POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

# Synthetic data augmentation for illegal landfill detection in remote sensing images

Tesi di Laurea Magistrale in
Computer Science and Engineering
Ingegneria Informatica

Author: **Federico Gibellini**

Student ID: 993479
Advisor: Prof. Piero Fraternali
Academic Year: 2022-23

# Abstract

Illegal landfills represent a serious health threat due to the numerous pollutants they may release through leachate or combustion fumes. Therefore, they must be detected as soon as possible and action must be promptly taken to appropriately dispose of their waste. For these reasons, the European Union recently launched the PERIVALLON project, which this thesis is part of, aimed at exploring AI-based solutions to counteract this phenomenon and its related crimes. Within this context, Computer Vision approaches have already proven promising for solving different tasks, such as illegal landfill detection via remote sensing images, even though generally suffering from data scarcity, stemming from the difficulty in retrieving positive samples.

To address this issue, this thesis proposes a synthetic data augmentation approach, eventually allowing the generation of a high number of images containing landfill instances. For this approach, we designed a pipeline to create multiple structures of objects from different waste categories and implant such structures in aerial or satellite images. This pipeline exploits the Blender modelling tool for rendering the graphical part and combines geographic and semantic information to define feasible locations for instance placement. This process allows to create synthetic images in averagely 10 seconds per image, with instances appearing in totally realistic locations in 85% of the cases. Furthermore, these images were used to fine-tune a neural network for the task of binary scene classification, leading to minor performance improvements.

**Keywords:** illegal landfills, waste detection, synthetic data, augmentation

# Sommario

Le discariche illegali rappresentano una seria minaccia per la salute a causa dei numerosi inquinanti che queste possono rilasciare tramite percolato o fumi di combustione. È quindi importante identificarle il prima possibile e agire prontamente per smaltire in modo opportuno i rifiuti che le compongono. Per questo motivo, l'Unione Europea ha recentemente avviato il progetto PERIVALLON, di cui è parte questa tesi, con l'obiettivo di esplorare soluzioni fondate sull'intelligenza artificiale per contrastare tale fenomeno e i crimini ad esso legati. In questo contesto, alcuni approcci nel campo della Computer Vision si sono già dimostrati promettenti per rispondere a diverse consegne, come l'identificazione di discariche illegali tramite immagini di telerilevamento, nonostante queste consegne soffrano spesso del problema di scarsità dei dati, originato dalla difficoltà di reperire campioni positivi.

Per affrontare questo problema, questa tesi propone un approccio di augmentation tramite dati sintetici tale da consentire la generazione di un numero elevato di immagini contenenti istanze di discariche. Per questo approccio, abbiamo progettato una pipeline per creare varie strutture di oggetti di diverse categorie di rifiuti e impiantare tali strutture in immagini aeree o satellitari. Questa pipeline sfrutta lo strumento di modellazione 3D Blender per il rendering grafico e combina informazioni geografiche e semantiche per definire posizioni accettabili dove introdurre le istanze. Questo processo consente di creare immagini sintetiche in mediamente 10 secondi per immagine, con istanze in posizioni totalmente realistiche nell'85% dei casi. Queste immagini sono poi state utilizzate per fare fine-tuning di una rete neurale per classificazione binaria delle scene, portando a lievi miglioramenti delle prestazioni.

**Parole chiave:** discariche illegali, identificazione di rifiuti, dati sintetici, augmentation

# Contents

**Abstract** **i**

**Sommario** **iii**

**Contents** **v**

**1 Introduction** **1**

**2 Related work** **5**

2.1 Illegal landfill detection . . . . . 5

2.1.1 State-of-the-art approaches . . . . . 5

2.2 Synthetic data . . . . . 9

2.2.1 Fundamental works . . . . . 9

2.2.2 Approaches in Remote Sensing . . . . . 13

**3 Method** **17**

3.1 Inputs . . . . . 17

3.1.1 AerialWaste . . . . . 18

3.1.2 3D models . . . . . 19

3.1.3 DUSAF7.0 . . . . . 22

3.1.4 DTM5x5 . . . . . 23

3.1.5 OpenEarthMap . . . . . 24

3.2 Scene setup and rendering . . . . . 25

3.3 Instance creation . . . . . 27

3.3.1 *Stacks* and *Scattered objects* . . . . . 27

3.3.2 *Heaps* . . . . . 28

3.4 Instance placement . . . . . 29

3.4.1 Land cover . . . . . 30

3.4.2 DTM . . . . . 34

3.4.3 Semantic Segmentation . . . 35
3.4.4 Shadow Detection . . . 35
3.4.5 Google overlays . . . 39
3.4.6 Online phase . . . 41

**4 Experiments** **43**
4.1 Computation environment . . . 44
4.1.1 Semantic Segmentation network . . . 44
4.1.2 Shadow detection threshold . . . 45
4.2 Computation efficiency evaluation . . . 48
4.3 Placement evaluation . . . 48
4.4 Inference on synthetic images . . . 50
4.5 Training with synthetic data . . . 50

**5 Conclusions and future work** **55**

**Bibliography** **57**

**A Land cover categories** **65**

**B Geometry nodes** **67**
B.1 *Heaps* . . . 68
B.2 *Scattered objects* . . . 69
B.3 *Stacks* . . . 70

**C Rubble collection creation** **73**

**Acronyms** **77**

**List of Figures** **79**

**List of Tables** **81**

**Acknowledgements** **83**

**Ringraziamenti** **85**

# 1 | Introduction

Environmental risk assessment is the set of practices and measures to evaluate the likelihood and descending effect entity of the potential harm caused by human activities to the environment and all of its inhabitants, with the aim of safeguarding the planet as well as guiding decision-making to counteract such harm. [1, 2]. In this context, illegal landfills represent a paramount issue for multiple reasons. First of all, such waste sites are significantly dangerous for the environment and for peoples' health, since massive amounts of untreated waste might severely pollute soil by leaching great quantities of heavy metals, thus inducing a severe risk where these landfills are located close to agricultural areas [3]. In addition, illegal landfills usually contain hazardous materials and frequently happen to be burnt for disposal, thus leading the aforementioned materials to release in the air thousands of cube meters of lethal compounds and fumes [4]. Taking this into account, it is not surprising that a recent study [5] also identified a positive correlation between the presence of illegal landfills in a municipality and the mortality and hospitalization in the same area for multiple serious diseases, such as leukemia, cancer and tumors. Finally, beside being accountable for all these health-related issues, illegal landfills also represent a major artifact of waste trafficking, a frequent crime in Europe, which is centered on bypassing regulations in order to reduce waste disposal costs [4].

For all these reasons, in December 2022, the European Union launched a 3-year project, named PERIVALLON, aimed at developing solutions to counteract environmental crimes by leveraging the most cutting-edge technologies in fields such as Artificial Intelligence (AI) and Computer Vision (CV) [6]. These fields, often exploiting methods from the Deep Learning (DL) area, such as Convolutional Neural Networks (CNN), have recently been thriving with works aimed at promoting modern approaches in waste management, with topics ranging from chemical analyses of waste to the design of innovative applications such as smart bins or waste-sorting robots [7]. Numerous studies from all over the world have also addressed the complex task of illegal landfill detection [7], which consists in trying to determine the presence of illegal landfills in Remote Sensing (RS) images, i.e. images captured by UAVs or satellites, with the eventual aim of accelerating the site identification process while relieving the necessity for physical site inspection [8].

Recent studies approach this task with the implementation of custom CNNs, sometimes supported by a Feature Pyramid Network (FPN) [9] to account for the multiple scales at which a landfill might appear in an aerial image [8, 10, 11]. However, these approaches face the serious issue of data scarcity since obtaining the necessary labels to effectively train a CNN usually requires technical expertise [8] and might be a significantly time-consuming process [10]. To overcome this limitation, multiple approaches have already been attempted; for instance, some studies focused on creating benchmark datasets [10, 12], whereas others applied methods from the area of Weakly/Self-Supervised Learning [13]. In this context, the adoption of synthetic data to augment the training dataset has already proven promising for waste detection [13, 14], even though hindered, sometimes, by a poor realism in the artificially generated images [13].

This thesis aims at proposing an alternative approach in the field of Synthetic Data Augmentation (SDA) for illegal landfill detection, based on the implantation in RS images of various object structures for multiple waste categories.

The contributions of this work can be summarized as follows:

- We design a pipeline leveraging Blender, a popular 3D modelling tool [15], to introduce synthetic waste instances in RS images. This pipeline also combines geographic and semantic information to determine realistic locations where to insert instances.
- We evaluate qualitatively and quantitatively the performances and outputs of such approach and observe that it allows to generate synthetic images in averagely $\approx$ 10 seconds per image, with instances appearing in totally realistic locations in 85% of the cases. Furthermore, fine-tuning experiments highlight that extending the training set with synthetic images usually results in increases in F1 and recall, in spite of decreases in accuracy and precision.

The rest of the thesis is organized as follows:

- **Chapter 2** discusses a set of studies related to the work presented in this thesis. A first part focuses on approaches for landfill and waste detection, whereas a second one describes recent approaches adopting synthetic data.
- **Chapter 3** presents the designed method for data augmentation focusing on the details of the input information and of all the phases that allow to introduce synthetic waste instances in RS images.
- **Chapter 4** describes the conducted experiments for quantitative and qualitative evaluation of the proposed method. An introductory part is dedicated to the experiment environment and to practical configurations of some of the inputs.

