-Net (*Cross-View Matching Network*), which provides significant improvements by adding a NetVLAD layer [2] to the Siamese network. This converts local image features extracted by the convolutional layers into global image descriptors and is shown to improve matching accuracy. The idea is to obtain global representations that are *independent* of the locations of their local features. Even though the results are good, it is not clear that this approach is optimal, as the location of features could be helpful in finding patterns between ground and satellite views. They also propose a new loss function, weighted soft-margin ranking loss, which performs better than the more common triplet loss and also speeds up training convergence. This loss is used in several following works. Their work is extended and

[1]This version of CVUSA is the most widely used in recent research [30, 41, 12, 66, 58], but the naming might be misleading since both versions are usually referred to as simply CVUSA. In this work, we will use the term CVUSA to refer to the new version proposed by [84], unless stated otherwise.

improved upon by Hu and Lee [31]. Liu and Li [41] take inspiration from the way humans usually localize themselves using a map (by first aligning the direction of the map with the direction they are looking at) and propose a Siamese network that encodes the orientation of each pixel in the images. This provides a boost in performance compared to previous methods and differs from other works because problem specific information is embedded in the solution. They also create a widely used dataset, CVACT, which is 10 times bigger than CVUSA, and contains higher resolution images. Cai *et al.* [12] propose an in-batch re-weighting triplet loss to emphasize the effect of hard exemplars (each triplet inside a batch is given a weight depending on the difficulty – samples that are too easy or too hard have low weights, and thus lower importance). Sun *et al.* [66] extend the most common architecture (a Siamese network with CNNs in each branch) by adding capsule layers [52] that enhance the representation power of the features generated. The authors claim that the capsule network is able of modelling spatial relationships of extracted features, and thus is particularly indicated for this kind of problem. Shi *et al.* [58] try to explicitly consider the geometric differences between ground and aerial views. The authors note that applying a polar coordinate transformation on the aerial images brings them closer to the ground images, as illustrated in Figure 2.2. Their method explicitly accounts for geometric differences in two ways: first, by applying the polar transformation to all aerial images; second, by adding a spatial-attention mechanism to the network so that corresponding deep features are brought closer in the embedding space. Shi *et al.* [60] explore spatial layout of local features by proposing a *Cross-View Feature Transport* layer that transports features from one domain to the other, preserving the spatial layout of local features and leading to more meaningful feature comparison. Shi *et al.* [59] achieve very good results by also taking the orientation of the images into account: they apply a polar transformation to the aerial image, extract features from both the polar-transformed aerial and ground view and then compute the correlation between these features, which they claim provides a more accurate measure of similarity. Zhu *et al.* [88] also study how orientation can affect the result and attempt to develop a method to estimate orientation together with the localization itself. They also point out that the comparison between some methods in the literature may not be fair, since some assume that the ground images and aerial images are always aligned [50, 58, 60][2], while some do not [30, 12, 74]. Wang *et al.* [76] claim that contextual information in neighboring areas can enrich features descriptors and improve matching. They propose a Local Pattern Network (LPN) that takes advantage of contextual information by extracting feature vectors from the center of the image and from surrounding areas. Rodrigues and Tani [51] point out an important factor: since we cannot guarantee that aerial and ground images were taken at the same time, the landmarks can change between one view and the other (for instance, a new building on the ground view might not be present in the aerial view). They propose a data augmentation method that uses semantic segmentation of ground images to create scene variations, e.g., removing a building or a road, which makes the network learn

[2]The images in CVUSA, the most used dataset, are always aligned, i.e., all aerial views have north as the up direction, and all ground panoramas have the north direction in the center of the image. Hence, unless explicitly taken into consideration, by performing random rotations, for instance, the methods will assume that the images are aligned.

correspondences that are robust to such temporal variation. Yang *et al.* [80] achieve state of the art results by proposing a novel model that uses Transformers [71] instead of the typical CNNs. They claim that the self-attention properties of a Transformer can better model global dependencies and make it easier for the network to learn the geometric correspondences. The main drawbacks of this method are related to its performance: the proposed network has considerably more parameters, making it more expensive in terms of memory and data required for training, and also slower in inference time (which can be a serious problem, particularly if real-time processing is desired).

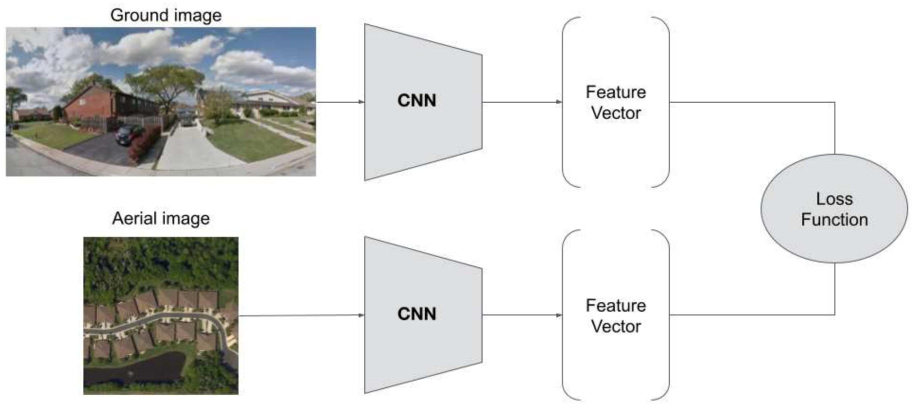
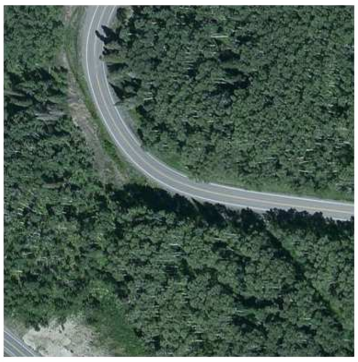


Figure 2.1: One of the most common approaches in the literature for the cross-view image matching problem: a Siamese network with a CNN in each branch.

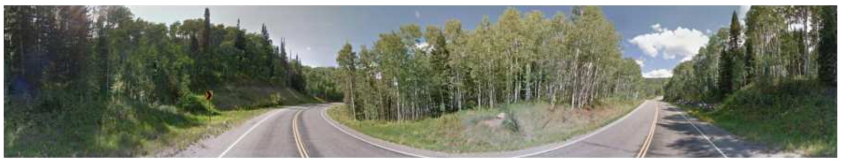
We can see that some of the works mentioned in the previous paragraphs, especially the initial ones, treat the images as “black boxes”, trying to extract information from them without using any kind of knowledge or information specific to the problem. Moreover, they analyze and describe the images as a whole, disregarding important information about the local features and their relationships among different viewpoints. This works to an extent, but more recent works such as [41], [88], [76], indicate that using additional information (such as orientation and the layout of local features, for instance) is an effective way to tackle this problem and can improve the performance of the methods.

Following recent advances on another research topic, conditional GANs, Regmi and Shah [50] train a GAN to synthesize aerial views given a ground image. They then train a Siamese-like network with three branches, one for the ground view, one for the aerial and one for the synthesized aerial view, claiming that the obtained representation is more robust compared to using just features of the original images. Toker *et al.* [70] use a similar approach, but in the other direction: their GAN synthesizes ground-level image from a polar-warped aerial image. Several other works in the literature do not tackle the matching problem itself, but propose GANs for *cross-view image synthesis*, some of which work in both directions [68, 49, 56, 24] (i.e., can generate aerial views from a ground image and vice-versa), while others only work in one direction [22, 57, 33, 23, 43], in most cases aerial-to-ground.

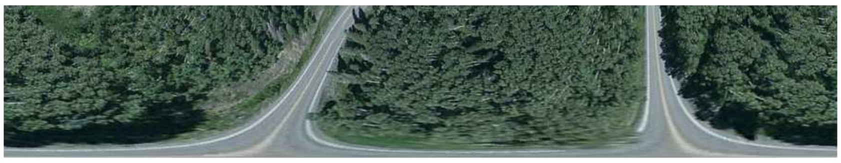
Some other works have been published with different approaches or applications that



(a) Aerial image taken from CVUSA dataset.



(b) Corresponding ground-level image.



(c) Aerial image shown in Figure 2.2(a) after polar coordinates transformation.

Figure 2.2: Applying a polar coordinate transformation to aerial images is a technique that can reduce the domain gap between ground and aerial views. It is applied in several works [58, 59, 70]. Notice that the alignment is obviously not perfect, but the main components (roads, for instance) are roughly aligned.

are not the focus of our work but are still worth mentioning. Weyand *et al.* [77] propose the localization problem as a classification one, instead of the usual image matching approach: they divide the world into cells and train a CNN that outputs, for a given query image, the likelihood that the image was taken in each cell. Their method, however, is limited to ground-to-ground image matching. Samano *et al.* [54] try to match ground-level panoramas to map tiles directly. They claim that the map representation is not sufficiently discriminative to allow for localization of a single image, but the results can be concatenated in a given route, using particle filtering approaches, to allow for localization of a moving vehicle, for instance. Chen *et al.* [16] focus on matching drone images (instead of ground images) to satellite images. The work by Wang *et al.* [76] that was mentioned previously can also be applicable to this scenario. Zhu *et al.* [89] propose a more realistic version of the problem: they claim that most of existing work assumes that there is always a reference image centered in the exact location of the input query image, and that the matching is always one-to-one. The authors introduce a more realistic definition of the problem in which query images and reference images are not always perfectly aligned, and there might be multiple reference images covering the same query location. A new dataset, VIGOR, built specifically for this approach, is also presented.

Table 2.1 contains a brief summary of the main research works cited throughout this section, listing the main characteristics and limitations of each work, and the datasets that they employ.

Work	**Characteristics and limitations**	**Datasets used**
Bansal *et al.* [7]	Uses bird's eye-view images to extract and match building facades with hand-craft features. Only works in urban areas.	Not published.
Baatz *et al.* [4]	Focuses on mountainous terrains, extracting contours and matching against digital elevation models. Input data has limited availability.	Not published.
Lin *et al.* [39]	Describes ground and satellite images using a set of hand-craft features. Also use land cover attributes as input, which have limited availability.	Not published.
Lin *et al.* [40]	First proposal of a Siamese network with CNNs in each branch.	Not published.
Workman *et al.* [78]	Uses different networks for different scales of aerial images.	Introduces CVUSA (original version).
Vo and Hays [74]	Investigates rotation invariance, new architectures and a new loss function, the DBL (distance-based logistic).	Introduces GTCrossView.
Zhai *et al.* [84]	Learns a transformation to map features extracted from aerial images to ground images.	Introduces a new, widely used version of CVUSA.
Hu *et al.* [30], Hu and Lee [31]	Introduces CVM-Net, which adds a NetVLAD layer to the Siamese network, converting local image features into global image descriptors. Also proposes the weighted soft-margin ranking loss.	GTCrossView, CVUSA.
Liu and Li [41]	Proposes a network that encodes orientation in the images, embedding problem specific information in the solution.	CVUSA. Introduces CVACT.
Cai *et al.* [12]	Proposes an in-batch re-weighting triplet loss to emphasize the effect of hard exemplars.	GTCrossView, CVUSA.
Sun *et al.* [66]	Adds capsule layers [52] that enhance the representation power of the features.	GTCrossView, CVUSA.
Shi *et al.* [58]	Explicitly considers geometric differences between views by adding a spatial-attention mechanism to the network.	CVUSA, CVACT.

Work	**Characteristics and limitations**	**Datasets used**
Shi *et al.* [60]	Proposes a layer specifically designed to transport features from one domain to the other.	CVUSA, CVACT.
Shi *et al.* [59]	Applies a polar transformation to the aerial image, extracts features from both the polar-transformed aerial and ground view and then computes the correlation between these features, which provides a more accurate measure of similarity.	CVUSA, CVACT.
Zhu *et al.* [88]	Tries to estimate orientation together with the localization itself.	GTCrossView, CVUSA.
Wang *et al.* [76]	Proposes a Local Pattern Network (LPN) that takes advantage of contextual information by extracting feature vectors from the center of the image and from surrounding areas.	CVUSA, CVACT, University-1652 (dataset for drone localization).
Rodrigues and Tani [51]	Proposes a network that can learn correspondences that are robust to temporal variations.	CVUSA, CVACT.
Yang *et al.* [80]	Proposes a model that uses Transformers [71] instead of CNNs, which can better model global dependencies and makes it easier for the network to learn the geometric correspondences.	CVUSA, CVACT.
Regmi and Shah [50]	Trains a GAN to synthesize aerial views given a ground image; train a Siamese-like network with three branches (ground, aerial and synthesized aerial).	GTCrossView, CVUSA.
Toker *et al.* [70]	Similar approach to Regmi and Shah [50], but their GAN synthesizes ground-level image from a polar-warped aerial image.	CVUSA, CVACT.
Zhu *et al.* [89]	Assumes that query images and reference images are not always perfectly aligned, and there might be multiple reference images covering the same query location.	Introduces VIGOR.

Table 2.1: A brief summary of published works that study the cross-view image matching problem

2.2 Additional Topics

In this section, we briefly discuss works in topics that are not directly related to our problem of interest but are employed in our solution and thus worth mentioning.

The first topic is semantic segmentation, which can be defined as the problem of classifying every pixel of an image as belonging to a particular class with semantic meaning (not to be confused with *instance segmentation*, where each different instance of a class must be labelled uniquely).

Semantic segmentation of ground-level images is a widely studied topic, especially because of its application in areas such as self-driving cars. Badrinarayanan *et al.* [5] propose SegNet, a deep fully convolutional neural network with an encoder-decoder architecture, followed by a pixel-wise classification layer at the end. Chen *et al.* [13] claim that the sequence of max-pooling and downsampling layers common in other approaches, such as [5], results in feature maps with reduced spatial resolution, and propose a network that uses upsamling filters (*atrous convolution*) to achieve denser feature maps. Zhang *et al.* [86] introduce a multimodal approach that takes as input several image modalities, such as near-infrared and depth images besides RGB images. They propose an architecture to extract and fuse information from all different modalities. Chen *et al.* [15] perform a detailed study on how common network architectures, such as VGG [61] and ResNet[29], affect segmentation in terms of accuracy, speed and storage size.

Semantic segmentation of aerial images is less explored in the literature. Zhai *et al.* [84] try to learn a transformation to convert the segmentation of a ground view of the same area into the aerial segmentation. Their method is particularly interesting because it does not require annotated data for aerial images, but is limited to three classes only (vegetation, roads and buildings). Costea *et al.* [18] propose an approach based on an ensemble of CNNs to detect roads only. Li *et al.* [38] focus on detecting buildings and roads with CNNs to extract a map representation from an image. Kaiser *et al.* [34] employ another neural network as base of the architecture, but their work is also limited to roads and buildings. Kuo *et al.* [36] focus on identifying land types and usages only, such as urban, forest, agricultural, etc. Popescu and Ichim [48] propose a solution based on handcrafted features that is aimed towards disaster-monitoring, detecting floods and roads. Zhang *et al.* [85] propose a dual-path network, containing a spatial branch that encodes local and global semantic information, and an edge path, that learns to detect semantic borders and improves the performance on the boundaries between classes. Their proposed network segments aerial images in 6 classes: roads, buildings, low vegetation, trees, cars, and background). Marmanis *et al.* [44] also propose the usage of an ensemble of CNNs to improve segmentation accuracy. It is also worth mentioning that there is a limited availability of datasets for this problem: ISRPS [1] is limited to 6 classes and only 71 images; Yuan *et al.* [81] provide data that is focused on finding borders between semantically different objects, but the objects themselves are not classified; DeepGlobe [20], introduced by Demir *et al.*, focuses on building and road extraction and land cover classification; Humans in The Loop [32] published a dataset, containing 72 images of Dubai segmented in five classes, that could be applicable to our problem, but the images are limited in number and in variability, since they were all collected in a single city; Chen *et al.* [14] introduce VALID, a dataset with 6690 high-resolution images annotated with segmentation data, depth information, bounding boxes, etc. This dataset is built from synthetic data, i.e., virtually generated scenes.

