DOCTORAT BRETAGNE LOIRE / MATHSTIC

THÈSE DE DOCTORAT DE

L'UNIVERSITÉ DE BRETAGNE SUD

ÉCOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Informatique*

Par
Ahmed NASSAR

Learning to map street-side objects using multiple views

Thèse présentée et soutenue à « Vannes », le « 21/05/2021 »
Unité de recherche : UMR 6074 IRISA
Thèse N° : 595

Rapporteurs avant soutenance :

Christian Heipke	Full Professor, Leibniz Universität Hannover
Friedrich Fraundorfer	Associate Professor, TU Graz

Composition du Jury :

Président :	Elisa Fromont	Full Professor, Université de Rennes 1
Examinateurs :	Konrad Schindler	Full Professor, ETH Zurich
	Nathan Jacobs	Associate Professor, University of Kentucky
	Alexandre Boulch	Researcher, Valeo AI
Dir. de thèse :	Sébastien Lefèvre	Full Professor, Université Bretagne Sud
Co-dir. de thèse :	Jan D. Wegner	Associate Professor, ETH Zurich

Abstract

Creating inventories of street-side objects and their monitoring in cities is a labor-intensive and costly process. Field workers are known to conduct this process on-site to record properties about the object. These properties can be the location, species, height, and health of a tree as an example. To monitor cities, gathering such information on a large scale becomes challenging. With the abundance of imagery, adequate coverage of a city is achieved from different views provided by online mapping services (e.g., Google Maps and Street View, Mapillary). The availability of such imagery allows efficient creation and updating of inventories of street-side objects status by using computer vision methods such as object detection and multiple object tracking.

This thesis aims at detecting and geo-localizing street-side objects, especially trees and street signs, from multiple views. Solving the problem using an object detector, as with any problem solved with computer vision, brings up the usual problems of invariances such as occlusion, lighting, pose, viewpoint, and background. We rely on multiple views coupled with coarse pose information to solve these problems and for the benefit of getting more information about the object from these different views. Using multiple views brings another challenge, namely how to re-identify the objects in these different views to aggregate the information and not get duplicates of a particular object. Another major challenge is that the data sets acquired or used in our work contain imagery captured at a larger baseline, contrary to other data sets employed for person re-identification or self-driving and made of sequences of video frames. We propose several deep learning-based approaches to better detect, re-identify, and geo-localize objects and tackle these different challenges.

In our first proposed approach, we aimed at investigating if using soft geometric constraints coupled with image evidence would provide a better re-identification or matching accuracy of objects across different views to overcome our large baseline obstacle. This method relied on image crops of the objects from ground-level imagery and geometric metadata acquired from the image and then given as an input to a novel Siamese convolutional neural network-based architecture that matches the image crops. Having confirmed that infusing our model with soft geometric constraints proved beneficial, our

second approach aimed at achieving the same objective through an end-to-end model. The model takes as input a full image instead of crops, and our output is geo-localized bounding box detections tagged with identities across different views. To achieve such a task, we had to build a tool to annotate and create a data set of urban trees. Our final approach introduces another end-to-end model that relies on graph neural networks to improve flexibility and efficiency compared to the previous one. Also, in this approach, we include aerial imagery as another input for the first time.

For all three proposed approaches in this thesis, we perform extensive experiments on curated data sets to demonstrate the proposed systems' effectiveness.

Résumé

La création d'inventaires d'objets urbains et leur suivi est un processus coûteux en main-d'œuvre. Les agents de terrain effectuent généralement ce processus sur place pour collecter les propriétés sur ces objets. Ces propriétés peuvent être liées à l'emplacement, l'espèce, la hauteur et la santé d'un arbre par exemple. La lourdeur du processus de collecte de telles informations rend difficile l'étude des villes. Avec l'abondance d'images fournies par les services de cartographie en ligne (Google Maps et Street View, Mapillary, etc.), une couverture adéquate d'une ville peut être obtenue à partir de différents points de vue, tels que les vues aériennes ou au niveau du sol. La disponibilité de telles images permet la création et la mise à jour efficaces des inventaires de l'état des objets urbains en utilisant des méthodes de vision par ordinateur telles que la détection d'objets et le suivi d'objets multiples.

Cette thèse traite du problème de la détection et de la géolocalisation des objets urbains, en particulier les arbres et les panneaux de signalisation à partir de vues multiples. La résolution du problème à l'aide d'un détecteur d'objet, comme pour tout problème résolu par la vision par ordinateur, soulève les problèmes habituels d'invariance à l'occlusion, l'éclairage, la pose, le point de vue et l'arrière-plan. Nous nous appuyons sur plusieurs vues pour résoudre ces problèmes, ce qui nous permet également d'obtenir plus d'informations sur les objets à partir de ces différentes vues. L'utilisation de plusieurs vues soulève un autre défi, à savoir comment ré-identifier les objets dans ces différentes vues pour agréger les informations et ne pas obtenir les doublons d'un objet particulier. Un autre défi majeur est que les données acquises ou utilisées dans notre travail contiennent des images extraites d'une base de référence plus large, contrairement aux données utilisées pour la ré-identification des personnes ou la conduite autonome qui consistent en des séquences d'images vidéo. Pour relever ces différents défis, nous proposons plusieurs approches basées sur l'apprentissage profond pour mieux détecter, ré-identifier et géolocaliser les objets.

Dans notre première approche, nous visions à déterminer si l'utilisation de contraintes géométriques douces couplées à des informations issues des images pouvait fournir une meilleure ré-identification ou une meilleure précision de correspondance des objets à

travers différentes vues. Cette méthode reposait sur des extraits visuels des objets à partir d'imagerie acquise au niveau du sol ainsi que sur des métadonnées géométriques acquises extraites des images, puis données comme entrée dans une nouvelle architecture de réseau de neurones convolutif siamois adapté aux extraits d'image. Ayant confirmé qu'utiliser des contraintes géométriques douces s'avérait bénéfique à notre modèle, notre seconde approche visait à atteindre le même objectif grâce à un modèle appris de bout en bout. Le modèle prennait alors en entrée une image entière au lieu des extraits, et notre sortie consistait en des détections de boîtes englobantes géolocalisées et ré-identifiées sur les différentes vues. Pour y parvenir, nous avons dû construire un outil pour annoter et créer un ensemble de données (arbres urbains). Notre dernière approche exploite finalement les réseaux de neurones par graphe pour offrir davantage de flexibilité et d'efficacité par rapport à l'approche précédente. De plus, dans cette approche, nous incluons pour la première fois l'imagerie aérienne comme une autre entrée du modèle.

Pour les trois approches proposées dans cette thèse, nous avons constitué des jeux de données et réalisé des expériences approfondies permettant de démontrer l'efficacité des systèmes proposés.

ACKNOWLEDGEMENT

My gratitude goes to everyone who has contributed to the completion of this dissertation, and for my journey during my doctoral studies. Whether it had to do with discussing scientific, research ideas, or dealing with administrative hurdles. I have been lucky to have been supported by you along the way.

First and foremost, I would like to thank **Prof. Sébastien Lefèvre** for giving me the opportunity to pursue my doctoral studies under his supervision at Université Bretagne Sud. He has supported me during my Master's internship and made circumstances easier to pursue my doctoral studies and internships with immense flexibility. He has given me the freedom to pursue research ideas while providing critical remarks and constructive criticism to help me shape my ideas and keep me focused on my goal. Also, I am very appreciative of his effort in making the collaboration happen between IRISA and PRS for my studies and the many letters he had to write during my continuous relocation between both countries. My thanks and appreciation in terms of supervision ought to be shared also with **Dr. Jan Wegner**. The RegisTree project was his idea, which was a solid ground for me to build upon. He has helped me improve my thinking, pointing me in the right direction, and exploring promising directions or ideas. I am grateful for his recommendations in order to help me secure an internship to benefit my future after my doctoral studies. I can't thank Jan enough for his continuous motivation, high spirit, and patience with me, it really made a difference to help me produce my results within a very tight schedule. The process of pursuing a Ph.D. has helped me to learn a lot and to develop my research and thinking process. Sébastien and Jan have given me the freedom and trust to produce this work, I consider myself lucky to have had such supervisors. It has been an utmost pleasure working with you both!

I would like to also thank **Prof. Konrad Schindler** for hosting me in his group, making the collaboration possible, and providing the resources needed. I would like to thank him for the environment of openness and for being thoughtful and polite to all of his students. Although I am not officially a student in his group, he has certainly made me feel so by including me in all events and manners.

My gratitude also goes to the Ph.D. committee members, who took the time to care-

fully review my thesis and were very flexible during this pandemic. It's an honor to have your names on my thesis.

Ms. Monique Berger Lande also deserves warm appreciation. She has managed my administrative issues and complications with great efficiency making sure my stays and moving process smooth many times. Also socially, she has organized many group activities, and winter retreats which I will miss!

I'll relish the time spent at both IRISA and PRS due to the coffee-breaks, lunches, trips, parties, foosball, and football matches. A special thank goes to my friends and colleagues **Nico Lang**, **Andres Rodriguez**, **Riccardo de Lutio**, **Stefano D'Aronco**, **Caglayan Tuna**, and **Minh-Tan Pham** for our various interesting discussions, activities, and help that has made this experience great. I would also like to thank the rest of my colleagues who have contributed to very welcome environments at both labs: **Bharath**, **Binbin**, **Cenek**, **Corinne**, **Jérôme**, **Katrin**, **Manu**, **Marc**, **Mikhail**, **Nadine**, **Nicolas**, **Nikolai**, **Ozgur**, **Priyanka**, **Titouan**, and **Yujia**.

My acknowledgments are extended to all my friends and especially **Fouly**, **Wafik**, and **Paula** who have suffered from my awful scheduling and procrastination.

Last but not least, I would like to thank my parents **Magda** and **Sami** for their continuous support, to whom I owe everything. Special thanks to my dad for proofreading and correcting the final version of the manuscript. I would also like to thank my aunt **Seham** for her encouragement during my academic endeavor. Moreover, I would like to thank my siblings **Aseel**, **Mohamed**, and **Morid** for their assistance and support through the years.

Thank you all very much!

Table of Contents

1 Introduction **17**

1.1 Motivation . . . 17
1.2 Research Aims . . . 19
1.3 Related work . . . 20
1.3.1 A review of object detection . . . 20
1.3.2 A review of urban object detection . . . 22
1.3.3 A review of multi-object tracking . . . 23
1.3.4 A review of pose estimation . . . 24
1.4 Methodology and Contributions . . . 25
1.4.1 Methodology in Chapter 3 . . . 25
1.4.2 Methodology in Chapter 4 . . . 26
1.4.3 Methodology in Chapter 5 . . . 26
1.5 Technical Relation of the Chapters . . . 27
1.6 Relevance for Science and Society . . . 28
1.6.1 Remote Sensing - Geographic Information Systems . . . 28
1.6.2 Computer Vision - HD Maps and Self-driving . . . 28
1.6.3 Publicly available research . . . 29

2 Background **31**

2.1 Deep Learning . . . 31
2.1.1 Neural Networks . . . 31
2.1.2 Backpropagation . . . 33
2.1.3 Deep Neural Network Challenges and Motivations . . . 35
2.2 Convolutional Neural Networks . . . 35
2.2.1 Convolutional Layer . . . 35
2.2.2 Dense Layers . . . 37
2.2.3 Pooling Layers . . . 37
2.2.4 CNN Architectures & Applications . . . 37
2.3 Geometric Deep Learning . . . 41

2.3.1 Spectral Graph Convolutional Networks 42
2.3.2 Spatial Graph Convolutional Networks 43

3 Multi-View Instance Matching with Learned Geometric Soft-Constraints 45
3.1 Abstract 45
3.2 Introduction 46
3.3 Related Work 47
3.4 Instance Matching with Soft Geometric Constraints 49
3.4.1 Model Architecture 51
3.4.2 Loss Functions 53
3.5 Experiments 54
3.5.1 Datasets 54
3.5.2 Evaluation Strategy 58
3.5.3 Does Geometry Help? 59
3.5.4 Results 59
3.6 Conclusions 62

4 Simultaneous multi-view instance detection with learned geometric soft-constraints 63
4.1 Abstract 63
4.2 Introduction 64
4.3 Related Work 65
4.4 Multi-view detection and instance re-identification 67
4.4.1 Multi-view object detection 68
4.4.2 Loss function 73
4.5 Experiments 74
4.5.1 Data sets 74
4.5.2 Multi-view object annotation tool 77
4.5.3 Detection 77
4.5.4 Re-identification with pose information 78
4.5.5 Geo-localization 79
4.6 Conclusion 79

5 GeoGraphV2: Graph-Based Aerial & Street View Multi-view Object Detection with Geometric Cues End-to-End 85
5.1 Abstract . . . 85
5.2 Introduction . . . 86
5.3 Related Work . . . 89
5.4 Method . . . 92
5.4.1 Object Detection . . . 94
5.4.2 Object Re-identification . . . 95
5.4.3 Geo-localization . . . 96
5.5 Experiments . . . 98
5.5.1 Datasets . . . 98
5.5.2 Implementation Details . . . 100
5.5.3 Object Detection & Re-identification . . . 100
5.5.4 Geo-localization . . . 103
5.5.5 Ablation Studies . . . 104
5.6 Conclusion . . . 106

6 Conclusion 109
6.1 Discussion of Contributions . . . 109
6.1.1 Multi-View Instance Matching with Learned Geometric Soft-Constraints 109
6.1.2 Simultaneous Multi-View Instance Detection with Learned Geometric Soft-Constraints . . . 110
6.1.3 GeoGraphV2: Graph-Based Aerial & Street View Multi-View Object Detection with Geometric Cues End-to-End . . . 110
6.2 Discussion of Limitations . . . 111
6.2.1 Ground-truth . . . 111
6.2.2 Processing Speed . . . 111
6.2.3 Improvement of aerial detections . . . 112
6.3 Outlook . . . 112
6.3.1 Future Directions. . . . 112

Bibliography 115

List of Publications

— AS Nassar, S Lefèvre, JD Wegner: **Multi-View Instance Matching with Learned Geometric Soft-Constraints**. – *ISPRS International Journal of Geo-Information 9 (11)*, 2020, 687

— AS Nassar, S D'Aronco, S Lefèvre, JD Wegner: **GeoGraph: Graph-Based Multi-view Object Detection with Geometric Cues End-to-End**. – *European Conference on Computer Vision (ECCV)*, 2020, 488-504

— AS Nassar, N Lang, S Lefevre, JD Wegner: **Learning geometric soft constraints for multi-view instance matching across street-level panoramas**. – *Joint Urban Remote Sensing Event (JURSE)*, 2019, 1-4

— AS Nassar, S Lefèvre, JD Wegner: **Simultaneous multi-view instance detection with learned geometric soft-constraints**. – *IEEE International Conference on Computer Vision (ICCV)*, 2019, 6559-6568

Not related to the PhD thesis:

— N Livathinos, C Berrospi, M Lysak, V Kuropiatnyk, AS Nassar, A Carvalho, M Dolfi, C Auer, K Dinkla, P Staar: **Robust PDF Document Conversion Using Recurrent Neural Networks**. – *Thirty-Third Annual Conference on Innovative Applications of Artificial Intelligence (IAAI-21)*, 2021

— AS Nassar, AH Montasser, N Abdelbaki: **A deep CNN-based framework for enhanced aerial imagery registration with applications to UAV geolocalization**. – *International Conference on Advanced Intelligent Systems and Informatics*, 2018, 1513-1523

— AS Nassar, AH Montasser, N Abdelbaki: **A survey on smart cities' IoT**. – *International Conference on Advanced Intelligent Systems and Informatics*, 2017

— S Lefèvre, D Tuia, JD Wegner, T Produit, AS Nassar: **Toward seamless multi-view scene analysis from satellite to street level**. – *Proceedings of the IEEE 105 (10)*, 2017, 1884-1899

List of Tables

3.1 Matching Results. 60

4.1 Re-identification results. 78
4.2 Detection and Re-identification results. 82
4.3 Chapter 4: Geo-localization results. 83

5.1 GeoGraphV2 Results for detection, re-identification, and geo-localization. . 105
5.2 Quantitative assessment of pipeline. 106

List of Figures

2.1 The biological structure of a neuron . . . 32
2.2 Representations of common activation functions. . . . 32
2.3 Illustration of an MLP. . . . 33
2.4 A convolutional layer operation. . . . 36
2.5 Max pooling operation . . . 37
2.6 Architecture of AlexNet. . . . 38
2.7 ResNet "skip-connections". . . . 39
2.8 Siamese Network Architecture. . . . 40
2.9 Triplet Network Architecture . . . 41
2.10 Irregular data structures . . . 42
2.11 Graph Pooling. . . . 43
2.12 SplineCNN aggregation. . . . 44

3.1 Chapter 3 problem setup. . . . 47
3.2 The tree instance matching problem. . . . 48
3.3 Chapter 3: The overall pipeline which encompasses our proposed model. . . . 50
3.4 Diagram showing the overall network architecture . . . 52
3.5 Pasadena dataset panormas. . . . 56
3.6 Mapillary dataset examples. . . . 57
3.7 Mapillary traffic sign dataset example. . . . 58
3.8 Pasadena matching examples. . . . 61
3.9 Mapillary matching examples. . . . 62

4.1 Chapter 4: Pipeline. . . . 66
4.2 Chapter 4: Problem setup. . . . 68
4.3 Tree instance re-identification problem. . . . 69
4.4 Chapter 4: Network design. . . . 71
4.5 Illustration of projection functions. . . . 72
4.6 Consecutive frames from Mapillary data set. . . . 75

4.7 Our annotation tool labeling multi-view panoramas from GSV. 76
4.8 Detection and Re-identification results on Mapillary. 80
4.9 Geolocalization results visualized. 81

5.1 Problem setup. 87
5.2 Tree re-identification problem illustration. 89
5.3 Architecture of GeoGraphV2 93
5.4 Subset of the Pasadena dataset to highlight challenges. 98
5.5 Subset of the Mapillary dataset to highlight challenges. 100
5.6 Sample results obtained on the Pasadena dataset. 101
5.7 Sample results obtained on the Mapillary dataset. 102
5.8 Visualization of geo-localization results on Pasadena. 104

Chapter 1

Introduction

1.1 Motivation

As cities are growing and becoming denser, automated cataloging of city objects is getting ever more important. Creating or updating inventories of street-side objects (e.g., trees, street signs) in urban areas is a labor-intensive and expensive process in practice today that is mainly conducted via in situ surveys of field crews. These catalogs can be employed to produce HD Maps for self-driving efforts or serve as data layers to give residents, businesses, and city planners insights for decision-making.

One of the unfortunate consequences of cities development and expansion is the increase of impervious surfaces (e.g., concrete and buildings) which absorbs heat. Such expansions largely contribute to the "urban heat island" effect. The urban heat island effect is a term that refers to urban areas that have higher temperatures and pollution in contrast to rural areas. To combat this effect, vegetation such as trees helps in multiple ways. Surfaces shaded by trees and vegetation can decrease the temperature compared to unshaded materials by 11–25°C [1] and can lead up to a 1–5°C reduction of peak summer temperatures [2], [3].

In the U.S.A. alone, urban forests contain approximately 3.8 billion trees [4]. One of our main interests in this work is public street trees. Although they constitute a small segment out of urban forests, they significantly impact cities or urban areas. California streets hold approximately 7.2 million street trees [5]. Surveying such many trees, including health monitoring, maintaining species diversity, and general maintenance, adds a burden to a city's tree budget. Furthermore, many cities do not have an inventory of their trees, and updating them is a huge endeavor that can be automated. Using aerial imagery and ground-level imagery of cities allows cities' coverage in a much faster and more economical approach.

Regardeless of how laborious creating public tree inventories is many efforts exist.

Tree Canopy Lab [1], Google also figures out where every tree is in a city by relying only on aerial imagery and presenting that information on an interactive map. OpenTreeMap [2] intends to build a publicly available tree inventory for the greater Los Angeles area by relying on professional arborists to update and combine existing tree inventories.

The ever-growing volume and coverage of crowdsourced geo-tagged images provide an effortless variance of perspectives. Apart from social media platforms (e.g., Twitter, Instagram, Flickr) that enable the attachment of geocoordinate uploaded or shared images, other platforms exist to provide a solution in which various users upload geotagged imagery. Mapillary [3] contains 500 million geo-tagged images that are processed using computer vision techniques to provide data layers of object detections (e.g., cars, crosswalks, trash cans) and scene classes (e.g., parking lot, roundabout, gas stations), these images being uploaded through an app by users. OpenStreetCam [4] is another platform part of the OpenStreetMap foundation that lets users upload open street-level imagery. 360cities [5] provides a stock of high-resolution geo-tagged panoramic photos.

Given a set of images from different views, humans can match corresponding objects or items in a given scene. Such an ability is possible since they can infer many details using visual cues or landmarks in the vicinity. They can also keep track of which objects are in the scene, their poses, and their spatial layout. Through the eyes, the human brain is able from multiple views to find a particular "street sign" if prompted by inducing it is in front of a storefront, beside a fire hydrant, on the sidewalk, and by what type it is. The eyes would dart in other views to find the street sign while gathering more context and clues from other views to identify the object. They would not look mid-air for it.

These various data sources offer different views (satellite, multiple ground-level perspectives) that help detecting 3D objects. Additionally, they offer us more information to be able to much more precisely detect an object, localize, and classify it compared to a single view approach.

1. https://insights.sustainability.google/labs/treecanopy
2. https://www.opentreemap.org
3. http://www.mappillary.com
4. https://openstreetcam.org/map/
5. https://www.360cities.net

1.2 Research Aims

In this thesis, we investigate using RGB images and geometry to recognize and geo-localize street-side objects across multiple views and modalities by using deep learning and computer vision techniques with the challenges and opportunities mentioned previously. In this context, this thesis explores the following overriding research questions:

— The project explores the usage of multi-modal and multi-view images of a city to detect street side objects. These street side objects are mainly trees and street signs. The aim is to exploit the different perspectives and information that the aggregation of these data sources can provide to achieve a more precise inventory. Can photogrammetry and computer vision techniques prove beneficial despite challenges this aggregation of different data sources introduce? Perhaps the biggest challenge and difference between our work, and other approaches is that their datasets are composed of consecutive frames (images) from cameras placed on moving vehicles. However in our work, our images are captured with a larger baseline or crowdsourced, therefore severe changes exist between them.

— Another question we explore is whether the metadata accompanying the images can be useful to compensate in our setup. The metadata includes information such as the camera's geo-coordinates and heading. We refer to this metadata in this work as "geometric features" or "coarse pose estimations". These geometric features can be used to compensate not having the cameras intrinsic or extrinsic values. Another major challenge with re-identifying urban street trees on the pavement is that they are ordinarily the same species, planted at the same time, and within close distance, which makes it hard to distinguish between them even for humans. Therefore, one of our fundamental questions is are these geometric features, coupled with appearance features helpful in detecting and re-identifying objects?

— We are determined to rely on RGB images solely compared to other methods that combine, e.g., point clouds. Therefore in our work, we strive only to use RGB images from ground-level, or aerial imagery in particular. Smartphones exist widely even in Global South regions, making the minimum requirement to capture imagery of a city possible without having to deal with LiDAR equipment per se. Can RGB images alone prove to be enough to accomplish the tasks of detection, re-identification, and geo-localization of an object?

— Finally, to put it all together, our main question revolves around: If using the abundance of different data sources and geometric features is enough to train a practical deep learning model that is end-to-end and can perform an automated street side object mapping process? This machine learning motivated approach intends to generalize well on different imagery types without bias and be practical in different cities worldwide.

With the dissertation goals defined and put into context, it is apparent that the scope of the research aims and questions realm beyond only remote sensing applications, geoinformatics, and photogrammetry but profoundly also in computer vision. With the camera advancements, an increase of content sharing platforms, and the updated satellite/aerial imagery continuously, it is affordable to provide coverage of an area. This once again helps with our primary aim which is automating street-side objects' detection to create GIS data layers for cities, and with our secondary aim of creating HD maps for self-driving efforts.

1.3 Related work

The proposed methods in this thesis touch upon many different research topics in computer vision: pose estimation, urban object detection, multi-object tracking, object geo-localization, and instance re-identification. We perform a full review of the key ideas related to the literature in the following sections and elaborate more in the core chapters as our published works are recent.

1.3.1 A review of object detection

Since our work is based on object detector architectures, we review several of the most popular approaches and those considered state of the art. The review is split into a brief overview of non-CNN "traditional" approaches and CNN based approaches.

Non-CNN approaches. One of the earliest works in visual object class recognition was the "Bag of words" [6] approach which was superior for a decade (2000-2010). In this method, an object represents a collection of "visual words", which later become known as local invariant features. These features are key points in the scene that can be detected in different images where invariances such as illumination, viewpoint, and scale might occur. Generalized Hough Voting [7] is another method that is inspired by BoW, and attempts to

group together features that belong to the same object instance. This grouping is achieved by a voting procedure in a test image's features decide where is the object and it's scale. Afterwards, by finding the mode in the voting space, a bounding box is achieved.

Another significant object detection approach is the group of slide window detectors. Slide-window detectors' design is to pass a sliding window (box) over the image with various scales. There are two sub-categories of sliding window methods rigid models and part-based models. Rigid sliding methods such as the Histogram of Oriented Gradients (HOG) [8] seeks to classify objects from the background by calculating an image gradient at many areas in the image. Part based models such as the Deformable Parts Model [9] search for parts of the object within the sliding window. In this model, a HOG template is learned over training data, and each object part number is defined manually. After being detected using the sliding window and HOG template, the object parts are then matched using some dynamic programming techniques.

CNN-based approaches. There are numerous CNN-based object detectors. However, in this section, we mention the most prominent frameworks. Perhaps the most famous object detector is Faster R-CNN [10], which resulted from several iterations of work [11], [12] and has been the inspiration for other object detectors. In Faster R-CNN, a CNN provides a convolutional feature map from an image. Another separate network predicts the region proposals. A Region of Interest (RoI) pooling layer reshapes the predicted region proposals to classify the image inside the proposed region using a set of predefined boxes of different scales and aspect ratios called "anchors". Afterward, the offset values prediction from the anchors for the bounding boxes occurs. Mask R-CNN [13] is an extension to Faster R-CNN by adding a branch that achieves instance segmentation by adding a few more convolutional layers to create the mask. YOLO [14] (You Only Look Once) is another popular framework with many versions that are well known to run in real-time. YOLO divides an image into a specifically sized grid, with predefined bounding boxes and their confidences. The confidence indicates the accuracy of the bounding box containing an object. The network predicts for each bounding box a classification score too. SSD [15] (Single Shot Detector) achieved a good middle ground between speed and accuracy. Typically, an image passes through a convolutional neural network to create a feature map. Afterward, a convolutional kernel goes over the feature map to predict the bounding boxes and classification probability. Similar to Faster R-CNN, SSD uses anchor boxes. YOLO and SSD are one-shot object detectors. They detect an object at once without intermediate stages (e.g., region proposal networks), making them fast and main-

taining a good accuracy, but not as good as two-stage detectors (e.g., Faster R-CNN). SSD and YOLO's problem is that these methods have a problem of class imbalance during the training. This class imbalance is due to many negative samples, as the methods consider the whole grid of the feature map. Retinanet [16] proposes a new loss from a modified cross-entropy loss, to focus on a hard set of examples and ignore the numerous easy negatives during training. In an effort to be more efficient in terms of model parameters, EfficientDet [17] introduces an object detector with different backbones. The backbone networks named "EfficientNet" [18] is a family of 7 networks that achieved SOTA accuracy with 10x efficiency. EfficientDet also introduces "BiFPN", a weighted bi-directional feature pyramid network that learns weights to learn input features' significance. DETR [19] (End-to-End Object Detection with Transformers) is a different approach to object detection compared to the methods mentioned above. DETR uses a CNN backbone to produce a 2D representation of the input image. Afterward, the representation is flattened and supplemented with a positional encoding and then passed to a transformer encoder. A transformer decoder tries to predict a fixed number of detections called "object queries". Typically this fixed number should be larger than the number of detections sought in an image. The transformer decoder then produces an output embedding to a feed-forward network (FFN) that predicts the detection class and bounding box.

1.3.2 A review of urban object detection

This section is a review of approaches in the literature that attempt at urban object detection with geo-localization. The review is split based on the type of data used in the methods.

Image sequences. Urban object detection has been addressed by a large body of literature related to autonomous driving. Many existing public benchmarks (e.g., KITTI [20], CityScapes [21], or Mapillary [22]) are captured as multiview image sequences with the smallest change in viewpoint. Such a setup is particularly useful to detect nonstatic objects such as cars and pedestrians. Therefore, such data sets and benchmarks are not primarily concerned about re-identifying static objects as it does not fit within the autonomous vehicle's mission. [23] detects road objects from sequences and maps them using semantic segmentation and a topological binary tree. In [24], a system of 5-DoF (Depth of Field) object pose estimation coupled with multi-object tracking produces the geo-localization of static objects in image sequences. Our setup is composed of coarse relative pose information, and the viewpoints between the images are large, which makes the view

noticeably different, which makes matching corresponding objects a more troublesome task.

Street-view Images. Similar to our work, other works aim at detecting street-side objects from ground-level imagery. In previous work on the RegisTree project [25], [26], a method combining GSV (Google Street View) panoramas and aerial images in a multi-stage workflow was proposed. The trees are detected in all different views, then the detections are projected into each view with their detection scores. Afterward, a conditional random field combines all image evidence with other learned priors. Similarly, [27] detects and geo-locates poles in GSV panoramas using a pipeline that segments the object, estimates depth, then uses an MRF model and clustering techniques to obtain the location. [28] likewise detects and geolocalizes poles in GSV by employing a brute-force line-of-bearing. [29] presents a method for updating inventories related to telecom assets by intersecting 3D rays. The methods mentioned previously all propose in common multi-stage hierarchical workflows in which the pose and appearance features are processed independently, unlike our approaches, which treat them simultaneously.

1.3.3 A review of multi-object tracking

Many similarities and common ground between the challenges faced in our problem and the ones of Multiple Object Tracking (MOT) and person re-identification (Re-ID) exist. MOT relates to the task of tracking multiple objects in an image sequence. The most prominent MOT task in literature is person re-identification. Person re-identification aims at tracking different identities of persons, which is commonly a surveillance task. Perhaps the most significant difference is that in MOT, the objects are nonstatic while the cameras are fixed, which is the opposite scenario to our problem.

Recent deep learning solutions have been extensively used in MOT. Siamese CNNs [30] have been widely used to measure similarities between image patches. Learning CNN features using a siamese CNN and temporally constrained metric simultaneously to train a tracklet affinity model were presented in [31]. In [32], the authors employ a method that uses imagery and optical flow maps to achieve MOT. Other works [33] uses a combination of a CNN network, and recurrent neural networks (RNN) to match pairs of detections. [34] presents a center loss function that tries to minimize the distance between candidate boxes in feature space. [35] suggests a framework that uses jointly visual semantic information and spatial-temporal information with a Logistic Smoothing metric. A work similar, despite its application, to ours is [36]. The authors tackle the

problem of person re-identification by devising the problem into a graph setup. The graph contains nodes of CNN features generated from image crops of person identities. Then a message passing, Graph Neural Network (GNN), is used to propagate the node features. The resulting node features' edges with other nodes are then classified to assign if the nodes correspond to the same person identity. However, most of the works mentioned above, use a single camera setup when, in this study, we must re-identify objects from within different views.

1.3.4 A review of pose estimation

Deep learning has made great strides in predicting camera poses using CNNs. A pose estimation network uses an RGB image as input and yields an object's position and orientation in the camera space or world without the need for any depth, stereo vision, or point cloud input.

Camera Pose Estimation PoseNet [37] paved the way for camera pose estimation using a deep end-to-end architecture that regresses 6-DoF from a single RGB image. This was achieved by truncating a GoogleNet architecture [38] and replacing the softmax layer with fully connected layers to output the pose. Another work [39] also uses a CNN to estimate a relative pose between two cameras and provides the relative rotation and translation as output by using a Siamese network architecture. [40] introduces a real-time camera pose estimation framework for Simultaneous Localization and Mapping (SLAM) systems. This work is the first "hybrid" pipeline as it manages both bundle adjustment (BA) and motion averaging to estimate and update the camera poses on-the-fly.