- **Chapter 5** presents the conclusions by resuming the work of this thesis. This chapter ends with a presentation of opportunities for future work.

# 2 | Related work

## 2.1. Illegal landfill detection

Illegal landfills represent a serious problem for people's health and for the environment safety, having been proven accountable for numerous hazards [16]. A major threat is represented by waste leachate, which might contaminate soil and pollute underground water by releasing significant quantities of heavy metals [3, 16]. In addition, criminal organizations frequently happen to set fire to waste disposals with the aim of deleting evidence of the presence of hazardous materials, thus allowing for diffusion in the air of a wide range of dangerous chemicals, such as carbon monoxide and dioxide, which are obtained from the combustion of such materials [4]. All these polluting substances are severely dangerous for human health and represent the main motivation for the increase of serious diseases in locations close to illegal disposals [5].

For all these reasons, it is paramount to promptly detect illegal landfills and to steadily counteract their diffusion or expansion. In this regard, governments have already performed various attempts, such as installing cameras to ease monitoring and patrolling [17] or requesting citizens to signal potential anomalies [18]. However, these methods generally result in failures while requiring also a significant contribution from human entities, a contribution which can nowadays be relieved thanks to the latest technological advances in the field of landfill detection. A historical overview of such advances is presented in the following section.

### 2.1.1. State-of-the-art approaches

Early approaches to landfill detection date back to the ending decades of the previous century, when studies still leveraged human interpretation and suggested that aerial images allowed to determine size, location and composition of waste sites, along with their chronological evolution where historic images were available [19–21]. In those years, it became also apparent that support might have arrived from aerial thermographic images obtained via infra-red cameras, leveraging the discovery that temperature in waste sites

was anomalous with respect to their surroundings [22]. Then, in the early years of the new millennium, remotely sensed data and digital ortho-photos allowed to shift to approaches based on image processing, with particular attention to spatial and spectral features of areas close to the dumping sites [23, 24]. This trend in spatial analysis continued until the last years of the decade, when these approaches were integrated with data from external sources, such as demographic and geographic data from GIS systems [25, 26], and satellite images started to gain popularity and to be reckoned promising [27].

However, in 2015, the state of the art for illegal landfill detection had not progressed significantly from what just mentioned, as Glanville et al. [28] asserted in their article. Moreover, these scholars also observed that most of the previous studies had been conducted in Europe, Japan and in the US, thus completely neglecting other parts of the world, such as the country they lived in, Australia. Given these shortcomings, the authors tried to identify the technical requirements for advances in landfill detection and asserted that Remote Sensing images had a great potential yet not fully explored. However, they also observed that, most likely, serious advances would have required the adoption of very high resolution images, such as aerial images or those which satellites were expected to provide only a few years later.

High resolution images became central in the work by Selani et al. [29], which is also one of the first studies to apply Machine Learning (ML) approaches to illegal landfill detection by addressing this problem as a classification task. To solve this task, the authors created a custom dataset of 610 multi-spectral satellite images, which they labelled into 6 categories, with 2 of them referring to waste. Subsequently, the scholars trained 2 different models, a Support Vector Machine [30] and a Random Forest (RF) [31] and notice that the first outperforms the latter. Nevertheless, RF have been adopted also recently by Ulloa-Torrealba et al. [32], in a method that combines a RF model with a segmentation approach to detect waste on the streets via aerial images and GIS data.

Selani et al. [29] inaugurated what has now become the main trend in landfill detection, i.e., the use of AI methodologies. In this sense, the last few years of research have been dominated by approaches leveraging CNNs and DL methodologies.

In 2019, Abdukhamet [33] considered illegal landfill detection as an object detection task and he addressed it with a RetinaNet [34] in which the backbone had been replaced by a DenseNet [35]. Then, he built a dataset of more than 2000 images from the Shanghai area and labelled them manually with bounding boxes to highlight the garbage. This approach resulted in an average accuracy of 84.7% when setting the IoU parameter to 0.3. Furthermore, he observed that (i) results could be improved with some geometric

data augmentation, such as flipping and rotation, as well as that (ii) different results could be obtained by changing the size of the input, since a different amount of context information was available.

Two years later, in 2021, Youme et al. [36] followed a similar approach to detect instances in images caught via drones in Saint Louis, Senegal. For this work, 5000 images were manually labelled with bounding boxes to train a Single Shot Detector [37] which demonstrated to perform well even though detecting a high number of false positives.

In the same year, Devesa and Brust [38] addressed the problem as an object segmentation task leveraging the increasing availability of open satellite images from the Buenos Aires area. The authors trained a UNet [39] on a dataset of almost 2000 6-band manually annotated images and achieved 63% IoU. They also conducted ablation studies by removing some of the bands in each image, but the best model still resulted to be the one trained with all 6 bands.

In 2022, Shahab and Anjum [40] designed a multi-path CNN to address the detection problem as a binary classification task (waste/no-waste). Since no benchmark dataset was available at the time, the authors created their own custom dataset with 6000 images per class by collecting the candidates from various sources on the Internet and placing in each category only those images that were classified as such by both the authors after visual inspection. Images in the non-waste class contained no instance of waste, whereas positive images contained instances occupying at least a fourth of the whole picture. Upon training, 9000 images (75% of each class) were used for training and validation, whereas the remaining 3000 were used for testing. The model demonstrated to perform exceptionally well, achieving 98.33% overall accuracy.

Finally, in 2023, Sun et al. [10] spent 6 months, supported by 12 experts, to create and label a dataset with 2200 sub-meter GSD images, containing 2500 dumpsites from multiple cities in the world, divided into 4 classes. Subsequently, they designed and trained a custom deep CNN, called BCANet, using the Faster R-CNN [41] as baseline. Such network managed to detect more than 98% of the landfills when trained with some geometric augmentation to relieve class imbalance. The authors claim that this method might reduce of 96.8% the inspection time to detect illegal landfills.

In this rich environment of works, two notable studies were published by Torres et al. [8, 12]. The first of these studies addresses the problem of illegal landfill detection as a multi-scale scene classification task. In order to solve it, the authors created, with the aid of experts from a local governmental environmental agency, a dataset of 3000 sub-meter-resolution images from Northern Italy, purposely designed with a higher number

of negatives to specifically account for real imbalance. Then, they trained a CNN with a ResNet50 [42] backbone, pre-trained on the ImageNet [43] dataset and augmented by an FPN, which should allow it to better account for multiple scales. This model achieved 88% precision and 87% recall on the test set. Subsequently, the obtained results were evaluated both quantitatively and qualitatively. From the quantitative point of view, the authors considered standard metrics, such as average precision and F1 score, as well as the Expected Calibration Error, which describes how reliable a model output confidence is with respect to the actual confidence of an image belonging to a certain class. The authors assert that the found error levels are valid and, therefore, that the model may be adopted by analysts. For qualitative analysis, the authors extracted and analysed Class Activation Maps to detect which portions of an image affect the output confidence the most. The model seems to behave well in this context, identifying in most cases the same areas that guided the decisions of expert annotators. This led the authors to eventually assert that their model might be a useful support to optimize the process of illegal dump detection.

In their second work [12], Torres et al. published AerialWaste (AW), an extended version of the dataset adopted in the previous study. This dataset contains more than 10,000 sub-meter resolution images from 3 different sources and it is suitable for multiple tasks: binary/multi-label classification and weakly supervised localization. In order to verify the validity of these images, the authors trained the same model presented in the previous study on images from AW, achieving 82% precision and 80% recall.

Following the future work suggestions by Torres et al. [8], Fasana et al. [13] addressed illegal landfill detection with weakly supervised approaches. In their work, the authors considered the detection problem as a multi-label classification task and evaluated several methodologies, such as training set extension, transfer learning and self-supervised learning, to counteract the major problem of data scarcity. Then, they trained the model proposed by Torres et al. [8] for each of those methodologies, achieving the highest results when augmenting the dataset with synthetic data.

This type of data has been adopted also by Padubidri et al. [14], who proposed, in 2022, a different DL approach addressing high-resolution airborne images from the city of Houston, USA. The authors trained 2 different models: a basic CNN with 3 hidden layers and a model composed of residual blocks. Given the unavailability of positive data, the authors adopted a synthetic augmentation approach leveraging the 3D modelling tool Blender [15] to implant waste instances in aerial images. Then, they could train the models using as positives the synthetically augmented images and few real positives, while leaving the other real positives for the test set. After training the models, the authors

achieved 98% precision and 90% recall with the CNN model as well as 97% precision and 92% recall with the residual one.