Another topic of interest for our work is graph matching and the use of graphs for visual place recognition. The general problem of exact graph matching is NP-hard [11] and finding exact node and edge matches is a combinatorial problem that can grow very quickly. To simplify this problem, inexact graph matching approaches have been proposed, such as edit-distance, detailed by Bunke and Riesen [10], which attempts to find the minimum cost based on edit operations (insertion, deletion, etc.) between two graphs. Other examples are spectral methods [75] and graph kernels [73]. More recently, Zanfir and Sminchisescu [83] proposed a novel approach based on deep learning, including the mathematical formulation of layers specifically designed for this problem. Stumm *et al.* [63, 64] introduce the concept of *covisibility graphs*, structures proposed for the field of robotics, more specifically in autonomous navigation systems: maps of feature points are used to provide a synthetic description of the scene, allowing the robot to recognize a location that it already visited. Two feature points, or landmarks, are connected by an edge in the graph if they appear co-visible within the field-of-view of the robot. This structure helps retain important relation information about the landmarks. The authors also propose an efficient method for graph comparison in this scenario. Their work is based on SLAM (Simultaneous Localization And Mapping) techniques, which is also a relevant field for our problem, with the possibility that other solutions could be adapted from one field to the other. SLAM is widely studied in the literature, and surveys such as [3] and [67] provide introductions to this topic.

Chapter 3

Localization Through Graph Matching

As mentioned in Chapter 2, most of the current work in geographical localization is data-driven. We decided to propose a different, unsupervised approach that attempts to leverage the view-independent adjacency properties of *landmarks* in aerial and ground images (we define as landmarks any significant object or structure in the scene, such as a building, a tree, a road). Our hypothesis is that, although distances and perspectives are clearly disrupted when changing from a ground to an aerial point of view, the general adjacency relationships among nearby objects, or *landmarks*, is preserved. Our proposal is to generate graphs for both ground and aerial views, modelling the spatial distribution of such landmarks and their relationships. If such graphs indeed maintain the adjacency relationships of landmarks, then we can try to match them and to extract insights from their similarity.[1]

3.1 View-point Localization and Landmark Graphs

To study how landmark graphs work and if they can be used in localization problems, we investigated the *view-point localization* problem, that consists in localizing a 360° ground image within a broader aerial view of the same area. This is a simplification of the original problem (geographical localization), since it does not involve matching one ground image to one among several aerial images. In this case, we start with a pair of matching ground and aerial images and our goal is to answer the following question: what place in the aerial image corresponds to the viewpoint of the ground image?

Our proposed approach for this problem is based on building and matching *landmark graphs*, a relational graph structure that models the adjacency and spatial relationships of key objects in the scene. To develop a mathematical model for landmark graphs, we took inspiration from the field of robotics, mainly from the concept of *covisibility graphs*[63, 64] described in Section 2.2. With some adaptations, this concept can be extended and applied to landmark graphs, as we will show in the next sections.

Using landmark graphs to study the localization problem involves, broadly speaking, two separate steps: extracting the landmarks and matching the graph itself. In this chapter, we will focus on the latter, assuming that the landmarks are already known. We

[1]Some parts of this chapter were directly adapted from previously published work [72].

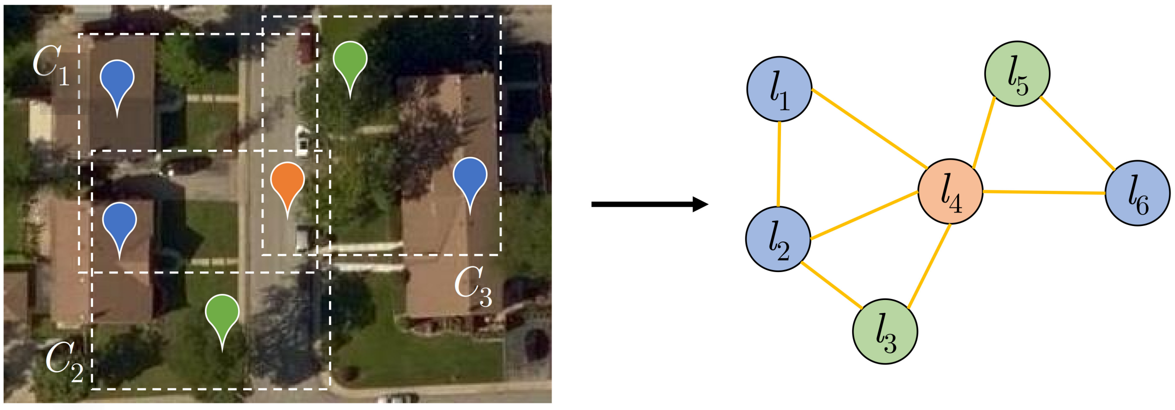


Figure 3.1: Landmark graph generation from a set of landmark points of three classes: *building* (blue), *road* (orange) and *tree* (green). Each subset of points co-observed by the moving window produces a connected clique in the graph.

will defer the study of how landmark points can be extracted to Chapter 4.

Our proposed method starts with a 360° ground image and an aerial view of the same area, and the corresponding lists of their landmark coordinates and classes. In Sections 3.1.1 through 3.1.4 we will present in detail the steps of our proposed pipeline.

3.1.1 Graph Generation

Let (l_k, w_k), $k = 1, \dots, K$, be a list of landmark points l_k on a ground or aerial image, each associated to a class w_k from a finite dictionary $\mathcal{W}$ (classes could be, for instance, buildings, roads, vegetation, water, and so on). A *landmark graph* is an undirected graph $G = (V, E)$, where $V = \{l_k\}$, and there exists an edge between two nodes if the related landmarks are co-visible, i.e., visible together within a window of predefined size, called *covisibility window*.

A landmark graph is obtained by making observations on the image while moving the covisibility window with a one-pixel stride. For each i-th observation, all landmarks l_k lying within the current window are stored as a *clique*, C_i, represented by a K-element binary vector. While cliques are extracted, a matrix C is continuously updated by adding each vector C_i as a new column. Duplicate cliques are discarded. Once the observations are concluded, the adjacency matrix A of the overall landmark graph can be calculated from the clique matrix C as

$$A = H(C \cdot C^\top), \tag{3.1}$$

where H denotes the element-wise Heaviside step function.

In the example of Figure 3.1, we have three observations, producing cliques $C_1 = \{l_1, l_2, l_4\}$, $C_2 = \{l_2, l_3, l_4\}$, $C_3 = \{l_4, l_5, l_6\}$ and resulting in the following clique matrix:

$$C^\top = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}.$$

Note that in the case of 360° images, a special treatment is required: landmarks are marked as *covisible* also in the case they appear on opposite ends of the picture. In fact, the picture must be considered as closed on itself, applied on the inner surface of a cylinder.

3.1.2 Candidate Locations

Up to now, we have treated ground and aerial images in the exact same way. As we are interested in localizing the viewpoint in the aerial image that corresponds to the ground view, our next step is to divide the aerial graph into several subgraphs, each one representing a possible location. We will then look for, among these candidate subgraphs, the one that better matches the graph of the ground image.

First, a subset of *relevant cliques* is extracted from the columns of C. A clique C_i is considered relevant if all the classes visible in the query ground image appear on it at least once. This task can be made computationally more efficient by using an inverted index representation, as described in [62].

Given the set of relevant cliques, these are then *extended* according to a covisibility parameter p representing the ratio of landmarks that are re-observed between pairs of neighboring cliques [63]. The rationale for this parameter is to collect within the same location subsets of landmarks that are sufficiently close to one another and thus likely to be seen together in the ground image. A clique C_i is extended with landmarks of C_j if the following condition is true:

$$\frac{\|C_i \odot C_j\|_0}{\|C_i\|_0} > p, \tag{3.2}$$

where $\odot$ denotes the element-wise binary multiplication and $\|\cdot\|_0$ is the L0 norm, i.e., the number of non-zero elements. The left part of (3.2) thus denotes the number of common landmarks in C_i and C_j over the total number of landmarks in C_i.

In the example of Figure 3.1, by setting $p = 0.5$, C_1 is extended with C_2 (as they share 2/3 of their landmarks), but C_3 remains separated (since the shared ratio is 1/3).

The extended cliques are collected in the set of *candidate locations*, $\mathcal{L}_i$, to be compared to the query ground image, $\mathcal{Z}$.

3.1.3 Class Adjacency Matrices

The last step before location retrieval consists in deriving a suitable feature representation for comparing $\mathcal{Z}$ and $\mathcal{L}_i$. Since individual landmarks cannot be directly matched across the two views, a node-wise comparison of landmark graphs is not viable. Moreover, graph sizes and shapes are different, in general, as the number of visible landmarks may vary even between the ground observation and the corresponding aerial location. Nevertheless, we have yet to take into account the class information associated with each landmark. An effective representation to harness class information for matching heterogeneous graphs is the *class adjacency matrix* [64].

Given the adjacency matrix $A^{\mathcal{L}}$ from the subgraph of a generic location $\mathcal{L}$, the associated class adjacency matrix $W^{\mathcal{L}}$ is a $|\mathcal{W}| \times |\mathcal{W}|$ matrix such that $W^{\mathcal{L}}_{u,v}$ is equal to the number of edges connecting a u-class to a v-class node, normalized by the total number of edges (as described in pseudocode in Algorithm 1). Such matrix is symmetric and the upper/lower triangle (including the diagonal) sums to 1. The algorithm works in the following manner: in the main nested loop (lines 5 to 13),the lower triangle of the landmark adjacency matrix A is swept, and if landmarks i and j are connected (i.e., if $A_{i,j} > 0$), the value of W (initialized with all zeroes) at the corresponding position is increased. Since the graph is undirected (i.e., A is symmetrical, $A_{i,j} = A_{j,i}, \forall i, j$), we can sweep only the bottom triangle of the matrix, instead of all the elements. The result will contain, in element $W_{r,c}$ the count of occurrences of edges connecting a landmark belonging to class r to a landmark belonging to class c. Again, due to the property that A is symmetrical, only the lower triangle of W is initially calculated. Lastly, on line 14, operations are performed to make sure that W is also symmetrical and that the elements sum to one.

Algorithm 1 Class adjacency matrix

```
Require:
 1: $A$, landmark adjacency matrix ($K \times K$);
 2: $w_k$, landmark classes, $k = 1, \dots, K$;
 3: $\mathcal{W}$, class-dictionary.
Ensure:
 4: $W$, class adjacency matrix ($|\mathcal{W}| \times |\mathcal{W}|$).

 5: for $i$ in range($K$) do
 6:     for $j$ in range($i - 1$) do
 7:         if $A_{i,j} > 0$ then
 8:             $r \leftarrow \max(w_i, w_j)$
 9:             $c \leftarrow \min(w_i, w_j)$
10:             $W_{r,c} \leftarrow W_{r,c} + 1$
11:         end if
12:     end for
13: end for
14: $W \leftarrow \left(W + W^\top - \mathrm{diag}(W)\right) / \mathrm{sum}(W)$
```

In the example of Figure 3.1, there are two edges connecting blue and green nodes out of a total of eight edges. Therefore, $W_{\text{blue,green}} = 2/8 = 0.25$.

Class adjacency matrices are calculated for each candidate location $\mathcal{L}_i$ and the ground query $\mathcal{Z}$.

3.1.4 Location Likelihood

The likelihood model $P(\mathcal{Z}|\mathcal{L})$ for a ground query $\mathcal{Z}$ given an aerial location $\mathcal{L}$ is represented by the normalized cross-correlation between class adjacency matrices, as in (3.3),

$$P(\mathcal{Z}|\mathcal{L}) = \frac{\sum_{u,v} W^{\mathcal{Z}}_{u,v} \cdot W^{\mathcal{L}}_{u,v}}{\sqrt{\sum_{u,v} \left(W^{\mathcal{Z}}_{u,v}\right)^2 \cdot \sum_{u,v} \left(W^{\mathcal{L}}_{u,v}\right)^2}}, \tag{3.3}$$

where $W^{\mathcal{Z}}_{u,v}$ and $W^{\mathcal{L}}_{u,v}$ are the class adjacency values between classes u and v, in $\mathcal{Z}$ and $\mathcal{L}$, respectively.

The posterior probability of a candidate aerial location $\mathcal{L}_i$ given the ground query $\mathcal{Z}$ is given by the Bayes' rule,

$$P(\mathcal{L}_i|\mathcal{Z}) = \frac{P(\mathcal{Z}|\mathcal{L}_i)P(\mathcal{L}_i)}{P(\mathcal{Z}|\mathcal{L}_i)P(\mathcal{L}_i) + P(\mathcal{Z}|\neg\mathcal{L}_i)P(\neg\mathcal{L}_i)}, \tag{3.4}$$

whereby the prior is set to $P(\mathcal{L}_i) = 1/N$, being N the total number of candidate locations.

The likelihood of the query being matched to another location is calculated as the average likelihood over all other candidates:

$$P(\mathcal{Z}|\neg\mathcal{L}_i) = \sum_{j \neq i} \frac{P(\mathcal{Z}|\mathcal{L}_j)}{N-1} \tag{3.5}$$

.

Finally, the best candidate for the current query is the one satisfying the maximum a posteriori (MAP) criterion,

$$\mathcal{L}_{\text{MAP}} = \arg\max_{\mathcal{L}_i} P(\mathcal{L}_i|\mathcal{Z}) \tag{3.6}$$

Note that with the choice of a uniform prior $P(\mathcal{L}_i)$, the estimation criterion becomes a maximum likelihood (ML), so maximizing (3.3) or (3.4) is equivalent. However, we report the description of the complete MAP model either way, in view of a possible extension to non-trivial prior distributions.

3.1.5 Hierarchical Search Refinement

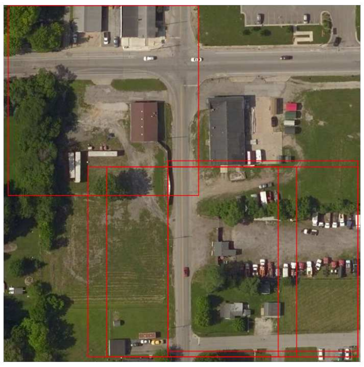
A possible extension to the algorithm described would be to apply it in a hierarchical way: as the size of the covisibility window directly affects the size of the searched area, it might be plausible to start the search on a large area and then refine the search multiple times using a smaller window in each iteration. We then modified our algorithm to work in a recursive manner: given a window of size W, it will find the candidate locations and return the likelihoods of each of them. The search will then be repeated inside of each window, using a smaller window w, and so on. This procedure is illustrated in Figure 3.2, with window sizes of 400, 200 and 100 pixels. To compute the likelihood of each final candidate locations, we must properly consider and propagate the likelihoods of the bigger windows that came before it. This can be achieved by applying Bayes' theorem as follows:

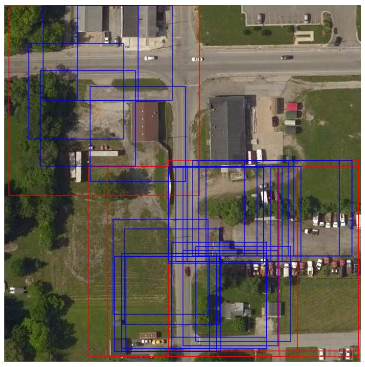
$$P(w|W) = \frac{P(W|w)P(w)}{P(W)}, \tag{3.7}$$

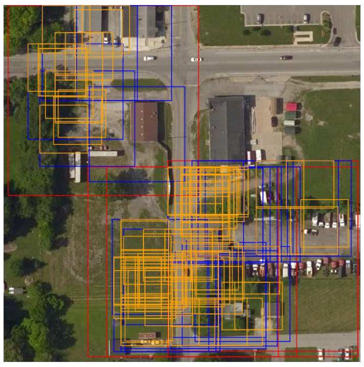
where $P(W)$ and $P(w)$ are the probabilities of the candidate locations in the bigger window W and in the smaller window w, respectively. In this equation, $P(W)$ is the output of the first run of the algorithm; $P(w|W)$ corresponds to the output of the second run of the algorithm, with the smaller window, since we assume that the query is inside the bigger window W; $P(W|w) = 1$ by definition, since the query is obviously inside the bigger window if it is inside of the smaller one; finally, $P(w)$ is what we are trying to find, the probability that the query image is inside the candidate location of size w. Re-arranging the terms of Equation 3.7, we get

$$P(w) = P(W)P(w|W), \tag{3.8}$$

which means that at each step we must multiply the probability provided by the algorithm for window w to the probability provided by the previous iteration in window W.