Object Pose Estimation Early deep learning approaches [41]–[44] that predicted estimations of 6-DoF poses of objects showed an enhanced performance in dealing with occlusion using synthetic training data. SSD-6D [41] proposed a method that builds upon the SSD object detector [15] to predict 6-DoF poses using synthetic training data. They decomposed the 6D pose into discrete viewpoints and approached the rotation estimation problem as a classification problem. In [42] the authors propose a method called "BB8" formed from a cascade of CNNs to achieve the object pose estimation task. BB8 first localizes objects using a segmentation network, then using a PnP algorithm, it finds the correspondences between 2D coordinates and 3D bounding boxes. [44] proposes PoseCNN, a method which estimates 3D rotation and 3D translation. The 3D translation is estimated by first localization, then predicting its distance from the camera. Afterwards, the 3D rotation is regressed to a quaternion representation. Another work [45] aims at

estimating human pose and action recognition by a multitask CNN that combines visual features, probability maps, and estimated poses. In [46], a human hand's appearance in any viewpoint can be predicted by coupling pose with image content. A recent method [47] introduces a real-time RGB-based method for 6D pose estimation that does not require pose-annotated training data and generalizes well to different sensors by learning an implicit representation in a latent space. However, in our work, we rely on public imagery that excludes fine-grained camera pose information.

1.4 Methodology and Contributions

This thesis is written as a cumulative dissertation. The main technical "core" chapters (Chapters 3, 4, and 5) are added as originally published, only with minor typographical and formatting corrections. They are presented with this introductory chapter, followed by a background chapter (Chapter 2) and a conclusion and outlook chapter (Chapter 6).

The core chapters are arranged in order of our exploration to answer our research aims and questions. To build our end-to-end deep learning network that automates street-side geo-localization pipeline, we experiment incrementally part by part by employing different computer vision techniques to achieve this.

1.4.1 Methodology in Chapter 3

Chapter 3 experiments with fusing image evidence and soft geometric constraints to verify whether geometric evidence helps while trying to match multi-views of cropped images of objects. As explained before, to implement the end-to-end pipeline, it is addressed by breaking it down into separate components. Therefore in this first approach, it is assumed that the objects in the scene are detected to investigate the process of matching different image crops. In these experiments, the multi-views are from ground-level imagery only.

Contributions:

1. A novel siamese based architecture that jointly learns distributions of appearance-based features and geometric cues to match image crops of different instances of the same object from multiple views.
2. Experiment with various siamese networks architectures for comparison on data sets pertaining to trees and street-signs.

1.4.2 Methodology in Chapter 4

Having concluded from chapter 3 that geometry coupled with appearance indeed benefits our goal of matching crops of multiple views. Detecting objects and re-identifying them in multiple views can be challenging due to the changes in viewpoint, lighting conditions, high similarity to the neighboring objects, and the variability in scale. The approach is based on a joint learning task that combines object detection, instance re-identification, and geo-localization. This task is reinforced by using soft geometric constraints to produce a learnable end-to-end network.

Contributions:

1. As to our knowledge, there is no available data set of trees labeled as instances with IDs. Therefore we built a new interactive, semi-supervised multi-view instance labeling tool that annotates Google Street View panoramas to create the data set.
2. Using the new annotation tool, we introduce a new object instance re-identification data set that contains 6,020 different tree identities, with a total of 25,061 bounding boxes of trees.
3. A novel multi-view object instance detection and re-identification method that jointly learns across camera poses and object instances by employing a siamese inspired object detector.

1.4.3 Methodology in Chapter 5

In chapter 5, another end-to-end network is introduced that is much more efficient and flexible than the previous network (chapter 4). The set up of the problem is formulated into a graph, with the nodes of the graph being the target objects to be re-identified, and the edges between them to be classified as matching or not. With the flexibility and efficiency introduced in this method, satellite imagery is combined alongside the ground-level multi-views to achieve our research aims.

Contributions:

1. An efficient end-to-end, multi-view detector for static street-side objects.
2. The method implements graphs inside any anchor-based object detector.
3. A Geometric Neural Network (GNN) approach that jointly learns to use soft geometric features and image evidence to detect distinct objects in the scene.

4. An extension to our dataset is introduced, including satellite imagery and annotations of labeled tree instances corresponding to ground-level instances.

1.5 Technical Relation of the Chapters

In the following subsections, the progression and technical relations between the three "core" chapters are presented.

Relation between Chapter 3 & 4.

Although chapter 3 has a recent publication date, it is an extension to the first work [48] performed during the PhD. This work is to question our hypothesis as mentioned before *"Does geometry help?"*. Therefore, in our first work, we attempted to compartmentalize a task from our aim. This task matches the correspondences of objects in ground-level imagery, which is a problem foreseen from literature and previous work. Matching correspondence is a core task since we are detecting objects in multiple views for geo-localization. It is essential to identify instances of the objects to avoid double counting objects. To test our hypothesis, the object detector has been removed not to influence the results and assume that the objects have been detected beforehand.

Consequently, chapter 4 builds on the results of chapter 3, which concluded that geometry coupled with appearance features, does help in matching objects. Hence, chapter 4 is our first attempt at achieving an end-to-end network that performs detection and geo-localization in ground-level imagery by including an object detector and employs geometric features.

Relation between Chapter 4 & 5.

Chapter 5 principally is an extension to one of our published works [49]. We excluded [49] from our core chapters to avoid being repetitive. In comparison to chapter 4, the main improvement is the addition of aerial imagery combined with ground-level imagery. Also, the object detector used in chapter 5 is much more efficient. This efficiency gives us reduced memory consumption, which helps us to 1) use more views and 2) for our network training to converge faster. Furthermore, the object detector's architecture is convenient for various network design choices, which we will discuss in chapter 5.

1.6 Relevance for Science and Society

This section discusses how the methods and ideas introduced in this thesis are beneficial and impactful on science and society.

1.6.1 Remote Sensing - Geographic Information Systems

The incorporation of computer vision and deep learning techniques into remote sensing is becoming more prevalent. These techniques are replacing handcrafted methods producing higher accuracy and efficiency. Therefore, the data layers in which Geographic Information Systems (GIS) rely on will evermore be dependent on outputs from deep learning models.

Such tools help cities' decisions and policymakers' insight to better manage their cities and beneficial to regions where they are challenging to access or cover. As mentioned beforehand, street-trees also affect the ecosystem in cities to reduce the urban heat island effect. Reducing the urban heat island effect diminishes the energy needed, which makes cities more sustainable. These efforts fall inline with the 11th goal of the United Nations Sustainable Development calling to *"Make cities and human settlements inclusive, safe, resilient and sustainable"*[6].

Economic potential. Our proposed approaches can provide smarter management of a city's resources which in return could affect a region's economy. As previously explained in our motivation (section 1.1), creating inventories of trees and street-side objects by field crews is time-consuming and inefficient to cover large cities. In California, it is estimated that the value of a street-tree is $111, amounting to $1 billion per year [5].

1.6.2 Computer Vision - HD Maps and Self-driving

Self-driving or autonomous vehicle research has been at the forefront of deep learning and computer vision due to it being a challenging scientific problem and its promising future. These promises include providing a cheaper, safer, and environmental impact. The perception component of a self-driving solution chiefly relies upon 3D object detection [50] to detect objects of interest or obstacles. High-Definition (HD) Maps hold landmarks or priors used by self-driving systems for geo-localization and route planning [51]. Having data layers in HD Maps containing static objects such as street-trees, street signs, bicycle

6. https://sdgs.un.org/goals/goal11

racks, and trashcans can help act as priors to boost the vehicle's object detector by knowing beforehand the surrounding scene.

1.6.3 Publicly available research

The methods introduced in this thesis for automatic street-side object discovery and geo-localization can be applied to any set of geotagged images, and therefore they are presented for science and society as well. The approaches and methods introduced in this thesis and detailed in its core chapters have been published and communicated to the public through an open access policy. Furthermore, the software code related to our tools, and the latest methods have been made available. The annotations of our curated data set have also been made accessible. One of the obstacles encountered in this work was the change of service terms at a certain online mapping platform that prohibited the dissemination of imagery even for research. We plead tech companies and leading online mapping platforms to support nonprofit research initiatives by making it possible to use their services for research without restrictions. The various required elements of our work can be found on the Internet:

— Pasadena Dataset 3.0: *http://registree.ethz.ch/*
— Multi-view Reprojection source code:
https://github.com/nassarofficial/registreetooldemo/
— Registree Annotation tool source code:
https://github.com/nassarofficial/registreetool/
— GeoGraphDETR: *https://github.com/nassarofficial/geographdetr/*

The resources listed above can entirely be modified and used for commercial and noncommercial purposes. Hopefully, the dataset can be further expanded using the annotation tool to label more trees in different regions and settings. The different implementations will be helpful for reproducibility and hopefully empower more research on that topic.

Chapter 2

Background

This chapter covers the relevant background information to prepare for the following "core" chapters (i.e. Chapter 3, 4, 5). We touch upon several topics that are the most relevant to our work to make it self-explicative. However, complete textbooks and plentiful resources exist that explain these topics to a great extent. The first section includes general deep learning techniques, including conventional neural network types, and present Convolutional Neural Networks (CNN). In the third section, we present Graph Neural Networks (GNN), which is essential to cover our latest work.

2.1 Deep Learning

The neural network-based deep learning branch of machine learning has gained popularity recently. Such prevalence has been showcased by the neural networks outperforming all traditional machine learning methods at various natural language processing, computer vision, and robotics tasks, merely to mention a few. Neural networks are composed of a sequence of layers representing mathematical equations that aim at taking data, process it, and give an output from a pattern it learned. In this section, we will present some of the main concepts in deep learning.

2.1.1 Neural Networks

It is clear from the name that the human brain inspires neural networks. In essence, the human brain is an aggregation of neurons connected to each other, creating a vast network. These neurons transmit small electric charges as a means to transmit information. Another property of the neurons' connections is that the connections could be strong or weak, depending on the transmission frequency. As shown in the Fig. 2.1, a neuron receives input from the dendrites, which pass through the membrane and then propagates across the axons to feed into other neurons. As mentioned earlier, the neural

Learning to map street-side objects using multiple views Ahmed Samy Nassar 2021

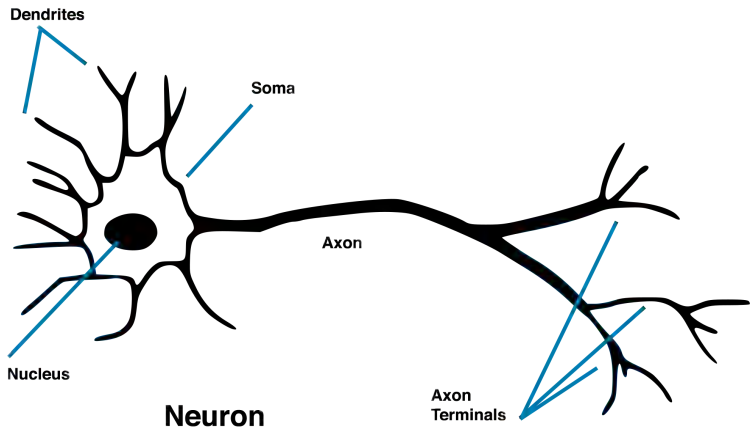


Figure 2.1: The biological structure of a neuron. Image downloaded from `https://commons.wikimedia.org/wiki/File:Neuron_-_annotated.svg`

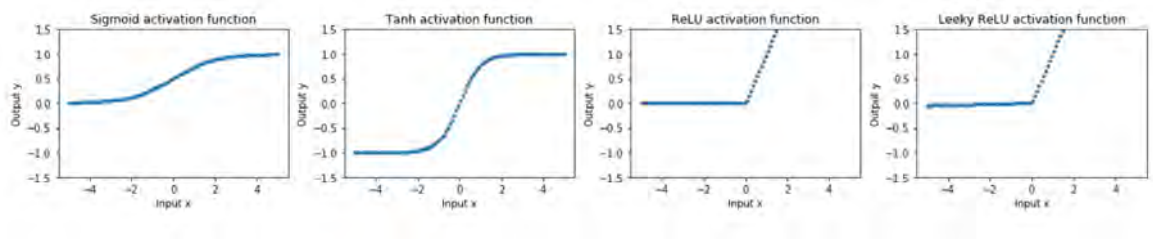


Figure 2.2: Representations of common activation functions. Image downloaded from `https://h1ros.github.io/categories/activation-function/`

network mathematically models a neuron. The neuron is represented as a calculation and summation of the input vector x, weight vector w, and a bias term b, encompassed frequently with a nonlinear function referred to as an activation function:

$$y = \sigma\left(\sum_i w_i x_i + b\right) = \sigma\left(w^T x + b\right) \tag{2.1}$$

The activation function decides the output of the neural network. It maps the resulting values between, for example, 0 to 1 or -1 to 1. There are various activation functions to be used, as shown in the Fig. 2.2. The most common and recommended activation function is the Rectified Linear Unit (ReLU). Another popular choice is the sigmoid function, which has different variations like the logistic and hyperbolic tangent. As no single neuron is capable of much, artificial neurons cannot do complex computations independently. Many artificial neurons are, therefore, structured to perform complex computations. The structures of the neuron can be arranged in any shape. However, the most common

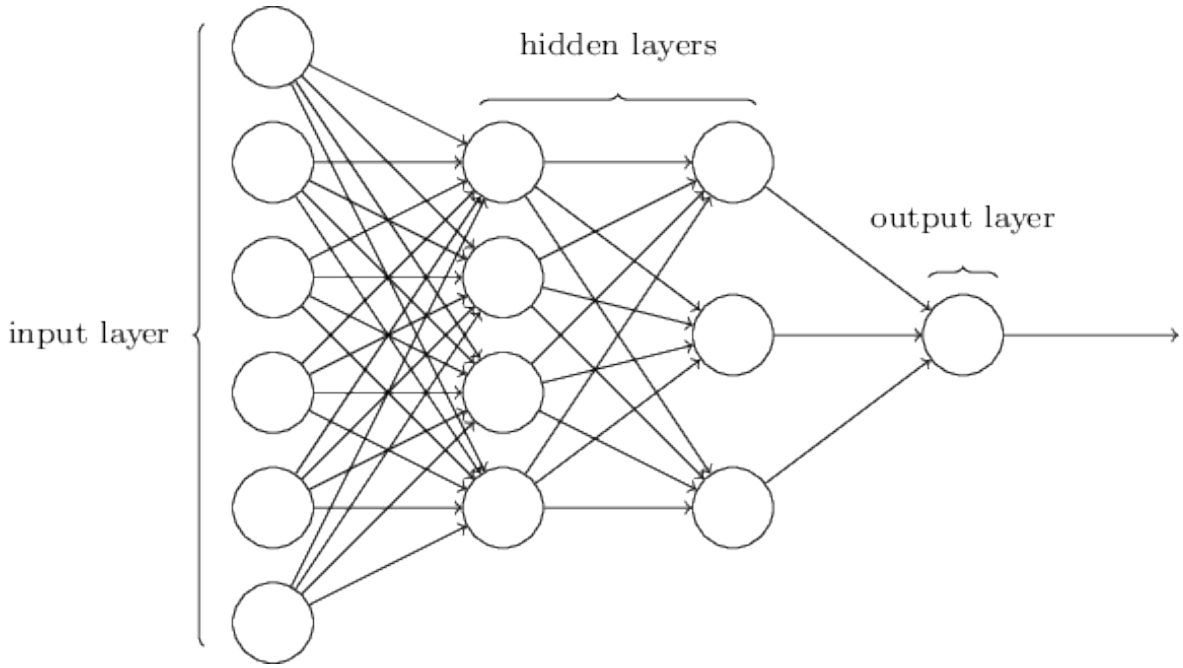


Figure 2.3: Illustration of an MLP (Multi-Layer Perceptron). Image downloaded from `https://github.com/ledell/sldm4-h2o/blob/master/sldm4-deeplearning-h2o.Rmd`

approach consists in a feed-forward topology, as it enables the activations to propagate forward in the entire network. Other topologies exist, such as networks containing cyclic connections in the case of recurrent neural networks, which are better at modeling for temporal sequences. Multi-Layer Perceptrons (MLP) used in this work are a conventional type of feed-forward neural networks. The typical structure of an MLP contains an input layer, a hidden layer, and an output layer. All neurons within a layer are all connected to a preceding layer, as shown in Fig. 2.3. To put it all together, the neural network combines all these different neurons to perform a particular function by tuning the entire network's parameters. This statement was founded in 1989 by [52] to prove that by using enough neurons of only sigmoid activation functions, it is possible to approximate any continuous function to a reasonable accuracy.

2.1.2 Backpropagation

Backpropagation [53], also known as the "backward propagation of errors", is the most popular and standard approach to train neural networks and deep neural networks. Backpropagation's role is to calculate the gradient to readjust the weights of the weight

matrices. The neuron weights are adjusted or set by calculating the loss or cost function's gradient. To be able to do so, a gradient descent optimization algorithm is used. The loss function E yields a value to be optimized. Naturally, there are many different cost function variations, but the simplest and most commonly used is the sum of the squared differences:

$$L = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.2}$$

The ground-truth is denoted as y, and predicted output as $\hat{y}$ from the neural network. For each of the n training samples, the summation of the square of differences between the ground-truth and predicted output is summed. The gradient regarding the loss function can be represented by three equations, one for the network's weight, bias, and activations:

$$\frac{\partial C}{\partial w^{(L)}} = \frac{\partial C}{\partial a^{(L)}}\frac{\partial a^{(L)}}{\partial z^{(L)}}\frac{\partial z^{(L)}}{\partial w^{(L)}} \tag{2.3}$$

$$\frac{\partial C}{\partial b^{(L)}} = \frac{\partial C}{\partial a^{(L)}}\frac{\partial a^{(L)}}{\partial z^{(L)}}\frac{\partial z^{(L)}}{\partial b^{(L)}} \tag{2.4}$$

$$\frac{\partial C}{\partial a^{(L-1)}} = \frac{\partial C}{\partial a^{(L)}}\frac{\partial a^{(L)}}{\partial z^{(L)}}\frac{\partial z^{(L)}}{\partial a^{(L-1)}} \tag{2.5}$$

Based on the outputs of the equations above, the step in the right direction to reduce the loss through optimization is taken. The gradient is usually computed by taking a batch or sample from the data. In batch training, we achieve an accumulated gradient by averaging the network's weights and biases for each batch over all it's training samples. In stochastic learning, the gradient is updated after each sample in our batch. These weights and biases are updated by using a learning rate α:

$$w^{(l)} = w^{(l)} - \alpha \times \frac{\partial C}{\partial w^{(l)}} \tag{2.6}$$

$$b^{(l)} = b^{(l)} - \alpha \times \frac{\partial C}{\partial b^{(l)}} \tag{2.7}$$

It is important to note that there are several determinants of how a neural network performs, the model's complexity, hyperparameters (e.g., learning rate), activation functions, dataset size, and more.

2.1.3 Deep Neural Network Challenges and Motivations

As the name suggests, deep neural networks are composed of multiple layers of neurons. As mentioned above, a single hidden layer conducts simple tasks such as a linear classifier, where two layers are universal approximators and three or more layers, which can be considered deep learning, are compact universal approximators. The motivation for using a deep neural network can undoubtedly be attributed to how efficient they are [54]. Another motivation is that with the many layers, the levels of abstraction increase, enabling learning a hierarchy of features. A neuron residing in one of the layers could be active depending if a feature is present. For example, in the case of image classification of a cat, these features could be a fur's texture and the shape of the ears. Naturally, the different layers and neurons get more sophisticated as deeper as the network continues.

2.2 Convolutional Neural Networks

Convolutional Neural Networks (CNN) are deep learning architectures produced toward computer vision tasks and have replaced many traditional methods in remote sensing [55]. Once again, basic concepts are briefly mentioned for the sake of completeness. However, for a detailed introduction, see [56]. For computer vision tasks, the input is usually an image comprised of thousands of pixels represented as a matrix. A neural network cannot process this input matrix using layers of fully connected neurons solely, which would lead to a considerable number of parameters to train.

2.2.1 Convolutional Layer

A convolutional layer performs a linear operation, which involves a matrix multiplication operation between the input matrix with a 2-dimensional array of weights commonly referred to as a filter or kernel. The filter's size is smaller than the input data matrix, covering only a small patch of the input as shown in Fig. 2.4. The multiplication performed is an element-wise dot product multiplication, which is then summed, resulting in a single value, that we store in a tensor. This process is referred to as the scalar product. The kernel is designed to be smaller than the input matrix to be able to multiply the kernel of weights at different positions of the image, orderly sliding over other patches of the image. The kernel is applied on the input matrix from left to right, top to bottom. This process of sliding the kernel over the input matrix allows the kernel to find features in the

input matrix. This sliding method also makes it capable of finding features "translation invariant". As referenced previously, a fully-connected layer is represented with (2.1), the

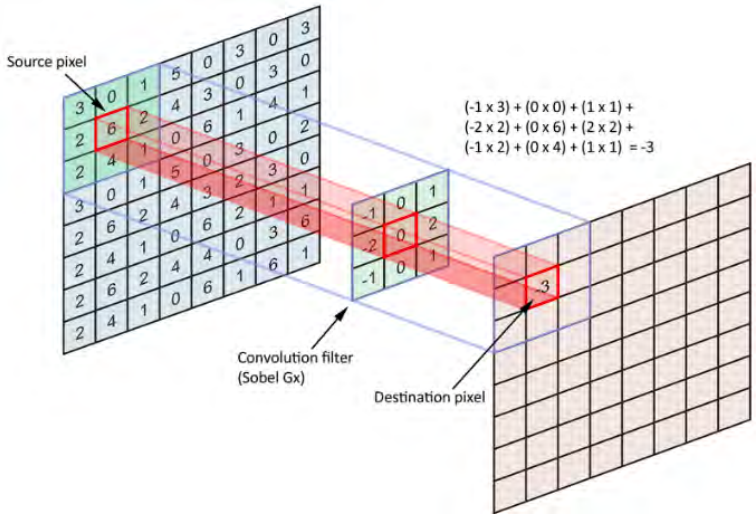


Figure 2.4: A convolutional layer operation, showing the kernel/filter sliding over an input tensor with the dot product multiplication between them. Image downloaded from `https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2`

convolution operator can also be represented similarly as:

$$y = \sigma(W * X + b) \tag{2.8}$$

where $*$ represents the convolution operation. As our input matrix is a two-dimensional image X, the convolution operator is defined as:

$$(W * X)(i, j) = \sum_m \sum_n X(m, n) W(i - m, j - n) \tag{2.9}$$

The full output of the convolutional layer, which is a matrix, is often referred to as an "activation map" or "feature map". These many kernels are capable of detecting different features. The kernel weights are all learned through backpropagation. A convolutional layer typically includes many different convolution kernels and producing feature maps.

2.2.2 Dense Layers

A dense layer is also referred to as a fully connected layer. The neurons within the layer are all connected to each other, which is similar to traditional multi-layer perceptrons [56]. The layers perform simple layer operations that are similar to matrix multiplication followed by vector addition.

2.2.3 Pooling Layers

One of the concepts introduced by CNNs is pooling, which enables the downsampling of the image. There are two standard non-linear downsampling functions, average and maximum pooling, being the most popularly used. Max pooling is an operation that calculated the maximum value for patches of feature maps, therefore creating a downsampled feature map, as shown in Fig 2.5. The subsampling process reduces the resolution of the

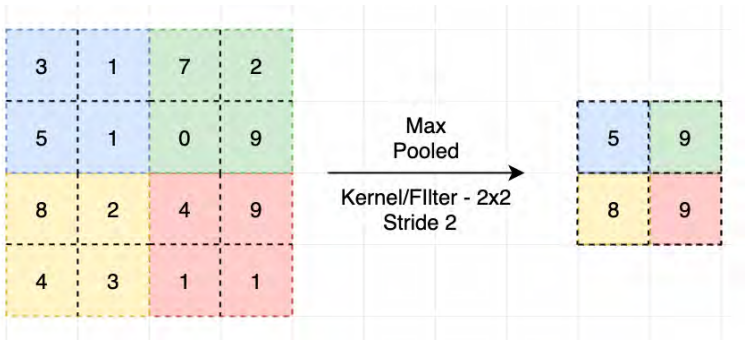


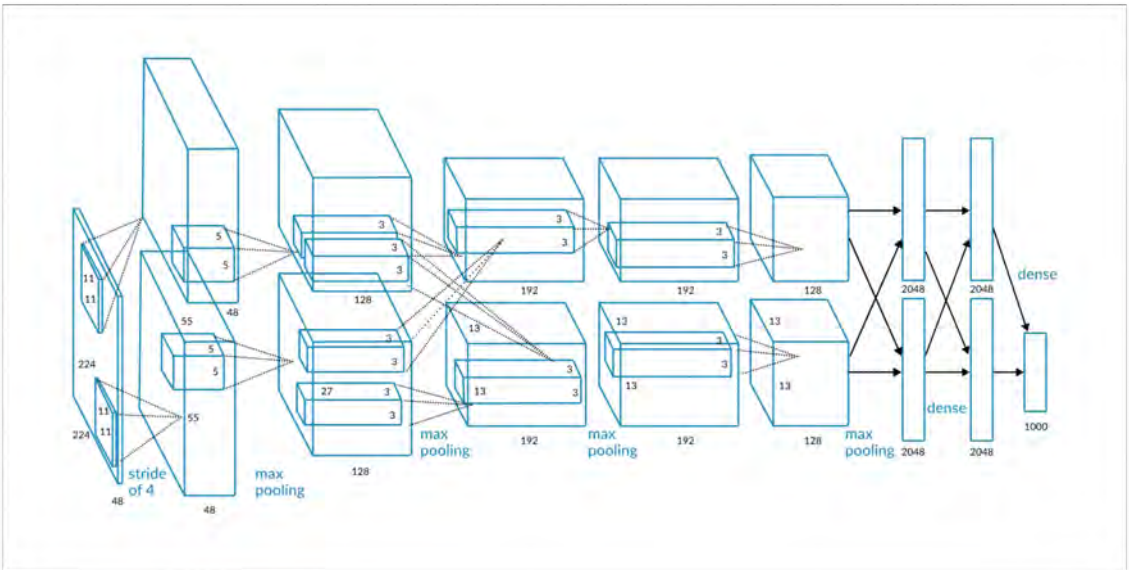
Figure 2.5: Max pooling operation function of a sample feature map. Image downloaded from `https://medium.com/ai-in-plain-english/pooling-layer-beginner-to-intermediate-fa0dbdce80eb`

image, which causes several advantages. The first advantage is that it makes the position of the features unimportant, rather than their relative positions. The second advantage is that with stacking pooling layers, the convolutions can have an effective field that gives the network the ability to learn complex spatial relations. Another advantage is that it reduces the number of parameters in the network, and therefore manages overfitting.

2.2.4 CNN Architectures & Applications

In this section, we highlight various commonly used CNN architectures that have been proven to be state of the art in recent history at various tasks and closely relevant to our

work. **AlexNet** [57] gained popularity after winning an image classification competition known as ImageNet Large Scale Visual Recognition Competition (ILSVRC) in 2012. The ImageNet dataset comprised of 15 million labeled images covering 1000 classes with each containing approximately 1000 images. The architecture of AlexNet is composed


224
224
11
11
11
11
stride
of 4
48
48
55
55
5
5
5
5
48
48
max
pooling
128
128
27
3
3
3

more kernels of the same size. Adding more kernels with a smaller size achieves a better performance with fewer parameters. Another boost of performance can be achieved with additional non-linear layers as it extends the network's depth, which enables it to learn more complex features. **ResNet** [59] is also known as deep residual networks, was introduced to combat a problem noticed when attempting to increase a network's depth. The problem encountered is that the network's accuracy saturates and deteriorates quickly, although a deep network should achieve similar accuracy to a shallow one. To solve this

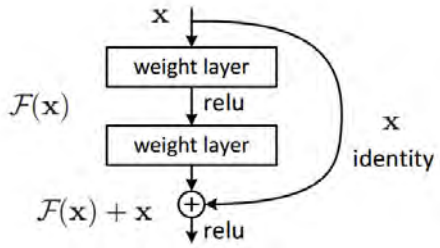


Figure 2.7: ResNet "skip-connections" or "identity shortcut connections". Image downloaded from `https://towardsdatascience.com/residual-blocks-building-blocks-of-resnet-fd90ca15d6ec`

problem, the authors introduce "skip-connections" as shown in Fig 2.7. During training, the network backpropagates through the identity function by using vector addition. This allows the gradient to flow smoothly from layer to layer and allows the lowest layers to receive activations from the top ones, which becomes useful in very deep networks. Another advantage of this technique is that it passes on useful information captured at early layers that would help the later layers learning from them.

Siamese Neural Networks were introduced by [60] to verify signatures. [61] also utilized it for face verification or recognition, later in person re-id, and various applications. In our work, we explore using siamese networks to re-identify objects across multiple views. Siamese networks are composed of two identical subnetworks, typically CNNs. The network receives a pair of inputs or samples and produces an output that measures the similarity in an output vector as shown in Fig. 2.8. This similarity output is learned by a non-linear metric training the network using pairs of positive and negative samples. In the case of face recognition, a popular application of siamese networks, a pair of positive samples can be two images of the same person under different circumstances, while a pair of negative images can be of different persons. The network then takes in those samples and tries to bring in positive samples together in "feature space" and push away negative samples. Therefore two similar images are supposed to yield a smaller Euclidean distance.

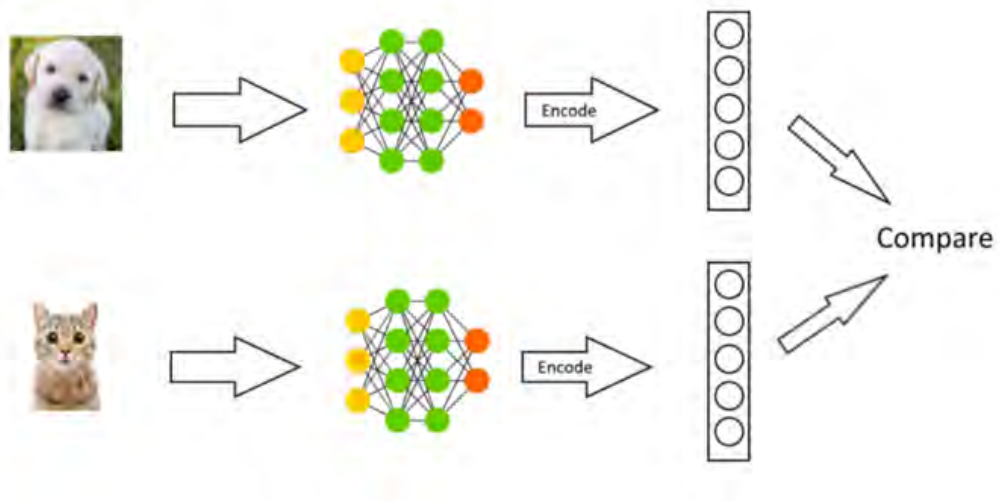


Figure 2.8: Siamese Network Architecture. Image downloaded from `https://medium.com/@crimy/one-shot-learning-siamese-networks-and-triplet-loss-with-keras-2885ed022352`

The most popular loss function used to train a siamese network is the "contrastive loss", which is motivated by the Euclidean distance metric, and can be defined as:

$$L_{\text{contrastive}} = \frac{1}{N}\sum^{N} y \cdot \|f(x_i) - f(x_j)\|_2^2 + (1-y) \cdot \max(m - \|f(x_i) - f(x_j)\|, 0)^2 \quad (2.10)$$

where x_i, x_j represents an input image pair, m is a constant margin, y is the ground-truth label of the pair, f is the CNN. It is important to note that both CNNs share the same weights, and both are updated the same way. Sharing weights between both networks ensures that the learned distance is symmetric. Triplet neural network is another different architecture inspired by siamese architecture. As the name suggests, the triplet network is composed of three subnetworks, as shown in Fig. 2.9. The samples provided as input to a triplet network consist of three images, referenced as an anchor image, a positive image similar to the anchor image, and a negative image of another identity. Similar to a siamese network, the weights between the three subnetworks are shared. During training, the triplet loss function pushes the negative sample away from the anchor sample and pull in the positive sample in feature space. The triplet loss function is defined as:

$$L_{\text{triplet}} = \frac{1}{N}\sum^{N} \max(\|f(x_a) - f(x_p)\| - \|f(x_a) - f(x_n)\| + m, 0) \quad (2.11)$$

where x_a, x_p, and x_n denote the anchor, positive and negative samples respectively.