## 2.2. Synthetic data

For the concept of *Synthetic data* no definition has been universally accepted yet. However, by combining multiple sources, it is possible to agree that synthetic data are artificially obtained as output of some mathematical process or algorithm without leveraging direct measurement, unlike real data [44–47]. Throughout the last decade, synthetic data have been used in several ML contexts, such as Natural Language Processing or bioinformatics, even though the field which most benefited from this practice is CV [47]. Such a wide adoption of this approach is motivated by two main reasons. Firstly, synthetic data allow to easily gather a significant quantity of information, enough to train complex supervised models, from even rare or dangerous situations and without major privacy concerns. Secondly, they offer the possibility to automatically annotate the generated data, an opportunity which is incredibly valuable in CV, where obtaining labels such as bounding boxes or object segmentations usually results in prohibitive costs [48]. In addition, this type of data have already proven to be really effective, achieving results comparable with those obtained by models trained with only real data, while being efficient in terms of economic and temporal costs [48].

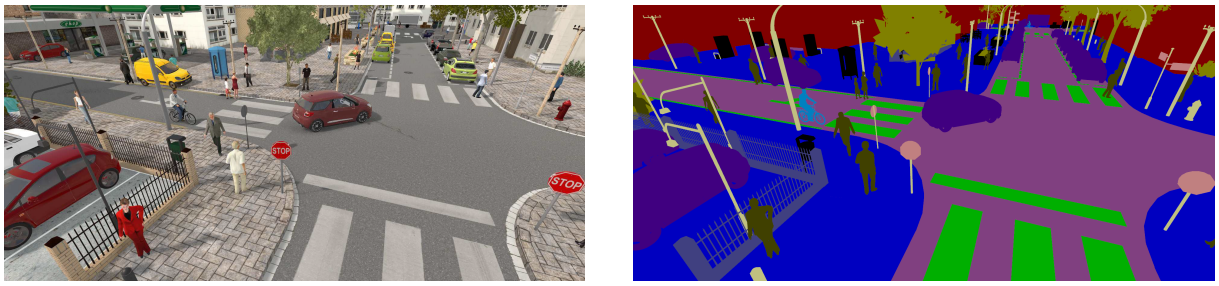
Given the scope of this thesis, this section focuses on the different approaches involving synthetic data in the field of CV with specific attention to the two main directions that can be followed when dealing with these data [47]: using them (i) to build standalone datasets or (ii) to augment real datasets with the aim of improving results from model training.

### 2.2.1. Fundamental works

To this day, synthetic data have already been adopted for a wide range of tasks, from optical flow estimation to depth estimation, from semantic segmentation to 3D scenario reconstruction [49]. However, throughout the last few decades, a specific field, i.e., autonomous driving, has always pioneered approaches in this context, with studies involving synthetic data being publicly released since the late 80s [48]. To support advances in this field and a major task for it, i.e., urban Semantic Segmentation (SS), a few significant synthetic datasets were released in 2016.

The SYNTHIA dataset [50] counted more than 213,000 photo-realistic snapshots of a

virtual city created with the Unity development platform [51] and showing significant realism and variability. In this virtual scene, almost every element could be changed dynamically, from object textures to lighting, from weather and seasons to the camera view-point, thus allowing the output images to be as diverse as possible while lowering the required effort for potential future extensions. Furthermore, being all such images obtained via rendering of a virtual environment, the authors were able to automatically generate SS labels with pixel-level accuracy for 13 classes, thus reducing sensitively the temporal costs for annotation. The dataset was then tested on 2 different networks and it demonstrated incredible performances when combined to real images, providing in some cases accuracy improvements of more than 10%.

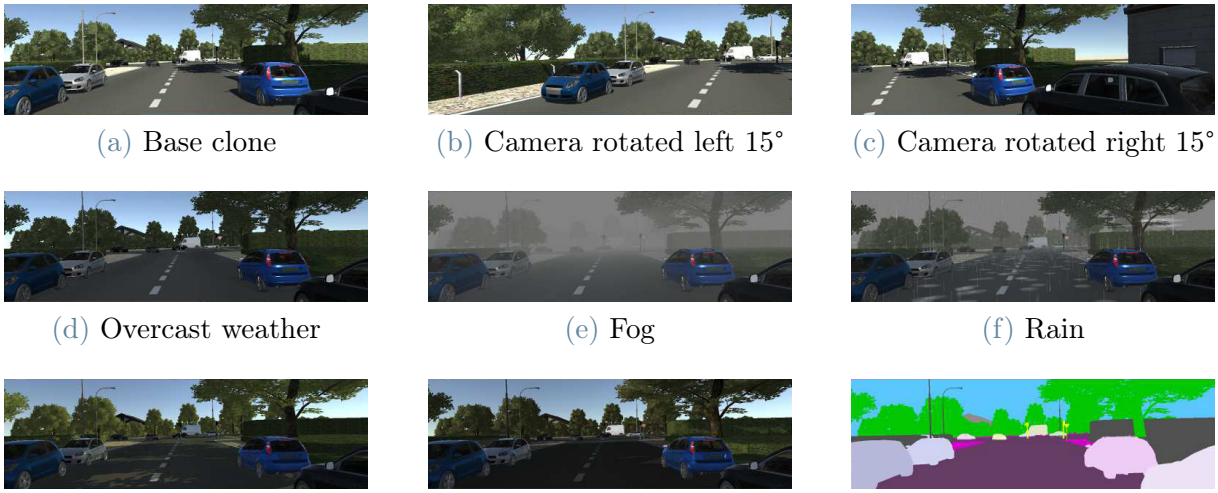


(a) Color image (b) Annotation

Figure 2.1: Example from the SYNTHIA dataset. From [50].

The Virtual KITTI dataset [52] contained only 17,000 frames collected in 35 photo-realistic synthetic video sequences for several CV tasks, such as scene segmentation and optical flow estimation. In order to create such sequences, the authors leveraged a selection of 5 real urban videos from the KITTI dataset [53] to initialize camera parameters and to virtualize the real scenes by cloning objects into similar publicly available assets. Then, the authors created 7 variations of each video, differing in environmental settings such as weather and illumination, and automatically generated their ground-truth annotations (Figure 2.2). The authors, assessing pedestrians to be complex to animate, focused only on the class of cars and conducted some experiments on it. With these experiments, they demonstrated that virtual pre-training followed by real-world fine-tuning allowed a significant increase in deep model performances, whereas the changing of scene conditions was accountable for serious deteriorations.

An additional paramount work for the task of urban SS is represented by a dataset of 25,000 images from the popular videogame Grand Theft Auto V [54]. To extract these images, the authors applied a technique called detouring, which allowed them to obtain the standard video game images and pixel-accurate annotations (Figure 2.3a). By introducing a wrapper at the graphics API level, the authors could intercept all the