(a) Initial search with window size set to 400 pixels.


(b) Second search with a window of 200 pixels.


(c) Last search with a window of 100 pixels.

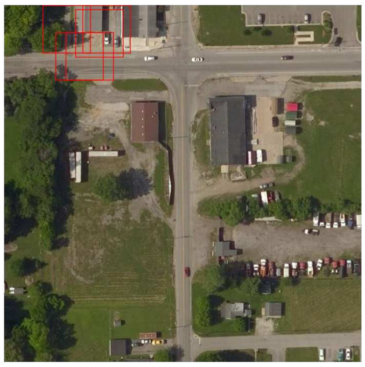

(d) Top 5 locations selected among the candidate locations.

Figure 3.2: Illustration of our search-refinement approach with windows of decreasing size. The 5 most likely locations in each step are shown by colored squares in each image.

Chapter 4

Semantic Segmentation

The algorithm described in Chapter 3 can be divided in two phases: extracting the landmarks and matching the generated graphs. In this chapter, we will focus on the first phase, investigating if *semantic segmentation* can be used to automatically obtain landmark points. As we have images of two significantly different domains (ground and aerial), we decided to investigate separate solutions for each. Sections 4.1 and 4.2 describe our investigations for ground and aerial images, respectively.

4.1 Ground Image Segmentation

As mentioned in Section 2.2, there are several networks available for ground image segmentation. For our work, we decided to use SegNet [5], because it provides reasonably good results, is fast, contains all the segmentation classes that we consider relevant, and also because several open source implementations are available. However, there are other options that would have sufficed for our purposes. Though we did not investigate other networks, this is planned as a future work, because improving the segmentation might yield an overall improvement in our method. Some networks that could be investigated are: the extension of DeepLabV3Plus proposed by Zhu *et al.* [90] and the Transformer-based solutions proposed by Chen *et al.* [17] and Li *et al.* [37]. The network proposed by the authors of SegNet has an encoder-decoder architecture, shown in Figure 4.1, with the encoder network using the same architecture as VGG16 [61], except for the fully connected layers. This has the advantage of making the network smaller, faster and easier to train.

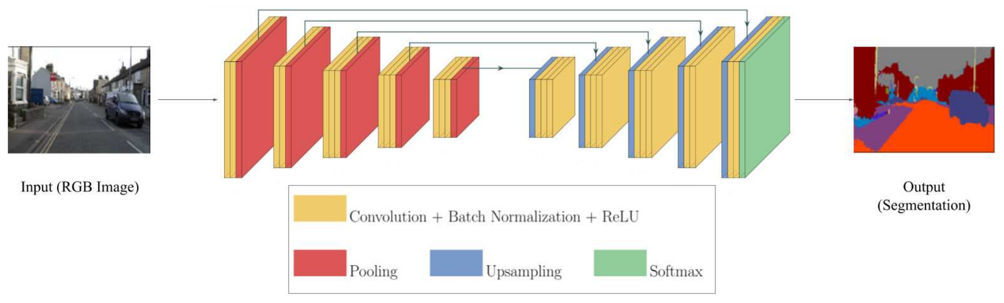


Figure 4.1: The SegNet encoder-decoder architecture (Image adapted from [5]).

The network was trained using the CamVid dataset[9], which contains 11 classes: building, tree, sky, car, sign-symbol, road, pedestrian, fence, column-pole, sidewalk and bicyclist. Some of these classes are not applicable to our problem since they are not visible from aerial images (for instance, the sky), but most of the expected classes are here, such as building, road and sidewalk.

4.2 Aerial Image Segmentation

Semantic segmentation of aerial images is a problem less explored in the literature. Most related work is of limited scope or not directly applicable to our problem. Furthermore, none of them provide source code, therefore we could not use any of them directly. Hence, we decided to implement an aerial segmentation method ourselves.

However, as also stated in Section 2.2, there are few datasets available for this problem. VALID [14] is, to our best knowledge, the closest dataset related to what we need; however, it was not available when we performed our research. Therefore, our decision was to build a dataset ourselves.

In Section 4.2.1 we describe how this dataset was built, and in Section 4.2.2 we detail how we trained our aerial segmentation method.

4.2.1 Building an Aerial Segmentation Dataset

To build our aerial segmentation dataset, we basically need pairs of images: the aerial image and its corresponding expected segmentation (ground truth).

Obtaining aerial images is more straightforward as there are several services that provide them. In our work, we choose to use Bing Maps [46], though several other sources could be used.

Obtaining the ground truth for segmentation is a more complicated, since, as mentioned, there are no publicly available datasets. Our solution was to employ OpenStreetMap (OSM) [47], a Wiki-like project that creates and distributes free geographic data for the world, allowing users to edit and create maps in a collaborative manner. OSM presents a conceptual model of the physical world that consists of three basic elements:

- **Nodes:** a single point in space, defined by its latitude and longitude.
- **Ways:** an ordered list of nodes, which can be used to represent an area or linear features. Ways can be closed (i.e., the end node and the begin node are the same) or open. Closed ways can be used to represent the area of a building, for instance, while open ways can represent roads, among several other things.
- **Relations:** an ordered list of nodes, ways or other relations. They are used to group elements and explain work they work together. For instance, a relation might contain several open ways that represent parts of the same highway, or several closed polygons that represent parts of the same building.

Besides these three basic elements, each element can have one or more *tags* associated with it. A tag is a pair of textual values (a 'key' and a 'value') that are used to describe an element. It can contain several kinds of information, for instance: the width of a road, the maximum speed of a highway, the name of a building, among many others.

OSM also provides an API (Application Programming Interface) for downloading all available data for a given area. This data can be used as input to build a ground truth segmentation, since it can contain all the elements that we expect: buildings, roads, vegetation, rivers, and so on. As expected, since the platform relies on user input, it is prone to error, and information might not be available in all areas. However, as will be detailed later in this section, the data seems reliable enough, particularly for urban areas, and filtering methods can be developed to reduce the errors.

With sources for both aerial imagery and segmentation data available, we developed an algorithm to download and process this data, generating an aerial segmentation dataset. This algorithm works with the following steps:

1. **Define points of interest:** the user provides the interest areas and its central coordinates (latitude, longitude).

2. **Download aerial images:** the algorithm downloads a pre-defined number of images in random locations near the points of interest from Bing Maps. Due to constraints in Bing Maps API, only images of 256×256 pixels can be downloaded. To overcome this and be able to obtain higher resolution images, we download 25 images (in a 5×5 grid around the central point) and then merge them, obtaining a final image of 1280×1280.

3. **Download and process segmentation data:** obtaining the segmentation is more complicated since the data from OSM must be properly parsed and filtered to obtain the information we seek. The process is the following: given an aerial image downloaded from Bing Maps, first, we download all the available information from OSM (all the nodes, ways and relationships, and the corresponding tags). As OSM is a general-use platform, a lot of this data is not of interest to the segmentation. We then discard all the unnecessary data and keep only what will be required for the next steps. The next step is to create a *visual representation* of the data: using Python and libraries such as Shapely[27], we start with a black image and then sequentially draw all the required elements (lines for roads, polygons for buildings, and so on). Each element will have a different color, according to the segmentation class it belongs to, and unlabeled elements will remain black. Particular attention must be paid to correctly convert the elements coordinates (given by latitude and longitude) to pixel coordinates, to make sure that the drawn elements match the aerial image. After this process, the resulting images (in the same format as the aerial images, 1280×1280) are saved, both in binary (each pixel containing an integer number from 0 to N according to the class it belongs to) and colored versions (each class has a specific color to represent it).

4. **Filter results:** we then employ some heuristics to try and eliminate results that are bad or uninformative: for instance, images which result in only one class, images that are mostly unlabeled, and so on.

5. **Save results:** lastly, the obtained pairs of images are organized in folders and zipped.

For the purpose of our work, after manually looking at the available data from OSM, we defined 13 classes of interest, which are detailed below (the description of each class is extracted from OSM):

1. **Building:** man-made structure with a roof, standing more or less permanently in one place.
2. **Construction area:** site which is under active development and construction of a building or structure.
3. **Industrial area:** predominantly industrial land uses such as workshops, factories, or warehouses.
4. **Parking:** facility used by the public, customers, or other authorized users for parking motor vehicles.
5. **Highway:** used for identifying any kind of road, street or path, except for the more specific sub-classes below.
6. **Footway:** for designated footpaths, i.e., mainly/exclusively for pedestrians or roads used mainly/exclusively for pedestrians.
7. **Dirt road:** highways made of surfaces such as sand, dirt, ground, etc.
8. **Beach:** loose geological landform along the coast or along another body of water consisting of sand, gravel, shingle, pebbles, cobblestones or sometimes shell fragments etc.
9. **Water:** Any body of water, from natural such as a lake or pond to artificial like moat or canal.
10. **Park:** an area of open space for recreational use, usually designed and in semi-natural state with grassy areas, trees and bushes.
11. **Tree:** A single tree, sometimes lone or significant.
12. **Grass:** areas of mown and managed grass or other forms of low vegetation.
13. **Natural:** wide variety of physical geography, geological and landcover features that do not fit in other categories above (examples: peak, wetland, cliff, conservation, nature reserve, among others).

An extra 14th class is used to represent unlabeled parts of the image. As all the available data is downloaded from OSM, the algorithm can be easily modified to add or remove classes.

For the purpose of our work, we downloaded data from 17 interest areas: nine urban areas (Berlin, Boston, Chicago, London, Manhattan, Paris, Rome, San Francisco and São

Paulo) and eight rural areas (located in distinct areas of the United States). A total of 30 images were downloaded from each location, amounting for a total of 510 images. After the filtering process described, a total of 277 images remained for the final version of our dataset.

In Figure 4.2 we show some example pairs of images for both urban and rural areas. We can see that, as expected, the quality of the final images highly depends on the quality of the data obtained from OSM. In general, data from central, highly populated areas, is more reliable, whereas data from rural areas is often missing important information. Even after our filtering, some bad quality image remains (such as the one shown in Figures 4.2(e) and 4.2(f)), mainly because there is no segmentation data available in OSM. However, in general, we believe that the result is acceptable and that the obtained dataset can be useful for our next steps, and perhaps for other future work in this area.

The dataset is available upon request at `https://zenodo.org/record/4927665` (DOI 10.5281/zenodo.4927665). For more information about the dataset, please refer to Appendix A, which presents the Datasheet (full description) of the dataset, according to the instructions provided by Gebru *et al.* [26].

4.2.2 Training an Aerial Segmentation Network

Our next step was to train SegNet specifically for aerial segmentation, using the dataset described above. Given that the domains of aerial and ground segmentation are so different, we chose not to perform a fine-tuning of the network trained from ground segmentation, but to train a completely new network instead.

Firstly, some adaptations were required in the dataset: to make the dataset possibly applicable to other scenarios, the images were downloaded in high resolution (1280×1280 pixels), while SegNet expects lower resolution images as input (360×480 pixels). We performed as following: for each original 1280×1280 image, we extract six non-overlapping patches of dimensions 360×480, leading to a total of 1662 images (277 images of the original dataset × 6 patches per image). The dataset was then randomly split in train, validation and test (2/3 for train and validation and 1/3 for test). This division was performed on an image basis, so all patches of a given image are either assigned to the train or to the test set. We also built a simplified version of the dataset, reducing the number of classes from the original 13 to 6 (building, highway, natural, water, footway and dirt road). To do this, some classes were merged (for instance, grass, natural, tree and park were merged into a single class "natural") and some classes with very low frequency were eliminated (construction, industrial and parking).

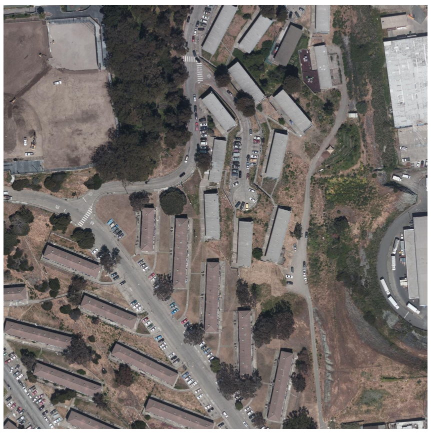
Finally, we trained two versions of the aerial segmentation network, using the same parameters suggested by the authors of the original network [5]: one using the dataset with 13 classes and another one with the simplified dataset. Results will be shown later in Chapter 5.

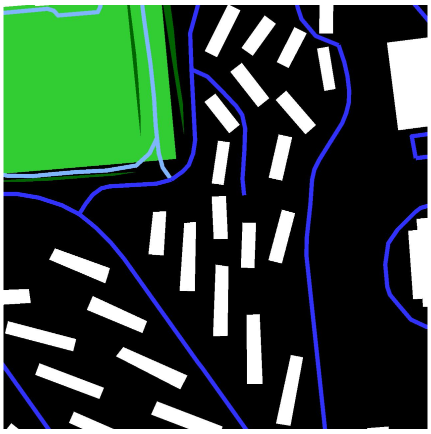
4.3 Extracting Landmarks from Segmentation

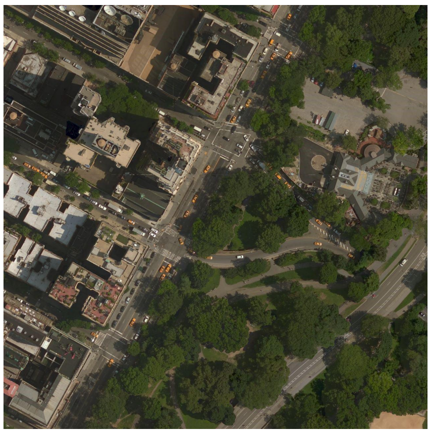
There are several ways in which we can obtain landmark points (and their corresponding classes) from a segmented image. We developed and tested several of them, but they all have one step in common: first, the segmented image is divided into connected components (*CC*, i.e., regions of adjacent pixels belonging to the same class). Given all the CC, we can proceed in ways such as:

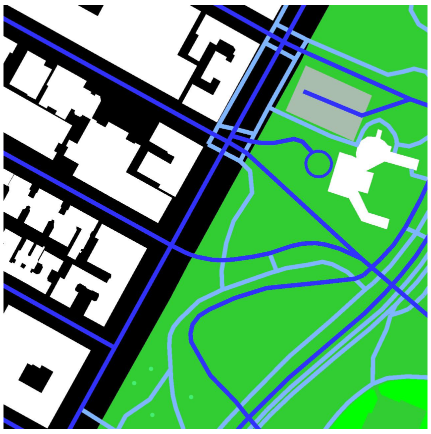
- **Centroid:** the centroid of each CC (i.e., the mean of its pixels coordinates) is considered as a landmark point. This is very simple to do but has the potential drawback that the landmark point extracted could lie outside the component itself (consider a *U* shape, for instance).
- **Grid:** we could extract several landmark points inside each CC, distributed over a grid of predefined size.
- **Component:** lastly, we could extend the concept of landmark points to landmark components. With this approach, each CC is considered a landmark itself, and landmarks are considered covisible if any pixel in one component is visible to any pixel in the other.