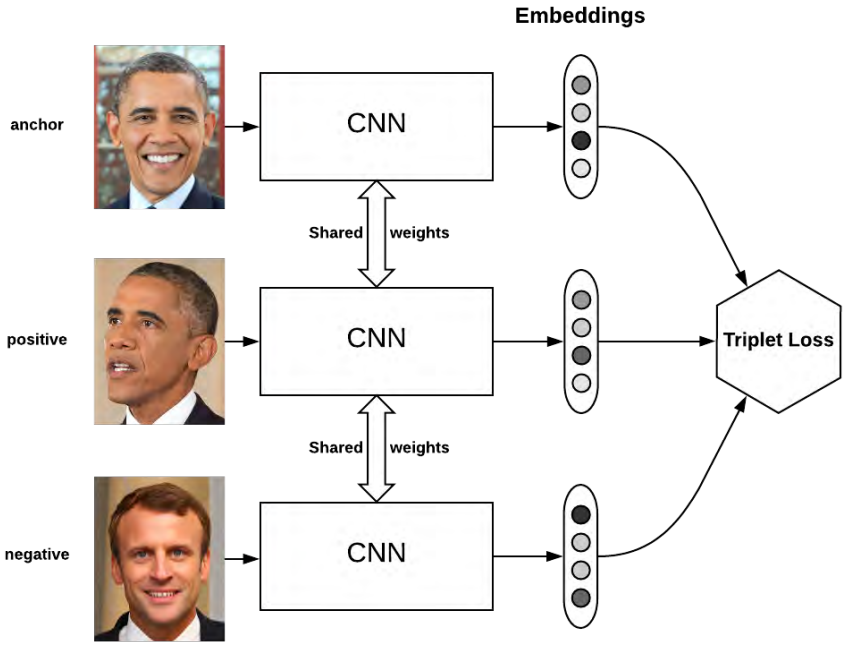
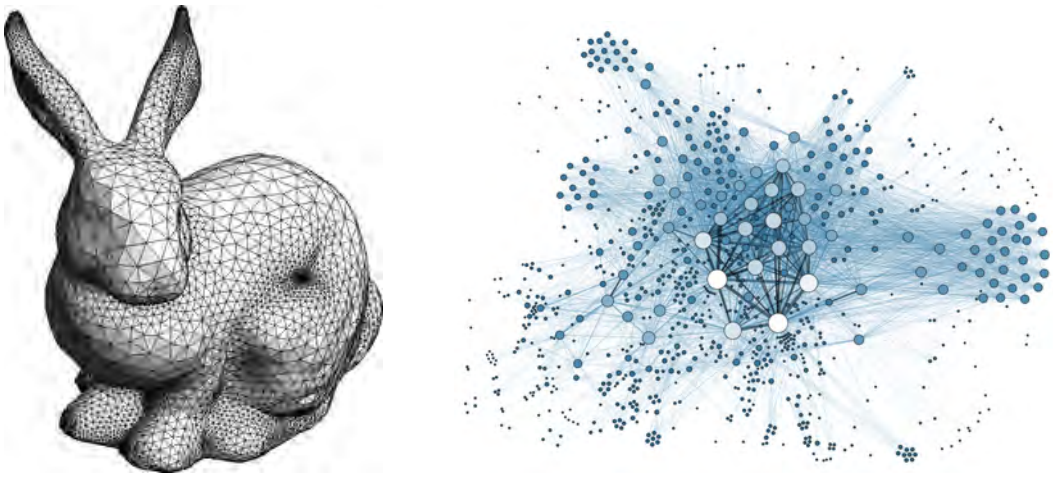


Figure 2.9: Triplet Network Architecture. Image downloaded from `https://omoindrot.github.io/triplet-loss`

2.3 Geometric Deep Learning

Geometric deep learning was first presented by [62] as another field of machine learning that applies Convolutional Neural Network methods and approaches to irregular data structures. Irregular data structures can be 3D point clouds, graphs, or manifolds (see Fig. 2.10). Graphs and manifolds represent social networks, molecular graph structures, biological protein networks, and recommender systems as nodes and edges. In this regard, geometric deep learning exploits the relationships between nodes and edges to discover new patterns, such as recommending a friend in a social network [30], classifying a 3D object, or classifying a protein's role in a biological interaction network [63]. The challenge from a machine learning perspective is that there is no straightforward approach to encode this non-euclidean information into a feature vector. A graph exists as undirected, connected, or weighted and is represented as a $\mathcal{G} = (\mathcal{V}, \mathcal{E}, W)$, where $\mathcal{V}$ denotes the vertices or nodes, $\mathcal{E}$ is the set of edges, and $\mathbf{W} \in R^{n \times n}$ is a weighted adjacency matrix. An edge between two



(a) Manifolds. (b) Graphs.

Figure 2.10: Examples of irregular data structures. Image from Grandjean, Martin (2014). "La connaissance est un réseau". Les Cahiers du Numérique

nodes (i and j) can be represented by $e = (i, j)$. Similar to CNNs, Graph Neural Networks (GNN) are also composed of convolutional, pooling, batch normalization, dropout, and activation layers. There are two main approaches in graph neural networks, spectral and spatial approaches.

2.3.1 Spectral Graph Convolutional Networks

The main concept for spectral approaches in graph neural networks is to apply the Fourier transform theorem on graph and manifold data, i.e. convolutions in the spectral domains. In graph convolutional networks, node features are represented as signals acquired from applying a function to a node feature. To perform convolutions on a graph, the eigenvectors and eigenvalues of a graph Laplacian are found by employing eigendecomposition, which transforms the graph into the spectral domain. Similarly, we apply the decomposition to the convolutional kernel or filter. Comparable to CNNs, we multiply the spectral graph with the spectral kernel. Many different methods in spectral graph learning exist. ChebNets [64] are one of the earliest works. As mentioned above, it applied a spectral convolution by multiplying node features by a kernel or filter. The kernel of a ChebNets is composed of the sum of Chebyshev polynomials from the diagonal matrix of

the Laplacian eigenvalues up to a certain order. The kernel is represented as:

$$g_\theta(\Lambda) = \sum_{k=0}^{K-1} \theta_k T_k(\tilde{\Lambda}) \tag{2.12}$$

where g_θ denotes the kernel, θ represents the vector of Chebyshev coefficients, $\tilde{\Lambda}$ represents scaled Laplacian eigenvalues diagonal matrix. k, K, T represent the smallest order neighborhood, largest order neighborhood, and Chebyshev polynomials of a certain k order respectively. ChebNets, also similar to CNNs, introduced graph pooling (see Fig. 2.11), which coarsens the graph to increase efficiency.

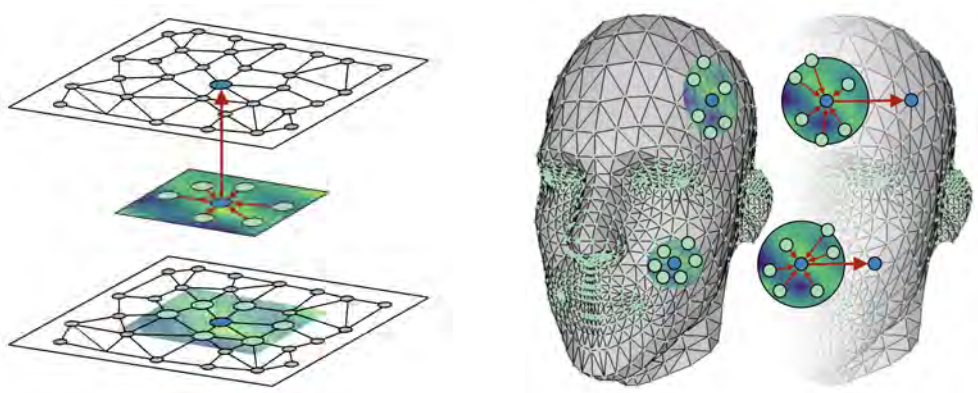


Figure 2.11: Before and after picture of graph pooling. Image downloaded from `https://andreasloukas.blog/2018/05/26/demystifying-graph-coarsening/`

However, the most prominent disadvantages of spectral graph convolutional networks are that they cannot generalize well over different graphs, also with large graphs (millions of nodes) they sum up to a huge number of trainable parameters, therefore inefficient [65].

2.3.2 Spatial Graph Convolutional Networks

In our work, we rely on spatial graph convolutional networks. Spatial approaches are also based largely on CNNs, they represent a neighborhood using a fixed structure, however the difference is that in images the structure is constant for all vertices. Therefore, spatial approaches use the distance and angle on manifolds or the degree of neighboring nodes for graphs. Perhaps the earliest work in spatial graph CNNs was presented by [66],which applied convolutions on manifolds using kernels represented as geodesic polar coordinates. Another interesting work is SplineCNN [67], which is currently considered state of the art. SplineCNNs aim at capitalizing on the advantages of B-spline bases



(a) Filtering of Graphs. (b) Filtering of Meshes or Manifolds.

Figure 2.12: SplineCNN aggregation method using trainable continuous kernel functions.

to create their kernels. They also use d-dimensional pseudo-coordinates **u(i,j)** around a node or vertex to discover how the features are to be aggregated as shown in Fig. 2.12. The B-spline kernel can be defined as:

$$\mathcal{K}_{\mathbf{p}\in\mathcal{P}}(\mathbf{u}) = \prod_{i=1}^{d} N_{i,p_i}^{m}\left(u_i\right) \tag{2.13}$$

where N_{i,p_i}^{m}, m represent B-spline bases, and the degree respectively. The filter function Γ is defined as:

$$\Gamma_{l,l'}(\mathbf{u}) = \sum_{\mathbf{p}\in\mathcal{P}} \mathcal{W}_{\mathbf{p},l,l'} \cdot \mathcal{K}_{\mathbf{p}\in\mathcal{P}}(\mathbf{u}) \tag{2.14}$$

where $\mathcal{W}_{\mathbf{p},l,l'}$ represents the B-spline control points, which are learnable. Given the Γ and the input node feature f, an output of a node i can be represented as:

$$g_{l'}(i) = \sum_{l=1}^{p} \sum_{j\in\mathcal{N}(i)} f_l(i) \cdot \sum_{\mathbf{p}\in\mathcal{P}} \mathcal{W}_{\mathbf{p},l,l'} \cdot \Gamma_{l,l'}(\mathbf{u}(i,j)) \tag{2.15}$$

With the advantage of being in the spatial domain, they do not have the same disadvantage of spectral methods, being that they do not generalize well across different domains. As they work intrinsically, this makes them more robust and capable of discovering local invariant information in 3D shape analysis.

Chapter 3

Multi-View Instance Matching with Learned Geometric Soft-Constraints

Nassar, Ahmed Samy, Sébastien Lefèvre, and Jan Dirk Wegner.
"Multi-View Instance Matching with Learned Geometric Soft-Constraints."
ISPRS International Journal of Geo-Information 9.11 (2020): 687
(Author version; for typeset version please refer to the original journal paper.)

3.1 Abstract

We present a new approach for matching urban object instances across multiple ground-level images for the ultimate goal of city-scale mapping of objects with high positioning accuracy. What makes this task challenging is the strong change in view-point, different lighting conditions, high similarity of neighboring objects, and variability in scale. We propose to turn object instance matching into a learning task, where image-appearance and geometric relationships between views fruitfully interact. Our approach constructs a Siamese convolutional neural network that learns to match two views of the same object given many candidate image cut-outs. In addition to image features, we propose utilizing location information about the camera and the object to support image evidence via soft geometric constraints. Our method is compared to existing patch matching methods to prove its edge over state-of-the-art. This takes us one step closer to the ultimate goal of city-wide object mapping from street-level imagery to benefit city administration.

3.2 Introduction

Automated methods for mobile mapping to generate inventories of urban objects at large scale have received significant attention lately [68]–[72]. While most systems have laser scanners as a major part of their measurement device, a significant number of research efforts try to match objects across multiple views based solely on imagery. Some traditional methods [73] rely on SIFT [74] to perform the matching. Several methods [75] similarly employ Siamese CNNs to solve the problem. However, our case is different in that objects are static, but appear from very different viewing angles and distances in contrast to other works.

In this work, we propose to augment image evidence with soft geometric constraints to learn object instance matching in street-level images at large scale end-to-end. Our ultimate goal is to improve geo-positioning of urban objects from ground level images, particularly street-trees and traffic signs. To achieve this, we rely on using multi-views to have more information on the objects inside the scene. We acquire our geometric constraints from the metadata accompanying our images. The set up of the problem as demonstrated in Figure 3.1, which includes a scene with multiple cameras with a large distance between them. This introduces the problem of matching the objects inside the scene across multiple views which can be difficult due to how similar the objects can look while sharing the same background, as presented in Figure 3.2. Our method builds upon a Siamese architecture [60] that constructs two identical network branches sharing (at least partially) their weights. Features are computed for both input images and then compared to estimate the degree of similarity. This can be achieved by evaluating either a distance metric in feature space or the final classification loss. We build a Siamese CNN to match images of the same objects across multiple street-view images. Google street-view and Mapillary provide access to a huge amount of street-level images that can be used to construct very large datasets for deep learning approaches. Here, we use the former to build a multi-view dataset of street-trees and use a dataset provided by the latter for traffic signs. Both are then employed as testbeds to learn instance matching with soft geometric constraints based on a Siamese CNN model. Our main contribution is a modified Siamese CNN architecture that jointly learns geometric constellations from multi-view acquisitions jointly with the appearance information in the images. This will further on help us in our main pipeline to better geo-position objects in the wild, and to subsequently assign them with predefined semantic classes. As such, our problem encompasses several research

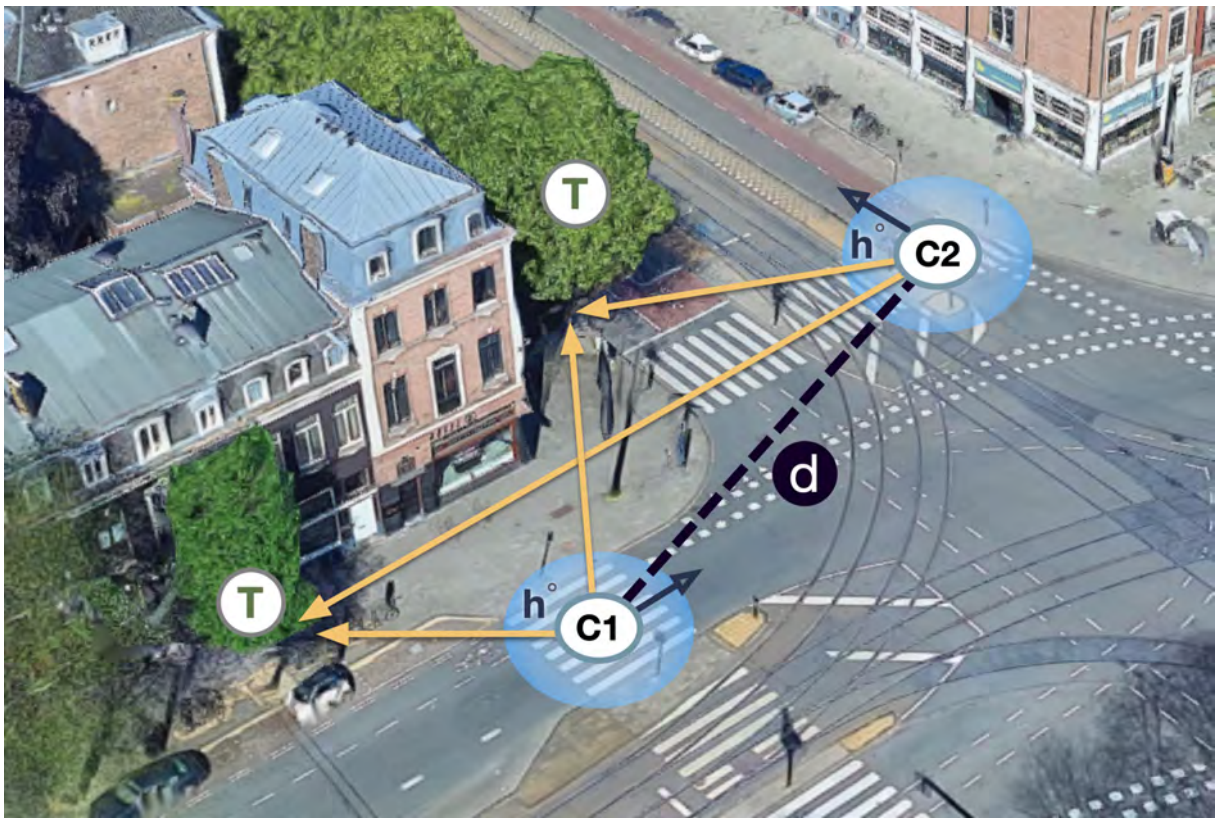


Figure 3.1: C*, camera with geo-position; T, the tree has its actual geographic coordinates and location within the panorama; $h°$, heading angle inside panorama; d, distance between cameras.

topics in computer vision, such as multi-view object tracking, instance re-identification, and object localization. We highlight some examples in the literature per field and draw comparisons between these problems and ours in the following section.

3.3 Related Work

Siamese CNNs, as introduced by [60], propose the matching of signatures using a neural network architecture with two usually identical network branches that partially share their weights at some stage. Siamese networks are used for a wide range of applications such as face verification [61], [76], [77], ground-to-aerial image matching [55], [78], object tracking [79], [80], local patch matching [81], [82], and patch descriptors [83], [84]. In this work, we explore Siamese networks to jointly learn robust appearance-based and geometric features and improve instance matching across multiple views.

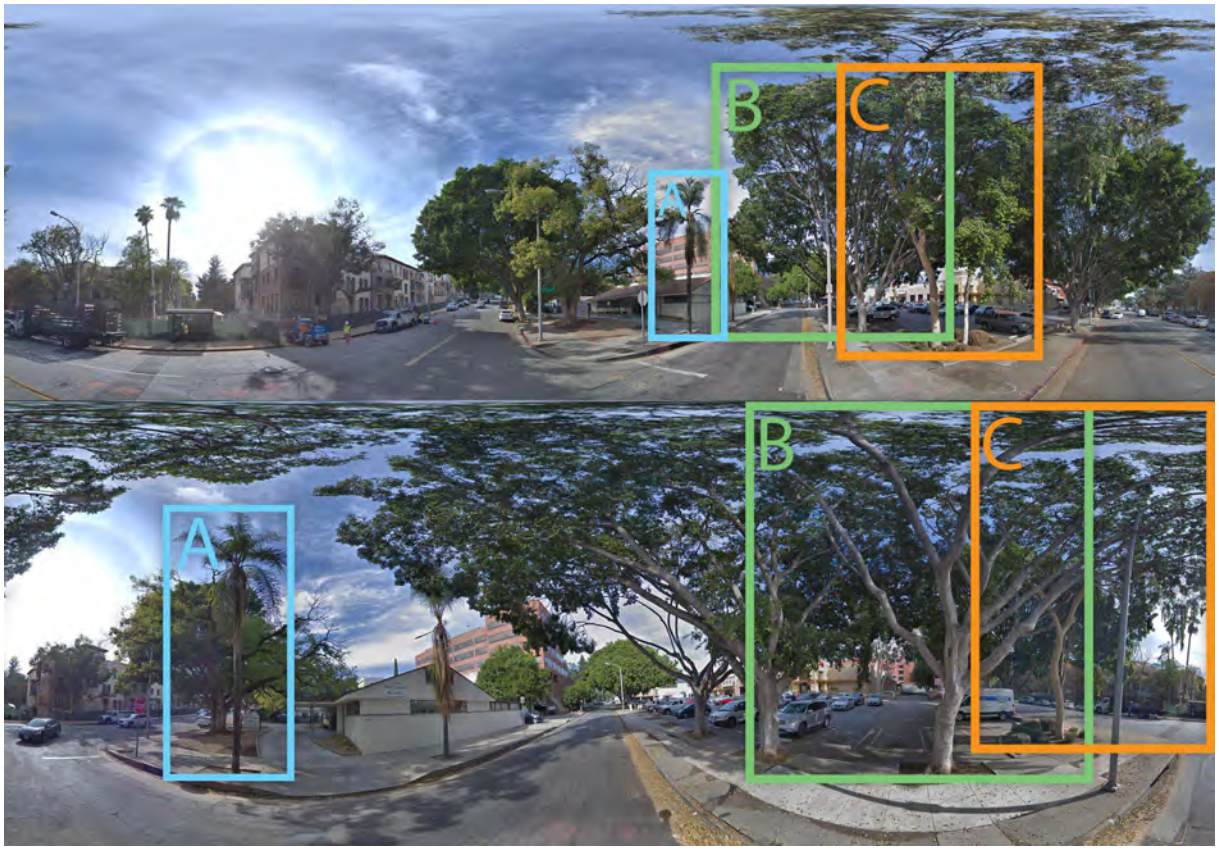


Figure 3.2: The tree instance matching problem (color and letters indicate the matches of identity): each tree is photographed from multiple different views, changing its size, perspective, and background. Note that many trees look alike and are in close proximity (Imagery © 2019 Google).

Multi-view Object Tracking (MOT) has been tackled by many different deep learning approaches, e.g., the method of [32] learns features using a Siamese CNN from multi-modal inputs from images and optical flow maps. In [31], a Siamese CNN and temporally constrained metrics are jointly learned to create a tracklet affinity model. Another approach [33] uses a combination of CNNs and recurrent neural networks (RNN) in order to match pairs of detections. In our setup and dataset, objects are not tracked from consecutive frames unlike the works mentioned, causing a background change which makes the problem more difficult.

Object geo-localization from ground imagery such as Google street-view has been a major research interest for several years, for example, for street-tree detection [25], [26]. Other works aim at geo-localizing poles in Google street-view images [28] and use state-of-the-art object detectors along with a modified brute-force-based line-of-bearing to estimate the locations of the poles. [27] used semantic segmentation of images alongside a monocular depth estimator that feed into an MRF model to geo-localize traffic signs. [23] detected road objects from ground level imagery and placed them into the right location using semantic segmentation and a topological binary tree.

Instance re-identification matches image patches for object re-identification purposes. The most similar problem to our task is the person re-identification problem, which has become a major research interest recently [34], [75], [85]–[89]. Another interesting application includes vehicle re-identification. For example, [90] used a CNN to extract features that are input to a temporal attention model. [91] used a CNN to extract appearance attributes, a Siamese CNN for license plate verification and the vehicle search is refined using re-ranking based on spatiotemporal relations. The authors of [92], [93] used key points or descriptors to find matches between images; however, we try to find if image patches are of the same object, therefore finding descriptors is irrelevant. Again, our task differs significantly because consecutive images have large baselines (Google street-view panoramas) or are acquired from a moving platform (Mapillary dashcam dataset), which leads to high perspective change and varying background.

3.4 Instance Matching with Soft Geometric Constraints

An overview of the proposed pipeline is shown in Figure 3.3. The main idea is that corresponding images of the same object should follow the basic principles of stereo- (or multi-view) photogrammetry if the relative orientation between two or more camera

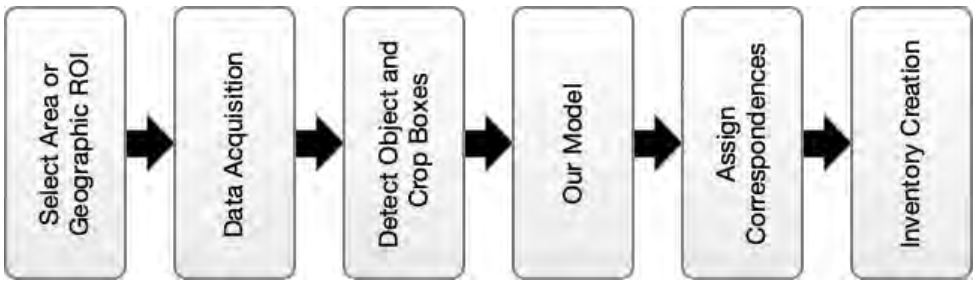


Figure 3.3: The overall pipeline which encompasses our proposed model. First, a geographic region of area is selected. Then, for that geographic region, the imagery and metadata pertaining to it are downloaded. An object detector detects the objects in this imagery. Our proposed model takes in pairs of image crops of the object and decides if they are matching or not. An inventory is created of the objects, with a pool of image crops from different views for each instance.

viewpoints can be established. Directly imposing hard constraints based on the rules of, for example, forward intersection is hard. An unfavorable base-to-height ratio, i.e., trees on the street-side get very close to the camera but the distance between two panorama acquisitions is significantly larger, makes dense matching impossible. The perspective of the object changes too much to successfully match corresponding image pixels, as presented in Figure 3.2. Moreover, the heading and geolocation (that are recorded in the metadata of street-view panoramas) are often inaccurate due to telemetry interference or other causes. As for traffic signs, the image crops vary depending on the acquisition due to Mapillary's dataset being crowdsourced. The crops can be acquired from a camera mounted on a vehicle or by pedestrians therefore providing an inconsistent setup. We thus propose to implicitly learn the distribution of geometric parameters that describe multi-view photogrammetry together with the image appearance of the objects. Our assumption is that this approach will enable cross-talking between image evidence and geometry. For example, if the same object appears with the same size in two images (but very different perspective), the triangle that connects both camera positions and the object must be roughly isosceles. That is, the object is located in between both camera standpoints. Conversely, the object in question that is viewed from the same perspective (very similar image appearance) but appears rather small will point at a pointy triangle with one very long leg (longer than the baseline) and another shorter leg. More literally speaking, the target object will most likely be situated outside the baseline between the two cameras.

3.4.1 Model Architecture

Our method employs a modified Siamese CNN that processes image crops, and geometric features jointly. We use geometric features composed of $\{[C^*_{lat}, C^*_{lng}, h^\circ]\}$, where C represents the image geolocation and h° is the heading angle of the object inside the image. We add these geometric cues to image evidence inspired from [94], who merged multi-modal data inside a single CNN architecture to minimize a joint loss function.

We feed two geometric vectors to our network in addition to the image crops containing the object instance, resulting in six channels in total, as shown in Figure 3.4. That creates an extra channel with the dimensions of the image for each geometric feature containing only the value of the feature. This is where this model differs from our previous work [48], in which the geometric vectors pass through a different subnetwork. Consequently, two feature subnetworks extract features from the geometric vectors and the image crops. Performing convolutions on the six channels provides enhanced descriptive features by applying the filter on both the RGB, and the geometric values conjointly. After that, the "Feature Subnetwork" produces "Feature Embeddings" that we provide as input to the "Decision Networks" that determines whether the images are similar or not.

In general, any state-of-the-art architecture could be used to extract the features. We experimented with shallow networks such as AlexNet and deeper networks such as ResNet for the different tasks in order to investigate how efficient (in terms of parameters) and deep the base network should be to extract the features. After preliminary experiments with common architectures such as AlexNet [95], ResNet34 [96], and MatchNet [82], we found ResNet34 to perform best in our scenario and thus kept it for all experiments. For future work, ResNet is a better choice implementation-wise when integrating with object detectors [16], [97] that use ResNet as the backbone.

As shown in Figure 3.4, the features generated from the feature subnetworks are fed into the "decision networks" component, which is the decision-making part of the network that computes the similarity. This "decision networks" can be either a contrastive loss or made of fully connected layers [82] with classification (depending on the experiment, as we explain in the Results Section). The decision component is composed of four FC (fully connected) networks. Our Siamese CNNs shares the weights of the feature subnetworks, as suggested by [76] when dealing with the same modality. Thus, both of our feature subnetworks are identical and come with shared network parameters.

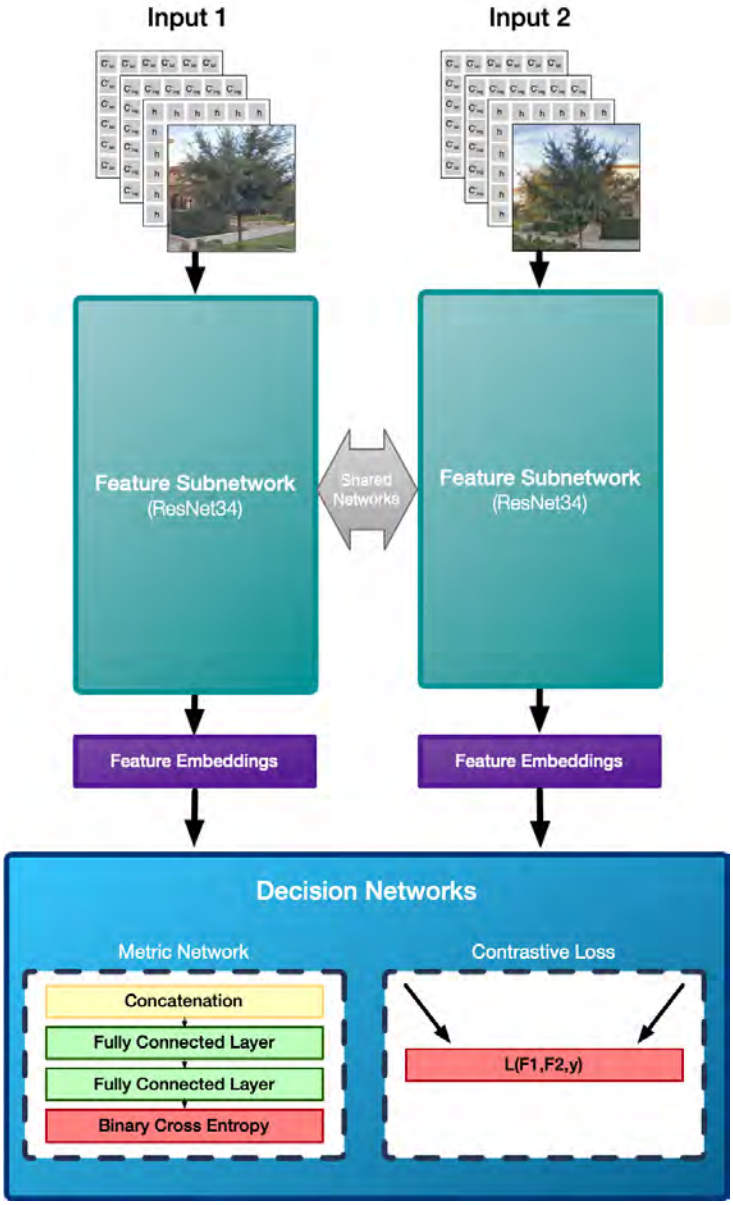


Figure 3.4: Diagram showing the overall network architecture. The Feature Subnetwork receives an image with three extra channels ($\{[C^*_{lat}, C^*_{lng}, h^\circ]\}$) as input (shown as three additional matrix layers in grey). In general, the feature subnetworks could be of any state-of-the-art architecture but preliminary tests showed that ResNet consistently yielded best performance. Generated feature embeddings are passed to the decision networks that classify whether two image patches are matching or not.

3.4.2 Loss Functions

We tried three different variants of loss functions for our multi-view instance matching approach and explain their details in the following.