(a) Base clone (b) Camera rotated left 15° (c) Camera rotated right 15°

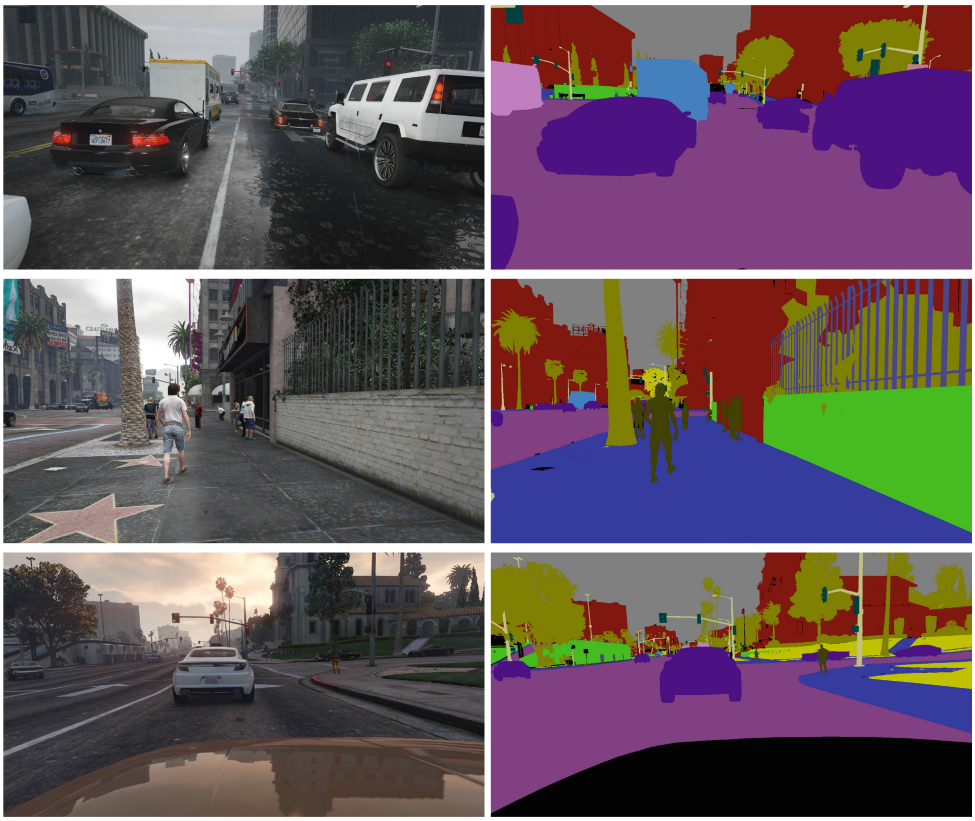
(d) Overcast weather (e) Fog (f) Rain

(g) Morning (h) Sunset (i) Ground-truth label

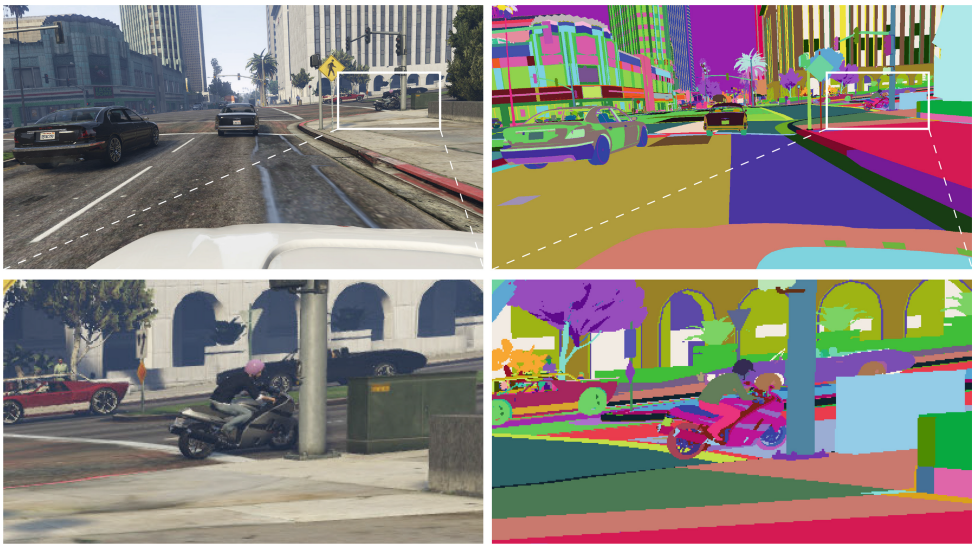
Figure 2.2: Example from the Virtual KITTI dataset. Original image, variations and segmentation label. Adapted from [52].

rendering commands and the related resources, i.e., mesh, shaders and textures. Then, they could duplicate the render pass to produce two images: the usual color image from the videogame and an image where each pixel was given a value depending on a hash of its <Mesh-Texture-Shader> (MTS) combination (Figure 2.3b). Subsequently, they applied a rule mining method to group the MTS combinations by occurrence frequency and they adopted a simple interface to manually annotate the MTS into the final classes. This interface also allowed them to propagate the mapped classes to all other images where the just-annotated MTS appeared, thus requiring only a single annotation for each MTS. As a result of this, the authors were able to obtain pixel-perfect annotations in only 49 hours, a time which is almost 3 magnitude orders lower than what they could have expected from manually labelling each pixel. In addition, experiments showed that with just 1/3 of the CamVid dataset [55] and all the synthetic images it is possible to outperform models trained solely on the CamVid dataset.

A similar approach was adopted by Butler et al. [56] to create the MPI-Sintel dataset, which addressed the task of optical flow estimation, another task which is significantly affected by data scarcity, given the absence of sensors for natural motion measurement. The MPI-Sintel dataset was built leveraging *Sintel*, an open-source short movie produced with Blender [15]. Given the open-source nature of such video, it was easy for the authors to access all the code to generate it. Therefore, they were immediately able to edit render passes in the Blender [15] motion blur pipeline and obtain different rendered images, including the necessary ground-truth annotations. In particular, to obtain such annota-



(a) Original images and their final ground-truth annotations. .



(b) Segmented images with pixels grouped by MTS.

Figure 2.3: Examples from the Grand Theft Auto V dataset. From [54].

tions, the authors replaced, in a specific render pass, color vectors with motion vectors, thus producing images that represented the motion of each pixel (Figure 2.4). Finally, to build the dataset, they extracted 35 clips of 50 frames each, thus providing 49 flow images

each. On this dataset, the authors experimented common estimation methods and noted that those who used to work well with available benchmarks performed definitely worse on this dataset, thus opening a wide space for future research.

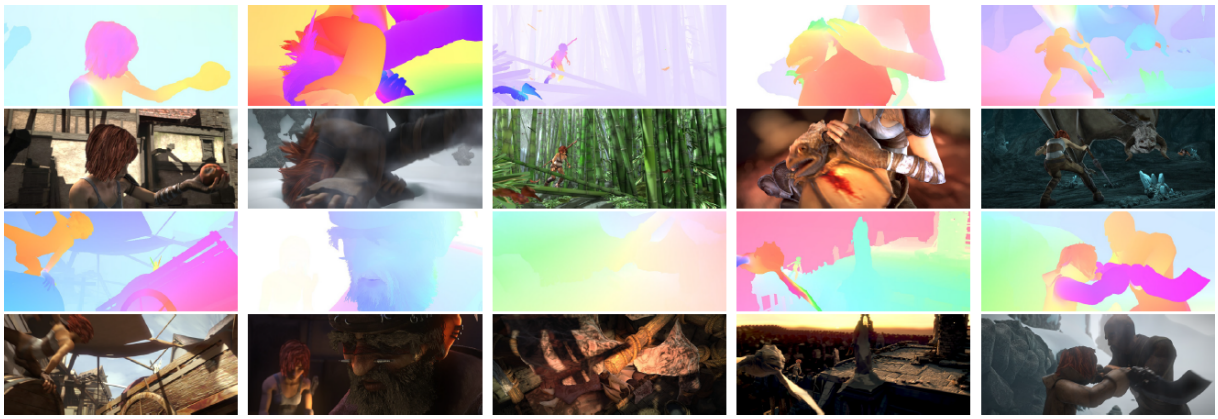


Figure 2.4: Examples of frames and related motion images from the MPI-Sintel dataset. Adapted from [56].

### 2.2.2. Approaches in Remote Sensing

In the recent years, synthetic data have become more and more relevant also in the field of RS, another field in which data acquisition and publication on a large scale have proven significantly complex for several reasons. Among these, the costs for image collection and major privacy concerns are paramount causes of the general availability of small datasets [49]. In this context, synthetic data have emerged as a valuable resource to overcome current limitations and, therefore, they have been adopted for multiple different tasks.

One of the first works in this field [57] consisted in the creation of Synthinel-1, a dataset for building segmentation released in 2020 and composed of more than 2,000 synthetic overhead images at 0.3 m/px resolution. These images were obtained via CityEngine [58], a 3D modelling software specifically designed to create urban aerial images, allowing the creation of large virtual worlds with multiple styles. Among these, the authors selected a set of 9 styles (Figure 2.5) to simulate cities all over the world, thus overcoming geographic shortcomings of most of the available real datasets. Cities are created with a 2-step process: first, the road network is randomly generated, then the areas in between the roads are populated with models such as buildings, trees or landscapes. The resulting images were used to train 2 deep learning architectures by augmenting real datasets. The authors found, after ablation studies, that the best models can be obtained by considering a subset of all the images which excludes those generated with specific styles. In such case, they noticed improvements both in within-domain and out-of-domain tasks.

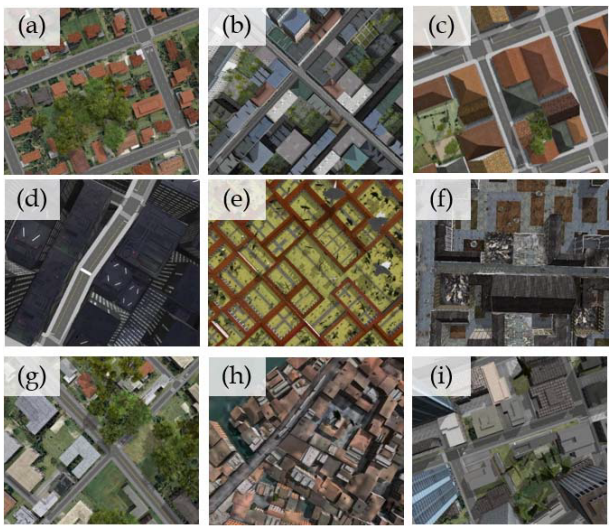



Figure 2.5: Selected styles for Synthinel-1 aerial synthetic images. From [57].

One year later, Shermeyer et al. [59] released RarePlanes, a dataset composed of both real and synthetic images (Figure 2.6) and targeting the task of aircraft detection. In this dataset, the 253 real images contain 14,700 annotated aircraft whereas the 50,000 synthetic images account for 630,000 annotations. To obtain the synthetic images, the authors first combined a proprietary GIS framework and geospatial vectors to create airport sites. After adding runways, they imported the newly-created airport scene in Unreal Engine [60], and spawned aircrafts in it. Then, they adjusted parameters defining weather, illumination and the surrounding biome to increase the image variability. After performing some experiments, the authors noticed that the dataset they created allowed to achieve the same training results as a 100% real dataset with one composed for 90% of synthetic images.