In practice, we might need to consider using a *segmentation size threshold* to avoid noise from the segmentation algorithm (i.e., CC that are too small to be meaningful, probably due to errors). This process can be applied for both aerial and ground images. In Chapter 5, we present the results of our experiments with each method.

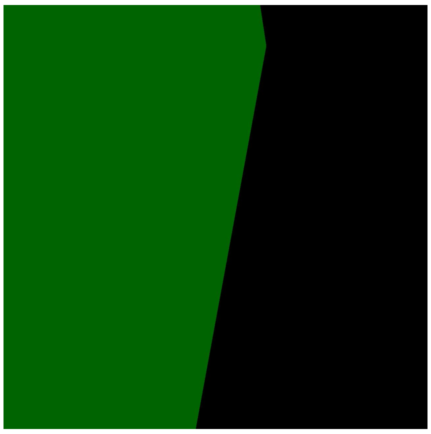

(a) Aerial image of San Francisco.


(b) Corresponding segmentation in our dataset.


(c) Aerial image of Manhattan.


(d) Corresponding segmentation in our dataset.

(e) Aerial image of a rural area in the US.


(f) Corresponding segmentation in our dataset.

⬜	building	🟩	grass	🟫	construction	🟨	beach	🟦	footway	🟩	park	🟩	tree
🟦	highway	🟩	natural	🟪	industrial	🟦	water	▨	parking	🟧	dirt_road	⬛	unlabelled

(g) Color code corresponding to each class in the segmented images.

Figure 4.2: Some pairs of aerial images and their corresponding segmentation extracted from our dataset.

Chapter 5

Experiments and Results

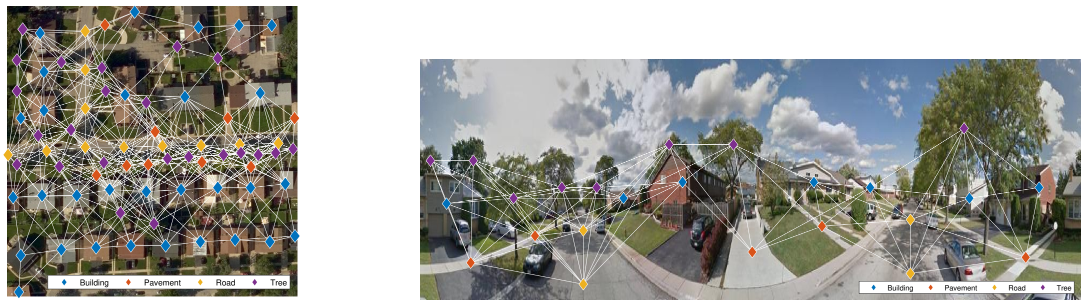
In this section, we provide and analyze results of the proposed methods and solutions. First, in Section 5.1, we present results of our graph-matching localization algorithm described in Chapter 3 using manual input of keypoints; then, in Section 5.2, we show the results of the network described in Chapter 4, trained for semantic segmentation of aerial images; lastly, in Section 5.3, we present the results of a fully automated pipeline, combining the graph-matching algorithm with the landmark extraction techniques described in Section 4.3.

Though there are several works in the literature that analyze the *cross-view image matching problem*, we could not find any research that discusses our proposed simplified problem, *viewpoint localization*. Hence, in this section, we will not present comparisons of our method with the literature.

5.1 Localization with Manual Landmarks

Our first step was to test the algorithm proposed in Chapter 3 using manually automated data. To do so, we arbitrarily selected 15 images from the CVUSA dataset [84], mixing urban and rural areas (no specific criteria was used to select these images, we just attempted to include multiple scenes, such as heavily populated and less populated urban areas, and rural areas with several or few landmarks). Ground images are 1232×224 360-degree panoramas, while aerials are 750×750 satellite images. All of the images were manually annotated with the following procedure: each visible landmark was pointed, and then, for each image, we stored a $K \times 3$ array, where each triplet denotes the pixel coordinates and class of a landmark. We considered four classes: *building*, *tree*, *road* and *pavement*. For the first two, related to objects limited in space, we selected a pixel within the object perimeter to represent the landmark position. For *road* and *pavement*, instead, we picked a set of points at regular intervals along their length. Figure 5.1 depicts an example of landmarks extracted from a ground-aerial matching pair, with the respective graphs in overlay. Since CVUSA images are centered and aligned, in our case, the expected output of the algorithm, i.e., the viewpoint in the aerial image that corresponds to the ground view, is always the center of the aerial image.

Landmark graphs were obtained following the method described in Section 3.1.1.



(a) Aerial view. (b) 360° ground image.

Figure 5.1: Landmarks extracted from a pair of associated aerial and ground images from the CVUSA dataset [84]. Four classes considered: *building*, *pavement*, *road*, and *tree*. The landmark graphs obtained via the algorithm in Section 3.1.1 are also displayed in overlay.

Given that the density of landmarks is strongly dependent on the considered area – being typically higher in urban regions – landmark graphs composed of multiple disconnected components are possible if the covisibility window is not large enough. To overcome this issue, we adopted an automated variable-size covisibility window. For each image, we first computed the distance from each landmark to its nearest neighbor. Then, being μ and σ the mean and standard deviation of such distances, respectively, we set the covisibility window size to $\mu + 5\sigma$. This choice helps minimizing the chance of having groups of landmarks disconnected from the main graph.

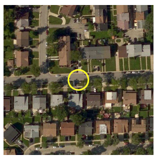
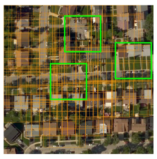
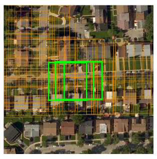
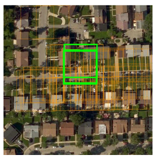
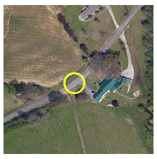
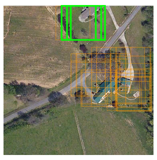
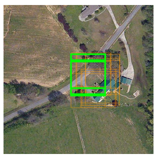
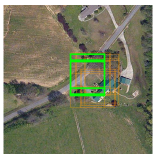
The covisibility parameter p was fixed to 0.5 throughout all the experiments. Mei *et al.* [45] present a discussion on the influence of p on another problem, with conclusions that hold for this paper. In [63], the authors list a series of alternative methods for location extension that may be tested in future developments to eliminate the need for a pre-tuned parameter.

Table 5.1 shows two examples of viewpoint localization with the proposed method. The first two columns show the aerial views (accompanied by the ground-truth viewpoint location) and the associated 360° images. For each image pair, we run the algorithm with an increasing number of landmark classes: the third column reports the results obtained for two classes, namely *building* and *tree*; in the fourth and fifth columns, we add *road* and *pavement* classes, respectively. Candidate locations are shown as orange squares, while green ones denote the top-3 locations, according to the ML criterion described in Section 3.1.4.

In both cases, the addition of new classes yields improvements in terms of localization accuracy and a better convergence of top candidate locations to a limited portion of the aerial view. This is crucial for areas that appear self-similar when restricting the focus to few landmark types (think of a urban area where only buildings are considered). The effect is particularly evident in the first case, where the satellite view depicts various self-similar locations, making the localization nontrivial even for a human observer. The correct location is recognized only after the introduction of the *road* class.

Figure 5.2 reports a quantitative evaluation of the system over 15 manually annotated

Table 5.1: Example results on CVUSA dataset [84] with manually annotated data.

Input views		Localization (top-3 results)		
Aerial	Ground 360°	2 classes	3 classes	**4 classes**
				
				

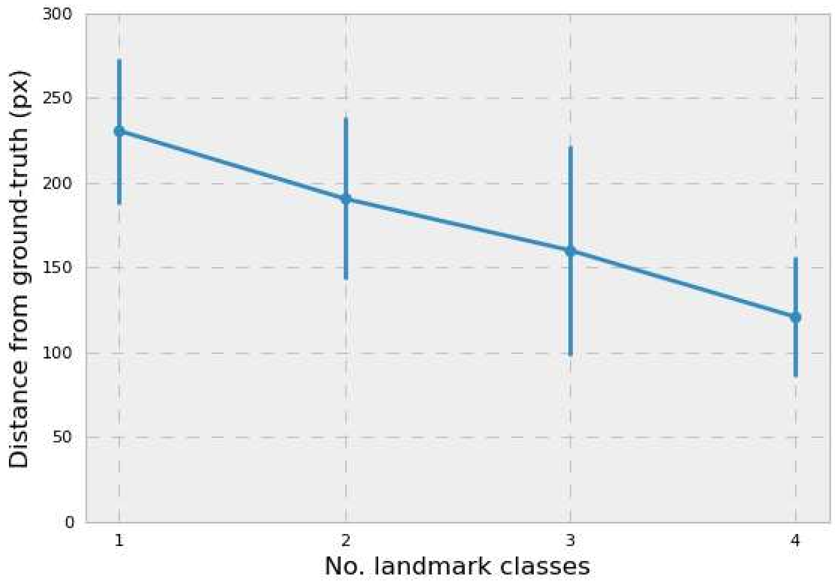


Figure 5.2: Average distance from the ground-truth viewpoint (in pixel) for increasing number of landmark classes in the dictionary $\mathcal{W}$. Error bars denote the standard deviation among top-5 locations.

image pairs. We calculate, for each pair, the average distance (in pixels) from the ground-truth viewpoint to the top-5 locations predicted by our algorithm (if there are ties, we select 5 locations arbitrarily among the top candidate locations). The results show a clear descending trend for increasing numbers of classes, corroborating the assumption that a greater diversity in the landmark graphs leads to more discriminative results. Finally, note that the standard deviation across top-5 locations reaches its minimum when all four classes are included in the dictionary.

5.2 Semantic Segmentation of Aerial Images

As mentioned in Section 4.2.2, we trained two versions of the aerial segmentation network, using the two versions of our dataset (the simplified version with 6 classes and the full version with 13). For brevity, we will refer to these networks as AirSegNet6 and AirSeg-

Net13 from here on. We used a publicly available TensorFlow implementation of SegNet [5], with adaptations, and default parameters suggested by the authors. Both networks were trained until convergence using an NVIDIA GTX 1080 Ti GPU. The network generates as output a 360×480 pixels, single channel image. Each pixel contains an integer value from 0 to N, where N is the total number of classes, indicating to which class it belongs (the value 0 is reserved for unlabeled pixels). To make visualization easier, we developed methods to convert such images into colored images, substituting each pixel value to a corresponding RGB color.

Tables 5.2 and 5.3 show some results provided by AirSegNet6 and AirSegNet13, respectively, in the test dataset (1/3 of the dataset). We can see that the results are promising and that the network indeed seems to be learning to segment aerial images. One of the main bottlenecks seems to be the quality of the input data itself: in the third row of both tables, we see a huge portion of unlabeled vegetation, while the network was able to identify (through knowledge from other examples) elements in that area; on the second row of Table 5.3, we can see a block labelled as a construction zone, which was misclassified as building, probably because there are many more examples of buildings than of construction areas in the dataset.

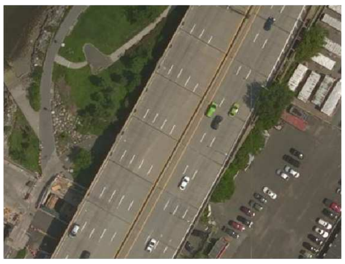
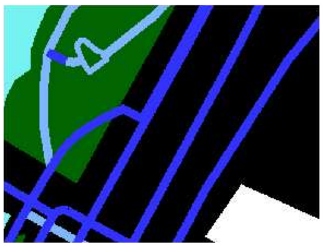
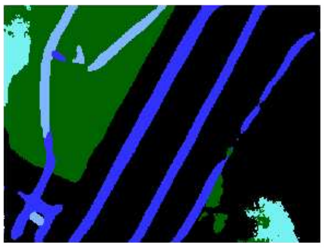
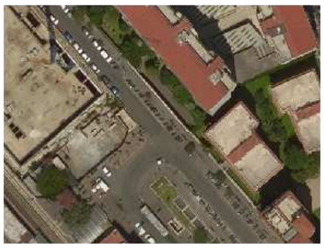
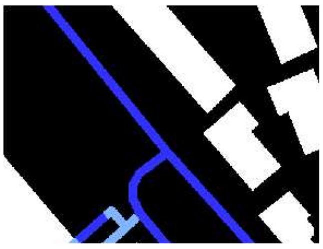
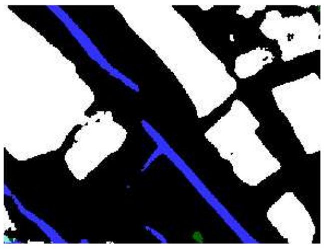
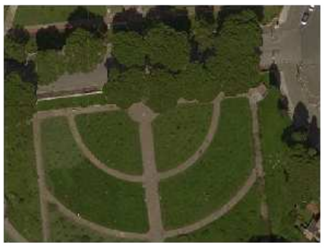
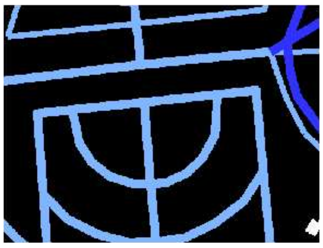
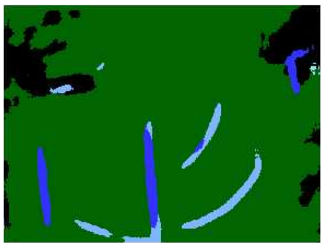
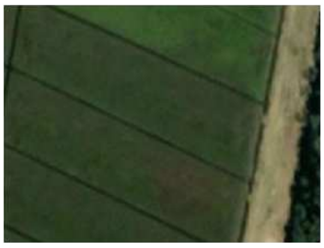
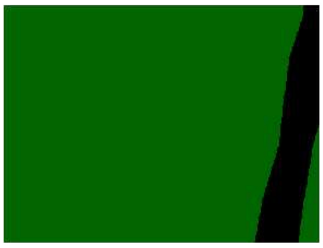
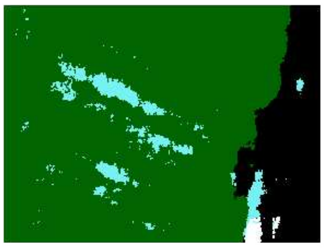
After the networks were trained, we tested them in a *cross-dataset* scenario, by using aerial images from CVUSA. However, some adaptation is required, since our networks take images with dimensions 360×480 as input, while CVUSA images are 750×750. We experimented with two different methods:

1. **Method 1 - split and join:** split the original 750×750 image into patches of 360×480 pixels each, segment each of them and then join the results to form a 750×750 segmentation. Notice that the dimensions are not exactly divisible, so our patches have some overlap. There are multiple ways to deal with this, but we chose a simple approach: when joining the results, we simply discard the overlapping portion of one of the patches. This approach has the disadvantage that the transition might not be smooth near to where the patches are joined.

2. **Method 2 - re-scaling:** re-scale the 750×750 into a 360×480 image, adding black strips in the corner. Segment this reduced image and then scale it back to 750×750. Special care is required here to avoid interpolation errors when scaling up the image (since all pixels are expected to be integers ranging from 0 to the number of classes).