Contrastive: Our first approach is a Siamese CNN composed of two identical subnetworks that are trained with a contrastive loss function [98]. A contrastive loss (Equation (3.1)) takes a pair of features created from the two branches of the network as input, unlike other loss functions that evaluate the network across the training dataset. The loss function's purpose is to bring matching or positive embeddings closer and push non-matching embeddings away in feature space. Therefore, the loss function encourages the network to output features that are close in feature space if samples are similar, or different features if they are not similar. This is achieved by penalizing the model depending on the samples. The contrastive loss function is defined as

$$L = \frac{1}{2N}\Big(\sum_{n=1}^{N} y_n d_n^2 + (1-y_n) + (1-y_n)\max(m-d_n,0)^2\Big) \tag{3.1}$$

where y is the ground truth label, m is a margin, and d_n is any distance function between the two output features.

Metric: This is another Siamese CNN approach composed of similar subnetworks that provide the metric network with concatenated features. The metric network is composed of three fully connected layers with ReLU activation, except the last layer which encodes the binary cross entropy function (Equation (3.2)). The outputs of the last layer are two non-negative values within [0,1] that sum up to 1. Each value corresponds to the probability of the samples being classified as similar or not. Binary cross entropy is defined as

$$L = -\sum_{i=1}^{C'=2} y_i \log(s_i) = -y_1 \log(s_1) - (1-y_1)\log(1-s_1) \tag{3.2}$$

where we only have two classes. y_1 is the ground truth label and s_1 is the probability score for C_1. Consequently, $y_2 = 1 - y_1$ and $s_2 = 1 - s_1$ are the ground truth and probability score for C_2.

TripleNet: This is a triplet network architecture [99] composed of three identical subnetworks rather than two. Each feature subnetwork receives a different image to generate an embedding. The inputs are an anchor image (our main image or image in

question), a positive image (an image similar to the anchor image), and a negative input (which is an image dissimilar to the anchor image). Similar to contrastive loss, the network is trained to minimize the anchor and positive embeddings while maximizing the distance between the anchor and negative embedding with a triplet loss (Equation (3.3)). The triplet loss is defined as:

$$L = \max(\|f(A) - f(P)\|^2 - \|f(A) - f(N)\|^2 + m, 0) \tag{3.3}$$

where m is the margin, f is the feature output, A is the anchor feature, P is the positive feature, and N is the negative feature. Note that all three architectures can be combined with different feature subnetworks such as AlexNet, MatchNet, ResNet34, etc.

3.5 Experiments

Our experiments were implemented in PyTorch. The weights of the network were initialized using the "Glorot uniform initializer" [100], the initial learning rate was set to 0.0001 with ADAM [101] as the optimizer, and the dropout rate was set to 0.3. All image patches were resized to 224 × 224 pixels and fed to the two network streams separately. Note that we applied standard pre-processing (mean subtraction and normalization) to the input images as well as the geometric features. We used normalization values calculated from ImageNet for our experiments since we used the pre-trained weights to initialize our models.

3.5.1 Datasets

We evaluated our method on two different datasets. Both datasets differ in terms of objects, image geometry, and acquisition strategy. The *Pasadena* dataset consists of panorama images from Google street-view, whereas the *Mapillary* dataset contains mostly images acquired with various dash cams in moving vehicles. Objects of interest are trees in *Pasadena* and traffic signs in *Mapillary*. Baselines between consecutive panoramas of *Pasadena* are usually larger (≈50 m, Figure 3.5) than those between consecutive frames of *Mapillary* (usually a few meters depending on the speed of the vehicle, Figure 3.6). While panoramas of *Pasadena* show a 360° view around the mapping vehicle, *Mapillary* images are acquired with a forward looking camera in a moving vehicle, resulting in a much narrower field-of-view. In addition, this leads to different mappings of the same

object in consecutive images, as shown in Figure 3.7. While objects in *Mapillary* images mainly experience a scale change while the vehicle is driven towards them, objects in panoramas also undergo a significant perspective change. In the following, we describe both datasets in more detail.

Pasadena

We tested our approach on a new dataset of Pasadena, California, USA, which extends the existing urban trees dataset of our previous work [26], [55]. It is generated from an existing KML-file that contains rich information (geographic position, species, and trunk diameter) of 80,000 trees in the city of Pasadena. For every tree, we downloaded the closest four panoramic images of size 1664 $\times$ 832 pixels from Google street-view, as shown in Figure 3.5. A subset of 4400 trees with four views each was chosen, leading to 17,600 images in total plus meta-data. Note that the Pasadena inventory contains only street-trees, which makes up roughly 20% of all city trees. We drew bounding boxes around all street-trees per panorama image, which resulted in 47,000 bounding boxes in total. A crucial part of the labeling task was to label corresponding images of the same tree in the four closest views, as shown in Figure 3.1. As presented also in Figure 3.5, the perspective changes are drastic in some cases, making it even difficult for the human eye to tell if they are of the same location. In addition, distortions often occur when the trees overcast the 360 camera. Our final dataset is composed of panoramic images containing labeled trees (and matches between four tree images per tree), the panorama meta-data (geographic location and heading of the camera), and the geo-position per tree. Note that the geo-position per tree was used during training to generate ground truth parameters of our geometric features. It was not used during testing, but geometric parameters were directly derived from the individual panoramas.

Mapillary

We ran the baseline methods and our methods on a new dataset provided by Mapillary (`www.mapillary.com`) in order to verify our results. This dataset is not to be confused with the Mapillary Vistas dataset [102] which is provided for a semantic segmentation challenge. The dataset contains 31,342 instances of traffic signs that are identified within 74,320 images in an area of approximately 2 km^2. On average, two traffic signs appear in each image. The dataset format is in GeoJSON, where each "feature" or identity has the following properties: (i) the geo-coordinate of the object that is attained by using 3D



Figure 3.5: Four consecutive panoramas from the *Pasadena* dataset (Imagery © 2019 Google).



Figure 3.6: Consecutive frames of two example scenes of the Mapillary dataset.



Figure 3.7: A single instance of a traffic sign from the *Mapillary* dataset acquired from different sensors, angles, and dates.

structure from motion techniques, which is thus affected by the GPS, and the density of the images; (ii) the objects distance in meters from the camera position; (iii) image keys to identify which the object appears in and which is used to retrieve the image using their API; (iv) geo-coordinates of the image location; (v) the object's altitude and angle; and (vi) the annotation of the sign in polygon form.

The Mapillary dataset is quite different from our Pasadena dataset in many ways. The images are crowd-sourced with forward looking dash cameras on moving vehicles, smartphones, or even panoramic rigs on hobbyists cars. Therefore, the image sizes and quality are very inconsistent, as well as the time the images were captured. Because most of the images were captured from car dash cams, the viewpoint changes are only a few meters (Figure 3.6) due to the images being consecutive frames in comparison to a GSV panorama (Figure 3.5). As shown in Figure 3.6, because the camera is mostly forward looking, the objects are viewed almost from the same viewpoint with scale changes, and the objects are of a very small size in comparison to trees for instance. In addition, it is important to note that unlike trees the traffic signs are much smaller, and the best angle to capture them is from the front and not sideways due to how thin they are, as shown in Figure 3.6.

3.5.2 Evaluation Strategy

We performed a 10-fold cross-validation for all experiments to avoid any train-test split bias and over-fitting. Each tree comes with four image patches from different views, where every image patch is associated with a feature vector that contains geometric cues, as described in Section 3.4. For training the positive match category, we inserted matching image patch pairs from the same object with the geometry feature vectors to our model.

Negative pairs of the rejection category were generated by randomly picking two image patches from two different objects. Initial tests showed that most mismatches occur at neighboring objects because geometry is least discriminative in such cases (i.e., the warping function is very similar) and objects share the same background. In the case of *Pasadena*, neighboring trees often belong to the same species, too, leading to very similar visual appearance in the images. Therefore, we added many negative example pairs from neighboring objects to make the classifier more robust.

3.5.3 Does Geometry Help?

We evaluated whether geometric evidence helps by comparing against a baseline without geometric features for the *Pasadena* and *Mapillary* datasets (Table 3.1). All three model architectures, *Contrastive*, *Metric*, and *TripleNet*, were evaluated per dataset with (*w/ Geometry*) and without geometric features (*w/o Geometry*).

The only difference from *w/ Geometry* to *w/o Geometry* is that we concatenated the geometric features to our image-based features right before the decision networks, i.e., after the feature subnetwork. Note that, for this experiment, we added geometric features at a later stage than for our full model (*Ours*) in order to allow a fair comparison. Adding geometric features consistently improves accuracy across datasets and architectures (Table 3.1). The *Metric* model architecture achieves the best results for *Pasadena*, whereas *Contrastive* works best for *Mapillary*.

3.5.4 Results

Superior performance of simple concatenation of geometric features to visual features (*w/ Geometry*) in comparison to using only visual features (*w/o Geometry*) leaves room for more discriminative, joint feature embedding. *Ours* adds geometric features as a second input in addition to image patches resulting in jointly convolving across geometric and visual cues with the feature subnetworks (ResNet34) at an earlier stage. In fact, using both sources of evidence simultaneously as input results allows the network to reason about their joint distribution. For reasons of consistency, we report the results of *w/o Geometry* and *w/ Geometry* for *Pasadena* and *Mapillary* using the different losses, with the same datasets. Since the *TripleNet* model architecture clearly performed worse for the baseline experiments, we keep only *Contrastive* and *Metric* for evaluating *Ours* (two bottom rows of Table 3.1).

Table 3.1: Matching accuracy (in %) and standard deviation matching results for *w/o Geometry*, *w/ Geometry*, and *Ours* on the two datasets and losses. The best results are marked bold.

	Loss	**Pasadena**	**Mapillary**
w/o Geometry	*Contrastive*	78.0 ± 0.611	93.6 ± 0.04
	Metric	80.1 ± 0.5	67.0 ± 0.73
	TripleNet	72.2 ± 0.67	66.1 ± 0.94
w/ Geometry	*Contrastive*	79.6 ± 0.61	94.4 ± 0.46
	Metric	81.1 ± 0.62	67.6 ± 0.52
	TripleNet	75.6 ± 0.812	67.1 ± 1.1
Ours	*Contrastive*	$\mathbf{81.75 \pm 0.82}$	$\mathbf{96.5 \pm 0.33}$
	Metric	$\mathbf{82.3 \pm 0.22}$	$\mathbf{69.3 \pm 0.96}$



Figure 3.8: Pairs of *Pasadena* candidate matches (**top** and **bottom** rows) that are correctly classified using our method (*Ours*) in comparison to the appearance-based only method (*w/o Geometry*). The first three columns show difficult situations correctly resolved as matches by *Ours* despite significant change in perspective, illumination, and background. Columns 4–6 (from left) show similar looking, neighboring trees correctly classified as not matching by *Ours*. (Imagery © 2019 Google).

Ours consistently outperforms all baseline methods regardless of the architecture. Adding geometric features at input to image patches, thus allowing the network to reason about the joint distribution of geometry and visual evidence, helps further reduce matching errors. Learning soft geometric constraints of typical scene configurations helps differentiate correct from incorrect matches in intricate situations.

Examples for both correct classifications as not matching and matching for hard cases are shown in Figures 3.8 and 3.9. Our method is able to correctly classify pairs of similar looking, neighboring trees as not matching (Figure 3.8), which was the major goal of this work to achieve more reliable object detections for multiple views. In addition, *Ours* also helps establish correct matches in difficult situations of very different viewing angles and occlusion. As for *Mapillary*, *Ours* helps in difficult situations if images are blurred, objects are partially occluded, or a significant perspective change happens, as shown in Figure 3.9. Furthermore, *Ours* correctly classifies image pairs of traffic signs of the same type as not matching even if these are located closely to one another (Figure 3.9).

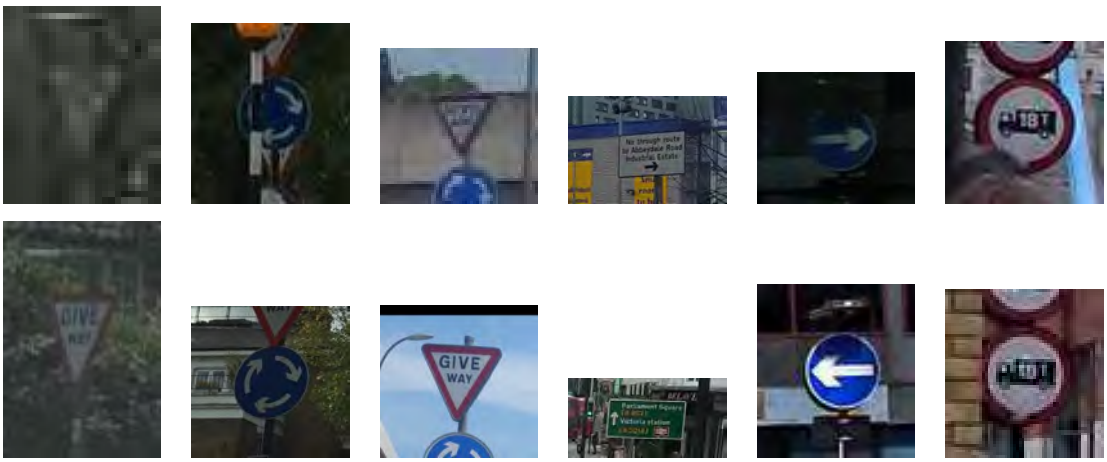


Figure 3.9: Pairs of *Mapillary* candidate matches (**top** and **bottom** rows) that are correctly classified using our method (*Ours*) in comparison to the appearance-based only method (*w/o Geometry*). The first three columns show difficult situations correctly resolved as matches by *Ours* despite significant change in perspective, illumination, and background. Columns 4–6 (from left) show similar looking, neighboring signs correctly classified as not matching by *Ours*.

3.6 Conclusions

We present a Siamese CNN architecture that jointly learns distributions of appearance-based warping functions and geometric scene cues for urban object (i.e., trees and traffic signs) instance matching in the wild. Instead of sequentially imposing hard thresholds based on multi-view photogrammetric rules, joint learning of appearance and geometry enables cross-talking of evidence inside a single network. While our network design is only a slightly adapted version of existing Siamese CNN architectures, adding geometry to image evidence consistently improves object instance matching results for both *Pasadena* and *Mapillary* datasets. Our hope is that this idea of "learning soft photogrammetric constraints" and combining them with object appearance will unleash a whole new line of research that models warped image content and relative sensor orientation jointly. For example, learned, soft photogrammetric constraints can help improving object detection across multiple views [103], for which in this study we explored different methods of incorporating the soft photogrammetric constraints in order to further experiment with learned end-to-end methods. Learning geometric constraints as soft priors jointly with image evidence will help in many situations where camera and object poses are ill-defined, noisy, or partially absent as well as for point cloud registration [104], [105].

Chapter 4

Simultaneous multi-view instance detection with learned geometric soft-constraints

Nassar, Ahmed Samy, Sébastien Lefèvre, and Jan Dirk Wegner.
"Simultaneous multi-view instance detection with learned geometric soft-constraints."
IEEE International Conference on Computer Vision, 2019 (ICCV 2019).
(Author version; for typeset version please refer to the original journal paper.)

4.1 Abstract

We propose to jointly learn multi-view geometry and warping between views of the same object instances for robust cross-view object detection. What makes multi-view object instance detection difficult are strong changes in viewpoint, lighting conditions, high similarity of neighbouring objects, and strong variability in scale. By turning object detection and instance re-identification in different views into a joint learning task, we are able to incorporate both image appearance and geometric soft constraints into a single, multi-view detection process that is learnable end-to-end. We validate our method on a new, large data set of street-level panoramas of urban objects and show superior performance compared to various baselines. Our contribution is threefold: a large-scale, publicly available data set for multi-view instance detection and re-identification; an annotation tool custom-tailored for multi-view instance detection; and a novel, holistic multi-view instance detection and re-identification method that jointly models geometry and appearance across views.

4.2 Introduction

We propose a method to simultaneously detect objects and re-identify instances across multiple different street-level images using noisy relative camera pose as weak supervision signal. Our method learns a joint distribution across camera pose and object instance warping between views. While object detection in single street-level panorama images is straightforward since the introduction of robust, deep learning-based approaches like Faster R-CNN [106] for object detection or Mask R-CNN [107] for instance segmentation, establishing instance correspondences across multiple views with this wide baseline setting is very challenging due to strong perspective change between views. Moreover, Google street-view panoramas, which are our core data in this paper, are stitched together from multiple individual photos leading to stitching artefacts in addition to motion-blur, rolling shutter effects etc. that are common for these type of mobile mapping imagery. This makes correspondence search via classical structure-from-motion methods like [108], [109] impossible.

Our core motivation is facilitating city maintenance using crowd-sourced images. In general, monitoring street-side objects in public spaces in cities is a labor-intensive and costly process in practice today that is mainly carried out via in situ surveys of field crews. One strategy that can complement greedy city surveillance and maintenance efforts is crowd-sourcing information through geo-located images like proposed for street trees [25], [26], [55]. We follow this line of work, but propose an entirely new simultaneous multi-view object instance detection and re-identification method that jointly reasons across multi-view geometry and object instance warping between views. We formulate this problem as an instance detection and re-identification task, where the typical warping function between multiple views of the same tree (Fig. 4.2) in street-view panoramas is learned together with the geometric configuration. More precisely, instead of merely relying on image appearance for instance re-identification, we insert heading and geo-location of the different views to the learning process. Our model learns to correlate typical pose configurations with corresponding object instance warping functions to disentangle multiple possibly matching candidates in case of ambiguous image evidence.

Our contributions are (i) a novel multi-view object instance detection and re-identification method that jointly reasons across camera poses and object instances, (ii) a new object instance re-identification data set with thousands of geo-coded trees, and (iii) a new interactive, semi-supervised multi-view instance labeling tool. We show that learning geometry

and appearance jointly end-to-end significantly helps improving object detections across multiple views as well as final geo-coding of individual objects.

4.3 Related Work

We are not aware of any work that does simultaneous object detection and instance re-identification with soft geometric constraints. But our proposed method touches a lot of different research topics in computer vision like pose estimation, urban object detection, object geo-localization, and instance re-identification. A full review is beyond the scope of this paper and we thus provide only some example literature per topic and highlight differences with the proposed work.

Pose estimation: Learning to predict camera poses using deep learning has been made popular by the success of PoseNet [37] using single RGB images and many works have been published since then [110]–[112]. Tightly coupling pose with image content can, for example, be helpful for estimating a human hand's appearance from any perspective if seen from only one viewpoint [46]. Full human pose estimation is another task that benefits from combined pose reasoning across pose and scene content like [45] who employ a multi-task CNN to estimate pose and recognize action. In this paper, we rely on public imagery without fine-grained camera pose information.

Urban object detection: A large body of literature addresses urban object detection from an autonomous driving perspective with various existing public benchmarks, e.g. KITTI [20], CityScapes [21], or Mapillary [102]. In these scenarios, dense image sequences are acquired with minor viewpoint changes in driving direction with forward facing cameras. Such conditions make possible object detection and re-identification across views [113]–[115]. In our setup, significant changes occur between views, thus making the re-identification problem much more challenging.

Object geo-localization: Geo-localization of objects from Google street-view imagery with noisy pose information was introduced in [25], [26]. In a similar attempt, [116] geo-localize traffic lights and telegraph poles by applying monocular depth estimation using CNNs, then using a Markov Random Field model to perform object triangulation. The same authors extend their approach by adding LiDAR data for object segmentation, triangulation, and monocular depth estimation for traffic lights [27]. [28] propose a CNN-based object detector for poles and apply a line-of-bearing method to estimate the geographic object position. We rather suggest here to follow an end-to-end learning

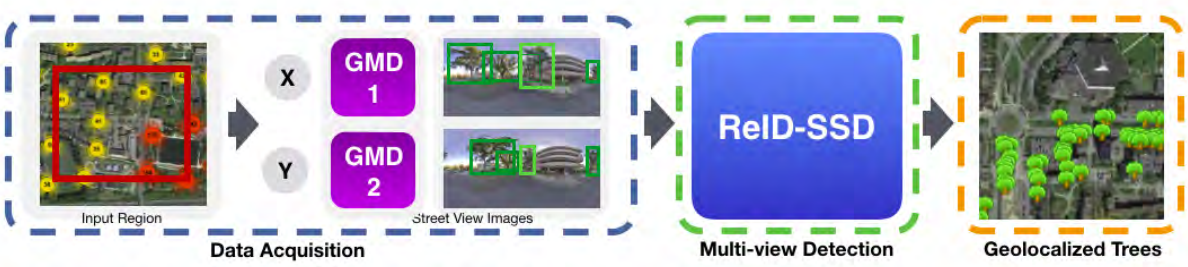


Figure 4.1: A pair of images is fed to our multi-view object detectors, matching projected predictions is learned, and the geo-coordinate of the object predicted.

strategy.

Instance re-identification: Matching image patches can be viewed as a simple version of re-identifying image content across different views, e.g. in structure-from-motion [82], tracking [117], super-resolution [118], depth map estimation [81], object recognition [119], image retrieval [120], and image classification [121]. Our scenario is closely related to works on re-identifying object instances across views. Siamese CNNs have been established as a common technique to measure similarity, e.g. for the person re-id problem that tries to identify a person in multiple views [75]). [89] detect and re-identify objects in an end-to-end learnable CNN approach with an online instance matching loss. [34] solve re-identification with a so-called center loss that tries to minimize the distance between candidate boxes in the feature space. In contrast to prior work [25]–[27], which does detection, geo-coding and re-identification in a hierarchical procedure, our method does it simultaneously in one pass. Methods based on Siamese models [75] alone are not a viable solution to our problem, since they need image crops of the object and can not fully utilize re-identification annotations due to their pairwise labeling training setup. [34] searches for a crop within the detections in a gallery of images, in comparison to our method which aims at matching detections from both full images. The key differences between our work and [89] is that we both ensure object geolocalization and avoid storing features from all identities since that is impractical in a real world application like the one considered in the paper where objects actually look very similar in appearance.

4.4 Multi-view detection and instance re-identification

Our method learns to detect and re-identify object instances across different views simultaneously. We compensate for inaccurate or missing image evidence by learning a joint distribution of multi-view camera poses together with the respective warping function of object instances. Intuitively, our method learns to correlate a particular geometric pose setup (e.g., an equilateral triangle, a right triangle, etc.) with the corresponding change of object appearance in the images. As shown in Fig. 4.3, trees in many situations being the same species, and planted the same time look very similar making it hard to detect or re-identify. Learned relative camera pose configurations help re-identifying object instances across views if appearance information in the images is weak while strong image evidence helps improving noisy camera pose. In general, one can view the relative camera pose estimation task as imposing soft geometric constraints on the instance re-identification task. This joint reasoning of relative camera poses and object appearance ultimately improves object detection, instance re-identification, and also the final object geo-coding accuracy.

A big advantage of this simultaneous computation of relative camera poses, object instance warping and finally object geo-coding is that the model learns to compensate and distribute all small errors that may occur. It thus implicitly learns to fix inaccurate relative poses relying on image evidence and vice versa. An overview of the architecture of our method is shown in Fig. 4.4. The basic layout follows a Siamese architecture as proposed originally by [60]. The main concept of Siamese CNNs is constructing two identical network branches that share (at least partially) their weights. Features are computed for both input images and then compared to estimate the degree of similarity. This can be achieved by either evaluating a distance metric in the feature space or by evaluating the final classification loss. Here, our primary data source are Google street-view (GSV) panoramic images along with their geographic meta data (GMD) because they are publicly available at a global scale, fit our purpose of city-scale object mapping for maintenance purposes, and constructing large data sets amenable to deep learning is straightforward. Fig. 4.2 illustrates the setup of the problem, where the GSV panoramas captured from C^* contain our object of interest T from different viewpoints. The GMD contains many useful properties of the panorama image at hand but location in latitude and longitude as well as yaw are rather inaccurate. Since we do not have any information regarding C's intrinsic or extrinsic properties, we rely on the GMD to use in our projection

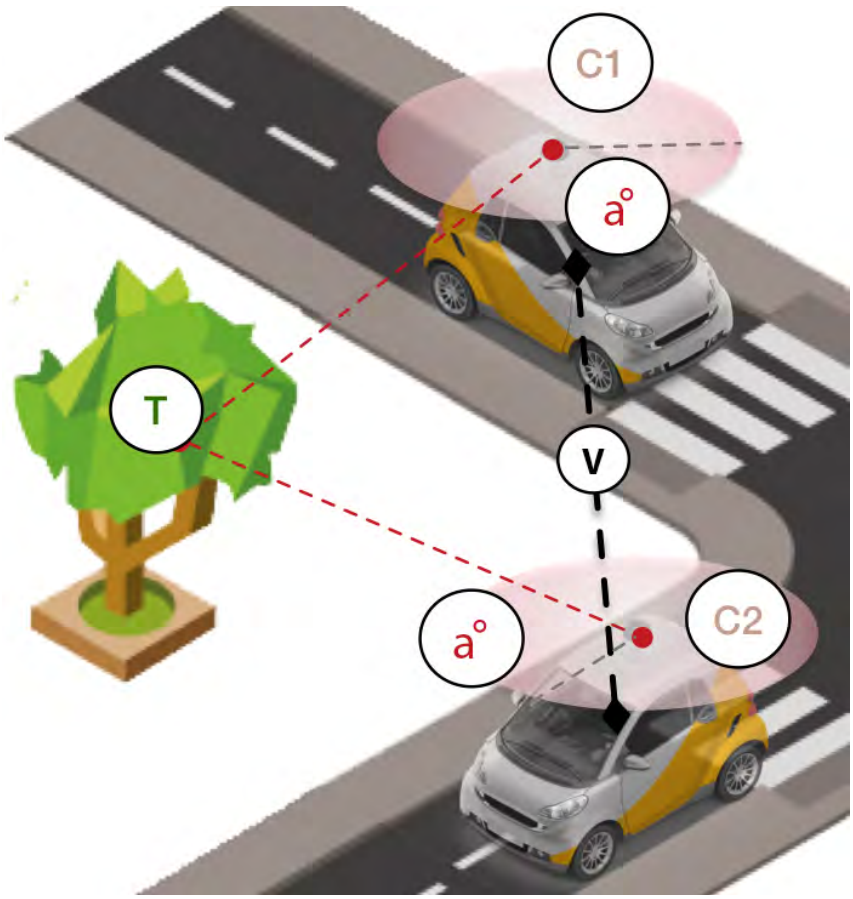


Figure 4.2: C^*: Camera with geo-position. T: The tree has its actual geographic coordinates, and location within the panorama. $a°$: heading angle inside panorama. v: Distance between cameras.

functions which plays an important role as we will present in the upcoming parts of the paper. GMD is also contains IDs of other images in vicinity.

4.4.1 Multi-view object detection

Our core object detection network component is based on the single shot detector (SSD) [15]. Our architecture is generally detector-agnostic and any detector could replace SSD if desired. We chose SSD over other prominent methods like Faster R-CNN [106] because SSD provides an easy implementation that allows intuitive modifications and it performs faster with fewer classes, like in our case, while achieving good accuracy [122]. We chose SSD512 [15] as our preferred architecture, which sacrifices a bit of computational speed for better accuracy.

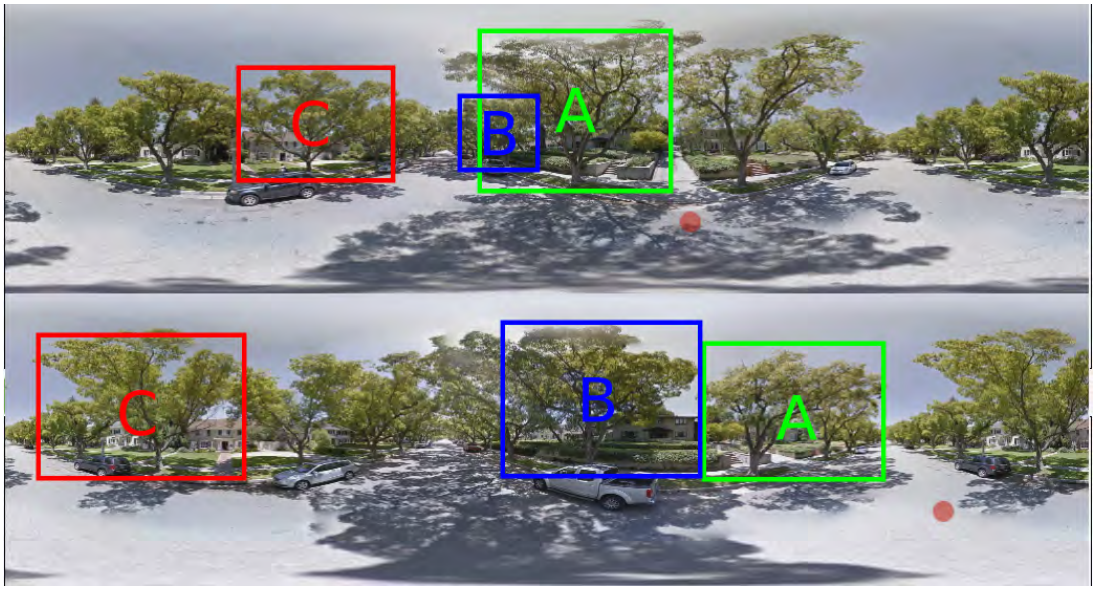


Figure 4.3: Tree instance re-identification problem (color indicates matches): each tree is photographed from multiple different views, changing its size, perspective, and background. Note that many trees look alike.

Our network is composed of two identical blocks denoted as $\boldsymbol{X}$ and $\boldsymbol{Y}$ (Fig. 4.1). As shown in Fig. 4.4, each block receives an image, camera pose information (geometric meta data, GMD), and the ground truth during training. Note that the camera's pose information C only contains its location $\ell = (\text{lat}, \text{lng})$, yaw, and height h, which is passed to the network denoted as GMD in Fig. 4.4. From the GMD data we are able to calculate the distance between the cameras, and the heading angle inside the panorama toward the object, see Eq. (4.1). Ground truth is composed of two types of bounding boxes: (i) regular object bounding boxes and (ii) bounding boxes that carry instance IDs labeled and geo-coordinates. Each image passes first through the SSD base network composed of ResNet-50 modules [96]. It is then subject to the convolutional feature layers that provide us with detections at multiple scales. In order to prepare for instance re-identification, each individual object detection is given a *local ID*, which will play a role in the multi-view instance matching stage later on. All detections per network block (i.e., panorama) are then projected during training into the other block's space using our geometric projection, see Eq. (4.1) & (4.2). Predictions generated from $\boldsymbol{X}$ and $\boldsymbol{Y}$ are passed through a projection function that estimates their real world geographic position. From this position it is again projected into pixels into the corresponding view. These projection functions assume that the local terrain is flat to simplify the problem. Objects T are represented inside street view images in local East, North, Up (ENU) coordinates that are calculated by providing

C_l, C_h and T_l using Eq. (4.1). To obtain the pixel location of the object $O_{x,y}$, Eq. (4.2) is used given R as the Earth's radius, W and H the image's width and height respectively, and z the estimated distance from C calculated by $z = \sqrt{e_x^2 + e_y^2}$.

$$(e_x, e_y, e_z) = \Big(R\cos[\mathrm{C}_{lat}]\sin[\mathrm{T}_{lng} - \mathrm{C}_{lat}], R\sin[\mathrm{T}_{lat} - \mathrm{C}_{lat}], -\mathrm{C}_h\Big) \tag{4.1}$$

$$x = (\pi + \arctan(e_x, e_y) - \mathrm{C}_{yaw})\, W/2\pi y = (\pi/2 - \arctan(-h, z))\, H/\pi \tag{4.2}$$

Blindly projecting bounding boxes between panoramas would, however, ignore any scale difference between different images of the same instance. Since the mapping vehicle is moving along while acquiring images, objects close to the camera in one image will likely be further away in the next. In addition, detected bounding boxes may sometimes be fitting an object inaccurately due to partial occlusions or simply poor detector performance. Using the above mentioned equations that assume flat terrain, these errors would results in projections meters away from the true position. We thus add a dense regression network to regress the predicted bounding boxes to the ground truth of the other block once projected. For example, $\boldsymbol{X}$'s projected predictions are regressed to $\boldsymbol{Y}$'s ground truth, and vice versa. This component (Geo Regression Net) aims at taking the predicted boxes, and projected boxes location, and regress them to their real world geo-coordinates through a densely layered network.