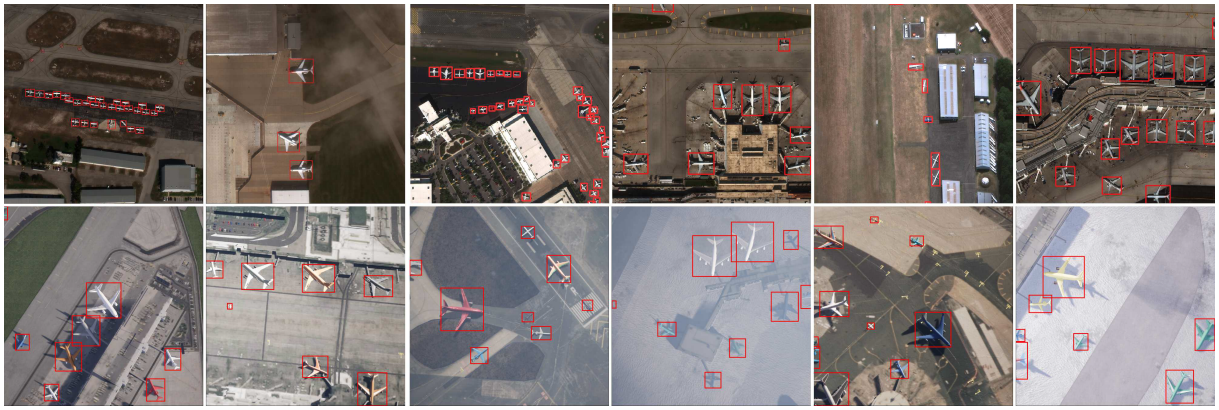


Figure 2.6: Examples of real (top row) and synthetic (bottom row) images from the RarePlanes dataset. Adapted from [59]

After noticing that most of the available works involved complex designs and could not be replicated, Xu et al. [61] decided to systematize a simple and reproducible method, called SIMPL, for creating synthetic overhead images. This method consists in *implanting*, via CityEngine [58], 3D models into real sub-meter-resolution satellite images, used as background to enhance realism. To compose the virtual scene, a ground plane is first introduced with an overhead image applied as texture; then, 3D models are randomly placed in the scene and some user-defined parameters are leveraged to define the object visual properties and other settings such as lighting. Finally, the image can be rendered and annotations can be automatically generated. The conducted experiments show that images created with this method improve training performances for the tasks of zero-shot and few-shot detection.

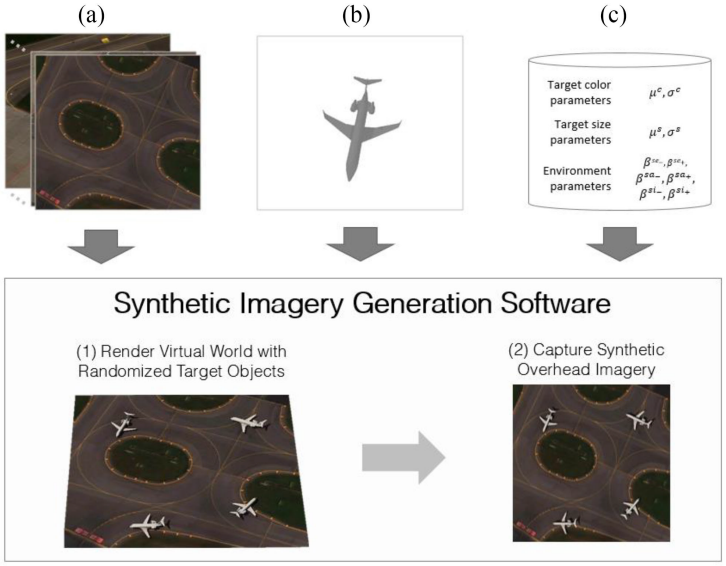



Figure 2.7: The SIMPL procedure for synthetic aerial images generation. (a) is a real image for background, (b) is a 3D object model, (c) is the set of parameters to build the scene. From [61]

Synthetic data have also been recently used for waste detection. In 2022, Padubidri et al. [14] addressed this problem as a binary classification task (dumping/no-dumping). They extracted and visually annotated 29,000 patches from a dataset of high-resolution aerial images from the city of Houston. Among these patches, they could annotate only 176 positives, thus highlighting the common problem of positive instance scarcity in the context of waste detection. In order to solve this issue, the authors randomly selected 2,000 negative images and edited them with the introduction of dumping instances by means of Blender [15]. Then, they trained two different networks on a training set composed of 16,000 negative patches, the 2,000 synthetic images and 35 real positives. The trained

models showed similar high performances, achieving both more than 97% precision and more than 90% recall on a test set completely composed of real images.

In the same field, also Fasana et al. [13] attempted an approach involving synthetic data. In particular, they adopted these data to augment images from the AW dataset [12] when exploring various weakly supervised approaches to overcome data scarcity for the multi-label classification task. The authors created synthetic images by extracting waste patches from positive samples and pasting them into negatives with small alterations, such as patch border blur and patch rotations. Despite noticing that this method usually created unrealistic positives, they also obtained the best results when training the chosen architecture with this augmentation.

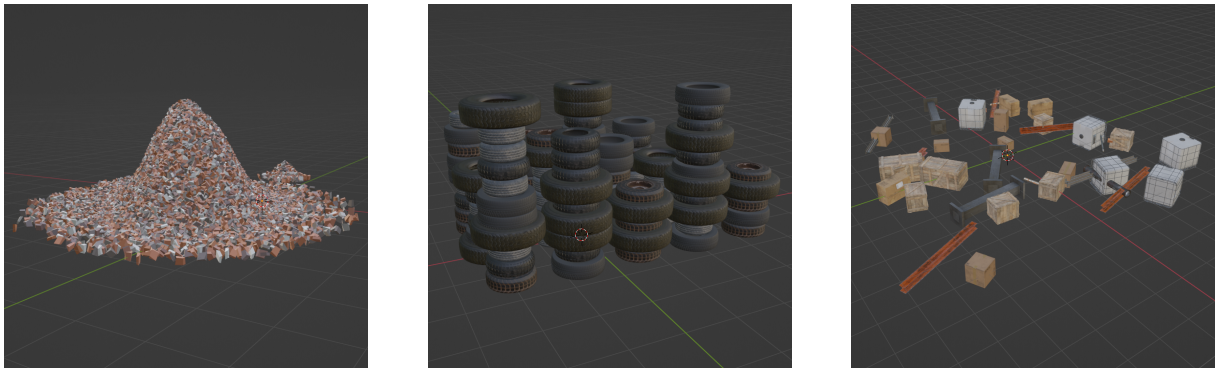
Finally, in 2023, Song et al. [49] published SyntheWorld, a large dataset composed of 40,000 synthetic images with sub-meter resolution addressing the tasks of land cover mapping, building segmentation and building change detection. The images were obtained by combining Blender [15], a 3D modelling tool, with several Python scripts. Among these, two specific scripts allowed the authors to generate urban scenes with a road grid and natural scenes with rivers or mountains, in which models were introduced by exploiting the *Geometry nodes* Blender feature. Other scripts accounted for randomization of the light and camera parameters, whereas Blender allowed to procedurally generate buildings and trees for an increased variability. Furthermore, the authors leveraged recent generative AI models to create seamless textures for various scene objects, such as roofs or roads. Finally, after conducting experiments for all the tasks addressed by the dataset, the authors noticed that SyntheWorld outperforms other synthetic datasets in building segmentation, it improves network performances when combined with portions of real datasets in land cover mapping, and it achieves acceptable results in building change detection.

(a) Semantic Segmentation (b) Building change detection

Figure 2.8: Examples from SyntheWorld. Adapted from [49].

# 3 | Method

This section describes the designed method to augment RS images, which is similar to the SIMPL method presented in [61] with the main differences being that, in this case, objects are arranged in complex structures and that their placement depends on the image context. With this method, we can generate 3 structures, *Heaps*, *Stacks* and *Scattered objects*, for 3 waste categories from the AW dataset [12], *Tires*, *Rubble/excavated earth and rocks* (henceforth, *Rubble*) and *Bulky items* (Figure 3.1).



(a) *Heap* of *Rubble* (b) *Stacks* of *Tires* (c) *Scattered Bulky items*

Figure 3.1: Object structures and waste categories.

The designed method was implemented, for the graphical part, with Blender [15], a popular 3D modelling tool which can also be controlled via Python scripts, thanks to the *bpy* library. Therefore, the whole procedure was completely automated, thus requiring no human intervention to complete.

The following sections describe more in detail the designed augmentation pipeline and all the necessary inputs.

## 3.1. Inputs

The designed pipeline leverages several external inputs, which are needed for different tasks in the pipeline: the AW images and the 3D models are used in the graphical

part, whereas all other inputs support the process of defining legal positions for instance placement.

This section describes all the inputs and their features, leaving the technicalities about their usage to the next sections, which focus on the specific processes involving such inputs.

### 3.1.1. AerialWaste

AerialWaste is a dataset of RS images developed as a collaboration between Politecnico di Milano and ARPA Lombardia, the environment monitoring agency of Region Lombardy, Italy. Being it mainly developed at the same institution where this thesis work was conducted, it has been possible to take advantage of additional information, such as location of the areas in the images, information which is not in the public domain for privacy concerns. Furthermore, at the time this work was conducted, an extended version of the dataset presented in [12] was already available within the research group. Therefore, for this thesis, such version was adopted and all the statistics in these pages refer to it.