In Table 5.4, we present results of our AirSegNet6 on CVUSA images: first column shows the input image, the middle column shows the output using method 1 described above, and last column shows the output obtained using method 2 (since this is a cross-dataset scenario, we do not have ground truth images here). We can see that the network performs fairly well in urban scenarios (top two rows): most of the buildings and roads were detected, and some of the vegetation. However, there are also some possibilities of improvement, since not all trees and vegetation were detected, and the roads do not form continuous lines. The third row shows a more challenging scene with several elements. While some results do make sense (the footway detected on the upper right corner and the dirt road on the upper left by method 1), there are some issues: the water of the

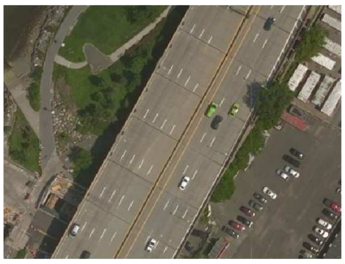
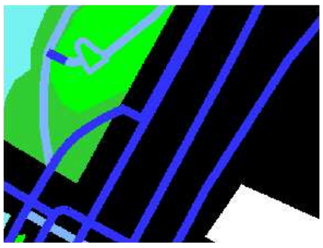
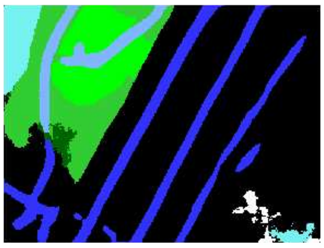
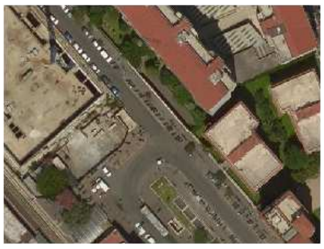
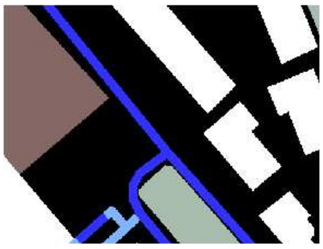
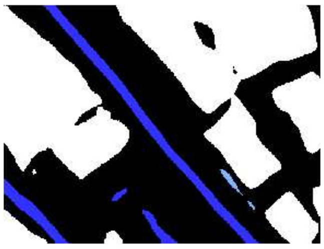
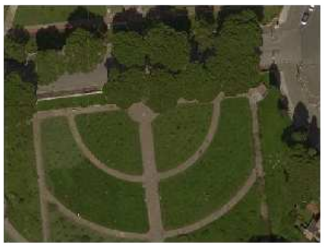
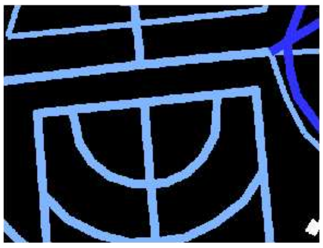
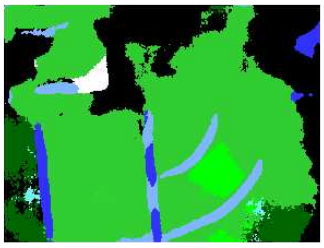
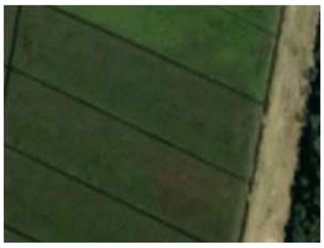
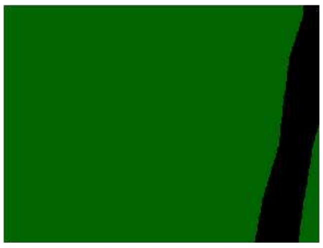
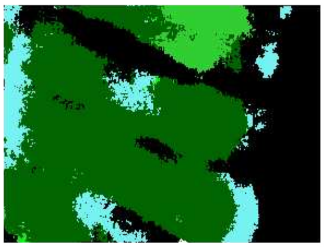
Table 5.2: Example results of our AirSegNet6 on the test dataset.

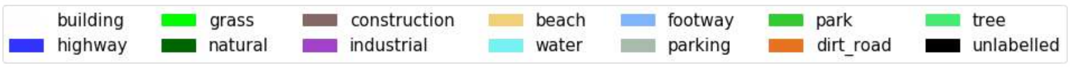
Input Image	Ground truth	Output
		
		
		
		

⬜ building 🟦 highway 🟩 natural 🟦 water 🟦 footway 🟧 dirt_road ⬛ unlabelled

river was mostly not detected, and some buildings were detected where there was none. Also, the effect of joining the patches is clearly visible for method 1 (notice the unnatural division in the center of the image). Lastly, the bottom line shows a scenario where the network performed particularly bad, since most of the image was classified as a road. We can notice this is a challenging scene, because the texture and colors of the vegetation do resemble a road, especially in the bottom corner. Comparing the results obtained by method 1 and 2, in general, the results of method 1 are visually better: for the urban images (top two rows), the results of method 2 are not bad, but some of the detail seems to be lost, probably due to the loss of information when we scale the original images from 750×750 to 360×480. For the images in the two bottom rows, method 2 seems to present the same issues than the method 1, with an extra downside that there seems to be a little

Table 5.3: Example results of our AirSegNet13 on the test dataset.

Input Image	Ground truth	Output
		
		
		
		

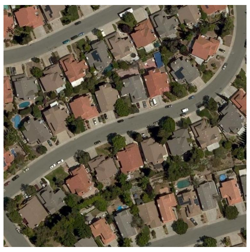
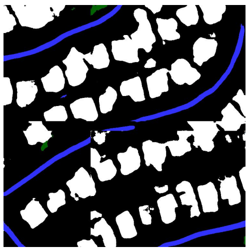
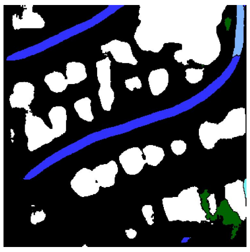
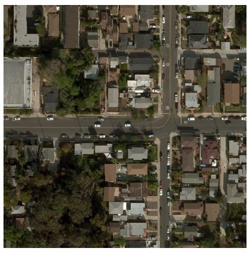
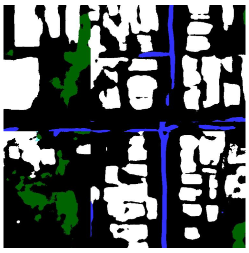
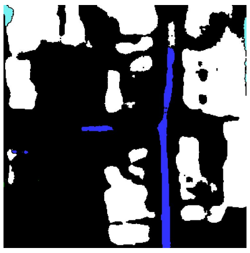
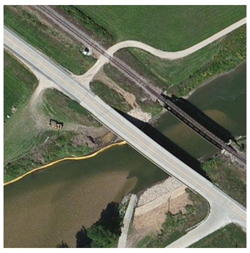
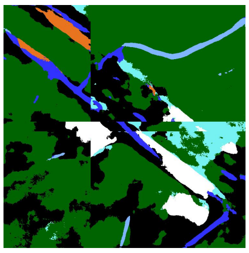
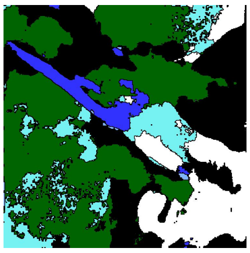
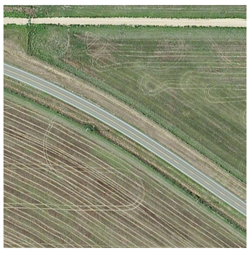
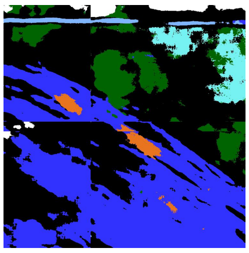
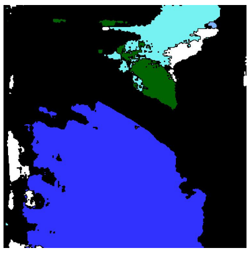


confusion near the borders of different components (there are a lot of unlabeled pixels in these areas). The overall best behavior for urban areas (in both methods), roads and buildings is expected, since the data obtained from OpenStreetMap seems to be more reliable and abundant for such classes.

Table 5.5 shows some results for AirSegNet13 with CVUSA images, organized in the same way as Table 5.4. Much of the conclusions for AirSegNet6 seem to be valid for AirSegNet13: the behavior is better in urban conditions, especially for roads and buildings, which are the most common and reliably labelled classes in the dataset; there seems to be a loss of detail and granularity in method 2 compared to method 1; the results for less common classes are not good (for instance, the water appearing in the two bottom rows).

The results show that both networks provide acceptable results for urban scenarios,

Table 5.4: Some segmentation results of our AirSegNet6 on CVUSA images using methods 1 (split and join) and 2 (re-scaling).

Input Image	Output of Method 1	Output of Method 2
		
		
		
		

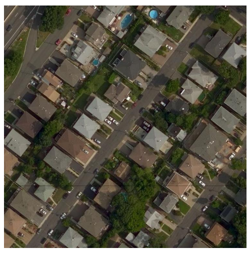
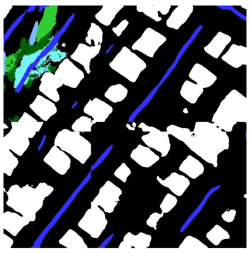
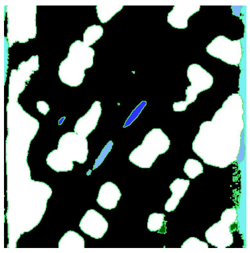
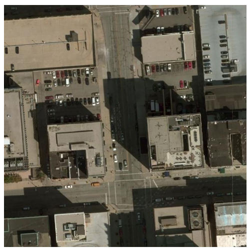
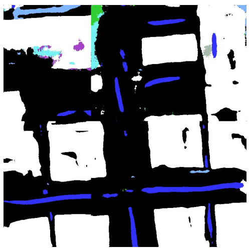
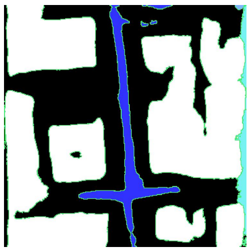
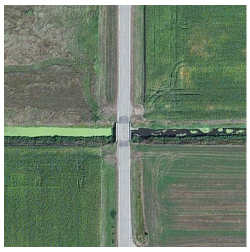
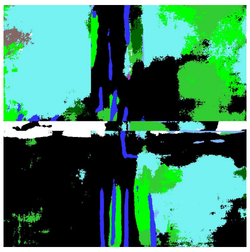
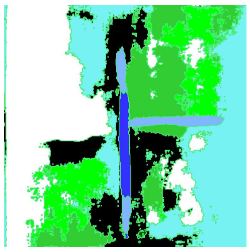
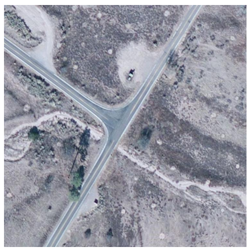
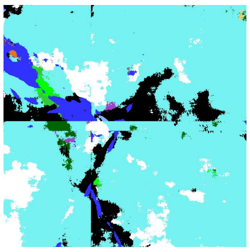
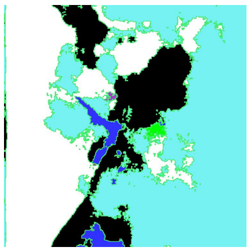
⬜ building 🟦 highway 🟩 natural 🟦 water 🟦 footway 🟧 dirt_road ⬛ unlabelled

and that the main issue is actually in the dataset, due to the lack of reliable data especially for some less common classes. Improving and extending the dataset could greatly improve the results of our networks. To minimize this issue, we chose to use AirSegNet6 for our following experiments, since it focuses on the most common classes and is therefore slightly less prone to this kind of errors than our other network, AirSegNet13.

5.3 Localization With Automated Landmarks

The last step of our work, aiming at an automated pipeline for geolocalization, was to employ the semantic segmentation, including the results obtained in Section 5.2, for automatic extraction of keypoints. Throughout this section, unless explicitly stated, we use our best network from Section 5.2 (AirSegNet6 with "join and split" method) for segmentation of aerial images, and off-the-shelf SegNet for segmentation of ground images.

Table 5.5: Some segmentation results of our AirSegNet13 on CVUSA images using methods 1 (split and join) and 2 (re-scaling).

Input Image	Output of Method 1	Output of Method 2
		
		
		
		

⬜ building	🟩 grass	🟫 construction	🟨 beach	🟦 footway	🟩 park	🟩 tree
🟦 highway	🟩 natural	🟪 industrial	🟦 water	⬜ parking	🟧 dirt_road	⬛ unlabelled

Our first set of experiments was directed at testing and understanding how each of the parameters of the algorithm described in Chapter 3 affects the results. The implemented algorithm has the following parameters:

- **Segmentation size threshold:** as mentioned in Section 4.3, connected components (CCs) that are too small are discarded, since they are much likely an error from the segmentation algorithm instead of a meaningful keypoint. This parameter controls the minimum size (in pixels) that a component must have in order to be considered.

- **Covisibility window size:** as explained in Section 3.1.1, there is an edge between two nodes in the landmark graph if the landmarks are visible together inside a square window of predefined size. This parameter controls the size of this window: the smaller it is, the closer the components must be in order to be considered connected. We also proposed a method to calculate this parameter, outlined in Section 5.1.

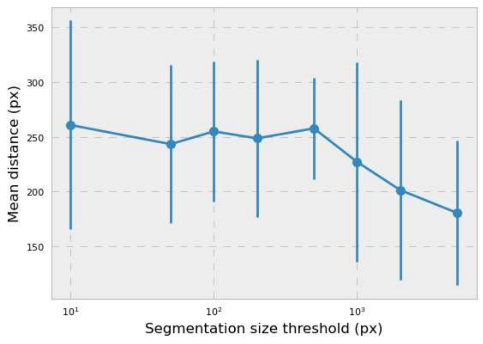
- **Covisibility stride:** this parameter denotes the size of the stride, in pixels, between observations of the covisibility window. Using a higher value might speed up the algorithm since less observations are required, but might miss important connections.

- **Landmark extraction method:** the method used to extract landmarks from the semantic segmentation. We propose three different approaches, as detailed in Section 4.3: component extraction, centroid extraction and grid extraction. For the grid extraction, there is an extra parameter, which is the size of the grid, in pixels.

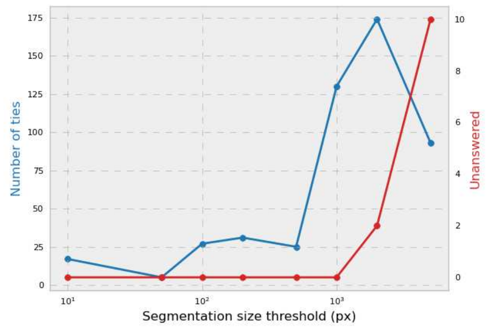
All experiments were run with the same 15 images used in Section 5.1, to make comparisons easier, unless explicitly noted. Also, following the same approach as before, if there are ties between candidate locations, the algorithm will select the top-k among them arbitrarily. We used, in all experiments, a total of four classes in the dictionary $\mathcal{W}$: building, road, vegetation and pavement; these classes are the ones that appear, perhaps with slightly different names, in both SegNet (which has 11 classes: building, tree, sky, car, sign-symbol, road, pedestrian, fence, column-pole, side-walk and bicyclist) and AirSegNet6 (building, highway, natural, water, footway and dirt road).

In Figure 5.3, we see the results of our algorithm over different segmentation size thresholds: in Figure 5.3(a) we show the mean error (distance in pixels), while in Figure 5.3(b) we show the number of ties and of unanswered samples. We can see that, as we increase the segmentation size threshold (which will make the graph contain progressively less nodes), we increase the number of unanswered queries by our algorithm, which makes sense since the graph becomes each time less informative. The number of ties also increases, though not monotonically, because, as we decrease our graph, we increase the possibility that the same or similar subgraph appears in several candidate locations, making it impossible for the algorithm do distinguish between them. Overall, the results are not very clear, a reasonable choice for this parameter seems to be 50 pixels, as it gives the best overall results (approximately 243px of mean error) with the minimum number of ties and no unanswered queries.