Geo Regression Net: Inspired by [123], [124], this network component estimates the geo-coordinates of the detected bounding boxes. Note that our "Projection Function" component (based on Eq. (4.1) and (4.2)) provides initial estimates for geo-coordinates, which are improved with this component. The Geo Regression Net consists of two dense layers with ReLU activations.

Projection Net: This network component fine-tunes projected predictions b'_* by learning to regress the discrepancy between them and the other block's ground truth as illustrated in Fig 4.5. Projection Net is constructed similar to the extra feature layers (Fig. 4.4), but uses only box regression layers.

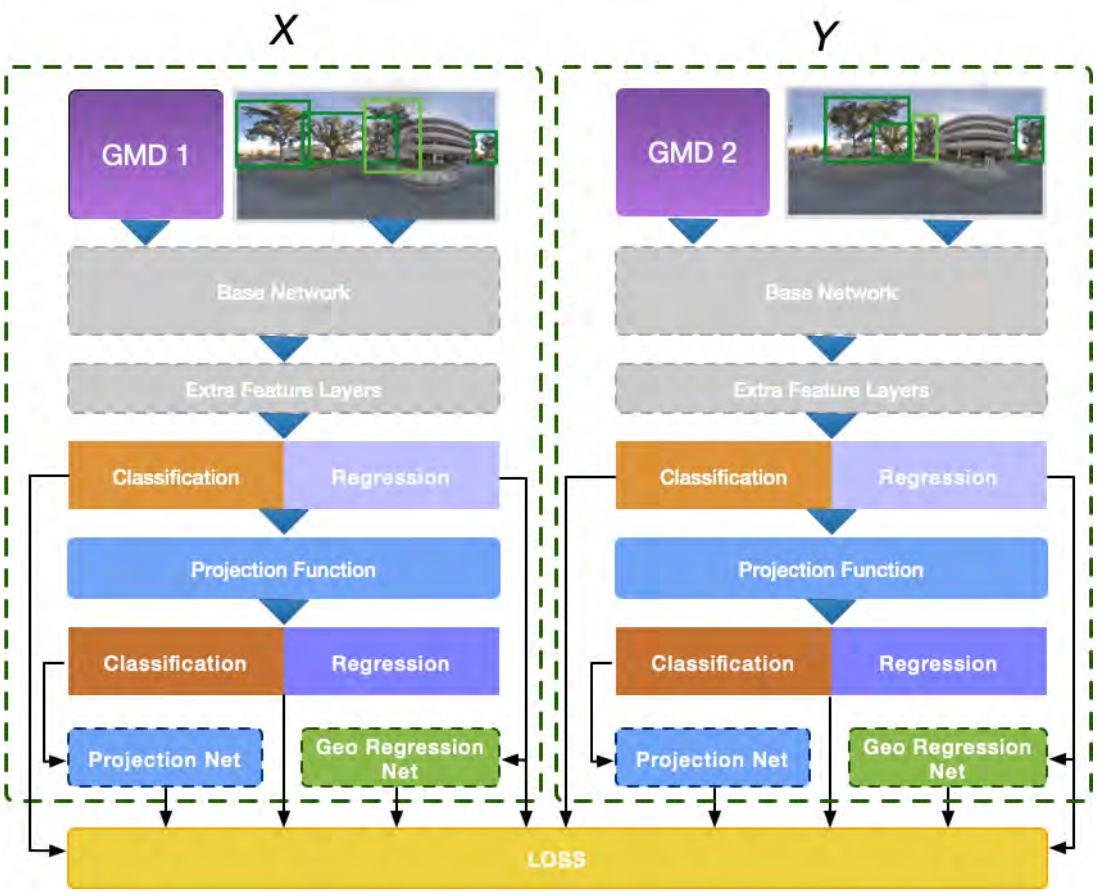


Figure 4.4: Our network design: Images along with their GMD are inputs to the network. However, the GMDs are only used inside the "Projection Function" component. Object bounding boxes and scores for each class are computed via extra feature layers (i.e., Conv4_3 [15]). These are projected into the other image's space (i.e. to the other network block, from X to Y and from Y to X), and input to a dense *Geo Regression Net* to estimate geographic coordinates. Finally, projected predictions are input to the *Projection Net* network component that does some fine-tuning of the projections.

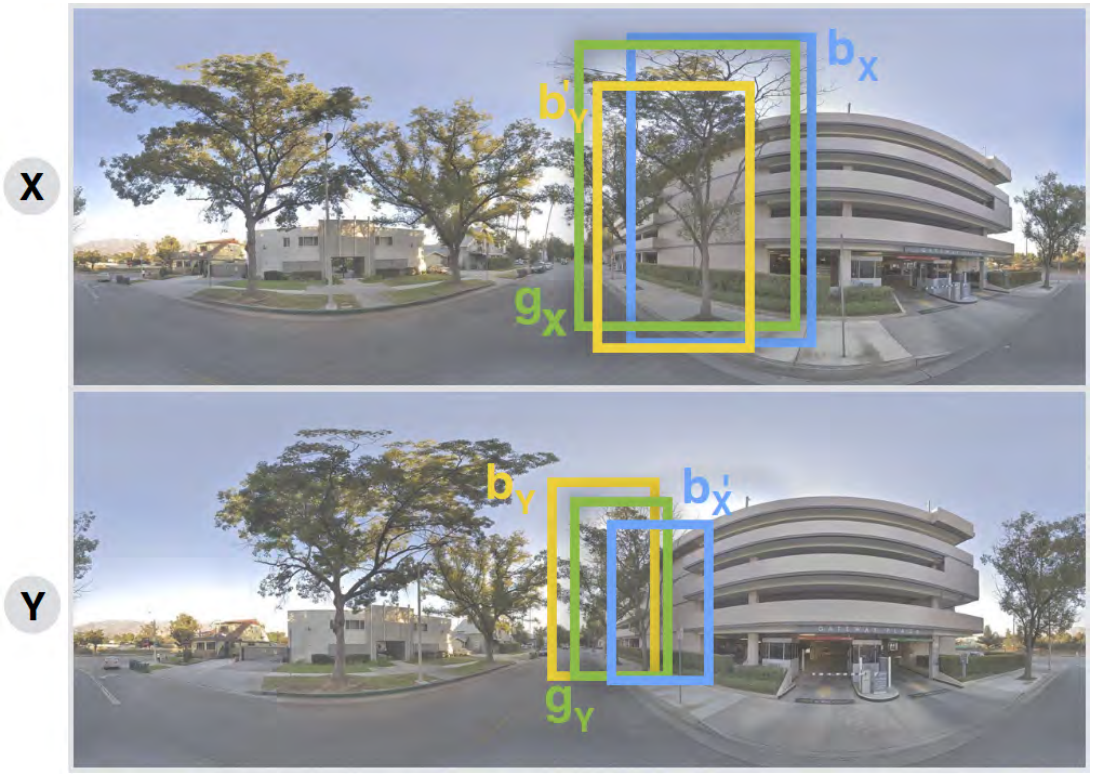


Figure 4.5: Illustration of predictions (b_*) projected (b'_*) in other views and their ground truth (g_*).

4.4.2 Loss function

We formulate a multi-task loss in Eq. (4.3) to train our network. Similarly to SSD [15], we use a softmax log loss L_{conf} for classification and a smooth-L1 loss L_{loc} for bounding box regression. As shown in Fig. 4.5, our predictions b are projected into b' using the projection function. We use again L_{loc} but this time using the ground truth g of the other image g', since it contains the actual bounding boxes we are trying to regress to in order for the *"Projection Network"* to regress the projected boxes. However, mapping which predicted bounding boxes correspond to which default boxes x in g' is not a straightforward task due to how the default boxes are generated systematically:

— boxes inside g and g' are filtered (f_g and f'_g) by keeping only the identified objects (ID'd) to ensure that we are regressing each instance to its corresponding box in the other image,
— b is matched using IoU with f_g to estimate which boxes are our target identities,
— indices of the boxes targeted are then selected to be used as inputs into our loss function, with f'_g as ground truth.

The Geo Regression Net network component is trained using a RMSE L_{RMSE} loss. For the re-identification task, we train both base networks $\boldsymbol{X}$ and $\boldsymbol{Y}$ using a contrastive loss L_{cont} by feeding features from x and x' that are of identified objects as input to learn discriminative features and pull them close if similar. Our complete, multi-task loss function is:

$$\begin{aligned} L_{com}(x, c, b, g) = \frac{1}{N}(L_{conf}(x, c) + \alpha L_{loc}(x, b, g) \\ +\alpha L_{loc'}(x, b', g, g') + L_{cont}(x, g) + L_{RMSE}(b, g)) \end{aligned} \tag{4.3}$$

During inference, predicted boxes b_* are combined in each view creating a large number of candidate boxes. As in the original implementation of SSD [15], we use a classification confidence threshold of 0.01 to filter redundant boxes. Afterwards non-maximum suppression (NMS) with Jaccard index (IoU) is employed using a 0.5 overlap. As mentioned in Sec. 4.4.1 the *local IDs* assigned are used to find which remaining candidate bounding boxes when projected, overlap's with the other view's candidate boxes (i.e. $b_X \cap b'_Y$), from which we can identify the corresponding boxes. Simultaneously, by calculating the distance between the candidate boxes from each view using Euclidean distance, we are able to match corresponding boxes.

4.5 Experiments

We validate our method with experiments on two different data sets with street-level imagery. The first data set consists of GSV panoramas, meta data, and tree object instance labels across multiple views. The second data set contains sequences of Mapillary images acquired with dash cams where object instances are labeled across multiple consecutive image acquisitions. In addition to presenting final results of our end-to-end learnable multi-view object instance detection and re-identification approach, we also do a thorough ablation study to investigate the impact of each individual component.

4.5.1 Data sets

Pasadena Multi-View ReID: We build a new multi-view data set of street-trees, which is used as a test-bed to learn simultaneous object detection and instance re-identification with soft geometric constraints. The original Pasadena Urban Trees data set [25] contains 1,000 GSV images labeled using Mechanical Turk without explicit instance labels across multiple views. We construct a new *Multi-View ReID* data set for our purpose where each tree appears in those four panoramas that are closest to a particular tree location. In total, we label 6,020 individual tree objects in 6,141 panoramas with each tree appearing in four different panoramas. Each panorama image is of size 2048 x 1024 px. This creates a total of 25,061 bounding boxes, where each box is assigned a particular tree ID to identify the different trees across panoramas. The annotations per image include the following: (i) bounding boxes identified (ID'd) and unidentified, (ii) ID'd bounding boxes include the geo coordinate position, distance from camera v, heading angle a, (iii) image's dimensions, and geo-coordinates. For validating our method experimentally, we split the data set into 4298 images for training, 921 for validation, and 922 for testing. Since we are not aware of any existing multi-view instance labeling tool for geo-located objects, we created a new one described in the sequel.

Mapillary: In order to verify if our method generalizes across different data sets, we run experiments on a data set provided by Mapillary [1]. Note that this particular data set is different from the well-known Mapillary Vistas dataset [102], which provides images and semantic segmentation. Our data set at hand is composed of 31,442 traffic signs identified in 74,320 images and carries instance IDs across views in an area of approximately $2km^2$.

1. www.mapillary.com

On average, two traffic signs appear per image. This data set comes as GeoJSON "FeatureCollection" where each "feature" or identity contains the following properties that were used: (i) the object's geo-coordinate that is achieved by using 3D structure from motion techniques, therefore it is affected by the GPS, and the density of the images, (ii) the distance in meters from the camera position, (iii) image keys in which the object appears in and which is used to retrieve the image using their API, (iv) geo-coordinates of the image location, (v) the object's altitude, and (vi) an annotation in polygon form of the sign.

The Mapillary data set significantly differs from our tree data set in several aspects. Images were crowd-sourced with forward looking dash cams attached to moving vehicles, and by walking humans using smart phones. Image sizes and image quality are thus inconsistent across the data set. Viewpoint changes between consecutive frames are only of a few meters, and the field of view per image is much smaller than a GSV panorama as shown in Fig. 4.6. Consequently, the distribution of relative poses between viewpoints is very different as well as the change in appearance of the same object instance across views. Because the camera is forward-looking, each object is viewed more or less from the same viewpoint, only scale changes. However, objects are generally smaller because unlike GSV, no orthogonal views perpendicular to the driving direction exist.



Figure 4.6: Consecutive frames of two example scenes of the Mapillary data set.

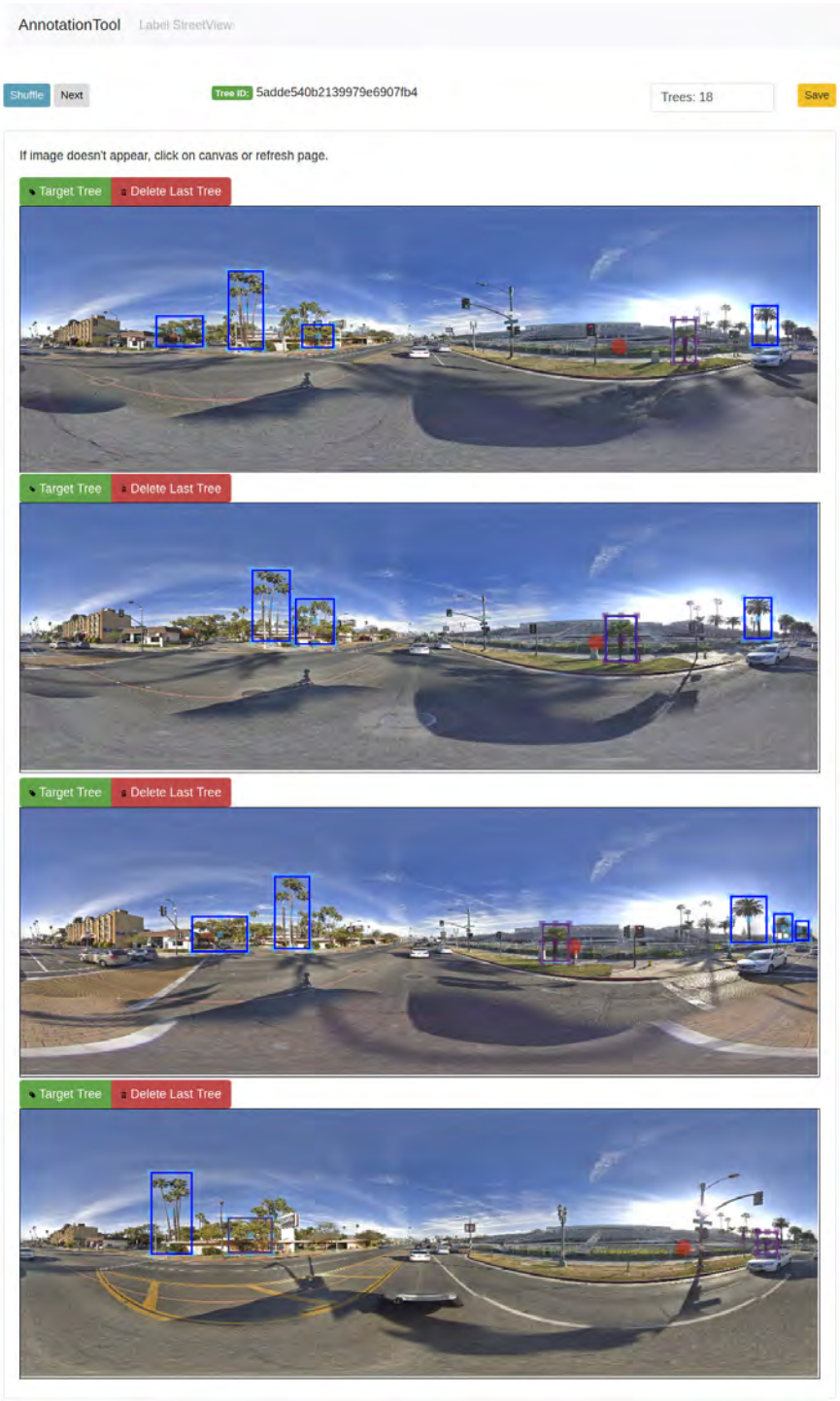


Figure 4.7: Our annotation tool provides 4 multi-view panoramas from GSV. Initial bounding boxes for the target object are predicted, in which annotators can then refine, or annotate the missing object. To help identifying the object in multiple views a red circle is drawn to estimate the location of the target object in all views to guide the annotator. Images in the figure are from our Multi-View ReID dataset in Pasadena.

4.5.2 Multi-view object annotation tool

Labeling object instances across multiple panoramas is a difficult task (Fig. 4.3) because many trees look alike and significant variations in scale and viewpoint occur. Our annotation tool aims at making multi-view instance labeling more efficient by starting from an aerial view of the scene. To begin labeling, the annotator first selects an individual object from the aerial image. The four closest panoramas are presented to the annotator and in each view a marker appears that roughly points at the object location inside each panorama. This projection from aerial view to street-view panorama approximates the object's position in each of the panoramas and is calculated using Eq. (4.1) and (4.2). This initial, approximate object re-identification significantly helps a human observer to identify the same tree across different images despite large scale and viewpoint changes. Moreover, SSD predicts bounding boxes around the objects of interest (here: trees) such that the annotator can simply refine or resize an existing bounding box in most cases (or create a new bounding box if the object remains undetected). Identity labeling or correspondence matching is done by selecting the best fitting bounding box (i.e., there may sometimes be more than one bounding box per object) per object per panorama. All multi-view instance annotations are stored in a MongoDB, which enables multiple annotators to work on the same data set at the same time. The database is designed to store annotated bounding boxes to each image, with a separate document storing which bounding boxes are identities. This reduces the effort of having to reannotate each image again for every identity. Our labeling tool is generic in terms of object category and can easily be adapted to any category by re-training the detector component for a different class. Also the detector can be exchanged for any other object detector implemented/wrapped in Python. As output the labeling tool provides annotations through its API in VOC and JSON format.

4.5.3 Detection

A significant benefit of projecting bounding boxes between blocks of our architecture is that it makes object detection much more robust against occlusions and missed detections due to missing image evidence in individual images. In order to validate the improvement that is due to simultaneously detecting objects across multiple view, we compare object detector results on individual panoramas (Monocular) with results from our model that combine object evidence from multiple views via projecting detections between blocks X

Method	w/o Pose [mAP]	w/ Pose [mAP]
FaceNet [125]	0.808	0.842
ResNet-50 [96]	0.828	0.863
MatchNet [82]	0.843	**0.871**

Table 4.1: Re-identification results without (w/o Pose) and with (w/ Pose) camera pose information (C_l^*, v, a), fed to the Siamese network architectures FaceNet, ResNet-50, and MatchNet.

and Y. Results are shown in Tab. 4.2. *Ours* improves detection mAP on the Pasadena tree data set by 8.5 percent points, while improving by 2.7 percent points on the Mapillary data set.

4.5.4 Re-identification with pose information

We verify if learning a joint distribution across camera poses and image evidence supports instance re-identification (regardless of the chosen architecture) with three popular Siamese architectures, namely FaceNet [125], ResNet-50 [96], and MatchNet [82]. Results shown in Tab. 4.1 indicate that *Ours* with camera pose information consistently outperforms all baseline methods regardless of the base network architecture. Any architecture with added geometric cues does improve performance. Learning soft geometric constraints of typical scene configurations helps differentiating correct from wrong matches in intricate situations. Overall, *Ours* with the MatchNet [82] architecture performs best. For implementation purposes ResNet-50 was chosen to be used as the base network for our final architecture to be able to load available pre-trained weights.

We evaluate the Re-ID mAP for our multi-view setting, which measures the amount of correct instance re-identifications if projecting detections between panoramas in the right column of Tab. 4.2. To measure the similarity between tree detections across multiple views projected onto one another, we use the distance between overlapping bounding boxes as explained in the inference stage. 73% of all tree instances labeled with identities are matched correctly, which is a high number given the high similarity between neighboring

trees and the strong variation in scale and perspective. As for Mapillary's dataset, 88% of the traffic signs were re-identified. In comparison to tree objects, traffic signs feature wise are much more discriminate, but the object scale is much smaller in size.

4.5.5 Geo-localization

We finally evaluate performance of our end-to-end trainable urban object mapping architecture by comparing predicted geo-locations of trees with ground truth positions. We compare our full, learned model (*Ours*) against simply projecting each detection per single panorama (Single) to geographic coordinates as well as combining detections of multiple views (Multi) (Tab. 4.3). We compute the discrepancy between predicted geo-coordinates and ground truth object position using the haversine formula given in Eq. (4.4) with r being the Earth's radius (6,372,800 meters):

$$d = 2r \arcsin\left(\left(\sin^2\left(\frac{b_{lat} - g_{lat}}{2}\right) + \cos(g_{lat})\cos(b_{lat})\sin^2\left(\frac{b_{lng} - g_{lng}}{2}\right)\right)^{0.5}\right) \quad (4.4)$$

Single view geo-localization was done by applying projection functions given in Eq. (4.1) and (4.2) to the individual detections. As for multi-view experiments, we use combined detections from multiple views without learning the projection and project to geographic coordinates as before. *Ours* is our full model as depicted in Fig. 4.4, which takes advantage of "Projection Net" and "Geo Regression Net" components. Learning multi-view object detection and instance re-identification significantly improves performance, bringing down the MAE to 3.13 meters for the Pasadena trees data set while achieving 4.36 meters for Mapillary. Fig. 4.9 shows tree detection results (red) for a small example scene in comparison to ground truth locations (orange) overlaid to an aerial view.

4.6 Conclusion

We have presented a new, end-to-end trainable method for simultaneous multi-view instance detection and re-identification with learned geometric soft-constraints. Quantitative results on a new data set (labeled with a novel multi-view instance re-identification annotation tool) with street-level panorama images showed promising performances. Experiments on a Mapillary data set with shorter baselines, tinier objects, smaller field of view, and mostly forward looking cameras indicates that our method generalizes to a

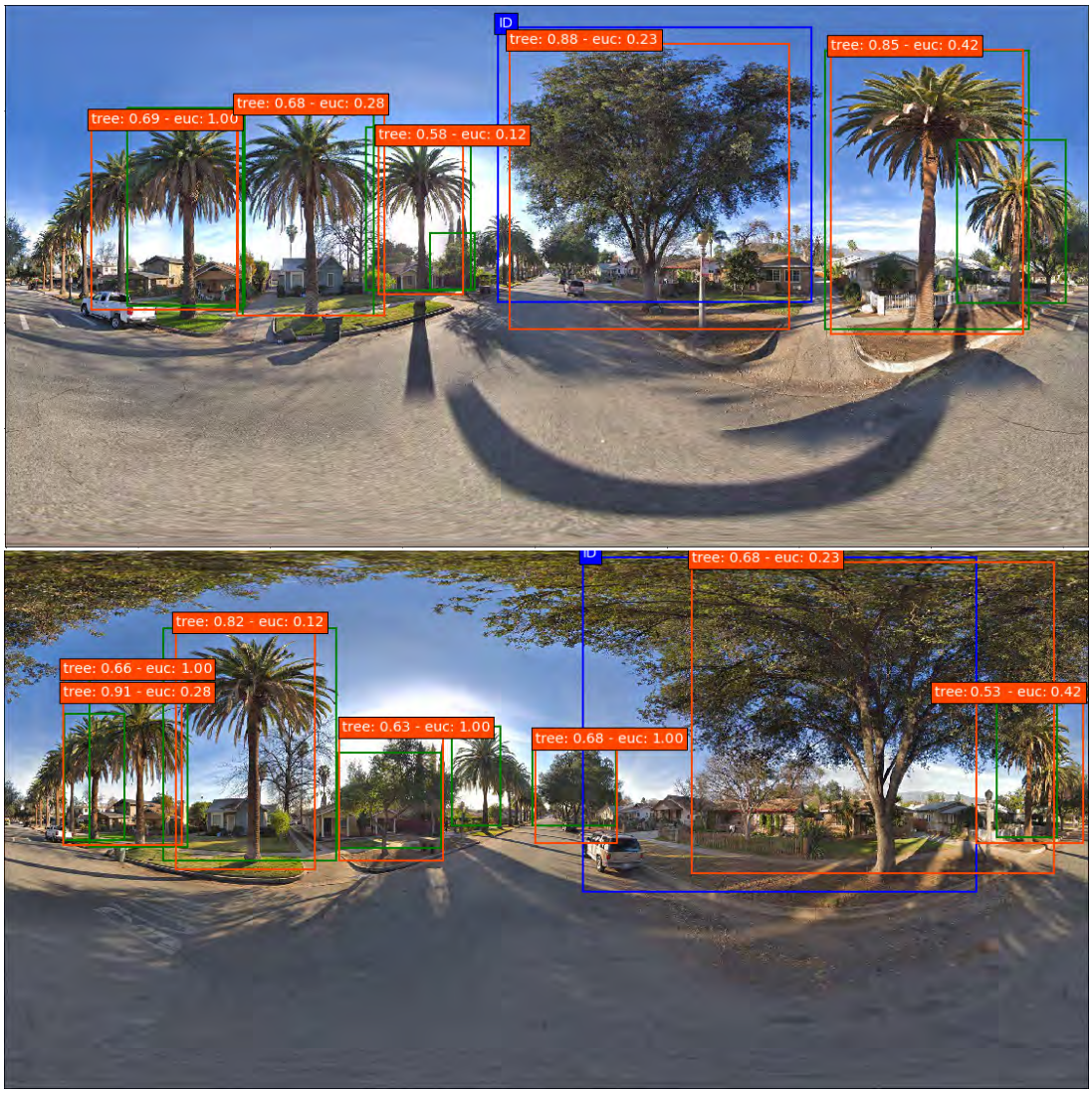


Figure 4.8: Detection and Re-identification using our method. Green: ground truth boxes. Blue: ground truth box of the identity instance. Orange: predictions with classification score and calculated feature distance from matching box in the other view.

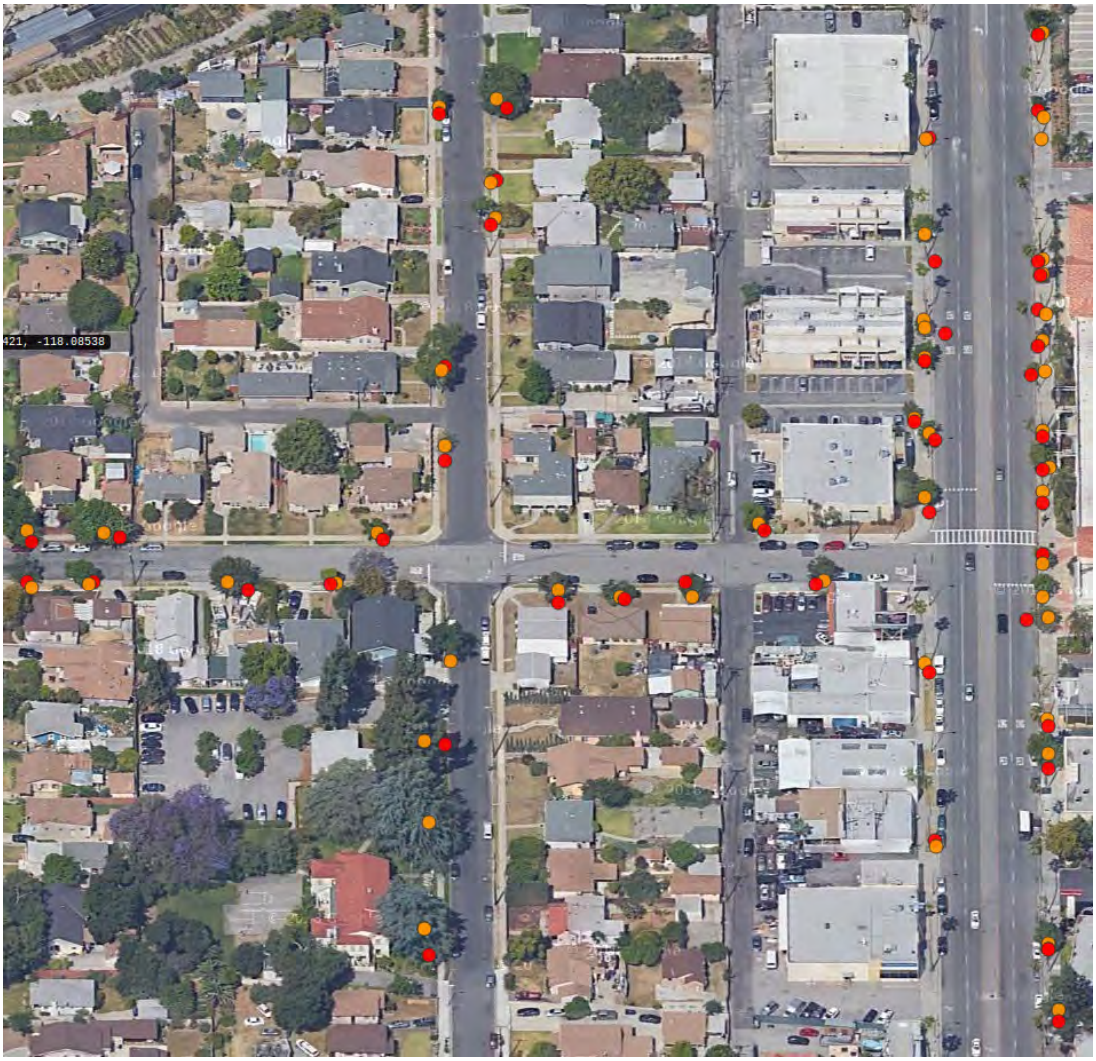


Figure 4.9: Small subset of tree predictions (red) overlaid to an aerial image (not used in our model) and compared to tree ground truth locations (orange).

Method	Data set	Det. mAP	Re-ID mAP
Monocular	Pasadena	0.597	-
	Mapillary	0.875	-
Ours	Pasadena	0.682	0.731
	Mapillary	0.902	0.882

Table 4.2: Detection and Re-identification results with individual, single-view object detections (*Monocular*) compared to our full, multi-view pipeline (*Ours*).

different acquisition design, too.

In general, integrating object evidence across views improves object detection and geo-localization. In addition, our re-identification ablation study proves that learning a joint distribution across camera poses and object appearances helps re-identification. We hope tight coupling of camera pose information and object appearance within a single architecture will benefit further research on multi-view object detection and instance re-identification in the wild. All source code, the tree detection and re-identification data set, and our new labeling tool will be made publicly available upon acceptance of this paper.

Method	Data set	MAE [m]
Single	Pasadena	77.41
	Mapillary	83.27
Multi	Pasadena	70.16
	Mapillary	64.0
Ours	Pasadena	**3.13**
	Mapillary	**4.36**

Table 4.3: Geo-localization results as mean absolute error (MAE) compared to geographic ground truth object positions.

Chapter 5

GeoGraphV2: Graph-Based Aerial & Street View Multi-view Object Detection with Geometric Cues End-to-End

---

Nassar, Ahmed Samy, Sébastien Lefèvre, and Jan Dirk Wegner. "GeoGraphV2: Graph-Based Aerial & Street View Multi-view Object Detection with Geometric Cues End-to-End"

(Unpublished)

Extension to:

Nassar, Ahmed Samy, Sébastien Lefèvre, Stefano D'Aronco, and Jan Dirk Wegner.

"GeoGraph: Graph-Based Multi-view Object Detection with Geometric Cues End-to-End."

European Conference on Computer Vision, 2020 (ECCV 2020).

5.1 Abstract

This paper presents a new approach that detects, reidentifies, and geo-localizes static urban objects in multiple ground-level and aerial views using an end-to-end learnable method. Our method constructs a network composed of an object detector and a Graph Neural Network (GNN). Our GNN receives image evidence from the object detector, coupled with relative pose information acquired from the image's metadata to reidentify an object instance across multiple views to aggregate the same object's different instances under one identity. By jointly coupling image evidence and relative pose, our method is robust to occlusion, similar appearance of objects in close proximity. We perform an experimental evaluation on two challenging extensive datasets of trees and street signs

and compare with state-of-the-art methods. We present substantial improvement in terms of re-identification accuracy, and efficiency. We also present an extension to a data set to include aerial imagery annotated with instances appearing in ground-level imagery.