The dataset contains 10,977 images from 3 different sources (Figure 3.2) but all covering a squared area with a side of 210 meters. However, since the different sources provided images with different Ground Sample Distance (GSD), images from each source vary in their average size. In particular:

- **AGEA** images are the result of an airborne campaign launched in 2018 by the Italian Agriculture Development Agency (AGEA). These images have a GSD of $\approx 20$ cm/px and a size of $\approx 1000 \times 1000$ pixels.
- **WorldView-3** images are high-resolution pan-sharpened RGB satellite images captured in 2021. They have a GSD of $\approx 30$ cm/px and a size of $\approx 700 \times 700$ pixels.
- **Google Earth** (GE) images were downloaded with the Google API. Their original sampling resolution is $\approx 50$ cm/px but they have been up-sampled with scale factor 2, thus resulting in having a size of $\approx 1000 \times 1000$ pixels.

AerialWaste is originally split in training set (75%) and test set (25%) and it has been annotated by professional photo interpreters to support 3 different tasks:

- For **binary classification**, images have been annotated as positives (containing a landfill) or negatives (no landfill), with the distribution shown in Table 3.1.
- For **multi-label classification**, some positive images have been further annotated to describe the type of object (15 categories) in the landfill and its storage mode (7


(a) AGEA (b) Google Earth (c) WorldView-3

Figure 3.2: Examples of AerialWaste images from different sources.

| Set | Positives | Negatives | Total |
|---|---|---|---|
| **Training** | 2793 | 5579 | 8372 |
| **Test** | 866 | 1739 | 2605 |
| **Total** | 3659 | 7318 | 10977 |

Table 3.1: Distribution of positives and negatives for binary classification in the AerialWaste dataset.

categories). All such categories, along with their distributions across training and test set, are displayed in Table 3.2. Visual examples can be found in Figure 3.3.

- For **Weakly-supervised localization** some images have been annotated with a segmentation mask surrounding the waste objects, as shown in Figure 3.3.

In this thesis, AW provides the background images for augmentation, which are also a necessary input for some phases of the instance placement process, such as Semantic Segmentation (SS) or Shadow Detection (SD).

### 3.1.2. 3D models

3D object models are paramount for the augmentation process, since they are the basic components of the structures to implant in the background images. When building such structures, the core idea is to leverage models of single objects instead of off-the-shelf models of the structures themselves. This choice was motivated by a pair of reasons: firstly, introducing in multiple images the same model of the entire structure would reduce variability in augmentation and, secondly, available models for these structures are usually obtained via 3D scans of real instances, thus introducing in the model also portions of the

| Name | Training | Test | Total |
|---|---|---|---|
| Rubble/excavated earth and rocks | 314 | 66 | 380 |
| Bulky items | 317 | 44 | 361 |
| Fire Wood | 163 | 38 | 201 |
| Scrap | 185 | 27 | 212 |
| Plastic | 133 | 24 | 157 |
| Vehicles | 44 | 26 | 70 |
| Tires | 39 | 13 | 52 |
| Domestic appliances | 21 | 5 | 26 |
| Paper | 28 | 5 | 33 |
| Sludge-Zootechnical waste-Manure | 19 | 4 | 23 |
| Foundry waste | 9 | 1 | 10 |
| Stone/marble processing waste | 15 | 1 | 16 |
| Asphalt milling | 9 | 3 | 12 |
| Corrugated sheets (presumed asbestos-cement) | 14 | 1 | 15 |
| Glass | 7 | 2 | 9 |
| Heaps not delimited | 491 | 93 | 584 |
| Full container | 115 | 54 | 169 |
| Big bags | 53 | 19 | 72 |
| Full pallets | 69 | 7 | 76 |
| Delimited heaps (by barriers/walls/etc) | 53 | 31 | 84 |
| Cisterns | 32 | 9 | 41 |
| Drums bins | 18 | 2 | 20 |
| **TOTAL** | 2148 | 475 | 2623 |

Table 3.2: Distribution of categories for multi-label classification in the AerialWaste dataset.

real scene or ground plane, which would eventually result in patches with a sharp border contrast with the background image. Therefore, structures are built by exploiting single objects and leveraging specific Blender features, as discussed in Section 3.3.

Nevertheless, concerning the adopted models, the designed method followed different approaches for the different waste types supported: for the cases of *Tires* and *Bulky items*, the models were downloaded from the Internet, whereas, for the case of *Rubble*, the singular shards were procedurally modelled. All the models from the Internet were freely downloaded from SketchFab [62] under the CC-BY license and manually adjusted

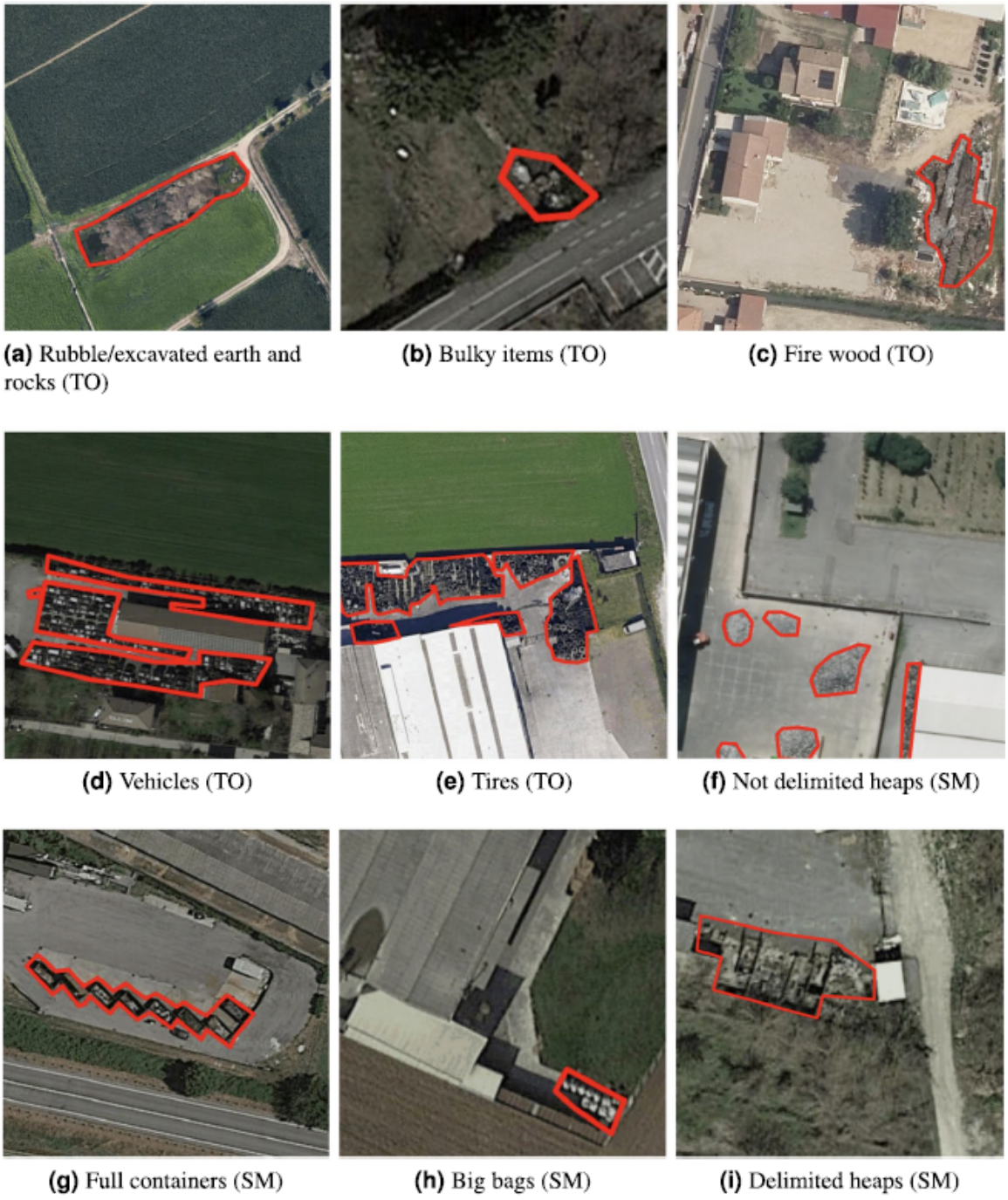



Figure 3.3: Examples of manual segmentations of waste instances in aerial images of the AW dataset. TO = Type of Object. SM = Storage Mode. From [12].

in Blender. These adjustments were mandatory, since the Blender scene is completely described with metric measures and most of the downloaded models were provided with excessive dimensions. Therefore, in order to obtain models with realistic sizes, these were applied some geometric transformations, such as scaling. As already said, instead, the

(a) *Tires* (b) *Bulky items* (c) *Rubble*

Figure 3.4: Blender *collections* for the different waste types.

atomic rubble shards were created via procedural modelling; the followed approach is described in Appendix C.

Finally, all the models belonging to a specific waste type were grouped in a Blender *collection*, thus allowing them to be used as a single input by the *Geometry nodes* described in Section 3.3.

### 3.1.3. DUSAF7.0

DUSAF7.0, acronym for Destination of Use of Soils for Agriculture and Forests, is the seventh version of a digital map project launched in 2000/2001 by Regione Lombardia, providing land cover information for the entire region updated to 2021. To realize this digital map, the annotators worked with AGEA orthophotos with a GSD of 20 cm/px.