Figure 5.4 presents the results of the experiments with the covisibility window size parameter (which represents the size of the window in which elements must be covisible to be connected in the graph): in Figure 5.4(a) we see the mean error, while in Figure 5.4(b) we see the ties. As expected, increasing this value too much, over 400 pixels, does not yield good results: we will start connecting practically all components, making the subgraphs too similar for the algorithm to decide between them. Below this threshold, there seems to be a tendency of improvement when we increase the covisiblity size, but there is an important detail: due to the design of our algorithm, the size of the covisibility window also defines the area of the candidate locations; if this value gets bigger, candidate locations tend to be dislocated towards the center of the image, which is the expected answer (since a bigger window would not fit the corner of the image, for instance). Figure 5.5 illustrates this behavior: notice how, in Figure 5.5(a) we have several candidates in



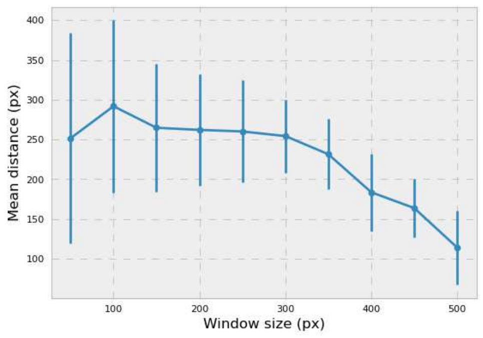
(a) Mean distance (in pixels) over different segmentation size thresholds.



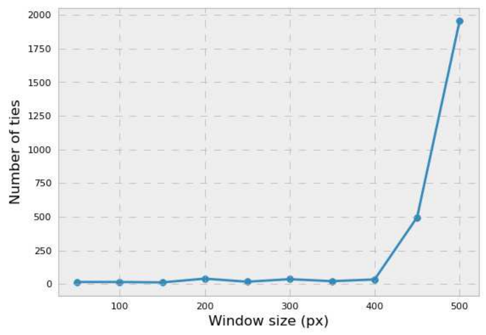
(b) Number of ties (in blue) and unanswered queries (in red) over different segmentation size thresholds.

Figure 5.3: How segmentation size threshold affects our algorithm.

the corner, while in Figure 5.5(b) the candidates are pushed towards the center, due to the bigger window size. Hence, a small error with a big covisibility window is neither beneficial, as it only happens because candidates are naturally closer to the answer, nor desired, as possibly meaningful candidates in the corners leave to be considered.



(a) Mean distance (in pixels) over different covisibility window sizes.

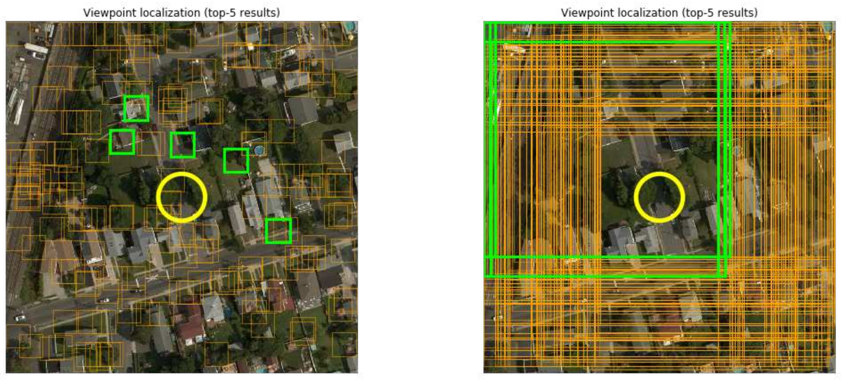


(b) Number of ties over different covisibility window sizes.

Figure 5.4: How covisibility window size threshold affects our algorithm.

In Figure 5.6, the results for our tests with covisibility stride are shown. The results vary, but there does not seem to be any particular logic or reason behind this. Hence, we decided to keep the default value of 1 for the next experiments, as it tends to make the graph as informative as possible.

Lastly, we performed tests with different methods for landmark extraction: using the components itself, its centroid, and a sampling points in a grid (we used strides of 20, 50 and 100 pixels). For this experiment, we used a slightly expanded dataset with 40 image pairs. Results are shown in Figure 5.7 and bring several interesting insights: though the change is not very significant, we can see that the error increases when we use the centroid instead of the whole component, which makes sense since the centroid loses important information about the shape of the component; the number of ties and unanswered queries explodes when we use a grid of stride 100, which also makes sense because we might


Viewpoint localization (top-5 results)
Viewpoint localization (top-5 results)

(a) Running the algorithm with a covisibility window size of 50px.

(b) The same image with a window size of 500px.

Figure 5.5: How covisibility window size affects the positions of the candidate locations in our algorithm. In light orange we see candidate locations, and in green the top-5 locations.

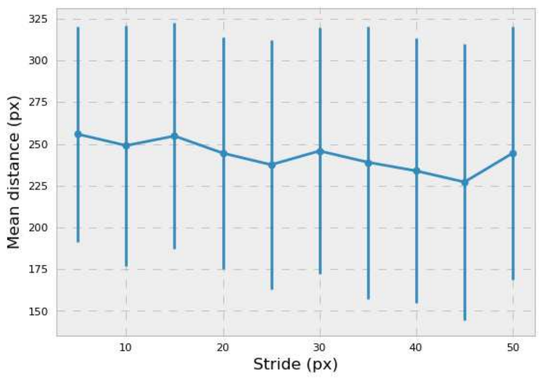
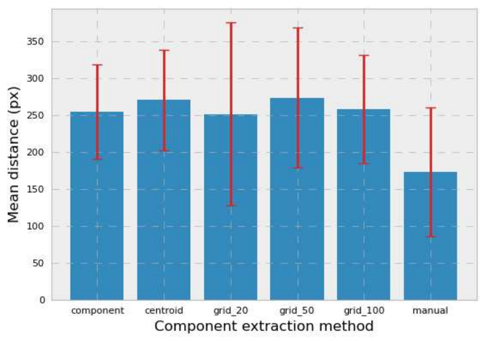


Figure 5.6: How covisibility stride affects our algorithm.

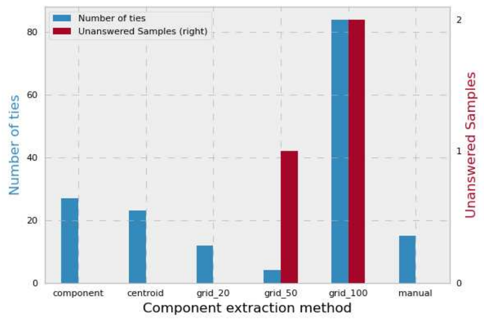
completely miss components that are too small. Overall, none of the automatic extraction methods came close to the manual results, but still using the components themselves or a grid of smaller size seem to be the two most promising approaches.

5.3.1 Hierarchical Search Refinement

In Section 3.1.5, we presented a possible extension of the graph matching algorithm that works in a hierarchical manner, refining the search in each iteration. Table 5.6 presents the results of this approach as compared to our baseline (standard algorithm with default parameters) using two different methods for landmark extraction (component and grid of size 20). The results are unfortunately worse than the baseline in all scenarios. This could be explained by the fact that, when we run our algorithm in a smaller window, we lose probably useful information about the areas not inside this window. In fact, our whole algorithm was devised in the hypothesis that information about how the landmarks are geographically distributed could be useful for localization. When we reduce the search



(a) Mean distance (in pixels) over different landmark extraction methods.



(b) Number of ties and unanswered samples over different landmark extraction methods.

Figure 5.7: How landmark extraction method affects our algorithm.

Table 5.6: Results of our search-refinement method, as compared to our baseline (standard algorithm with default parameters).

Mean Error (pixels)	Component Extraction	Grid Extraction (stride 20)
Search Refinement	265.2 ± 75.7	265.3 ± 87.9
Baseline	251.5 ± 132.2	243.5 ± 130.1

window and consider a smaller portion of the image, we lose such information, making the graphs less informative and leading to poorer results.

5.3.2 Removing Edges Connecting Similar Landmarks

We also considered the possibility of post-processing landmarks graph to rectify possible mistakes caused by errors in the segmentation. More specifically, we noticed that, in some cases, the generated graph contains edges connecting elements that are clearly from the same landmark. To avoid this, we tried to project a deep learning based approach to remove edges connecting elements that are too similar to each other. This approach works as follows: first, 64×64 patches are extracted for each connected component; if a patch cannot be found (for instance, if the component is too small), the component is ignored in the following steps; then, we extract feature vectors (FVs) of each path by forwarding them through a neural network and getting the output of the last layer before the fully connected layers; we use ResNet [29] for ground images and our own AirSegNet for aerial images; if a component contains more than one patch, we use the mean of the FVs; lastly, given the FVs describing each component, we check all edges of the graph and, if the distance between the corresponding FVs is smaller than a predefined threshold T, this edge is removed.

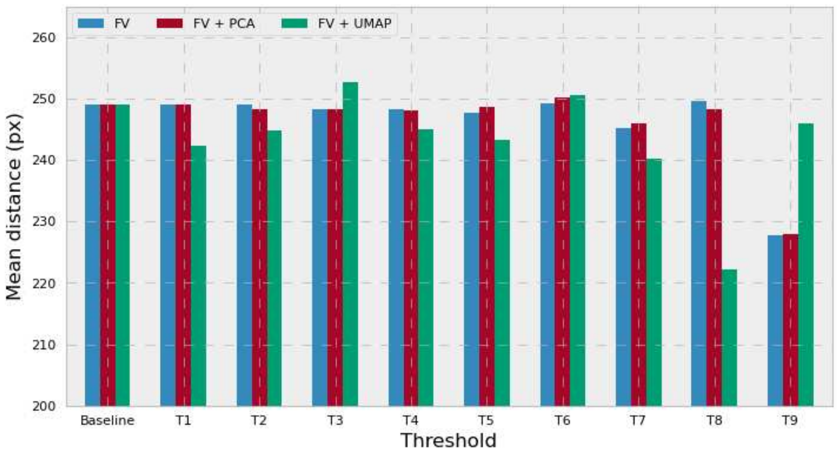
Using the procedure described above, we performed several experiments. Firstly, we tested several different values of the threshold T. This parameter is important because, if a very big threshold is used, too many edges will be removed, making the graph so uninformative that, sometimes, the algorithm cannot even find viable candidate locations. We also noticed that the dimensions of the feature vectors might be too large, particularly for AirSegNet (which has features of $12 \times 15 \times 512 = 92,160$ dimensions, while ResNet uses 2048 dimensions), so we experimented with two dimensionality reduction algorithms: PCA[25] and UMAP[53]. Since we have less than 2000 features, and it would not be possible to reduce FVs to 2048 dimensions, we decided to reduce both aerial and ground FVs to 1024 dimensions. For PCA, the total variance of the data explained by this reduction was 91.6% for aerial features and 99.8% for ground features.

Lastly, we performed tests with two different distances metrics: Euclidean distance and cosine distance. Figure 5.8(a) shows the mean distance, in pixels, over our test dataset, for different thresholds T (for brevity, the numerical values are omitted, but it suffices to say that they are in ascending order) and dimensionality reduction algorithms (the original FVs, in blue, and then after applying PCA and UMAP, in red and green, respectively). For this experiment, a subset of CVUSA with 100 images was used. The baseline result (original algorithm without edge removal) is also shown, to make comparison easier. We can see that the mean error starts reducing as the threshold grows bigger. However, there is a small caveat: as previously mentioned, too big thresholds will make the graph too uninformative and might make the algorithm fail to find any viable solution in some cases. In fact, for this experiment, all tests with threshold T_9 and UMAP tests with T_8 generated cases where no answer was found. Still, we can see that some results are interesting: in the best case here, T_7 with UMAP, the mean error was reduced from 249 of the baseline to 240, while still being able to provide an answer to all cases. In Figure 5.8(b), we see the same experiment setting, but using cosine distance. All tests with T_9 and T_{10} and also T_8 with the original feature vectors resulted in unanswered cases. However, in the best scenario that could still provide answers for all samples, T_8 with UMAP, the mean error was reduced from 249 to 231 pixels, a considerable improvement, and our best result so far. Overall, we can see that this approach of post-processing the graph to try to mitigate possible mistakes in the segmentation step are indeed promising and can lead to more informative graphs and better results.

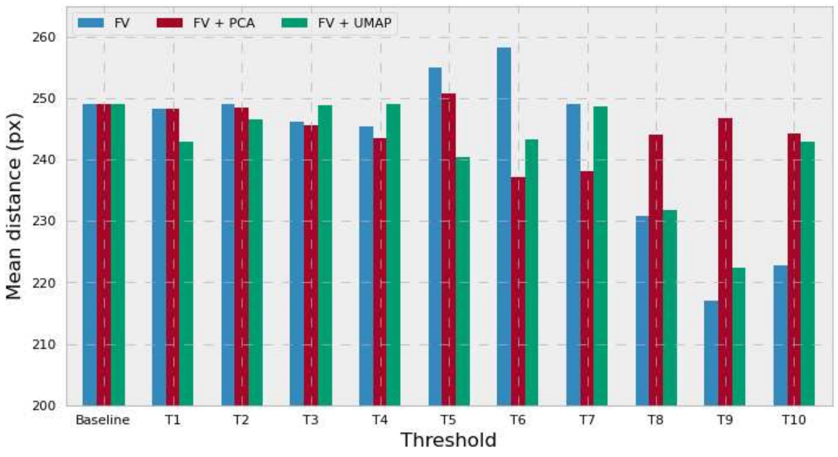
5.3.3 Non-matching Pairs

Up to this point, we have been analyzing our algorithm with matching pairs of aerial and ground images, the scenario for which it was developed. However, we also thought it could be meaningful to explore its behavior in the case of a non-matching pair. Our main goal here was to understand if the algorithm behaved significantly different for matching and non-matching pairs, and if it could somehow be used to discard non-matching pairs in the general cross-view image matching problem.

We projected the following experiment: using the 15 pairs of images in our small test dataset, we first ran the algorithm for all positive pairs (15 results), and then for all possible negative pairs ($15 \times 14 = 210$ results). We then plotted the distribution of the probabilities returned by the algorithm for the top-5 candidates and for the other candidates, for both the positive and negative pairs. Our intuition was that, if the algorithm could indeed be applicable to this scenario, the probabilities of the top-5 locations



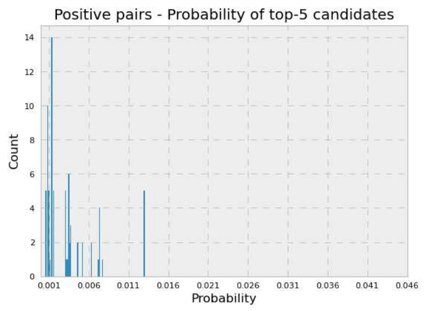
(a) Results using Euclidean distance.



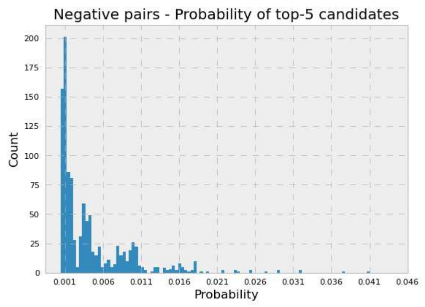
(b) Results using cosine distance.

Figure 5.8: Results of our method to remove edges connecting similar landmarks, using several different thresholds and dimensionality reduction algorithms.