5.2 Introduction

We present a novel end-to-end approach capable of detecting and re-identifying static urban objects throughout multiple views and modalities by using Graph Neural Networks (GNNs). Images are naturally classified as structural data in which its pixels are in a grid. However, many data exist in the form of unstructured, which can be represented in a graph form. The applications of GNNs to graph-related problems vary across different fields such as predicting molecular properties for chemical compounds [126], [127] and proteins [128], social influence prediction [129], object tracking [36], [130], or detection of fake news [131]. Given aerial and ground-level imagery of a scene or location, we propose to detect and re-identify objects (trees, street signs) in multi-view and assign them estimated geographic coordinates by representing the objects detected as a graph. We solve this problem by relying on Graph Neural Networks, which enables us to represent detections from multiple views as a graph. With the advent of many digital map services, urban maps have become part of our everyday life. Therefore, creating and updating maps of urban objects depends on various applications such as inventory management by local municipalities or generating HD Maps for autonomous driving. Producing such maps relies upon surveys from field workers. Fortunately, the large availability of street view images alongside recent computer vision advancements can primarily facilitate this task. In fact, it is possible to use street images to identify the different objects and, simultaneously, map them in a fully automatized way. This paper proposes a method that uses as input multiple panorama images, performs object re-identification over these images, and ultimately geo-localizes each identified object. Furthermore, covering large areas such as cities instigates the reliance on different data sources for a mass aggregation approach as suggested for street trees [23], [25], [55], [116], [132]. As for self-driving, high definition (HD) maps include a component that catalogs static urban objects such as street signs, which help navigate the vehicle from source to destination. Being able to regress objects' location in the 3D world from images is arguably one of the most studied problems in photogrammetry and computer vision. However, each different variant of this problem reveals some particular challenges that need to be specifically addressed.

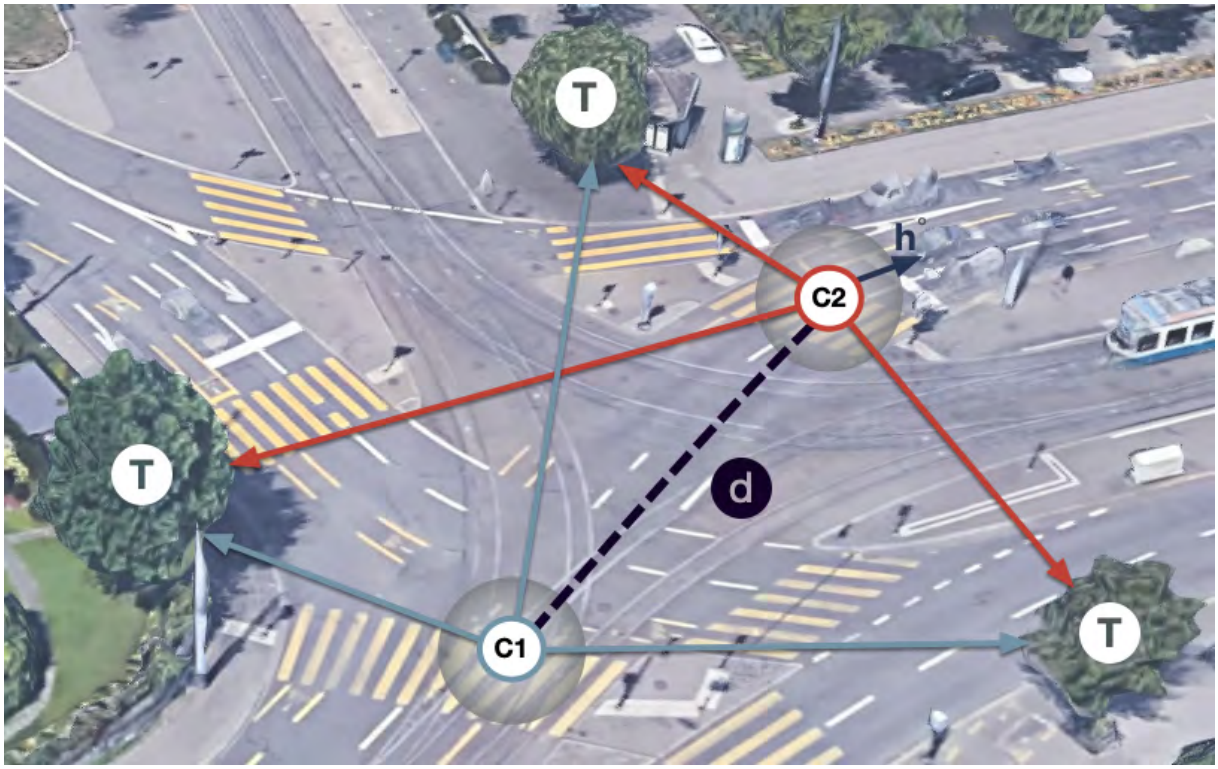


Figure 5.1: **Problem Setup:** T, the tree has its actual geographic coordinates and location within the panorama; C*, camera with its geographic coordinates; $h°$, heading angle of camera inside a panorama; d, the distance between cameras. (Imagery ©2019 Google)

One of the difficulties faced in our task is the quality of the images, which are poor in some cases. It is typical for street-level panoramas to find image-stitching artifacts, motion blur, or perspective distortions. Crowdsourced dashcam image sequences also suffer some issues (motion blur, narrow field of view, and imprecise GPS). The data is not acquired by standardized equipment through a dedicated mobile mapping campaign. As for the aerial imagery, conventional issues such as shadows and image resolution are also persistent. Due to the wide baselines between consecutive image acquisitions and inaccurate camera poses information, this setup does not tolerate pixel-by-pixel correspondences between images. To overcome these issues, we aim at fusing image evidence with pose information to train a single end-to-end neural network that employs a GNN subnetwork to identify and geo-localize object instances in a scene. In comparison from multi-staged approaches [25], [26], [55], [116] that tackled this problem by separating the task into several tasks that are trained independently, our end-to-end approach benefits from the simultaneous learning of detection, reidentification, and geo-localization. The underlying concept is that our network learns a joint distribution over many different samples of scenes of the same object instance appearing across different multiple views with its pose information. Another practical benefit is that our approach does not require multiple parameters tuning for each stage. We build upon and extend upon our work [49] by adding aerial imagery as another view, changing the object detector, and improving our GNN subnetwork to achieve higher accuracy in reidentification and geo-localization. We illustrate the scene or problem setup in Fig. 5.1 as follows: a set of ground-level images accompanied with their coarse camera pose information and an aerial view tile of the scene are passed as an input to the object detector to produce a preset number of bounding box detections for each view. Afterward, we take the bounding boxes and form a dense, fully connected graph with links between all detections in all images. We exploit the implicit rich information that can be derived from a graph structure and the relationship between the nodes to solve our problem. We then find which link defines the correspondence between instances bypassing the graph through the GNN. The GNN learns to use both the merging of image evidence and coarse pose information to give a similarity probability on each link. After the similarity is predicted, we proceed to aggregate these instances and estimate a geographic coordinate.

The contributions presented in this chapter are:

— An end-to-end multi-view detector for urban static objects.
— An extension to our dataset that now includes annotations for object instances in



Figure 5.2: **The tree re-identification problem (color indicates matches):** each tree is captured from multiple different views, changing its size, perspective, appearance and background. Note that many trees look alike and are in close proximity (Imagery ©2021 Google).

aerial view.

— Re-identification and geo-localization of object instances across aerial and street-level imagery using a GNN subnetwork.

Our method is experimentally evaluated on two datasets. In comparison to our previous method [49], the modified GNN extension alongside the choice of a different object detector and the introduction of aerial imagery has resulted in higher accuracy, for instance, re-identification and consequently geo-localization, therefore, outperforming existing methods. We hope that this new take on multi-view object detection via a graph neural network that reasons about image and pose evidence jointly will convince more researchers in this field to no longer view geometry and image interpretation as two separate steps for multi-view object detection.

5.3 Related Work

Our proposed method touches upon various research topics in computer vision such as urban object detection, object tracking, pose estimation, and instance re-identification. A full review is beyond this paper's scope, and we thus provide only some examples from literature per topic and emphasize the differences with the proposed work.

Urban object detection has been a significant research interest for several years. Early related work [25], [26], [133] aims at geo-localizing street-trees from Google street-

view panoramas and satellite images with a multi-stage hierarchical approach. A Faster RCNN object detector [10] is used to detect trees in all views. Using projection functions, the detections are then projected into the other views with estimated geographic coordinates. Afterward, a conditional random field probabilistic model is applied to the detections to aggregate image evidence from the detections with learned priors to generate the final detections. [116] geo-locates traffic signs by applying the semantic segmentation followed by monocular depth estimation as input features to a Markov random field model. To detect road objects, [23] performs semantic segmentation on ground level imagery and uses a topological binary tree to geo-locate. Another method [28] proposes to detect and geo-locate poles in Google street-view panoramas by first using an object detector then estimates pole locations using a modified brute-force line-of-bearing. [134] uses a Digital Surface Model and semantic segmentation to detect trees as blobs. The abovementioned methods are all similar in one aspect as they are all multi-stage and hierarchical. The features generated by the different models are used separately compared to our work, which uses them simultaneously.

A continuation upon previous work [132] proposed learning simultaneously relative camera pose information and image evidence with an end-to-end object detector with a siamese structure. Building upon that, [49] aims at being more efficient by using an EfficientDet backbone [17] for the object detector and matching using a GNN. Similar work [135] applies a pose regression network to estimate a 5D pose of objects present in an RGB image. Then, another model aims at matching objects in a consecutive sequence of ground-level frames. In this work, we set up our problem as a graph neural network as it provides us with the flexibility to have an arbitrary number of nodes (tree detections) and a variable number of views if needed. By doing so, our approach offers an advantage over siamese CNN as it makes the overall network much more efficient, and through evaluation, it has yielded better quantitative results. It is important to mention that a significant portion of the literature related to urban object detection is part of autonomous driving systems. Several popular benchmark datasets exist such as KITTI [20], CityScapes [21], or Mapillary [22], [102]. However, these datasets exclude static urban objects instances labeled, as their goal is to purposefully include labeled instances of cars. The image sequence includes minor changes in viewpoint as they are consecutive from a vehicle's path. It is also worth mentioning that such datasets' pose information is of good accuracy as part of the benchmark objective. Much work [113]–[115] achieves object detection and re-identification on such datasets. In contrast with our work, our relative pose information

is of lower quality, and the images aggregated have large baselines between which causes severe appearance changes.

Learning to predict camera poses using deep learning was made famous by the accomplishments of PoseNet [37] and has inspired numerous works. [110] proposes a camera pose estimation method by relying on databases of multiple feature descriptors for each viewpoint. [111] introduces RPNet, an end-to-end network that produces a full translation vector from a pair of images without the need of camera intrinsic/extrinsic values. [136] achieves 6-DoF object pose estimation method from a single RGBD image. HOPE-Net [137] using GNNs can jointly estimate hand and object pose estimation for 2D and 3D in real-time.

Multiple object tracking (MOT) and person re-identification are computer vision problems closely related to ours but they are of a different setup as the objects are in motion while the camera is stationary. [75] uses siamese CNN to find similarity between detected patches to assign matches. [32] uses a combination of images and optical flow maps as an input to a siamese CNN to track targets. A tracklet affinity mode devised by [31], that consists of a siamese CNN with temporally constrained metrics. Authors of [33] employ a combination of CNNs and recurrent neural networks (RNNs) to match pairs of detections. [34] tackles the re-identification problem by minimizing the distance in feature space between candidate boxes using a center loss. Further research [89] detects and re-identifies objects in an end-to-end learnable CNN approach with an online instance matching loss. Unlike in our scenario, where multi-view object detection and re-identification are done simultaneously, person re-identification assumes given detections. The work most closely related to ours is [36] as they also use graphs to represent their person re-identification setup. CNN features are generated from image crops of persons detected. Theses CNN features are then represented as nodes in a graph of persons over time with edges connected between all nodes. A GNN then performs message passing to propagate node features in the graph. The edges between the nodes are classified to track the target object (person) identity throughout time. Our work similarly represents the detections as nodes in a graph and classifies edges between them to assign correspondence or matches across views. However, we differ in that they rely upon a single camera setup, while in our problem we have multiple cameras. Also, in our work, we aim at performing detection and re-identification end-to-end.

Graph neural networks (GNN) operate on non-grid structures such as manifolds, and graphs represent molecules, social networks, point clouds, or road networks. Inter-

estingly, recurrent neural networks were the basis of early GNNs [138], [139], sequentially aggregating the neighborhood around nodes. Graph convolutional networks (GCNs) were first introduced in [140] that proposed a convolutional approach on spectral graphs, which has motivated various extensions in this direction such as [141]–[143]. As for the spatial domain, many methods [30], [67], [126], [144], [145] perform convolutions on graphs on spatial neighborhoods, which proved to generalize much better than spatial domains. Current methods, which are described as fixed-pooling methods, aim at making processing large graphs such as social networks more efficient by pooling nodes to perform subgraph-level classification [146], [147] based on the graph topology. Another coarser pooling graph topology is based on weighted aggregations [130], [148], [149] learned from the graph. In our work, we are more concerned about classifying edges between the nodes to assign the similarity between the nodes to determine the correspondence. We rely on spatial-based convolutions [146], [150] as they generalize better on other data. Several methods rely on edge convolutions [151], or a variational geometric neural network autoencoder [152]. Methods based on **Siamese models** [75] alone are not a viable solution to our problem, since they need image crops of the object and cannot fully use re-identification annotations due to their pairwise labeling training setup. [34] searches for a crop within the detections in a gallery of images compared to our method that aims at matching detections from both full images. The key differences between our work and [89] is that we both ensure object geo-localization and avoid storing features from all identities since it is impractical in real-world applications like the one considered in the paper where objects look similar in appearance.

5.4 Method

Our urban object localization method consists of several subnetworks that are trained end-to-end and backpropagated simultaneously as shown in Fig. 5.3. The main network is the backbone network, here ResNet [153], to generate the CNN features used by the other subnetwork. In this work, we rely on Detection Transformers (DETR) [19] from Facebook, which is a transformer encoder-decoder network. The final subnetwork is our Graph Neural Network (GNN), which is responsible for re-identifying objects and geo-localizing them.

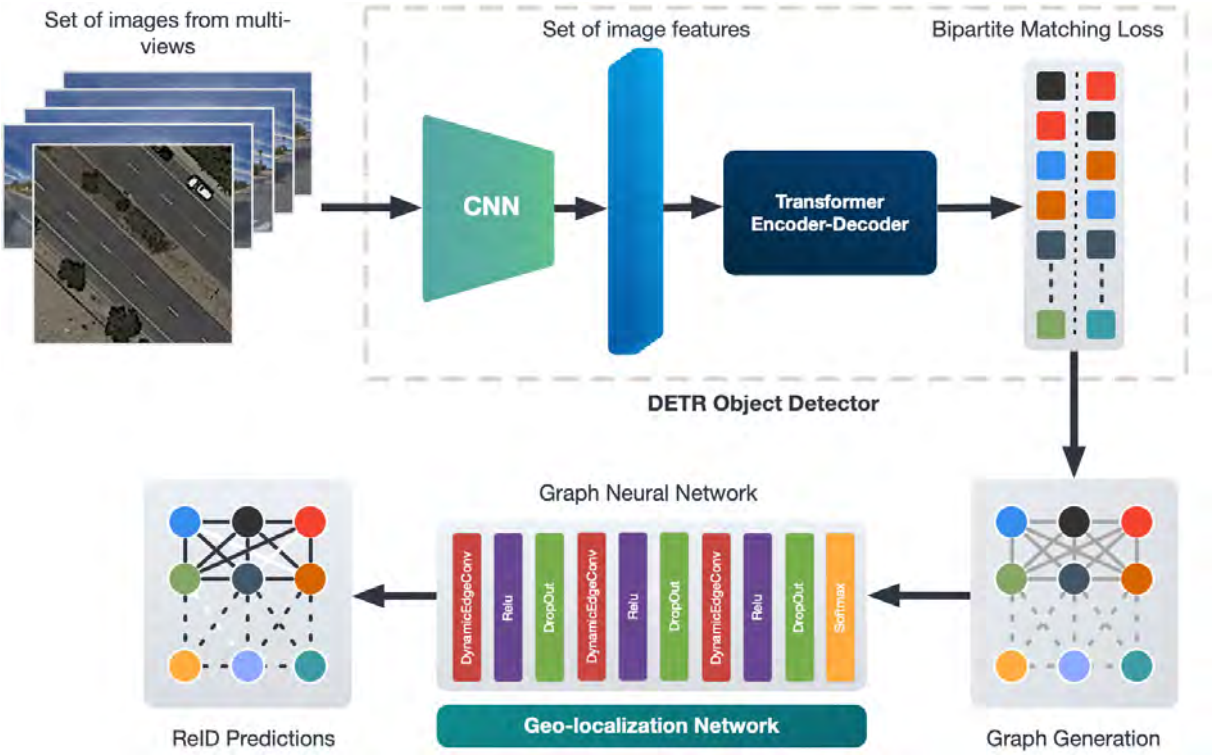


Figure 5.3: **Architecture of GeoGraphV2:** The network's input is a batch of images from multiple views accompanied by their pose information P. The images pass through the backbone network (ResNet-50) to produce a set of image features in the size of the *number of queries* per image. The image CNN features pass through the Transformer encoder-decoder network to produce a set of predictions. Following that, bipartite matching assigns the predictions to the ground truth to minimize accordingly. After the matching process, the graph generation process creates a graph out of the detections. The nodes of the graph contain the CNN features of the detection, along with the pose information. This GNN then receives the graph and proceeds to perform edge convolutions to classify the graph's edges to find the correspondence. Coincidentally, the nodes' geographic location is regressed using a shallow neural network.

5.4.1 Object Detection

DETR was chosen as our object detector for technical and practical purposes that benefit different parts of our network. DETR achieves state-of-the-art methods on challenging data sets such as COCO [154], but in our problem, we do not have numerous classes or objects in an image. The main advantage that led to it being chosen is its simple architecture. The CNN features generated from the ResNet-50 backbone network are passed to a transformer encoder-decoder network that directly provides a set of prediction boxes. In comparison to our previous methods, GeoGraph [49] and SSD-ReID-Geo [132], they have used object detectors that use "Region Proposal Networks (RPN)", and "Fast RCNN heads", and required "non-maximum suppression" (NMS) to provide the final detections. The set of predictions or *number of queries* is a predefined parameter, setting the total number of objects the network could detect. However, this is not a disadvantage in this study since we are not detecting numerous objects. It is indeed an advantage for our GNN, which will be discussed later. The object detector receives as input a batch of images accompanied by their pose information. The pose information $P = \{C^*_{lat}, C^*_{lng}, h^\circ\}$ where $C^*_{lat}, C^*_{lng}, h^\circ$ denote the cameras's geographic latitude, longitude coordinates, and heading angle, respectively. The pose information represents the location of the camera in the 3D world. The images are then provided as an input to the *backbone* similar to most object detectors, which generates feature maps. The feature maps are reshaped into one-dimensional feature maps which are passed to the *transformer encoder* network. The feature output from the *transformer encoder* is then passed to the *transformer decoder*, that generates bounding box coordinates by employing encoder-decoder transformer and self-attention modules. To predict the normalized center coordinates with height and width dimensions using feed-forward neural networks, with the class label predicted using a softmax function. DETR also uses bipartite matching loss to consider the differences in the predictions' permutations regarding the ground-truth. It tries finding the best permutation for the predictions that achieves the minimum total loss. The matching is then achieved using the *Hungarian Algorithm*. The main loss function for bounding box localization is linear combination of L1 loss and generalized IoU loss. Also, to output the correct number of objects per class, auxillary losses [155] are used.

Learning to map street-side objects using multiple views Ahmed Samy Nassar 2021

5.4.2 Object Re-identification

Detecting objects in multiple different views raises a natural problem. In those views, the objects appear more than once. Therefore, counting them at this stage leads to double counting and geo-localizing the same instances. Some data set scene setups have a heterogeneous number of images. A flexible network must make our method more generalizable. In our work, we represent our data structure as a graph to accommodate these difficulties. As mentioned previously, graphs represent irregular data structures; consequently, we represent the detections from all images as nodes in a graph. Graph algorithms are capable of managing a non-fixed number of nodes. Therefore, from the DETR object detections, a graph is generated, in which a GNN takes as an input to find which nodes are of the same object and geo-localizes them. In the following, we provide a detailed explanation of the graph generation method and discuss how the GNN re-identifies nodes.

Graph Generation

A graph is denoted as $\mathcal{G} = (V, E)$ where V represents the set of N nodes, and E the set of edges connecting the nodes. An advantage of DETR is that it provides a fixed number of predictions (number of queries) for each image, which makes the graph generation process less tedious than our previous works [49], [132]. In our previous works, to produce the graph nodes, IoU was applied over the proposals generated from *the region proposal network* and the ground-truth. Also, negative proposals were acquired randomly. These proposals' numbers can vary depending on the image from hundreds to thousands.

Since DETR's number of predictions is predetermined during training, an undirected graph is easily generated by creating a fully connected graph with edges between all nodes. Each node comprises the CNN feature the DETR generated for this particular prediction. Also, additional features such as pose information and an estimated geographic location are created for each node. Using the output from the *Hungarian Algorithm* assigns a ground-truth box ID to each prediction. Afterward, the graph's ground-truth, which denotes a binary label [0,1] for whether $e_{ij} = 1$ if nodes i and j is a match, is built by the ID the predictions got assigned.

Graph Neural Network

Our input to the GNN is a graph $\mathcal{G}$ representing all object detections across different views with edges connecting all nodes. Unlike CNNs, which serves regular grid structures, GNN can deal with irregular data structures such as graphs using message passing between the graph's nodes. Our main objective with the GNN is to train it to receive $\mathcal{G}$ and classify which nodes of the graph belong to each other against the graph's ground-truth $\mathcal{G}_{gt}$. The architecture of the GNN consists of 3 edge convolutional layers [151] (DynamicEdgeConv) followed by a multi-layer perceptron (MLP). The DynamicEdgeConv layer processes graphs using maximum aggregation and can be expressed mathematically by

$$\mathbf{x}_i^{(k)} = \max_{j \in \mathcal{N}(i)} h_{\mathbf{\Theta}} \left(\mathbf{x}_i^{(k-1)}, \mathbf{x}_j^{(k-1)} - \mathbf{x}_i^{(k-1)} \right), \tag{5.1}$$

where $h_{\mathbf{\Theta}}$ represents the MLP, $\mathbf{x}_i$ denotes the node's feature, and $\mathbf{k}$ symbolizes the number of neighborhood nodes. Ordinarily, message passing propagates the layers throughout the graph using

$$\mathbf{x}_i' = \gamma_{\Theta} \left(\mathbf{x}_i, \nabla_{j \in \mathcal{N}(i)} \phi_{\Theta} \left(\mathbf{x}_i, \mathbf{x}_j, \mathbf{e}_{j,i} \right) \right), \tag{5.2}$$

where ∇, and $\phi_{\mathbf{\Theta}}$ denotes differentiable aggregation functions and MLP, respectively. After each DynamicEdgeConv, a dropout [156] layer for regularization is added. In summary, developing the data structure as a graph grants the feasibility of efficiently handling a variable number of distinct objects. The graph's edges are then classified to assign matches or correspondence of the objects represented as nodes. This process is oblivious to the number of objects or views that are not hardcoded and can disentangle the graph into separate objects.

5.4.3 Geo-localization

At this stage, using the GNN, the graph nodes' identities are assigned. The remaining task is to estimate the geo-coordinates of the identified objects similarly to [25], [49], [132]. From the regressed bounding boxes values provided by DETR, projection equations (5.3) and (5.4) estimate real-world geographic coordinates. Having the pose information of the cameras in the scene makes it achievable. For the projection equations to function, an underlying assumption is that the scene's terrain is locally flat. The projection equations can map the object's bounding box pixel locations x and y in East, North, Up (ENU)

coordinates e_x, e_y, e_z and then vice versa to recover the position of the object in the real world geographic coordinates O_{lat}, O_{lng}:

$$\begin{aligned}(e_x, e_y, e_z) = \Big(&R\cos[\mathrm{C}_{lat}]\sin[\mathrm{O}_{lng} - \mathrm{C}_{lat}],\\ &R\sin[\mathrm{O}_{lat} - \mathrm{C}_{lat}], -\mathrm{h}^\circ\ \Big)\end{aligned} \tag{5.3}$$

$$\begin{aligned}x &= (\pi + \arctan(e_x, e_y) - \mathrm{h}^\circ)\, W/2\pi\\ y &= (\pi/2 - \arctan(C_h, z))\, H/\pi\end{aligned} \tag{5.4}$$

where R denotes the Earth's radius, W and H are the image's width and height, C_h denotes the camera's height and $z = \sqrt{e_x^2 + e_y^2}$ is an estimate of the object's distance from the camera. To improve the geo-localization accuracy, the predicted geo-coordinates provided by these equations are passed to a neural network, whose purpose is to regress and obtain much more refined geo-coordinates by using a Mean Square Error (MSE) loss.

5.5 Experiments

5.5.1 Datasets



Figure 5.4: Subset of the Pasadena dataset highlighting challenges to be faced by multi-view object detection, re-identification and geo-localization. The trees are in close proximity and share very similar appearances due to their same species and size (trees on the curb are planted together at the same time). Some objects are annotated with their instance label (red), some without (orange).(Imagery ©2021 Google)

Pasadena Multi-View ReID. Most autonomous driving and urban object datasets do not contain instance labeled trees but label all plants in the scene as one class "vegetation". The Pasadena Multi-View ReID data set [132] was introduced as the first data set with labeled instances of different trees obtained in Pasadena, California. The dataset contained initially 4 Google Street View panoramic views for each tree instance. One of our main contributions in this work is the addition of satellite imagery as a fifth view acquired from Google Satellite. On average, there are at least 2 labeled tree instances per scene. In total, there are 6,020 labeled individual tree instances. A total of 6,141 panorama images of size 2048 x 1024 px and 6020 aerial images of size 256 x 256 px. In entirety, this sums up to a total of 31,081 annotated objects in the dataset. Each panorama image annotation includes: 1) Bounding box values for the trees, 2) Pose information of the

image, and 3) IDs for labeled trees instance along with their geo-coordinates. As for the satellite images: 1) Bounding box values for the trees, 2) Geographic coordinate of the center pixel of the image, and 3) IDs for labeled trees instances. We keep the same data split as proposed in [132] by allotting 298 images for training, 921 for validation, and 922 for testing in our experiments.

Mapillary. Mapillary is a crowdsourced dataset [1], which is not to be confused by the popular segmentation dataset "Mapillary Vistas" [102]. This dataset is a subset of the Mapillary platform covering an area of $2km^2$ in London, England. Typically, the dataset includes various urban classes, but only one traffic signs class is included. In contrast with the Pasadena Multi-View ReID, the images in this dataset differ in the method of capturing. Mapillary's images have been acquired from various sources. These sources can be pedestrians' smartphones or car dash cameras. Most parts of the dataset are sequences of consecutive imagery as they are captured by forward-faced cameras mounted on vehicles. Due to this setup, objects change in scale as they are relatively seen from the same moving viewpoint. As the dataset is crowdsourced, this presents exciting challenges. The major challenge is that various camera sensors capture the dataset, producing as well different image sizes, in contrast with Pasadena Multi-View ReID, which is uniform. Another challenge is that the images are captured at different times of the day. The dataset contains 31,442 instances of traffic signs labeled and a total of 74,320 images. On average, there are two traffic signs instances in each image, and there are at least for each instance an average of 4 images. In all images, almost entirely all instances of signs are annotated. The annotations in each traffic sign's data sign comprise the bounding box values, object ID, image geo-coordinates, instance geo-coordinates, heading of the camera, and height. 3D Structure From Motion techniques (SFM) is how Mapillary has geo-localized the traffic signs. In comparison to Pasadena Multi-View ReID, the traffic signs are much smaller objects than trees and harder to detect as they are much thinner objects and couldn't be clear at some angles. However, fortunately, the dataset contains many samples allowing the object detector to deal with such cases.

1. www.mapillary.com



Figure 5.5: Subset of the Mapillary dataset. This scene depicts a dashcam sequence where nearly all signs in the scene are annotated with their instance label (red), and shows strong object scale changes across the different views.

5.5.2 Implementation Details

GeoGraphV2 is implemented using PyTorch [157]. For the DETR object detector, the official code implementation [2] has been used. For the GNN component, it was implemented using the PyTorch Geometric package [158]. The Dropout layers were set with a probability of 0.2. The optimizer used was ADAM [101] with the learning rate initialized as 0.001. Each epoch takes approximately 45 minutes during training time on a NVIDIA 1080 Ti GPU ([132] needs 270 minutes). Our network uses 34.75M parameters while [132] uses 50M. We significantly reduce inference time per image from 0.78ms [132] to 0.32ms.

5.5.3 Object Detection & Re-identification

We report the results obtained from our experiments on different data sets, setups, and related methods in Tab. 5.1. The approach's effectiveness is tested on the accuracy of detecting the objects in the images, correctly re-identifying them (see Fig. 5.4 and 5.5 for visual explanation), and how far they are from the geo-coordinate in terms of distance.