The dataset is officially provided by Regione Lombardia through its online geographic portal [63] as a set of vectorial data (in the *shapefile* format), thus representing the region in terms of polygons described by geographic coordinates. Each of these polygons, which has, by design, a minimum perimeter of 20 meters and a minimum area of 1600 $m^2$, is assigned a label describing its land use. All the available land uses, listed in Appendix A, have been collected and organized in a 5-layer hierarchy extending the original 3 layers of the CORINE Land Cover [64] legenda to account for peculiarities of the territory of region Lombardy.

In this work, polygons could be extracted for each of the images in the AW dataset by leveraging the geographic information available solely within Politecnico di Milano. For each image and for each polygon, then, the geographic coordinates have been converted into pixel coordinates, in order to foster integration with the information extracted from other datasets. This integration will be described in detail in Section 3.4.

Beside this polygonal information, DUSAF7.0 describes also hedges and tree lines. However, the representation given to these entities is that of a simple line, thus considering

only their length and neglecting other relevant dimensions, such as their width. In addition, as revealed by Regione Lombardia in the official description to the DUSAF7.0 dataset, these entities have been annotated only when their length was greater than 5 meters, thus ignoring all other instances of hedges or tree lines. Given these limitations, this kind of information was considered unhelpful and, therefore, it has not been included in any step of the augmentation method.


Figure 3.5: Examples of images from the AW dataset with overlaid polygons from DUSAF7.0.

### 3.1.4. DTM5x5

The DTM5x5, where DTM stands for Digital Terrain Model, is a dataset officially released by Regione Lombardia through its online geographic portal [63] and describing the distribution of altitudes throughout the Lombardy region. This dataset provides raster data, i.e., a grid of data where each point is associated a value. On this grid, each point is 5 meter distant from its 4 direct neighbors, thence the "5x5" in the dataset name. The altitude value refers to the terrain level for both urban and extra-urban locations, whereas it refers to the water level for geographic points on lakes or water reservoirs.

According to the official description provided on the online geographic portal, this dataset is mainly obtained by combining data from a topographic database with those from Lidar campaigns, filling regions without data with information from a previous DTM with a resolution of 20 meters. The topographic database was divided in lots with multiple scales and resolution, with a higher scale implying a higher altitude accuracy. The combination of data was then harmonized according to level curves, thus creating a vectorial terrain model.

Despite being realized with the most updated data in 2015, to the best of our knowledge, this is still the best dataset conforming to our needs. Since images in the AW dataset

refer to locations in different provinces of the Lombardy region, it was necessary to find a DTM or DEM (Digital Elevation Model) covering the whole region. However, all the datasets complying to this requirement had a lower resolution, usually $\approx 30$ meters. On the other hand, Regione Lombardia provides also a continuous value map for DTM data, which however covers only the Sondrio province.

### 3.1.5. OpenEarthMap

OpenEarthMap (OEM) [65] is a dataset consisting of 5000 images suited for the task of land cover mapping, a SS task for RS images. The dataset images were extracted from several different sources and represent areas from all over the world, with the aim of composing a dataset with a high geographic variability and incline to increased generalization towards unseen regions. For these reasons, the authors introduced in the dataset also images openly available, thus providing samples also for those regions which had been neglected in most-used datasets.

Given the different provenance of the images, these usually have a different size and a different GSD even though, for the latter aspect, all the sources provide images in the range of 25-50 cm/px. In addition, since the dataset is intended for semantic segmentation, it also contains a set of labels, which classify each pixel into 8 possible classes. These are:

- ***Bareland***: a land without vegetation but dominated by rocks, sands or other earthen materials;
- ***Rangeland***: any land covered by shrubs or other non-cultivated vegetation;
- ***Developed space***: an area covered with artificial materials such as asphalt, concrete, bricks or tiles;
- ***Road***: any area on which a vehicle might move, such as streets, airport lanes or bicycle lanes;

Figure 3.6: Examples of images and labels from the OEM dataset.

- ***Tree***: single instances and groups of trees;
- ***Water***: any water body, such as rivers, lakes and swimming pools;
- ***Agriculture land***: crops or pastures;
- ***Building***: residential, commercial and industrial buildings.

The dataset labels were obtained after a meticulous annotation process involving 16 people, with 8 people directly performing annotation while the other 8 were employed only for double-checking. The result is a carefully annotated dataset with a significant degree of spatial detail. Furthermore, the dataset has already been split in training (3000 images), validation (500 images) and test set (1500 images), though labels have been published only for the first two portions. To verify training results on the test set, it is necessary to upload an archive containing the output predictions on the related CodaLab page [66].

For this thesis, OEM was used to train a network for SS, which contributes to the process of instance placement.

## 3.2. Scene setup and rendering

This section provides an overall description of the designed method, which consists of a sequence of phases (Figure 3.7) to render a Blender virtual scene where instances of waste structures are created and positioned on top of a background image. In this section, the focus is on the generic scene construction and on the final rendering phase; the *Instance placement* and *Instance creation* parts are carefully described in the following sections. As a note for the sake of precision, in order to totally comply with the reference system adopted by Blender, all the following descriptions will assume a right-handed $z$-up reference system.

The scene setup starts with the creation of a *Plane* mesh, which visually is a simple rectangle with only 4 vertices, one for each corner. Since this mesh will act as background

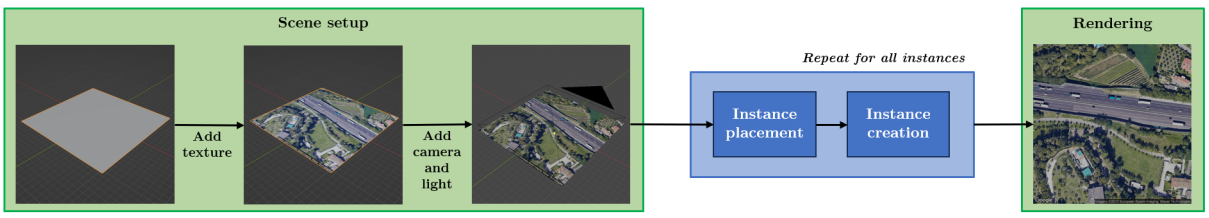



Figure 3.7: The pipeline and its phases, with focus on *Scene setup* and *Rendering*.

for all the synthetic waste instances, it can be positioned to lie on the $xy$ plane, orthogonal to the $z$ axis. Given that all images in the AW dataset [12] cover a squared area with a side of 210 meters, the introduced plane is created with such a size. Subsequently, to complete the background construction, the plane is applied as texture the chosen aerial image.

At this point, a camera is introduced, since it will be mandatory for rendering the scene. In order to center the camera projection plane with the center of the background image, the camera is placed on the $z$ axis, in the positive region and pointing downwards, sufficiently far from the plane to never intersect any eventual waste instance. Furthermore, given that the background image is originally from a satellite or an aerial vehicle, the camera is chosen to be orthographic, i.e., a camera that renders images according to an orthographic projection, thus simulating well the behavior of an observer significantly far from its target. Finally, to ensure the rendering will cover the whole image and nothing more, the field of view of the camera is set to be square and to return an image with the same size as the original one.

In order to guarantee realism in the scene and, above all, to allow the instances in it to cast shadows, a light source must be added. In this case, considering that the content of the background image is always an open-air scene, it would be useful to introduce a light source to emulate the Sun. For this reason, a directional light is introduced, i.e., a light whose beams hit the surface in all points with the same direction, a good choice to emulate the effect of a real source significantly far from its target. Since most of the images in AW have shadows directed North-West, the light source is rotated and translated to be positioned in the fourth quadrant of the $xy$ plane, with a positive $z$ coordinate. Furthermore, in order to improve realism and avoid sharp shadows on the background, a white ambient light is added as well.

Once the scene is set up, the process can proceed with the determination of the location for instance placement along with the dimensions and rotation of the instance, as described in Section 3.4. After defining these parameters, the instance is created with the method thoroughly described in Section 3.3.

Once all instances have been created and placed, the scene is rendered from the perspective of the orthographic camera introduced before (*Rendering* phase).

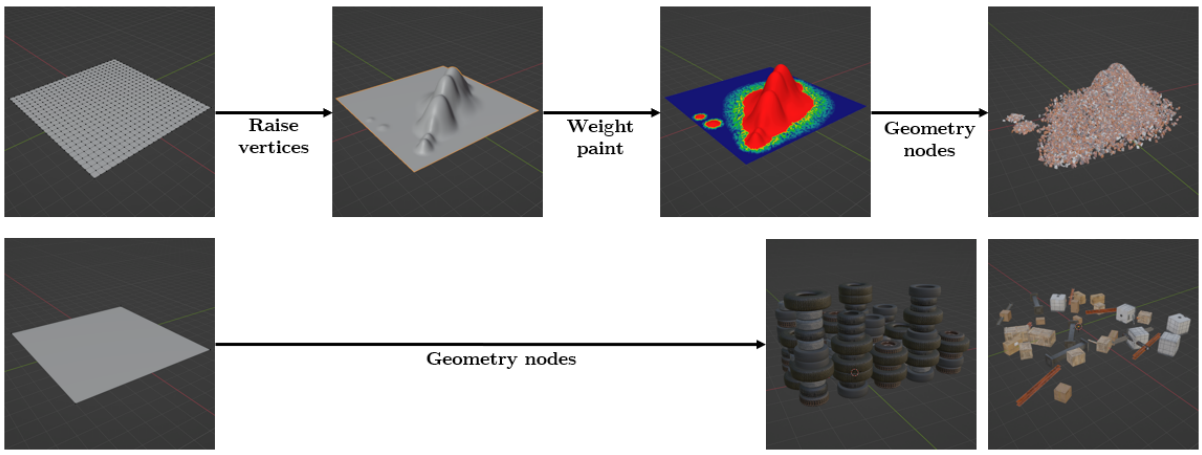



Figure 3.8: Representation of the *Instance creation* phase. Top line: the procedure to create *Heaps*. Bottom line: the procedure for *Stacks* and *Scattered objects*.