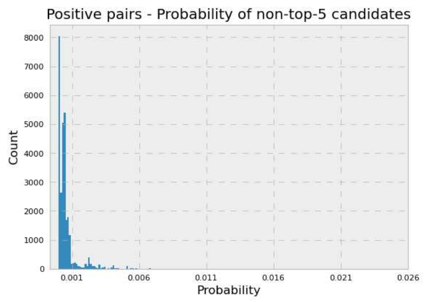
would be higher for the positive pairs (since the algorithm would be more certain that the candidate locations belong to that image); on the other hand, the probabilities for the negative pairs would be more evenly distributed, since the algorithm would not have high confidence in any candidate (because there is no correct candidate here). In Figure 5.9, we present the results of such experiments. Unfortunately, the distribution of probabilities does not seem to have any significant difference of behavior when the algorithm is ran with non-matching pairs instead of matching ones. This is somewhat expected: as the algorithm was not projected for this scenario, it will still try to find the most likely candidates among the available ones, even if none is indeed a match.



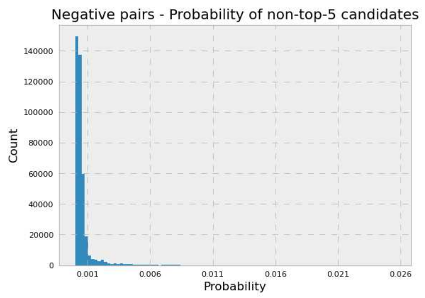
(a) Probability distribution for top-5 candidates in positive pairs.



(b) Probability distribution for top-5 candidates in negative pairs.



(c) Probability distribution for non-top-5 candidates in positive pairs.



(d) Probability distribution for non-top-5 candidates in negative pairs.

Figure 5.9: Probability distribution comparison between matching and non-matching pairs, for most probable candidates (top-5) and non-top-5 candidates.

Chapter 6

Conclusion

In this work, we studied the problem of geolocalization of a given ground query image. Due to the current worldwide availability of geo-tagged satellite imagery, the problem can be posed as a *cross-view image matching* problem: the ground query image has to be matched against a large database of geo-tagged aerial images. This is the approach used by many works in the literature [74, 84, 12, 31].

Differently from the most common approach, which relies upon a Siamese network architecture with convolutional neural networks (CNNs) in each branch, we propose a graph-matching based approach that attempts to take advantage of the view-invariant local relationships between landmark points visible in both aerial and ground images. We investigated the simpler problem of viewpoint localization: localizing a 360° ground image within a broader aerial view of the same area. We developed and described an unsupervised algorithm to solve this problem based on graph matching, which follows the steps of generating landmark graphs from a given set of salient points (or landmarks) in both views, extracting candidate locations from the aerial image and then calculating a probability distribution over them, according to a likelihood model specifically studied and proposed for this problem. We then proposed a method that can automatically find and extract the set of landmarks, which initially were manually annotated. To do so, we studied semantic segmentation of ground and aerial images. While segmentation is more advanced for ground images, we could not find suitable solutions for aerial images, so we created our own solution by gathering a dataset for semantic segmentation of aerial images and training a neural network for this purpose from scratch. By doing so, we are able to propose a fully automated algorithm for graph-based localization.

We studied our proposed algorithm for cases of manually and automatically extracted landmarks, and some possible extensions such as investigating if the algorithm could be applied to non-matching pairs of images and if post-processing techniques could be applied to the landmark graphs to generate more reliable answers. Some of the results were promising enough to indicate that the approach of graph-view matching could be employed in the problem of localization, if not as a standalone method, then perhaps as an auxiliary step to more traditional, deep learning based approaches. Though we could not reach results with a fully-automated algorithm as good as those obtained with manual annotation, we still believe that further improvements could help reduce this gap, as was achieved by our experiments with the removal of similar edges, for instance. Additionally, as a result of our early work, focusing on manual extraction of landmark points, we had one published paper [72]. According to our tests and experiments, the recommended pipeline to investigate this problem would be: use as landmark extraction method either the component or the grid approach (with a small stride); lastly, employ our proposed method for removal of edges connecting similar landmarks, applying a dimensionality reduction algorithm to reduce the dimension of the feature vectors.

From our research, we can see that most of the current research on the topic of image geolocalization poses the problem as a cross-view image matching problem: matching ground images to satellite images with known location. The matching problem, in its turn, is usually approached with Deep Learning techniques. Some approaches employ additional data, such as elevation maps [4] or drone imagery [76]: while using such data can be very useful, it is important to consider the availability of the required data, since using input that is not available worldwide could impair the applicability of the method to larger areas. Regarding the cross-view image matching problem, the vast majority of the research work employs the architecture of a Siamese network with two branches, one for the aerial images and one for the ground images. This works well and we can see that further improvements can be achieved by adding elements and customizations to this architecture (such as the addition of the NetVLAD layer [2] or capsule layers [66]). We can also conclude that, while some works in this area, specifically in the early stages, treated images as "black boxes" and did not use any problem-specific information, recent research [41, 88, 76] shows that developing solutions that combine classic Deep Learning methods with problem-specific information seems to be a more promising approach. Another strategy that seems worth exploring is to consider more realistic versions of the problem, such as taking temporal differences into account [51] or assuming that ground and aerial images are not always perfectly aligned [89]. Lastly, investigating alternative methods that do not employ Deep Learning, such as the algorithm we proposed, could prove fruitful: though we were not able to extend our approach to the cross-view image matching problem, we can still conclude that it shows potential to extract meaningful information, especially regarding the spatial relationships among significant objects. In this sense, studying and proposing solutions that can combine the well-established Deep Learning methods with alternative techniques is a line of research that can be pursued.

Finally, a natural extension of this work would be to extend it, possibly with the development of new techniques, for the *cross-view image matching* problem itself, studying if, and how, the generated graphs from different views can be compared and analyzed to extract meaningful insights. Also, since semantic segmentation is a key aspect of our proposed method, further studies are necessary to investigate alternatives to SegNet, since a better segmentation method could improve the overall performance of our algorithm. Lastly, thought it was not the focus of our work, the topic of graph neural networks (GNNs) [79], and particularly graph convolutional networks (GCNs) deserves to be studied: since our proposed method already models the problem as graphs, these techniques could prove very useful and provide new strategies to solve the localization problem.

Bibliography

[1] 2D Semantic Labeling Challenge. ISPRS (International Society for Photogrammetry and Remote Sensing). `http://www2.isprs.org/commissions/comm3/wg4/semantic-labeling.html`. Accessed: 2022-02-08.

[2] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5297–5307, 2016.

[3] Josep Aulinas, Yvan Petillot, Joaquim Salvi, and Xavier Lladó. The slam problem: a survey. *Artificial Intelligence Research and Development*, pages 363–371, 2008.

[4] Georges Baatz, Olivier Saurer, Kevin Köser, and Marc Pollefeys. Large scale visual geo-localization of images in mountainous terrain. In *European conference on computer vision*, pages 517–530. Springer, 2012.

[5] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017.

[6] Mayank Bansal and Kostas Daniilidis. Geometric urban geo-localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3978–3985, 2014.

[7] Mayank Bansal, Harpreet S Sawhney, Hui Cheng, and Kostas Daniilidis. Geo-localization of street views with aerial image databases. In *Proceedings of the 19th ACM international conference on Multimedia*, pages 1125–1128. ACM, 2011.

[8] BBC News. Anatomy of a killing. `https://www.youtube.com/watch?v=4G9S-eoLgX4`, 2018.

[9] Gabriel J Brostow, Julien Fauqueur, and Roberto Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, 30(2):88–97, 2009.

[10] Horst Bunke and Kaspar Riesen. Towards the unification of structural and statistical pattern recognition. *Pattern Recognition Letters*, 33(7):811–825, 2012.

[11] Tibério S Caetano, Julian J McAuley, Li Cheng, Quoc V Le, and Alex J Smola. Learning graph matching. *IEEE transactions on pattern analysis and machine intelligence*, 31(6):1048–1058, 2009.

[12] Sudong Cai, Yulan Guo, Salman Khan, Jiwei Hu, and Gongjian Wen. Ground-to-aerial image geo-localization with a hard exemplar reweighting triplet loss. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8391–8400, 2019.

[13] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):834–848, 2018.

[14] Lyujie Chen, Feng Liu, Yan Zhao, Wufan Wang, Xiaming Yuan, and Jihong Zhu. Valid: A comprehensive virtual aerial image dataset. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2009–2016, 2020.

[15] Ping-Rong Chen, Hsueh-Ming Hang, Sheng-Wei Chan, and Jing-Jhih Lin. Dsnet: An efficient cnn for road scene segmentation. *APSIPA Transactions on Signal and Information Processing*, 9, 2020.

[16] Shuxiao Chen, Xiangyu Wu, Mark W Mueller, and Koushil Sreenath. Real-time geo-localization using satellite imagery and topography for unmanned aerial vehicles. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2275–2281. IEEE, 2021.

[17] Zhe Chen, Yuchen Duan, Wenhai Wang, Junjun He, Tong Lu, Jifeng Dai, and Yu Qiao. Vision transformer adapter for dense predictions. *arXiv preprint arXiv:2205.08534*, 2022.

[18] Dragos Costea, Alina Marcu, Emil Slusanschi, and Marius Leordeanu. Roadmap generation using a multi-stage ensemble of deep neural networks with smoothing-based optimization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 220–224, 2018.

[19] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[20] Ilke Demir, Krzysztof Koperski, David Lindenbaum, Guan Pang, Jing Huang, Saikat Basu, Forest Hughes, Devis Tuia, and Ramesh Raskar. Deepglobe 2018: A challenge to parse the earth through satellite images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 172–181, 2018.

[21] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[22] Xueqing Deng, Yi Zhu, and Shawn Newsam. What is it like down there? generating dense ground-level views and image features from overhead imagery using conditional generative adversarial networks. In *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 43–52, 2018.

[23] Xueqing Deng, Yi Zhu, and Shawn Newsam. Using conditional generative adversarial networks to generate ground-level views from overhead imagery. *arXiv preprint arXiv:1902.06923*, 2019.

[24] Hao Ding, Songsong Wu, Hao Tang, Fei Wu, Guangwei Gao, and Xiao-Yuan Jing. Cross-view image synthesis with deformable convolution and attention mechanism. In *Chinese Conference on Pattern Recognition and Computer Vision (PRCV)*, pages 386–397. Springer, 2020.

[25] Karl Pearson F.R.S. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[26] Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé Iii, and Kate Crawford. Datasheets for datasets. *Communications of the ACM*, 64(12):86–92, 2021.

[27] Sean Gillies et al. Shapely: manipulation and analysis of geometric objects, 2007–.

[28] James Hays and Alexei A Efros. Im2gps: estimating geographic information from a single image. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[29] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[30] Sixing Hu, Mengdan Feng, Rang MH Nguyen, and Gim Hee Lee. Cvm-net: Cross-view matching network for image-based ground-to-aerial geo-localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7258–7267, 2018.

[31] Sixing Hu and Gim Hee Lee. Image-based geo-localization using satellite imagery. *International Journal of Computer Vision*, 128(5):1205–1219, 2020.

[32] Humans in the Loop. Semantic segmentation of aerial imagery. `https://www.kaggle.com/datasets/humansintheloop/semantic-segmentation-of-aerial-imagery`, 2020.

[33] Jinhyun Jang, Taeyong Song, and Kwanghoon Sohn. Semantic-aware network for aerial-to-ground image synthesis. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 3862–3866. IEEE, 2021.

[34] Pascal Kaiser, Jan Dirk Wegner, Aurélien Lucchi, Martin Jaggi, Thomas Hofmann, and Konrad Schindler. Learning aerial image segmentation from online maps. *IEEE Transactions on Geoscience and Remote Sensing*, 55(11):6054–6068, 2017.

[35] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[36] Tzu-Sheng Kuo, Keng-Sen Tseng, Jia-Wei Yan, Yen-Cheng Liu, and Yu-Chiang Frank Wang. Deep aggregation net for land cover classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 252–256, 2018.

[37] Feng Li, Hao Zhang, Shilong Liu, Lei Zhang, Lionel M Ni, Heung-Yeung Shum, et al. Mask dino: Towards a unified transformer-based framework for object detection and segmentation. *arXiv preprint arXiv:2206.02777*, 2022.

[38] Zuoyue Li, Jan Dirk Wegner, and Aurélien Lucchi. Topological map extraction from overhead images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1715–1724, 2019.

[39] Tsung-Yi Lin, Serge Belongie, and James Hays. Cross-view image geolocalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 891–898, 2013.

[40] Tsung-Yi Lin, Yin Cui, Serge Belongie, and James Hays. Learning deep representations for ground-to-aerial geolocalization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5007–5015, 2015.

[41] Liu Liu and Hongdong Li. Lending orientation to neural networks for cross-view geo-localization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5624–5633, 2019.

[42] David G Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[43] Xiaohu Lu, Zuoyue Li, Zhaopeng Cui, Martin R Oswald, Marc Pollefeys, and Rongjun Qin. Geometry-aware satellite-to-ground image synthesis for urban areas. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 859–867, 2020.

[44] Dimitrios Marmanis, Jan D Wegner, Silvano Galliani, Konrad Schindler, Mihai Datcu, and Uwe Stilla. Semantic segmentation of aerial images with an ensemble of cnss. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 2016*, 3:473–480, 2016.

[45] Christopher Mei, Gabe Sibley, and Paul Newman. Closing loops without places. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3738–3744. IEEE, 2010.

[46] Microsoft Bing. Aerial view - bing maps. `https://www.bing.com/maps/aerial`.

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

[48] Dan Popescu and Loretta Ichim. Aerial image segmentation by use of textural features. In *2016 20th international conference on system theory, control and computing (ICSTCC)*, pages 721–726. IEEE, 2016.

[49] Krishna Regmi and Ali Borji. Cross-view image synthesis using conditional gans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3501–3510, 2018.

[50] Krishna Regmi and Mubarak Shah. Bridging the domain gap for ground-to-aerial image matching. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 470–479, 2019.

[51] Royston Rodrigues and Masahiro Tani. Are these from the same place? seeing the unseen in cross-view image geo-localization. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3753–3761, 2021.

[52] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. In *Advances in neural information processing systems*, pages 3856–3866, 2017.

[53] Tim Sainburg, Leland McInnes, and Timothy Q Gentner. Parametric umap embeddings for representation and semisupervised learning. *Neural Computation*, 33(11):2881–2907, 2021.

[54] Noe Samano, Mengjie Zhou, and Andrew Calway. You are here: Geolocation by embedding maps and images. In *European Conference on Computer Vision*, pages 502–518. Springer, 2020.

[55] Qi Shan, Changchang Wu, Brian Curless, Yasutaka Furukawa, Carlos Hernandez, and Steven M Seitz. Accurate geo-registration by ground-to-aerial image matching. In *2014 2nd International Conference on 3D Vision*, volume 1, pages 525–532. IEEE, 2014.

[56] Yan Shen, Meng Luo, Yun Chen, Xiaotao Shao, Zhongli Wang, Xiaoli Hao, and Ya-Li Hou. Cross-view image translation based on local and global information guidance. *IEEE Access*, 9:12955–12967, 2021.

[57] Yujiao Shi, Dylan John Campbell, Xin Yu, and Hongdong Li. Geometry-guided street-view panorama synthesis from satellite imagery. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[58] Yujiao Shi, Liu Liu, Xin Yu, and Hongdong Li. Spatial-aware feature aggregation for image based cross-view geo-localization. In *Advances in Neural Information Processing Systems*, pages 10090–10100, 2019.

[59] Yujiao Shi, Xin Yu, Dylan Campbell, and Hongdong Li. Where am i looking at? joint location and orientation estimation by cross-view matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4064–4072, 2020.