2. https://github.com/facebookresearch/detr

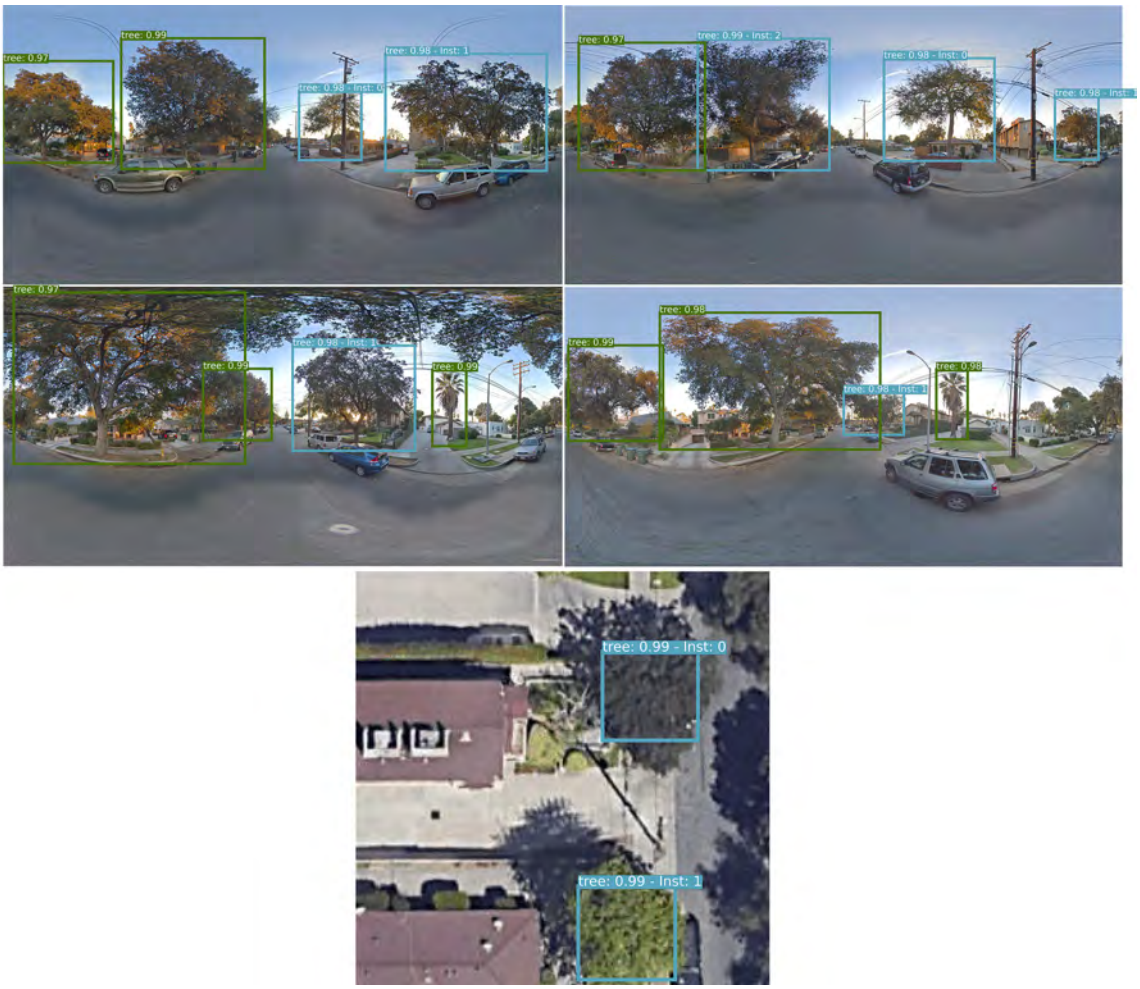


Figure 5.6: Sample results obtained on the Pasadena dataset for multi-view object detection and re-identification in the street view and aerial. Trees were correctly detected (green) and further accurately re-identified across different views (cyan) when possible. (Imagery ©2021 Google)

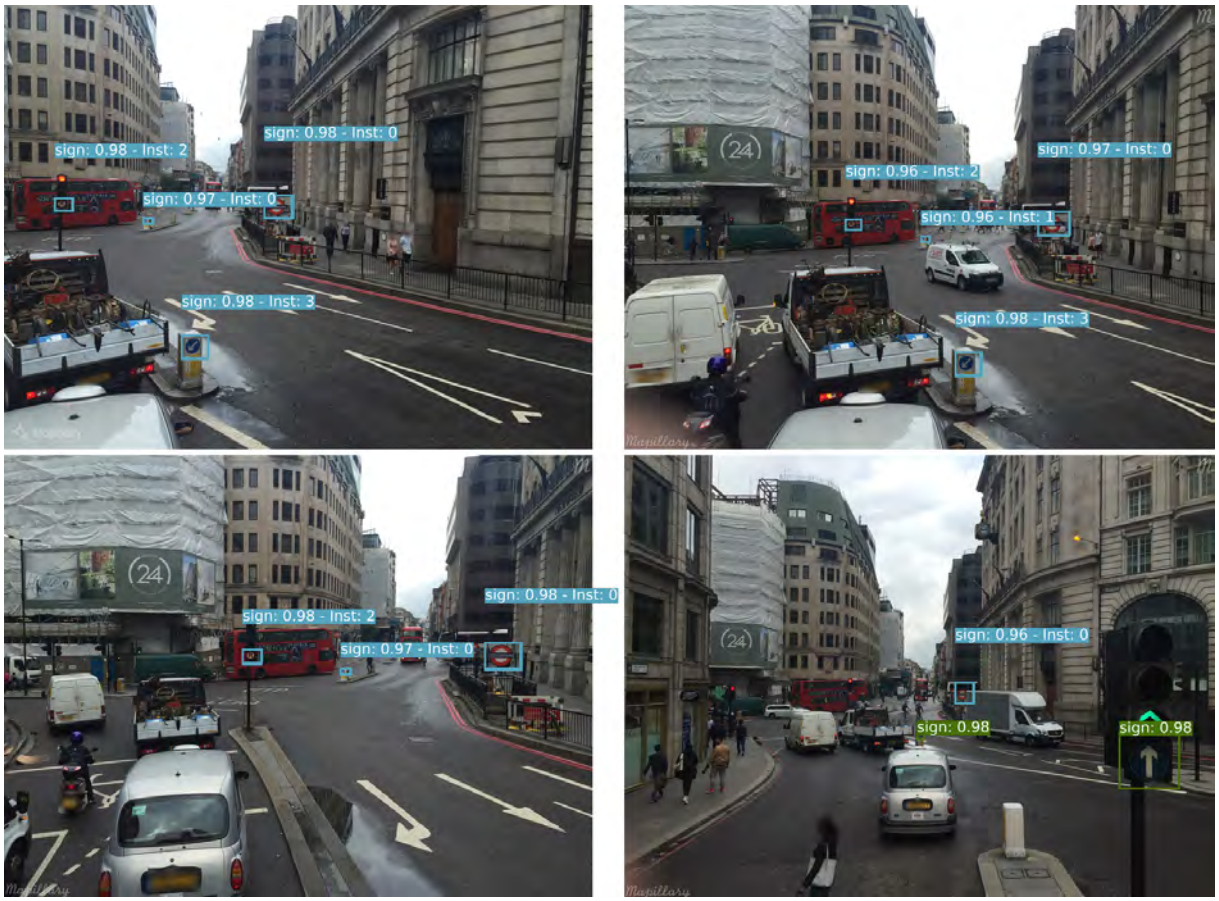


Figure 5.7: Sample results obtained on the Mapillary dataset for multi-view object detection and re-identification in the street view. Trees were correctly detected (green) and further accurately re-identified across different views (cyan) when possible. (Imagery ©Mapillary)

Naturally, the object detection performance is critical to the whole pipeline overall. The object detection Average Precision (mAP) impacts the re-identification score since the object instances must be detected with all images. Also, a bounding box that is correctly localized will improve the geo-localization score since the projection functions depend on the bounding box values. In our experiments, the DETR detector, on average, provided lower detection scores of ≈ 2 %. Despite that, our previous method [49] achieved a better detection score. This design decision was chosen for a couple of reasons. The first reason relates to that in this work we extend upon our previous work [49] by including aerial imagery, which implementation wise caused a GPU memory increase. To cope with the GPU memory increase, the object detector's transformer's encoder and decoder layers are reduced from 3 to 1, which affected the accuracy. With 3 to 6 layers each, the

object detector performs with similar accuracy to [17]. The second reason is that DETR offers a "set of predictions" approach that greatly helps in the re-identification process as mentioned before. For those two reasons, the advantages overweighed the ≈ 2 % decrease. We believe that technically this problem could be solved by improving the current pipeline in future work to accommodate more encoder and decoder layers. Our re-identification experiments validate our approach of using graphs to match objects in the scene by having the nodes carry our appearance information acquired from the object detector with the pose information. The GNN component is evaluated based on whether the edges between nodes are correctly classified or not, representing whether an object is matched across views. Our graph predictions are confidence values on each edge between the nodes, and values considered ≥ 0.5 are positive matches, while $\leq$ are considered negative ones. To calculate our mAP, true positives are considered for correctly classified edges between matching nodes, without any positive edges to other nodes (false positives). GeoGraphV2 has achieved a high re-identification mAP with an improvement of 10.3% for Pasadena and $\approx 2\%$ for Mapillary. This increase is attributed to: 1) using the "set of predictions" from DETR, instead of many proposals from anchor-based object detectors, that created many graphs of different sizes and the decrease in graph nodes. 2) The new architecture of our GNN. Our GeoGraphV2 achieved a lower Re-ID score for the 5 views experiment with Pasadena due to including an aerial view and the difficulty of severe discrepancy in appearance, as shown in Fig 5.6. As for the Mapillary dataset (see Fig. 5.7), as expected, the re-identification scores improved due to the new graph generation method. We could not experiment with aerial imagery, as street signs are not available with the dataset. They are barely visible in Google Satellite and are often occluded by trees.

5.5.4 Geo-localization

The geo-localization score evaluates the distance (in meters) between the predicted geo-coordinate and the ground-truth geo-coordinate for each instance in terms of a Mean Absolute Error (MAE). The predicted geo-coordinates are averaged over different views, as shown in Fig. 5.8. We use the Haversine distance as the metric to calculate the distance between the prediction and ground-truth:

$$d = 2R \arcsin\left(\left(\sin^2\left(\frac{O_{lat} - G_{lat}}{2}\right) + \cos(G_{lat})\cos(O_{lat})\sin^2\left(\frac{O_{lng} - G_{lng}}{2}\right)\right)^{0.5}\right), \quad (5.5)$$



Figure 5.8: Sample of geo-localization results for Pasadena comparing different methods. The circles represent the geo-coordinates. Green, purple, blue, yellow and red circles represent the ground-truth, GeoGraphV2, GeoGraph, SSD-ReID-Geo and MRF respectively.

where O_{lat}, O_{lng} represent the detection's predicted geo-coordinates, and G_{lat}, G_{lng} represent the object's ground truth. Our results reported in Tab. 5.1, show that for the Pasadena data set, the geo-localization error decreased with GeoGraphV2 for street-view only views. We associate this decrease with the improvement in re-identification, which resulted in more averaged geo-coordinate predictions per object to result in a final geo-coordinate. The higher localization with *GeoGraphV2 SV+AR* is because detections in aerial imagery provide bounding boxes, in which from their center pixel a much more accurate geo-coordinate can be obtained.

5.5.5 Ablation Studies

The graph generation component of our pipeline ties together all the other components. Therefore, each component is evaluated separately to assess its effect in re-identification and geo-localization on the Pasadena data set for completeness in Tab. 5.2.

We can observe from the results that monocular images alone leads to poor geo-localization results. Using the projection functions to estimate the geo-coordinates of trees detected from cameras that are far, causes significant errors. These significant errors are due to the assumptions of a flat terrain in the projection functions; therefore, a bounding

Table 5.1: Quantitative assessment of our GeoGraphV2 framework and related work on object detection, re-identification, and geo-localization tasks. Note: the object detection scores for the GeoGraph and GeoGraphV2 does not change between the 2 and 2+ views; this is due to the object detector component being the same.

Method	# Views	Type	Dataset	Detection mAP	Re-ID mAP	Geo-localization error (m)
MRF [116]	4	SV	Pasadena	0.742	-	3.83
SSD-ReID-Geo [132]	2	SV	Pasadena	0.682	0.731	3.13
GeoGraph [49]	2	SV	Pasadena	**0.742**	0.754	2.94
GeoGraph [49]	4	SV	Pasadena		0.763	2.75
GeoGraphV2	4	SV	Pasadena	0.721	**0.866**	2.32
GeoGraphV2	5	SV	Pasadena	0.71	0.815	**1.862**
GeoGraphV2		AR		0.42		
MRF [116]	4	SV	Mapillary	0.919	-	4.62
SSD-ReID-Geo [132]	2	SV	Mapillary	0.902	0.882	4.36
GeoGraph [49]	2	SV	Mapillary	**0.919**	0.902	3.88
GeoGraph [49]	6	SV	Mapillary		0.924	4.21
GeoGraphV2	2	SV	Mapillary	0.91	0.923	**3.14**
GeoGraphV2	6	SV	Mapillary		**0.941**	**3.33**

Table 5.2: Quantitative assessment by bypassing components of the GeoGraphV2 on the Pasadena dataset.

Method	Detection mAP	Re-Id mAP	Geo-localization MAE (m)
Monocular Street-view	0.71	-	10.2
Multi Street-view	0.71	0.866	2.32
Aerial view	0.42	-	1.34
SV + AR	-	0.815	1.8625

box that is localized on the ground can predict geo-coordinates in streets further away. Using multiple street-views with re-identification helps re-identifying the trees and having several geo-coordinate predictions that provide a better estimate when averaged. Aerial imagery alone does not provide a good detection score; however, for the detections they make, they assist in improving the geo-localization as it becomes easier to get a much more precise geo-coordinate. As mentioned before, it is indeed much more accurate to estimate the geo-coordinate from a top view bounding box than in a street-view one. As presented throughout this work, the combination of street-view and aerial imagery offers a smaller geo-localization error.

5.6 Conclusion

In this chapter, we have presented a full pipeline that tackles the problem of detecting, re-identifying, and geo-localizing object in multiple views using an end-to-end learnable framework. The method presented integrates both aerial and street-view imagery and can be extended to accept more views. The proposed method takes advantage of both appearance features and pose information to tackle the severe changes in views by relying upon a GNN to perform object re-identification. GeoGraphV2 has improved upon [49] to achieve much higher accuracy in re-identification and geo-localization. Similarly, our method identifies objects of similar appearance in close proximity and drastic changes in viewpoints. The proposed implementation could technically be improved, which would

lead to the ability to include more views efficiently. Furthermore, transferring attention from one view to the other using the projection functions would help detecting occluded objects better and improving other views' detection confidence simultaneously by steering the network to look at information it might have missed.

Chapter 6

Conclusion

This thesis explores detecting, re-identifying, and geo-localizing urban street objects using multi-view imagery. The thesis is written as a cumulative dissertation. The first chapter was the introduction chapter that presented our problem and work, followed by a background chapter that briefly touched on the relevant information required for our problem for completeness. Chapters 3, 4, and 5 are our core chapters in which we added published peer-reviewed papers only with minor typographical and formatting corrections. The core chapters' order reflects the way approached to solve the problem sequentially. After discussing our contributions, we address the future outlooks of this work.

6.1 Discussion of Contributions

6.1.1 Multi-View Instance Matching with Learned Geometric Soft-Constraints

Chapter 3 explores our hypothesis of fusing image evidence and soft geometric constraints to establish whether geometric evidence supports matching crops of images obtained from different multi-views. In this work, we aimed at simplifying the problem by assuming that object detection was achieved and experimented with some matching techniques for ground-level imagery. The contributions obtained were:

1. A deep learning siamese-based architecture that uses geometric cues alongside appearance information to match objects' image crops from multiple views.

2. Experimenting with various siamese network architectures for evaluation on data sets of trees and street-signs.

6.1.2 Simultaneous Multi-View Instance Detection with Learned Geometric Soft-Constraints

Chapter 4 builds upon the results attained from chapter 3 and begins to use pose information coupled with the appearance in an end-to-end manner. The challenges such as changes in viewpoint, lighting conditions, the high similarity between neighboring objects, and variability in scale were discussed. We presented a method that architecture wise was inspired by siamese networks, aiming to take two images and find correspondences between detected objects producing a learnable end-to-end network. Our contributions in this chapter were:

1. Introduced a first of its kind data set that contains tree instances labeled with IDs across multi-view imagery.
2. A novel labeling tool that annotates Google Street View panoramas in aerial and street view to generate the data set.
3. We introduced an end-to-end multi-view urban static detection method that re-identifies and geo-localizes objects across multiple views.

6.1.3 GeoGraphV2: Graph-Based Aerial & Street View Multi-View Object Detection with Geometric Cues End-to-End

Chapter 5 continues to improve upon the method introduced in Chapter 4 and addresses some of its shortcomings. We present another end-to-end network that is much more efficient and extensible than the previous network (chapter 4). We formulate our problem setup as a graph, which gives us the flexibility to add more views and assign correspondence without having a hardcoded value of how many views or how many objects. With this flexibility and efficiency, we added satellite imagery to accompany the street views. The contributions of this chapter can be summarized as:

1. We introduced another end-to-end multi-view detector for static street-side objects that is more efficient and accurate. This approach can extend on any anchor-based object detector.
2. A novel approach that uses Geometric Neural Networks (GNN) to learn to use pose information and appearance jointly to detect objects and match them in a scene.

3. We extend our dataset to include satellite imagery labeled with IDs and correspondences between satellite imagery objects and ground-level instances.

6.2 Discussion of Limitations

In this dissertation, and as summarized in the previous section, we have touched upon the various contributions to the street-side object detection and matching in multiple views. However, there is room for further enhancement as there are various issues that present some weaknesses and disadvantages in the approaches introduced. Here the significant factors of limitations are presented.

6.2.1 Ground-truth

Perhaps the most significant limitation to improving our accuracy in this dissertation is the lack of ground truth. Despite our significant effort at creating our data set, it still does not contain as many annotations as it should. Unfortunately, no free large-scale datasets containing static objects re-identified exist to our knowledge, and as tricky as our setup is, many annotations are required. In our efforts, the data was hand-labeled manually, which is not easy for even humans to re-identify the same object across different views with varying appearances. It is important to mention that the imagery exists, but not the annotations on a large scale. Another issue is the geo-coordinate ground-truth accuracy that already varies between $\sim$ 0.1 m in the south to $\sim$ 2.7 m in the north in Google Satellite [159]. We emphasize that more annotations, and preferably distributed around imagery from different parts of the world, would further increase the model's accuracy.

6.2.2 Processing Speed

Despite the many attempts to improve the implementations' efficiency presented in chapter 3 and chapter 4, the model takes a long time to converge. This problem arises due to the dependency on a set of images depending on each other's outputs. There are probably different approaches for solving this issue using LSTM or data parallelism, but those venues weren't explored due to time constraints. We believe that the implementations should not require multiple high-end GPU but should also run in real-time with limited resources to help areas without such facilities.

6.2.3 Improvement of aerial detections

One fundamental limitation is poor aerial detection scores. Improving that detection score would significantly impact the geo-localization score, as the aerial bounding box center is the geo-coordinate. Unfortunately, the satellite imagery used in the Pasadena data set is of low quality, which greatly impacts the detection score.

6.3 Outlook

The approaches discussed in this dissertation should bring current advance on the topic regarding computer vision and machine learning. It is important to note that the methods presented in this thesis are inspired by previous state-of-the-art in these fields. This section presents a few ideas that could help overcome the limitations discussed in the previous section. Also, we present open problems for future research or potentials solutions w.r.t. new technologies, methodology, and data.

6.3.1 Future Directions.

Building Data sets. As mentioned in the previous section, one of the main limitations is the lack of a huge dataset. There are several ways to acquire this data set. The first method proposed is to use the current methods presented in this thesis to get detections in different cities worldwide. After that, in a semi-supervised fashion, a human annotator finetunes the predictions achieved. These finetuned predictions can be used as training data in an iterative process to generate a vast data set. The second method proposed requires rigs containing multi-camera setups calibrated alongside Lidar, which could help generating the ground-truth for object re-identification, similar to [160].

Integrating Attention. An attention mechanism mainly aims at in a CNN to learn and focus on more useful information rather than learning irrelevant background information. Since our methods employ a multi-views setup, by using the detections in each view, they can be projected in the other views using the projection functions introduced in chapter 4 & chapter 5 to steer the attention of the detector to these spatial locations. Attention applied to these spatial locations would help improving the detection scores as it would help detecting occluded or potential objects the detector missed. In this work, we explored a similar direction in chapter 4. Due to time constraints, we were not able to explore that approach further. Such a method would be advantageous if the attention

is applied to all images simultaneously in parallel, but due to GPUs memory limitations with our implementation, this was not possible.

Depth Estimation. In our work, we have solely relied upon monocular imagery settings to rely only on available imagery that is possible for everyone to use. While we use multiple imageries, the reconstruction from motion techniques is not particularly helpful due to the relatively large distance between the images. However, depth estimation can provide more information about the scene, w.r.t. to 3D geometry. Having the depth of the images would enable the extraction of depth using the object's bounding box in the scene. Acquiring the object's depth could be a beneficial addition to our pose information feature vector.

Semi-supervised re-identification with deep learning. It is most common for deep learning methods to be trained in a supervised way. However, with the large amounts of unlabeled data that could be aggregated from the various mapping platforms, it would be interesting to explore if the unlabeled data can be used in an unsupervised or semi-supervised manner to improve existing methods further.

Climate Change. Remote sensing overall is vital for the observation of climate change. As mentioned in our introduction, we see that monitoring trees could help in the task of observation of climate change. With the continuous flow of data and improved sensors, earth observation becomes ever more critical for producing information that could help in decision making. This information can help climatologists gain new insights and perspectives to help with regards to the problems of climate change.

BIBLIOGRAPHY

[1] H. Akbari, D. M. Kurn, S. E. Bretz, and J. W. Hanford, « Peak power and cooling energy savings of shade trees », *Energy and buildings*, vol. 25, *2*, pp. 139–148, 1997.

[2] Y. J. Huang, H. Akbari, and H. Taha, « The wind-shielding and shading effects of trees on residential heating and cooling requirements », *ASHRAE proceedings*, vol. 96, *pt. 1*, 1990.

[3] H.-P. Siderius, « The End of Energy Efficiency Improvement= The Start of Energy Savings? 2004 ACEEE Summer Study on Energy Efficiency in Buildings », *Pacific Grove, CA, American Council for an Energy-Efficient Economy*, 2004.

[4] J. N. David, « Compensatory value of urban trees in the United States », *Journal of Arboriculture*, vol. 28, *40*, pp. 194–199, 2002.

[5] E. G. McPherson, N. van Doorn, and J. de Goede, « Structure, function and value of street trees in California, USA », *Urban Forestry and Urban Greening*, vol. 17, pp. 104–115, 2016, ISSN: 1618-8667. DOI: https://doi.org/10.1016/j.ufug.2016.03.013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1618866715301400.

[6] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray, « Visual categorization with bags of keypoints », *in Workshop on statistical learning in computer vision, ECCV*, Prague, vol. 1, 2004, pp. 1–2.

[7] A. Yao, J. Gall, and L. Van Gool, « A Hough transform-based voting framework for action recognition », *in 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010, pp. 2061–2068. DOI: 10.1109/CVPR.2010.5539883.

[8] N. Dalal and B. Triggs, « Histograms of oriented gradients for human detection », *in 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, 2005, 886–893 vol. 1. DOI: 10.1109/CVPR.2005.177.

[9] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, « Object Detection with Discriminatively Trained Part-Based Models », *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, *9*, pp. 1627–1645, 2010. DOI: 10.1109/TPAMI.2009.167.

[10] S. Ren, K. He, R. Girshick, and J. Sun, « Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks », *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, *6*, pp. 1137–1149, Jun. 2017, ISSN: 1939-3539. DOI: 10.1109/TPAMI.2016.2577031.

[11] R. Girshick, J. Donahue, T. Darrell, and J. Malik, « Rich feature hierarchies for accurate object detection and semantic segmentation », *in Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[12] R. Girshick, « Fast {R-CNN} », *in Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1440–1448.

[13] K. He, G. Gkioxari, P. Dollar, and R. Girshick, « Mask R-CNN », *in Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017.

[14] J. Redmon and A. Farhadi, « Yolov3: An incremental improvement », *arXiv preprint arXiv:1804.02767*, 2018.

[15] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, « Ssd: Single shot multibox detector », *in European conference on computer vision*, Springer, 2016, pp. 21–37.

[16] T. Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, « Focal Loss for Dense Object Detection », *in IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, 2020, pp. 318–327. DOI: 10.1109/TPAMI.2018.2858826. arXiv: 1708.02002.

[17] M. Tan, R. Pang, and Q. V. Le, « EfficientDet: Scalable and Efficient Object Detection », *arXiv preprint arXiv:1911.09070*, 2019. arXiv: 1911.09070. [Online]. Available: http://arxiv.org/abs/1911.09070.

[18] M. Tan and Q. V. Le, *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*, 2020. arXiv: 1905.11946 [cs.LG].

[19] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, « End-to-End Object Detection with Transformers », *in Computer Vision – ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds., Cham: Springer International Publishing, 2020, pp. 213–229, ISBN: 978-3-030-58452-8.

[20] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, « Vision meets robotics: The {KITTI} dataset », *The International Journal of Robotics Research*, vol. 32, *11*, pp. 1231–1237, 2013.

[21] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, « The Cityscapes Dataset for Semantic Urban Scene Understanding », *in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-December, 2016, pp. 3213–3223, ISBN: 9781467388504. DOI: 10.1109/CVPR.2016.350. arXiv: 1604.01685.

[22] D. Ma, H. Fan, W. Li, and X. Ding, « The state of mapillary: An exploratory analysis », *ISPRS International Journal of Geo-Information*, vol. 9, *1*, p. 10, 2019, ISSN: 22209964. DOI: 10.3390/ijgi9010010.

[23] C. Zhang, H. Fan, W. Li, B. Mao, and X. Ding, « Automated Detecting and Placing Road Objects from Street-level Images », *arXiv preprint arXiv:1909.05621*, 2019. arXiv: 1909.05621. [Online]. Available: http://arxiv.org/abs/1909.05621.

[24] M. Chaabane, L. Gueguen, A. Trabelsi, R. Beveridge, and S. O'Hara, « End-to-end Learning Improves Static Object Geo-localization in Monocular Video », *arXiv preprint arXiv:2004.05232*, 2020.

[25] J. D. Wegner, S. Branson, D. Hall, K. Schindler, and P. Perona, « Cataloging public objects using aerial and street-level images - Urban trees », *in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-December, 2016, pp. 6014–6023, ISBN: 9781467388504. DOI: 10.1109/CVPR.2016.647.

[26] S. Branson, J. D. Wegner, D. Hall, N. Lang, K. Schindler, and P. Perona, « From Google Maps to a fine-grained catalog of street trees », *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 135, pp. 13–30, 2018, ISSN: 09242716. DOI: 10.1016/j.isprsjprs.2017.11.008. arXiv: 1910.02675.

[27] V. A. Krylov and R. Dahyot, « Object Geolocation Using MRF Based Multi-Sensor Fusion », *in Proceedings - International Conference on Image Processing, ICIP*, IEEE, 2018, pp. 2745–2749, ISBN: 9781479970612. DOI: 10.1109/ICIP.2018.8451458.

[28] W. Zhang, C. Witharana, W. Li, C. Zhang, X. Li, and J. Parent, « Using deep learning to identify utility poles with crossarms and estimate their locations from google street view images », *Sensors (Switzerland)*, vol. 18, *8*, p. 2484, 2018, ISSN: 14248220. DOI: 10.3390/s18082484.

[29] R. Hebbalaguppe, G. Garg, E. Hassan, H. Ghosh, and A. Verma, « Telecom Inventory Management via Object Recognition and Localisation on Google Street View Images », *in 2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2017, pp. 725–733. DOI: 10.1109/WACV.2017.86.

[30] F. Monti, D. Boscaini, J. Masci, E. Rodolà, J. Svoboda, and M. M. Bronstein, « Geometric deep learning on graphs and manifolds using mixture model CNNs », *in Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-January, 2017, pp. 5425–5434, ISBN: 9781538604571. DOI: 10.1109/CVPR.2017.576. arXiv: 1611.08402.

[31] B. Wang, L. Wang, B. Shuai, Z. Zuo, T. Liu, K. L. Chan, and G. Wang, « Joint Learning of Convolutional Neural Networks and Temporally Constrained Metrics for Tracklet Association », *in IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2016, pp. 386–393, ISBN: 9781467388504. DOI: 10.1109/CVPRW.2016.55.

[32] L. Leal-Taixe, C. Canton-Ferrer, and K. Schindler, « Learning by Tracking: Siamese CNN for Robust Target Association », *in IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2016, pp. 418–425, ISBN: 9781467388504. DOI: 10.1109/CVPRW.2016.59. arXiv: 1604.07866.

[33] A. Sadeghian, A. Alahi, and S. Savarese, « Tracking the Untrackable: Learning to Track Multiple Cues with Long-Term Dependencies », *in Proceedings of the IEEE International Conference on Computer Vision*, vol. 2017-October, 2017, pp. 300–311, ISBN: 9781538610329. DOI: 10.1109/ICCV.2017.41. arXiv: 1701.01909.

[34] J. Xiao, Y. Xie, T. Tillo, K. Huang, Y. Wei, and J. Feng, « IAN: The Individual Aggregation Network for Person Search », *Pattern Recognition*, vol. 87, pp. 332–340, 2019, ISSN: 00313203. DOI: 10.1016/j.patcog.2018.10.028. arXiv: 1705.05552.

[35] G. Wang, J. Lai, P. Huang, and X. Xie, « Spatial-temporal person re-identification », *in Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 8933–8940.

[36] G. Brasó and L. Leal-Taixé, « Learning a Neural Solver for Multiple Object Tracking », *arXiv preprint arXiv:1912.07515*, 2019. arXiv: 1912.07515. [Online]. Available: http://arxiv.org/abs/1912.07515.

[37] A. Kendall, M. Grimes, and R. Cipolla, « PoseNet: A convolutional network for real-time 6-dof camera relocalization », *in Proceedings of the IEEE International Conference on Computer Vision*, vol. 2015 International Conference on Computer Vision, ICCV 2015, 2015, pp. 2938–2946, ISBN: 9781467383912. DOI: 10.1109/ICCV.2015.336.

[38] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, « Going Deeper With Convolutions », *in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015.

[39] I. Melekhov, J. Ylioinas, J. Kannala, and E. Rahtu, « Relative camera pose estimation using convolutional neural networks », *in International Conference on Advanced Concepts for Intelligent Vision Systems*, Springer, 2017, pp. 675–687.

[40] X. Li and H. Ling, « Hybrid Camera Pose Estimation With Online Partitioning for SLAM », *IEEE Robotics and Automation Letters*, vol. 5, *2*, pp. 1453–1460, 2020.

[41] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab, « Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again », *in Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1521–1529.

[42] M. Rad and V. Lepetit, « BB8: A scalable, accurate, robust to partial occlusion method for predicting the 3D poses of challenging objects without using depth », *in Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 3828–3836.

[43] B. Tekin, S. N. Sinha, and P. Fua, « Real-time seamless single shot 6d object pose prediction », *in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 292–301.

[44] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, « Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes », *arXiv preprint arXiv:1711.00199*, 2017.

[45] D. C. Luvizon, D. Picard, and H. Tabia, « 2D/3D Pose Estimation and Action Recognition Using Multitask Deep Learning », *in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5137–5146, ISBN: 9781538664209. DOI: 10.1109/CVPR.2018.00539. arXiv: 1802.09232.

[46] G. Poier, D. Schinagl, and H. Bischof, « Learning Pose Specific Representations by Predicting Different Views », *in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018, pp. 60–69, ISBN: 9781538664209. DOI: 10.1109/CVPR.2018.00014. arXiv: 1804.03390.

[47] Y. Labbé, J. Carpentier, M. Aubry, and J. Sivic, « CosyPose: Consistent Multi-view Multi-object 6D Pose Estimation », *in Computer Vision – ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds., Cham: Springer International Publishing, 2020, pp. 574–591, ISBN: 978-3-030-58520-4.

[48] A. S. Nassar, N. Lang, S. Lefevre, and J. D. Wegner, « Learning geometric soft constraints for multi-view instance matching across street-level panoramas », *in 2019 Joint Urban Remote Sensing Event, JURSE 2019*, May 2019, pp. 1–4, ISBN: 9781728100098. DOI: 10.1109/JURSE.2019.8808935.

[49] A. S. Nassar, S. D'Aronco, S. Lefèvre, and J. D. Wegner, « GeoGraph: Graph-Based Multi-view Object Detection with Geometric Cues End-to-End », *in Computer Vision – ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds., Cham: Springer International Publishing, 2020, pp. 488–504, ISBN: 978-3-030-58571-6.

[50] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Z. Kolter, D. Langer, O. Pink, V. Pratt, M. Sokolsky, G. Stanek, D. Stavens, A. Teichman, M. Werling, and S. Thrun, « Towards fully autonomous driving: Systems and algorithms », *in 2011 IEEE Intelligent Vehicles Symposium (IV)*, 2011, pp. 163–168. DOI: 10.1109/IVS.2011.5940562.

[51] B. Yang, M. Liang, and R. Urtasun, « Hdnet: Exploiting hd maps for 3d object detection », *in Conference on Robot Learning*, 2018, pp. 146–155.

[52] G. Cybenko, « Approximation by superpositions of a sigmoidal function », *Mathematics of control, signals and systems*, vol. 2, *4*, pp. 303–314, 1989.

[53] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, « Learning representations by back-propagating errors », *nature*, vol. 323, *6088*, pp. 533–536, 1986.

[54] J. Håstad and M. Goldmann, « On the power of small-depth threshold circuits », *Computational Complexity*, vol. 1, *2*, pp. 113–129, 1991.

[55] S. Lefevre, D. Tuia, J. D. Wegner, T. Produit, and A. S. Nassaar, « Toward Seamless Multiview Scene Analysis from Satellite to Street Level », *Proceedings of the IEEE*, vol. 105, *10*, pp. 1884–1899, 2017, ISSN: 15582256. DOI: 10.1109/JPROC.2017.2684300.