[60] Yujiao Shi, Xin Yu, Liu Liu, Tong Zhang, and Hongdong Li. Optimal feature transport for cross-view image geo-localization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 11990–11997, 2020.

[61] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[62] Josef Sivic and Andrew Zisserman. Video google: a text retrieval approach to object matching in videos. In *Proceedings Ninth IEEE International Conference on Computer Vision*, pages 1470–1477 vol.2, Oct 2003.

[63] Elena Stumm, Christopher Mei, and Simon Lacroix. Building location models for visual place recognition. *The International Journal of Robotics Research*, 35(4):334–356, 2016.

[64] Elena Stumm, Christopher Mei, Simon Lacroix, and Margarita Chli. Location graphs for visual place recognition. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5475–5480. IEEE, 2015.

[65] B. Sun, C. Chen, Y. Zhu, and J. Jiang. Geocapsnet: Ground to aerial view image geo-localization using capsule network. In *2019 IEEE International Conference on Multimedia and Expo (ICME)*, pages 742–747, 2019.

[66] Bin Sun, Chen Chen, Yingying Zhu, and Jianmin Jiang. Geocapsnet: Aerial to ground view image geo-localization using capsule network. *arXiv preprint arXiv:1904.06281*, 2019.

[67] Takafumi Taketomi, Hideaki Uchiyama, and Sei Ikeda. Visual slam algorithms: A survey from 2010 to 2016. *IPSJ Transactions on Computer Vision and Applications*, 9(1):1–11, 2017.

[68] Hao Tang, Dan Xu, Nicu Sebe, Yanzhi Wang, Jason J Corso, and Yan Yan. Multi-channel attention selection gan with cascaded semantic guidance for cross-view image translation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2417–2426, 2019.

[69] Yicong Tian, Chen Chen, and Mubarak Shah. Cross-view image matching for geo-localization in urban environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3608–3616, 2017.

[70] Aysim Toker, Qunjie Zhou, Maxim Maximov, and Laura Leal-Taixé. Coming down to earth: Satellite-to-street view synthesis for geo-localization. In *Proceedings of the*

IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 6488–6497, 2021.

[71] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[72] Sebastiano Verde, Thiago Resek, Simone Milani, and Anderson Rocha. Ground-to-aerial viewpoint localization via landmark graphs matching. *IEEE Signal Processing Letters*, 27:1490–1494, 2020.

[73] S Vichy N Vishwanathan, Nicol N Schraudolph, Risi Kondor, and Karsten M Borgwardt. Graph kernels. *Journal of Machine Learning Research*, 11:1201–1242, 2010.

[74] Nam N Vo and James Hays. Localizing and orienting street views using overhead imagery. In *European conference on computer vision*, pages 494–509. Springer, 2016.

[75] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[76] Tingyu Wang, Zhedong Zheng, Chenggang Yan, Jiyong Zhang, Yaoqi Sun, Bolun Zheng, and Yi Yang. Each part matters: Local patterns facilitate cross-view geo-localization. *IEEE Transactions on Circuits and Systems for Video Technology*, 2021.

[77] Tobias Weyand, Ilya Kostrikov, and James Philbin. Planet-photo geolocation with convolutional neural networks. In *European Conference on Computer Vision*, pages 37–55. Springer, 2016.

[78] Scott Workman, Richard Souvenir, and Nathan Jacobs. Wide-area image geolocalization with aerial reference imagery. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3961–3969, 2015.

[79] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

[80] Hongji Yang, Xiufan Lu, and Yingying Zhu. Cross-view geo-localization with evolving transformer. *arXiv preprint arXiv:2107.00842*, 2021.

[81] Jiangye Yuan, Shaun S. Gleason, and Anil M. Cheriyadat. Systematic benchmarking of aerial image segmentation. *IEEE Geoscience and Remote Sensing Letters*, 10(6):1527–1531, 2013.

[82] Amir Roshan Zamir and Mubarak Shah. Accurate image localization based on google maps street view. In *European Conference on Computer Vision*, pages 255–268. Springer, 2010.

[83] Andrei Zanfir and Cristian Sminchisescu. Deep learning of graph matching. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2684–2693, 2018.

[84] Menghua Zhai, Zachary Bessinger, Scott Workman, and Nathan Jacobs. Predicting ground-level scene layout from aerial imagery. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 867–875, 2017.

[85] Gang Zhang, Tao Lei, Yi Cui, and Ping Jiang. A dual-path and lightweight convolutional neural network for high-resolution aerial image segmentation. *ISPRS International Journal of Geo-Information*, 8(12), 2019.

[86] Yifei Zhang, Olivier Morel, Marc Blanchon, Ralph Seulin, Mojdeh Rastgoo, and Désiré Sidibé. Exploration of deep learning-based multimodal fusion for semantic road scene segmentation. In *VISIGRAPP (5: VISAPP)*, pages 336–343, 2019.

[87] Bolei Zhou, Liu Liu, Aude Oliva, and Antonio Torralba. Recognizing city identity via attribute analysis of geo-tagged images. In *European conference on computer vision*, pages 519–534. Springer, 2014.

[88] Sijie Zhu, Taojiannan Yang, and Chen Chen. Revisiting street-to-aerial view image geo-localization and orientation estimation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 756–765, 2021.

[89] Sijie Zhu, Taojiannan Yang, and Chen Chen. Vigor: Cross-view image geo-localization beyond one-to-one retrieval. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3640–3649, 2021.

[90] Yi Zhu, Karan Sapra, Fitsum A Reda, Kevin J Shih, Shawn Newsam, Andrew Tao, and Bryan Catanzaro. Improving semantic segmentation via video propagation and label relaxation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8856–8865, 2019.

Appendix A

Datasheet for the Aerial Segmentation Dataset

This appendix contains the Datasheet for the Aerial Segmentation Dataset described in Section 4.2.1. This datasheet was created following the instructions provided by [26] and is also distributed with the dataset.

A.1 Motivation

For what purpose was the dataset created? Was there a specific task in mind? Was there a specific gap that needed to be filled? Please provide a description.
The dataset was created to enable research on the topic of semantic segmentation of aerial (satellite) images.

Who created the dataset (for example, which team, research group) and on behalf of which entity (for example, company, institution, organization)?
The dataset was created by Thiago Resek and prof. Anderson Rocha at the Institute of Computing of the University of Campinas.

Who funded the creation of the dataset? If there is an associated grant, please provide the name of the grantor and the grant name and number.
No funding.

Any other comments?
None.

A.2 Composition

What do the instances that comprise the dataset represent (for example, documents, photos, people, countries)? Are there multiple types of instances (for example, movies, users, and ratings; people and interactions between them; nodes and edges)? Please provide a description.

Instances represent aerial (satellite) images and their corresponding semantic segmentation in 13 classes: Building, Construction Area, Industrial Area, Parking, Highway, Footway, Dirt Road, Beach, Water, Park, Tree, Grass, Natural.

How many instances are there in total (of each type, if appropriate)?
There are 277 pairs (image and segmentation).

Does the dataset contain all possible instances or is it a sample (not necessarily random) of instances from a larger set? If the dataset is a sample, then what is the larger set? Is the sample representative of the larger set (for example, geographic coverage)? If so, please describe how this representativeness was validated/verified. If it is not representative of the larger set, please describe why not (for example, to cover a more diverse range of instances, because instances were withheld or unavailable).
The dataset is a sample of images collected at random locations centered in 9 urban areas (Berlin, Boston, Chicago, London, Manhattan, Paris, Rome, San Francisco and São Paulo) and 5 rural areas (Carver, Lincoln, Palmer, Waynesville, Zolfo Springs). The locations were manually selected so that the dataset would contain a mix of urban and rural areas. For the urban areas, big cities were manually selected in order to improve data quality. For rural areas, the selection was arbitrary.

What data does each instance consist of? "Raw" data (for example, unprocessed text or images) or features? In either case, please provide a description.
Satellite images consist of raw images. Segmentation images consist of RGB images colored according to the corresponding classes and the following code: Building: [255, 255, 255]; Highway: [50, 50, 255]; Grass: [0, 255, 0]; Natural: [0, 100, 0]; Construction: [133, 104, 102]; Industrial: [163, 66, 201]; Beach: [242, 209, 119]; Water: [119, 242, 242]; Footway: [128, 181, 255]; Parking: [168, 189, 173]; Park: [50, 205, 50]; Dirt Road: [232, 116, 32]; Tree: [69, 237, 114]; Unlabeled: [0, 0, 0].

Is there a label or target associated with each instance? If so, please provide a description.
The label is the corresponding segmentation (located in the 'segmented' folder, with the same name as the satellite image).

Is any information missing from individual instances? If so, please provide a description, explaining why this information is missing (for example, because it was unavailable). This does not include intentionally removed information, but might include, for example, redacted text.
No missing instances.

Are relationships between individual instances made explicit (for example, users' movie ratings, social network links)? If so, please describe how these relationships are made explicit.
None explicitly.

Are there recommended data splits (for example, training, development, validation, testing)? If so, please provide a description of these splits, explaining the rationale behind them.

None.

Are there any errors, sources of noise, or redundancies in the dataset? If so, please provide a description.

Segmentation data is obtained and processed from OpenStreetMap, which is a Wiki-like platform. Hence, the segmentation data is dependent on 3rd party user input, and hence prone to errors, particularly in rural areas.

Is the dataset self-contained, or does it link to or otherwise rely on external resources (for example, websites, tweets, other datasets)?

Self-contained.

Does the dataset contain data that might be considered confidential (for example, data that is protected by legal privilege or by doctor-patient confidentiality, data that includes the content of individuals' non-public communications)?

No.

Does the dataset contain data that, if viewed directly, might be offensive, insulting, threatening, or might otherwise cause anxiety?

No.

A.3 Collection process

How was the data associated with each instance acquired? Was the data directly observable (for example, raw text, movie ratings), reported by subjects (for example, survey responses), or indirectly inferred/derived from other data (for example, part-of-speech tags, model-based guesses for age or language)? If the data was reported by subjects or indirectly inferred/derived from other data, was the data validated/verified? If so, please describe how.

Satellite images were downloaded from Bing Maps using the API provided by the platform. Segmentation data was generated by downloading the data from OpenStreetMap in the same location as the satellite image (using their provided APIs) and processing the relevant information there to generate an image (drawing elements according to their descriptions). Segmentation images with low contrast or 70% or more of unlabeled area are discarded.

What mechanisms or procedures were used to collect the data (for example, hardware apparatuses or sensors, manual human curation, software programs, software APIs)? How were these mechanisms or procedures validated?

Bing and OpenStreetMaps APIs.

If the dataset is a sample from a larger set, what was the sampling strategy (for example, deterministic, probabilistic with specific sampling probabilities)?
Not applicable.

Who was involved in the data collection process (for example, students, crowdworkers, contractors) and how were they compensated (for example, how much were crowdworkers paid)?

Only the researchers were involved.

Over what timeframe was the data collected? Does this timeframe match the creation timeframe of the data associated with the instances (for example, recent crawl of old news articles)? If not, please describe the timeframe in which the data associated with the instances was created.

Data was collected from the corresponding platforms on July 2020, though there is no way of knowing when the platforms collected the data themselves.

Were any ethical review processes conducted (for example, by an institutional review board)? If so, please provide a description of these review processes, including the outcomes, as well as a link or other access point to any supporting documentation.

None.

A.4 Preprocessing/cleaning/labeling

Was any preprocessing/cleaning/labeling of the data done (for example, discretization or bucketing, tokenization, part-of-speech tagging, SIFT feature extraction, removal of instances, processing of missing values)? If so, please provide a description. If not, you may skip the remaining questions in this section.

Satellite images were downloaded from Bing in 256x256 resolution in a 5x5 grid around the interest point, then merged into a 1280x1280 image. Data from OpenStreetMap was processed using software developed by the authors to generate a visual representation. Images with low resolution or 70% or more of unlabeled area were discarded.

Was the "raw" data saved in addition to the preprocessed/cleaned/labeled data (for example, to support unanticipated future uses)? If so, please provide a link or other access point to the "raw" data.

Raw data was not saved.

Is the software that was used to preprocess/clean/label the data available? If so, please provide a link or other access point.

The software, developed by the authors, is currently not publicly available.

Any other comments?

None.

A.5 Uses

Has the dataset been used for any tasks already? If so, please provide a description.

The dataset was used to train a network for aerial segmentation as part of a Masters dissertation.

Is there a repository that links to any or all papers or systems that use the dataset? If so, please provide a link or other access point.

No.

What (other) tasks could the dataset be used for?

Mainly for semantic segmentation of aerial images, or subsets of this problem (e.g., segmentation of roads).

Is there anything about the composition of the dataset or the way it was collected and preprocessed/cleaned/labeled that might impact future uses? For example, is there anything that a dataset consumer might need to know to avoid uses that could result in unfair treatment of individuals or groups (for example, stereotyping, quality of service issues) or other risks or harms (for example, legal risks, financial harms)? If so, please provide a description. Is there anything a dataset consumer could do to mitigate these risks or harms?

There is minimal risk, since data was already public and available on other platforms.

Are there tasks for which the dataset should not be used? If so, please provide a description.

The data was collected solely for the aerial semantic segmentation problem.

Any other comments? None.

A.6 Distribution

Will the dataset be distributed to third parties outside of the entity (for example, company, institution, organization) on behalf of which the dataset was created? If so, please provide a description.

Yes, it will be available via request in Zenodo platform.

How will the dataset be distributed (for example, tarball on website, API, GitHub)? Does the dataset have a digital object identifier (DOI)?

Yes, the DOI is 10.5281/zenodo.4927665

When will the dataset be distributed?

It is already available.

Will the dataset be distributed under a copyright or other intellectual property (IP) license, and/or under applicable terms of use (ToU)? If so,

please describe this license and/ or ToU, and provide a link or other access point to, or otherwise reproduce, any relevant licensing terms or ToU, as well as any fees associated with these restrictions.
No. Specific licenses by Zenodo, Bing Maps or OpenStreetMap might apply.

Have any third parties imposed IP-based or other restrictions on the data associated with the instances? If so, please describe these restrictions, and provide a link or other access point to, or otherwise reproduce, any relevant licensing terms, as well as any fees associated with these restrictions.
No.

Do any export controls or other regulatory restrictions apply to the dataset or to individual instances? If so, please describe these restrictions, and provide a link or other access point to, or otherwise reproduce, any supporting documentation.
No.

Any other comments?
None.

A.7 Maintenance

Who will be supporting/hosting/maintaining the dataset?
The dataset will be hosted at Zenodo platform. Thiago Resek will support it.

How can the owner/curator/manager of the dataset be contacted (for example, email address)?
The author can be contacted at resek.thiago [@] gmail . com.

Is there an erratum? If so, please provide a link or other access point.
No.

Will the dataset be updated (for example, to correct labeling errors, add new instances, delete instances)? If so, please describe how often, by whom, and how updates will be communicated to dataset consumers (for example, mailing list, GitHub)?
There are no plans to update it yet.

If the dataset relates to people, are there applicable limits on the retention of the data associated with the instances (for example, were the individuals in question told that their data would be retained for a fixed period of time and then deleted)? If so, please describe these limits and explain how they will be enforced.
Not applicable.

Will older versions of the dataset continue to be supported/hosted/maintained? If so, please describe how. If not, please describe how its obsolescence will be communicated to dataset consumers.

Not applicable.

If others want to extend/augment/build on/contribute to the dataset, is there a mechanism for them to do so? If so, please provide a description. Will these contributions be validated/verified? If so, please describe how. If not, why not? Is there a process for communicating/distributing these contributions to dataset consumers? If so, please provide a description.

Please contact the author for contribution.

Any other comments?

None.