[56] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*, 2. MIT press Cambridge, 2016, vol. 1.

[57] A. Krizhevsky, I. Sutskever, and G. E. Hinton, « Imagenet classification with deep convolutional neural networks », *Communications of the ACM*, vol. 60, *6*, pp. 84–90, 2017.

[58] K. Simonyan and A. Zisserman, « Very deep convolutional networks for large-scale image recognition », *arXiv preprint arXiv:1409.1556*, 2014.

[59] K. He, X. Zhang, S. Ren, and J. Sun, « Deep Residual Learning for Image Recognition », *in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016.

[60] J. Bromley, J. W. Bentz, L. Bottou, I. Guyon, Y. LeCun, C. Moore, E. Säckinger, and R. Shah, « Signature verification using a "siamese" time delay neural network », *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 7, *04*, pp. 669–688, 1993.

[61] S. Chopra, R. Hadsell, and Y. LeCun, « Learning a similarity metric discriminatively, with application to face verification », *in Proceedings - 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005*, vol. I, 2005, pp. 539–546, ISBN: 0769523722. DOI: 10.1109/CVPR.2005.202.

[62] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, « Geometric deep learning: going beyond euclidean data », *IEEE Signal Processing Magazine*, vol. 34, *4*, pp. 18–42, 2017.

[63] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, « Convolutional networks on graphs for learning molecular fingerprints », *Advances in neural information processing systems*, vol. 28, pp. 2224–2232, 2015.

[64] R. Levie, F. Monti, X. Bresson, and M. M. Bronstein, « Cayleynets: Graph convolutional neural networks with complex rational spectral filters », *IEEE Transactions on Signal Processing*, vol. 67, *1*, pp. 97–109, 2018.

[65] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, « A Comprehensive Survey on Graph Neural Networks », *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–21, 2020. DOI: 10.1109/TNNLS.2020.2978386.

[66] J. Masci, D. Boscaini, M. M. Bronstein, and P. Vandergheynst, « Geodesic Convolutional Neural Networks on Riemannian Manifolds », *in Proceedings of the IEEE International Conference on Computer Vision (ICCV) Workshops*, Dec. 2015.

[67] M. Fey, J. E. Lenssen, F. Weichert, and H. Muller, « SplineCNN: Fast Geometric Deep Learning with Continuous B-Spline Kernels », *in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018, pp. 869–877, ISBN: 9781538664209. DOI: 10.1109/CVPR.2018.00097. arXiv: 1711.08920.

[68] J. Wu, W. Yao, and P. Polewski, « Mapping individual tree species and vitality along urban road corridors with LiDAR and imaging sensors: Point density versus view perspective », *Remote Sensing*, vol. 10, *9*, 2018, ISSN: 20724292. DOI: 10.3390/rs10091403.

[69] R. Wan, Y. Huang, R. Xie, and P. Ma, « Combined lane mapping using a mobile mapping system », *Remote Sensing*, vol. 11, *3*, 2019, ISSN: 20724292. DOI: 10.3390/rs11030305.

[70] E. Khoramshahi, M. B. Campos, A. M. G. Tommaselli, N. Vilijanen, T. Mielonen, H. Kaartinen, A. Kukko, and E. Honkavaara, « Accurate calibration scheme for a multi-camera mobile mapping system », *Remote Sensing*, vol. 11, *23*, pp. 1–22, 2019, ISSN: 20724292. DOI: 10.3390/rs11232778.

[71] M. Hillemann, M. Weinmann, M. S. Mueller, and B. Jutzi, « Automatic extrinsic self-calibration of mobile mapping systems based on geometric 3D features », *Remote Sensing*, vol. 11, *16*, 2019, ISSN: 20724292. DOI: 10.3390/rs11161955.

[72] J. Balado, E. González, P. Arias, and D. Castro, « Novel approach to automatic traffic sign inventory based on mobile mapping system data and deep learning », *Remote Sensing*, vol. 12, *3*, 2020, ISSN: 20724292. DOI: 10.3390/rs12030442.

[73] J. Joglekar, S. S. Gedam, and B. Krishna Mohan, « Image matching using SIFT features and relaxation labeling technique -A constraint initializing method for dense stereo matching », *IEEE Transactions on Geoscience and Remote Sensing*, vol. 52, *9*, pp. 5643–5652, 2014, ISSN: 01962892. DOI: 10.1109/TGRS.2013.2291685.

[74] D. G. Lowe, « Object recognition from local scale-invariant features », *in Proceedings of the IEEE International Conference on Computer Vision*, Ieee, vol. 2, 1999, pp. 1150–1157. DOI: 10.1109/iccv.1999.790410.

[75] W. Li, R. Zhao, T. Xiao, and X. Wang, « DeepReID: Deep filter pairing neural network for person re-identification », *in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2014, pp. 152–159, ISBN: 9781479951178. DOI: 10.1109/CVPR.2014.27.

[76] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, « DeepFace: Closing the gap to human-level performance in face verification », *in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1701–1708, ISBN: 9781479951178. DOI: 10.1109/CVPR.2014.220.

[77] A. J. Nathan and A. Scobell, « How China sees America », *in Foreign Affairs*, vol. 91, 2012, pp. 956–963, ISBN: 9788578110796. DOI: 10.1017/CBO9781107415324.004. arXiv: arXiv:1011.1669v3.

[78] T. Y. Lin, Y. Cui, S. Belongie, and J. Hays, « Learning deep representations for ground-to-aerial geolocalization », *in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 07-12-June-2015, 2015, pp. 5007–5015, ISBN: 9781467369640. DOI: 10.1109/CVPR.2015.7299135.

[79] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr, « Fully-convolutional siamese networks for object tracking », *in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lec-*

ture Notes in Bioinformatics), Springer, vol. 9914 LNCS, 2016, pp. 850–865, ISBN: 9783319488806. DOI: 10.1007/978-3-319-48881-3_56. arXiv: 1606.09549.

[80] Q. Guo, W. Feng, C. Zhou, R. Huang, L. Wan, and S. Wang, « Learning Dynamic Siamese Network for Visual Object Tracking », *in Proceedings of the IEEE International Conference on Computer Vision*, vol. 2017-October, 2017, pp. 1781–1789, ISBN: 9781538610329. DOI: 10.1109/ICCV.2017.196.

[81] J. Žbontar and Y. Lecun, « Stereo matching by training a convolutional neural network to compare image patches », *Journal of Machine Learning Research*, vol. 17, *1-32*, p. 2, 2016, ISSN: 15337928. arXiv: 1510.05970.

[82] X. Han, T. Leung, Y. Jia, R. Sukthankar, and A. C. Berg, « MatchNet: Unifying feature and metric learning for patch-based matching », *in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 07-12-June-2015, 2015, pp. 3279–3286, ISBN: 9781467369640. DOI: 10.1109/CVPR.2015.7298948.

[83] Y. Tian, B. Fan, and F. Wu, « L2-Net: Deep learning of discriminative patch descriptor in Euclidean space », *in Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-January, 2017, pp. 6128–6136, ISBN: 9781538604571. DOI: 10.1109/CVPR.2017.649.

[84] B. G. Kumar, G. Carneiro, and I. Reid, « Learning local image descriptors with deep siamese and triplet convolutional networks by minimizing global loss functions », *in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-December, 2016, pp. 5385–5394, ISBN: 9781467388504. DOI: 10.1109/CVPR.2016.581.

[85] Y. Lin, L. Zheng, Z. Zheng, Y. Wu, Z. Hu, C. Yan, and Y. Yang, « Improving person re-identification by attribute and identity learning », *Pattern Recognition*, vol. 95, pp. 151–161, 2019, ISSN: 00313203. DOI: 10.1016/j.patcog.2019.06.006. arXiv: 1703.07220.

[86] X. Liu, S. Bi, X. Ma, and J. Wang, « Multi-Instance Convolutional Neural Network for multi-shot person re-identification », *Neurocomputing*, vol. 337, pp. 303–314, 2019, ISSN: 18728286. DOI: 10.1016/j.neucom.2019.01.076. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231219301043.

[87] J. Meng, S. Wu, and W. S. Zheng, « Weakly supervised person re-identification », *in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2019-June, 2019, pp. 760–769, ISBN: 9781728132938. DOI: 10.1109/CVPR.2019.00085. arXiv: 1904.03832.

[88] W. Wu, D. Tao, H. Li, Z. Yang, and J. Cheng, « Deep features for person re-identification on metric learning », *Pattern Recognition*, vol. 98, p. 107 036, 2020, ISSN: 00313203. DOI: 10.1016/j.patcog.2020.107424.

[89] T. Xiao, S. Li, B. Wang, L. Lin, and X. Wang, « Joint detection and identification feature learning for person search », *in Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-January, 2017, pp. 3376–3385, ISBN: 9781538604571. DOI: 10.1109/CVPR.2017.360. arXiv: 1604.01850.

[90] T.-W. Huang, J. Cai, H. Yang, H.-M. Hsu, and J.-N. Hwang, « Multi-View Vehicle Re-Identification using Temporal Attention Model and Metadata Re-ranking », *in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 434–442. [Online]. Available: http://openaccess.thecvf.com/content%7B%5C%5F%7DCVPRW%7B%5C%5F%7D2019/html/AI%7B%5C%5F%7DCity/Huang%7B%5C%5F%7DMulti-View%7B%5C%5F%7DVehicle%7B%5C%5F%7DRe-Identification%7B%5C%5F%7Dusing%7B%5C%5F%7DTemporal%7B%5C%5F%7DAttention%7B%5C%5F%7DModel%7B%5C%5F%7Dand%7B%5C%5F%7DMetadata%7B%5C%5F%7DRe-ranking%7B%5C%5F%7DCVPRW%7B%5C%5F%7D2019%7B%5C%5F%7Dpaper.html.

[91] X. Liu, W. Liu, T. Mei, and H. Ma, « A deep learning-based approach to progressive vehicle re-identification for urban surveillance », *in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Springer, vol. 9906 LNCS, 2016, pp. 869–884, ISBN: 9783319464749. DOI: 10.1007/978-3-319-46475-6_53.

[92] H. Altwaijry and S. J. Belongie, « Ultra-wide Baseline Aerial Imagery Matching in Urban Environments. », *in BMVC*, Citeseer, 2013.

[93] D. DeTone, T. Malisiewicz, and A. Rabinovich, « Superpoint: Self-supervised interest point detection and description », *in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 224–236.

[94] E. Park, X. Han, T. L. Berg, and A. C. Berg, « Combining multiple sources of knowledge in deep CNNs for action recognition », *in 2016 IEEE Winter Conference on Applications of Computer Vision, WACV 2016*, 2016, pp. 1–8, ISBN: 9781509006410. DOI: 10.1109/WACV.2016.7477589.

[95] A. Krizhevsky, I. Sutskever, and G. E. Hinton, « ImageNet classification with deep convolutional neural networks », *in Communications of the ACM*, vol. 60, 2017, pp. 84–90. DOI: 10.1145/3065386.

[96] K. He, X. Zhang, S. Ren, and J. Sun, « Deep residual learning for image recognition », *in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-December, 2016, pp. 770–778, ISBN: 9781467388504. DOI: 10.1109/CVPR.2016.90. arXiv: 1512.03385.

[97] K. He, G. Gkioxari, P. Dollár, and R. Girshick, « Mask {R-CNN} », *in Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2961–2969.

[98] R. Hadsell, S. Chopra, and Y. LeCun, « Dimensionality reduction by learning an invariant mapping », *in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE, vol. 2, 2006, pp. 1735–1742, ISBN: 0769525970. DOI: 10.1109/CVPR.2006.100.

[99] E. Hoffer and N. Ailon, « Deep metric learning using triplet network », *in 3rd International Conference on Learning Representations, ICLR 2015 - Workshop Track Proceedings*, Springer, 2015, pp. 84–92.

[100] X. Glorot and Y. Bengio, « Understanding the difficulty of training deep feedforward neural networks », *in Journal of Machine Learning Research*, vol. 9, 2010, pp. 249–256.

[101] D. P. Kingma and J. L. Ba, « Adam: A method for stochastic optimization », *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015. arXiv: 1412.6980.

[102] G. Neuhold, T. Ollmann, S. R. Bulo, and P. Kontschieder, « The Mapillary Vistas Dataset for Semantic Understanding of Street Scenes », *in Proceedings of the IEEE International Conference on Computer Vision*, vol. 2017-October, 2017, pp. 5000–5009, ISBN: 9781538610329. DOI: 10.1109/ICCV.2017.534.

[103] A. S. Nassar, S. Lefèvre, and J. D. Wegner, « Simultaneous multi-view instance detection with learned geometric soft-constraints », *in International Conference on Computer Vision (ICCV)*, 2019, pp. 6559–6568.

[104] Z. Gojcic, C. Zhou, J. D. Wegner, and A. Wieser, « The perfect match: 3D point cloud matching with smoothed densities », *in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2019-June, 2019, pp. 5540–5549, ISBN: 9781728132938. DOI: 10.1109/CVPR.2019.00569.

[105] Z. Gojcic, C. Zhou, J. D. Wegner, L. J. Guibas, and T. Birdal, « Learning multiview 3D point cloud registration », *arXiv e-prints*, arXiv:2001.05119, arXiv:2001.05119, Jan. 2020. arXiv: 2001.05119 [cs.CV]. [Online]. Available: http://arxiv.org/abs/2001.05119.

[106] S. Ren, K. He, R. Girshick, and J. Sun, « Faster r-cnn: Towards real-time object detection with region proposal networks », *in Advances in neural information processing systems*, 2015, pp. 91–99.

[107] K. He, G. Gkioxari, P. Dollár, and R. Girshick, « Mask R-CNN », *in IEEE International Conference on Computer Vision*, 2017, pp. 2980–2988.

[108] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, and R. Szeliski, « Building Rome in a Day », *in IEEE International Conference on Computer Vision*, 2009, pp. 72–79.

[109] S. Agarwal, Y. Furukawa, N. Snavely, B. Curless, S. M. Seitz, and R. Szeliski, « Reconstructing Rome », *IEEE Computer*, pp. 40–47, 2010.

[110] Y. Nakajima and H. Saito, « Robust camera pose estimation by viewpoint classification using deep learning », *Computational Visual Media*, vol. 3, *2*, pp. 189–198, 2017, ISSN: 20960662. DOI: 10.1007/s41095-016-0067-z.

[111] S. En, A. Lechervy, and F. Jurie, « RPNet: An end-to-end network for relative camera pose estimation », *in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11129 LNCS, 2019, pp. 738–745, ISBN: 9783030110086. DOI: 10.1007/978-3-030-11009-3_46.

[112] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, « PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes », *in Robotics: Science and Systems*, 2018. DOI: 10.15607/rss.2018.xiv.019. arXiv: 1711.00199.

[113] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, « Multi-view 3D object detection network for autonomous driving », *in Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-January, 2017, pp. 6526–6534, ISBN: 9781538604571. DOI: 10.1109/CVPR.2017.691. arXiv: 1611.07759.

[114] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. L. Waslander, « Joint 3D Proposal Generation and Object Detection from View Aggregation », *in IEEE International Conference on Intelligent Robots and Systems*, Oct. 2018, pp. 5750–5757, ISBN: 9781538680940. DOI: 10.1109/IROS.2018.8594049. arXiv: 1712.02294.

[115] J. Zhao, X. N. Zhang, H. Gao, J. Yin, M. Zhou, and C. Tan, « Object Detection Based on Hierarchical Multi-view Proposal Network for Autonomous Driving », *in Proceedings of the International Joint Conference on Neural Networks*, IEEE, vol. 2018-July, 2018, pp. 1–6, ISBN: 9781509060146. DOI: 10.1109/IJCNN.2018.8489196.

[116] V. A. Krylov, E. Kenny, and R. Dahyot, « Automatic discovery and geotagging of objects from street view imagery », *Remote Sensing*, vol. 10, *5*, p. 661, 2018, ISSN: 20724292. DOI: 10.3390/rs10050661. arXiv: 1708.08417.

[117] R. Tao, E. Gavves, and A. W. Smeulders, « Siamese instance search for tracking », *in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-December, 2016, pp. 1420–1429, ISBN: 9781467388504. DOI: 10.1109/CVPR.2016.158. arXiv: 1605.05863.

[118] C. Y. Yang, J. B. Huang, and M. H. Yang, « Exploiting self-similarities for single frame super-resolution », *in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Springer, vol. 6494 LNCS, 2011, pp. 497–510, ISBN: 9783642193170. DOI: 10.1007/978-3-642-19318-7_39.

[119] E. Shechtman and M. Irani, « Matching local self-similarities across images and videos », *in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE, 2007, pp. 1–8, ISBN: 1424411807. DOI: 10.1109/CVPR.2007.383198.

[120] Q. F. Zheng, W. Q. Wang, and W. Gao, « Effective and efficient object-based image retrieval using visual phrases », *in Proceedings of the 14th Annual ACM International Conference on Multimedia, MM 2006*, ACM, 2006, pp. 77–80, ISBN: 1595934472. DOI: 10.1145/1180639.1180664.

[121] X. Zhou, K. Yu, T. Zhang, and T. S. Huang, « Image classification using super-vector coding of local image descriptors », *in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Springer, vol. 6315 LNCS, 2010, pp. 141–154, ISBN: 3642155545. DOI: 10.1007/978-3-642-15555-0_11.

[122] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, *et al.*, « Speed/accuracy trade-offs for modern convolutional object detectors », *in Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7310–7311.

[123] S. Nilwong, D. Hossain, S.-i. Kaneko, and G. Capi, « Outdoor Landmark Detection for Real-World Localization using Faster R-CNN », *in 6th International Conference on Control, Mechatronics and Automation*, ACM, 2018, pp. 165–169.

[124] S. Sun, R. Sarukkai, J. Kwok, and V. Shet, « Accurate Deep Direct Geo-Localization from Ground Imagery and Phone-Grade GPS », *in IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 1016–1023.

[125] F. Schroff, D. Kalenichenko, and J. Philbin, « FaceNet: A Unified Embedding for Face Recognition and Clustering », *in IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 815–823.

[126] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, « Neural message passing for quantum chemistry », *in 34th International Conference on Machine Learning, ICML 2017*, vol. 3, 2017, pp. 2053–2070, ISBN: 9781510855144. arXiv: 1704.01212.

[127] R. Salakhutdinov, « Learning Deep Generative Models », *Annual Review of Statistics and Its Application*, vol. 2, *1*, pp. 361–385, 2015, ISSN: 2326-8298. DOI: 10.1146/annurev-statistics-010814-020120.

[128] A. Fout, J. Byrd, B. Shariat, and A. Ben-Hur, « Protein interface prediction using graph convolutional networks », *in Advances in Neural Information Processing Systems*, vol. 2017-December, 2017, pp. 6531–6540.

[129] J. Qiu, J. Tang, H. Ma, Y. Dong, K. Wang, and J. Tang, « DeepInf: Social influence prediction with deep learning », *in Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2018, pp. 2110–2119, ISBN: 9781450355520. DOI: 10.1145/3219819.3220077. arXiv: 1807.05560.

[130] H. Gao and S. Ji, « Graph {U}-{N}ets », *arXiv preprint arXiv:1905.05178*, 2019.

[131] R. P. P and S. M V, « Fake News Detection On Social Media Using Machine Learning », *International Journal of Computer Trends and Technology*, vol. 67, *10*, pp. 35–38, 2019. DOI: 10.14445/22312803/ijctt-v67i10p106.

[132] A. Nassar, S. Lefevre, and J. D. Wegner, « Simultaneous multi-view instance detection with learned geometric soft-constraints », *in Proceedings of the IEEE International Conference on Computer Vision*, vol. 2019-October, 2019, pp. 6558–6567, ISBN: 9781728148038. DOI: 10.1109/ICCV.2019.00666. arXiv: 1907.10892.

[133] S. Lefèvre, D. Tuia, J. D. Wegner, T. Produit, and A. S. Nassar, « Toward Seamless Multiview Scene Analysis From Satellite to Street Level », *Proceedings of the IEEE*, vol. 105, *10*, pp. 1884–1899, Oct. 2017, ISSN: 1558-2256. DOI: 10.1109/JPROC.2017.2684300.

[134] M. Kampffmeyer, A. B. Salberg, and R. Jenssen, « Semantic Segmentation of Small Objects and Modeling of Uncertainty in Urban Remote Sensing Images Using Deep Convolutional Neural Networks », *in IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2016, pp. 680–688, ISBN: 9781467388504. DOI: 10.1109/CVPRW.2016.90.

[135] M. Chaabane, L. Gueguen, A. Trabelsi, R. Beveridge, and S. O'Hara, *End-to-end Learning Improves Static Object Geo-localization in Monocular Video*, 2020. arXiv: 2004.05232 [cs.CV].

[136] Y. He, W. Sun, H. Huang, J. Liu, H. Fan, and J. Sun, « PVN3D: A Deep Point-Wise 3D Keypoints Voting Network for 6DoF Pose Estimation », *in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020.

[137] B. Doosti, S. Naha, M. Mirbagheri, and D. Crandall, « HOPE-Net: A Graph-based Model for Hand-Object Pose Estimation », *in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020.

[138] M. Gori, D. Ingegneria, and G. Monfardini, « A New Model for Learning in Graph Domains », *in Ijcnn*, vol. 2, 2005, pp. 729–734.

[139] A. Micheli, « Neural network for graphs: A contextual constructive approach », *IEEE Transactions on Neural Networks*, vol. 20, *3*, pp. 498–511, 2009, ISSN: 10459227. DOI: 10.1109/TNN.2008.2010350.

[140] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, « Spectral networks and deep locally connected networks on graphs », *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*, 2014.

[141] M. Defferrard, X. Bresson, and P. Vandergheynst, « Convolutional neural networks on graphs with fast localized spectral filtering », *in Advances in Neural Information Processing Systems*, 2016, pp. 3844–3852. arXiv: 1606.09375.

[142] T. N. Kipf and M. Welling, « Semi-supervised classification with graph convolutional networks », *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 2019. arXiv: 1609.02907.

[143] R. Levie, F. Monti, X. Bresson, and M. M. Bronstein, « CayleyNets: Graph Convolutional Neural Networks with Complex Rational Spectral Filters », *IEEE Transactions on Signal Processing*, vol. 67, *1*, pp. 97–109, 2019, ISSN: 1053587X. DOI: 10.1109/TSP.2018.2879624. arXiv: 1705.07664.

[144] J. Atwood and D. Towsley, « Diffusion-convolutional neural networks », *in Advances in Neural Information Processing Systems*, 2016, pp. 1993–2001.

[145] M. Niepert, M. Ahmad, and K. Kutzkov, « Learning convolutional neural networks for graphs », *in 33rd International Conference on Machine Learning, ICML 2016*, vol. 4, 2016, pp. 2958–2967, ISBN: 9781510829008. arXiv: 1605.05273.

[146] W. L. Hamilton, R. Ying, and J. Leskovec, « Inductive representation learning on large graphs », *in Advances in Neural Information Processing Systems*, vol. 2017-December, 2017, pp. 1025–1035. arXiv: 1706.02216.

[147] P. Veličković, A. Casanova, P. Liò, G. Cucurull, A. Romero, and Y. Bengio, « Graph attention networks », *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, 2018. arXiv: 1710.10903.

[148] R. Ying, C. Morris, W. L. Hamilton, J. You, X. Ren, and J. Leskovec, « Hierarchical graph representation learning with differentiable pooling », *in Advances in Neural Information Processing Systems*, vol. 2018-December, 2018, pp. 4800–4810. arXiv: 1806.08804.

[149] F. Diehl, « Edge Contraction Pooling for Graph Neural Networks », *arXiv preprint arXiv:1905.10990*, 2019. arXiv: 1905.10990. [Online]. Available: http://arxiv.org/abs/1905.10990.

[150] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, « A Comprehensive Survey on Graph Neural Networks », *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–21, 2020, ISSN: 2162-237X. DOI: 10.1109/tnnls.2020.2978386. arXiv: 1901.00596.

[151] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, « Dynamic Graph CNN for Learning on Point Clouds », *ACM Trans. Graph.*, vol. 38, *5*, Oct. 2019, ISSN: 0730-0301. DOI: 10.1145/3326362. [Online]. Available: https://doi.org/10.1145/3326362.

[152] T. Kipf, E. Fetaya, K.-C. Wang, M. Welling, and R. Zemel, *Neural Relational Inference for Interacting Systems*, 2018. arXiv: 1802.04687 [stat.ML].

[153] K. He, X. Zhang, S. Ren, and J. Sun, « Deep Residual Learning for Image Recognition », *in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

[154] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, « Microsoft COCO: Common Objects in Context », *in Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds., Cham: Springer International Publishing, 2014, pp. 740–755, ISBN: 978-3-319-10602-1.

[155] R. Al-Rfou, D. Choe, N. Constant, M. Guo, and L. Jones, *Character-Level Language Modeling with Deeper Self-Attention*, 2018. arXiv: 1808.04444 [cs.CL].

[156] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, « Dropout: A simple way to prevent neural networks from overfitting », *Journal of Machine Learning Research*, vol. 15, *1*, pp. 1929–1958, 2014, ISSN: 15337928.

[157] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. D. Facebook, A. I. Research, Z. Lin, A. Desmaison, L. Antiga, O. Srl, and A. Lerer, « Automatic differentiation in PyTorch », *in Advances in Neural Information Processing Systems 32*, 2019, pp. 8024–8035. [Online]. Available: `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

[158] M. Fey and J. E. Lenssen, « Fast Graph Representation Learning with PyTorch Geometric », *in ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019. arXiv: `1903.02428`. [Online]. Available: `http://arxiv.org/abs/1903.02428`.

[159] N. Z. Mohammed, A. Ghazi, and H. E. Mustafa, « Positional accuracy testing of Google Earth », *International Journal of Multidisciplinary Sciences and Engineering*, vol. 4, *6*, pp. 6–9, 2013.

[160] J. A. Musk, A. K. Elluswamy, and S. K. Sahai, *Estimating Object Properties Using Visual Image Data*, 2020.

DOCTORAT BRETAGNE LOIRE / MATHSTIC

Titre : DeepGIS : Apprendre des informations géographiques à partir d'images multimodales et de crowdsourcing

Mot clés : Vision par ordinateur, détection d'objet, ré-identification, géolocalisation, observation de la terre, apprentissage profond

Résumé : La création d'inventaires d'objets urbains et leur suivi est un processus coûteux en main-d'œuvre. Les agents de terrain effectuent généralement ce processus sur place pour collecter les propriétés sur ces objets. Ces propriétés peuvent être liées à l'emplacement, l'espèce, la hauteur et la santé d'un arbre par exemple. La lourdeur du processus de collecte de telles informations rend difficile l'étude des villes. Avec l'abondance d'images fournies par les services de cartographie en ligne (Google Maps et Street View, Mapillary, etc.), une couverture adéquate d'une ville peut être obtenue à partir de différents points de vue, tels que les vues aériennes ou au niveau du sol. La disponibilité de telles images permet la création et la mise à jour efficaces des inventaires de l'état des objets urbains en utilisant des méthodes de vision par ordinateur telles que la détection d'objets et le suivi d'objets multiples.

Cette thèse traite du problème de la détection et de la géolocalisation des objets urbains, en particulier les arbres et les panneaux de signalisation à partir de vues multiples. La résolution du problème à l'aide d'un détecteur d'objet, comme pour tout problème résolu par la vision par ordinateur, soulève les problèmes habituels d'invariance à l'occlusion, l'éclairage, la pose, le point de vue et l'arrière-plan. Nous nous appuyons sur plusieurs vues pour résoudre ces problèmes, ce qui nous permet également d'obtenir plus d'informations sur les objets à partir de ces différentes vues. L'utilisation de plusieurs vues soulève un autre défi, à savoir comment ré-identifier les objets dans ces différentes vues pour agréger les informations et ne pas obtenir les doublons d'un objet particulier. Un autre défi majeur est que les données acquises ou utilisées dans notre travail contiennent des images extraites d'une base de référence plus large, contrairement aux données utilisées pour la ré-identification des personnes ou la conduite autonome qui consistent en des séquences d'images vidéo. Pour relever ces différents défis, nous proposons plusieurs approches basées sur l'apprentissage profond pour mieux détecter, ré-identifier et géolocaliser les objets.

Dans notre première approche, nous visions à déterminer si l'utilisation de contraintes géométriques douces couplées à des informations issues des images pouvait fournir une meilleure ré-identification ou une meilleure précision de correspondance des objets à travers différentes vues. Cette méthode reposait sur des extraits visuels des objets à partir d'imagerie acquise au niveau du sol ainsi que sur des métadonnées géométriques acquises extraites des images, puis données comme entrée dans une nouvelle architecture de réseau de neurones convolutif siamois adapté aux extraits d'image. Ayant confirmé qu'utiliser des contraintes géométriques douces s'avérait bénéfique à notre modèle, notre seconde approche visait à atteindre le même objectif grâce à un modèle appris de bout en bout. Le modèle prennait alors en entrée une image entière au lieu des extraits, et notre sortie consistait en des détections de boîtes englobantes géolocalisées et ré-identifiées sur les différentes vues. Pour y parvenir, nous avons dû construire un outil pour annoter et créer un ensemble de données (arbres urbains). Notre

dernière approche exploite finalement les réseaux de neurones par graphe pour offrir davantage de flexibilité et d'efficacité par rapport à l'approche précédente. De plus, dans cette approche, nous incluons pour la première fois l'imagerie aérienne comme une autre entrée du modèle.

Pour les trois approches proposées dans cette thèse, nous avons constitué des jeux de données et réalisé des expériences approfondies permettant de démontrer l'efficacité des systèmes proposés.

Title: DeepGIS: Learning geographic information from multi-modal imagery and crowdsourcing

Keywords: Computer Vision, Object Detection, Re-identification, Geolocalization, Earth Observation, Deep Learning

Abstract: Creating inventories of street-side objects and their monitoring in cities is a labor-intensive and costly process. Field workers are known to conduct this process on-site to record properties about the object. These properties can be the location, species, height, and health of a tree as an example. To monitor cities, gathering such information on a large scale becomes challenging. With the abundance of imagery, adequate coverage of a city is achieved from different views provided by online mapping services (e.g., Google Maps and Street View, Mapillary). The availability of such imagery allows efficient creation and updating of inventories of street-side objects status by using computer vision methods such as object detection and multiple object tracking.

This thesis aims at detecting and geo-localizing street-side objects, especially trees and street signs, from multiple views. Solving the problem using an object detector, as with any problem solved with computer vision, brings up the usual problems of invariances such as occlusion, lighting, pose, viewpoint, and background. We rely on multiple views coupled with coarse pose information to solve these problems and for the benefit of getting more information about the object from these different views. Using multiple views brings another challenge, namely how to re-identify the objects in these different views to aggregate the information and not get duplicates of a particular object. Another major challenge is that the data sets acquired or used in our work contain imagery captured at a larger baseline, contrary to other data sets employed for person re-identification or self-driving and made of sequences of video frames. We propose several deep learning-based approaches to better detect, re-identify, and geo-localize objects and tackle these different challenges.

In our first proposed approach, we aimed at investigating if using soft geometric constraints coupled with image evidence would provide a better re-identification or matching accuracy of objects across different views to overcome our large baseline obstacle. This method relied on image crops of the objects from ground-level imagery and geometric metadata acquired from the image and then given as an input to a novel Siamese convolutional neural network-based architecture that matches the image crops. Having confirmed that infusing our model with soft geometric constraints proved beneficial, our second approach aimed at achieving the same objective through an end-to-end model. The model takes as input a full image instead of crops, and our output is geo-localized bounding box detections tagged with identities across different views. To achieve such a task, we had to build a tool to annotate and create a data set of urban trees. Our final approach introduces another end-to-end model that relies on graph neural networks to improve flexibility and efficiency compared to the previous one. Also, in this approach, we include aerial imagery as another input for the first time.

For all three proposed approaches in this thesis, we perform extensive experiments on curated data sets to demonstrate the proposed systems' effectiveness.