## 3.3. Instance creation

The designed method allows to arrange objects in 3 different structures, *Heaps*, *Stacks* and *Scattered objects*, for which examples are provided in Figure 3.1. In order to create such structures, the method mostly leverages *Geometry nodes*, a Blender feature that allows the definition of specific configurations to procedurally edit an input geometry through a wide range of transformations. However, even though the application of these configurations as mesh modifiers constitutes the main step for obtaining a visually intelligible structure, this is not the only step needed for creating such structure, as shown in Figure 3.8.

This section aims at providing a detailed description of all such steps without specific attention to the actual node configurations. A thorough description of these configurations and of their components can be found in Appendix B.

### 3.3.1. *Stacks* and *Scattered objects*

The simplest structures to build are *Stacks* and *Scattered objects* which only require a sequence of basic phases and differ solely for the specific node configuration to apply.

To begin with, it is mandatory to introduce in the virtual scene a base mesh, which is a strict requirement for the application of any modifier. In this case, the chosen mesh is the *Plane* which visually is a simple rectangle with only 4 vertices, one for each corner. Since this *Plane* will act as ground surface for the object structures, it can be positioned to lie on the $xy$ plane, orthogonal to the $z$ axis. When inserting the *Plane*, it is also possible to

set its size, which is a key factor for the case of *Scattered objects*.

After importing in the scene the 3D object models to compose the structures, the node configuration can be created and applied with a modifier to the *Plane* formerly introduced. The chosen configuration will receive as input the needed objects and some parameters to adjust its appearance. Among these parameters, the case of *Stacks* reckons the number of objects to stack up as well as the number of rows and columns of stacks, whereas the case of *Scattered objects* has only the density of the objects to distribute on the surface.

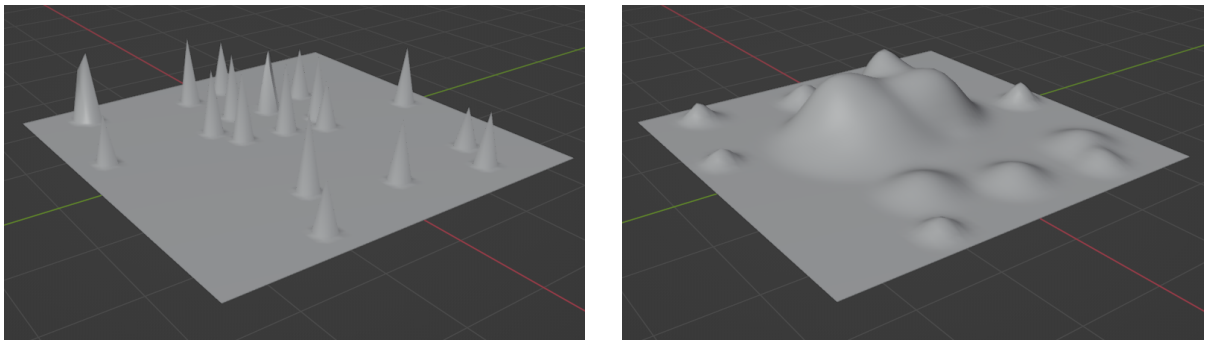
### 3.3.2. *Heaps*

Unlike the procedure for the structures presented in previous section, building *Heaps* is a rather complex process because its core idea is to distribute objects on a mesh presenting hill-like protrusions. However, Blender does not natively provide a mesh with such features and, therefore, this must be modelled with an ad-hoc procedure.

The Blender mesh which best fits for the required adjustments is the *Grid*, which visually is a rectangle, similarly to *Plane* used in the previous cases, but with vertices also inside the rectangle, not only on the 4 corners. As the mesh name suggests, inner vertices are distributed to form a grid, whose number of rows and columns can be defined upon creation together with the rectangle size. Similarly to the previous case, since the rectangle will be needed as a surface to lie covering objects on, it can be introduced on the $xy$ plane, orthogonal to the $z$ axis.

The required protrusions can be obtained by randomly selecting some of the *Grid* inner vertices and by positively changing their $z$ coordinate, thus moving such vertices higher than all others. However, following this editing procedure without additional adjustments would result in creating pin-like protrusions in the *Grid* (Figure 3.9a), which would be sensitively far from the expected results. To solve this issue, it is possible to exploit another Blender feature, called *Proportional edit*. By enabling this feature, when changing the height of a vertex, its neighbors, within a certain custom radius, are lifted as well by a value proportional to their distance from the selected vertex. In this way, depending on the chosen radius, it is possible to obtain protrusions with a variable degree of smoothness (Figure 3.9b).

Nevertheless, applying the *Geometry nodes* at this point would result in distributing objects all over the mesh, instead of covering only the created protrusions. Therefore, rendering such instance with an overhead camera would imply rendering a rectangle covered with the selected objects. To avoid this situation, it is necessary to constrain object placement only on the elevated portions of the base grid. This is achievable with the



(a) Without *Proportional edit* (b) With *Proportional edit*

Figure 3.9: Different approaches for lifting vertices

support a *Weight paint*, a Blender feature which allows to associate a value in the [0,1] range to each vertex of a specific mesh. Given the procedure followed to create the protrusions, it is easy to notice that vertices belonging to such protrusions are those with a $z$ coordinate greater than 0. Therefore, in the *Weight paint*, such vertices can be assigned value 1, whereas all the others can be given value 0, thus creating a sharp distinction between the vertices belonging to the *Heap* and those that do not belong.

From this point on, the process proceeds as in the previous case, with the import of the covering models and the application of the geometry modifier. Similarly to the other cases, the node configuration has an adjustment parameter, which is the density of the covering objects. This value is affected by the *Weight paint* introduced before: by multiplying the weight for the given density value, each vertex is assigned a density which is either the parameter value or 0, with the parameter value being applied only to the points on the protrusions. Therefore, points outside of the *Heap* are assigned density 0 and the mesh is not covered by objects in that part.

Finally, it is worth mentioning that the process described in this section is just a vanilla version of the actual implemented process. Indeed, in order to improve realism of the output instances, several small adjustments were added to the method: for instance, the *Weight paint* was extended with a halo effect at the heap basis in order to avoid a sharp contrast between the instance border and the background image.

## 3.4. Instance placement

One of the main objectives for the whole augmentation process is to create images with the highest degree of realism. Given this aim, a crucial phase to consider is *Instance placement*, i.e., that phase in which various decisions are made to define in which posi-

tion to implant a synthetic instance. In this context, the simplest approaches are often either impractical or unfruitful. For example, a potential solution could be to manually select the implantation location, resulting however in requiring human intervention in a process which aims at being completely automatic. Otherwise, it is possible to distribute instances in completely random locations, thus avoiding the need for human intervention but creating a process incline to place instances in totally unfeasible locations, such as on water in lakes or reservoirs. To avoid this issue, it is possible to manually annotate all the images with the legal placement areas; however, this method would result in being significantly time-consuming, though potentially incredibly accurate. As a consequence, it is necessary to revert to other approaches which can be totally automated, even though potentially more complex or less accurate.

The procedure proposed in this work is composed of 2 parts, one offline and one that takes place during the augmentation process. In the offline part, several sources are combined to define all the image areas where it would be legal to place a waste instance. Then, during augmentation, a rectangular portion is identified within such legal areas to provide dimensions and rotation for the base mesh for creating an instance as described in Section 3.3.

This section describes in detail both these parts. In the first one, each source contributes by defining some legal and illegal areas, handled as polygons similar to those obtained from the DUSAF7.0 dataset, described in Section 3.1.3. Then, the legal polygons from each source are intersected with those from all other sources, thus defining areas that in all sources are considered legal. In order to foster visualization, this process is described in this section in terms of binary masks, where legal areas are associated to color white, whereas illegal areas are black. A summary of this phase, including the mask combination, is provided in Figure 3.10.

### 3.4.1. Land cover

When defining the location where to place a waste instance in an image, it is important to consider where, in real life, such instance could be located. In this context, the positive samples in the AW dataset can be a paramount reference, since they show real examples of landfills, which might guide the placement of synthetic ones. However, these images provide per se no information about the represented context, thus complicating the extraction of placement guidelines. This information can be introduced with the DUSAF7.0 dataset, described in Section 3.1.3, and leveraged to study which land cover uses are most frequent in positive images.

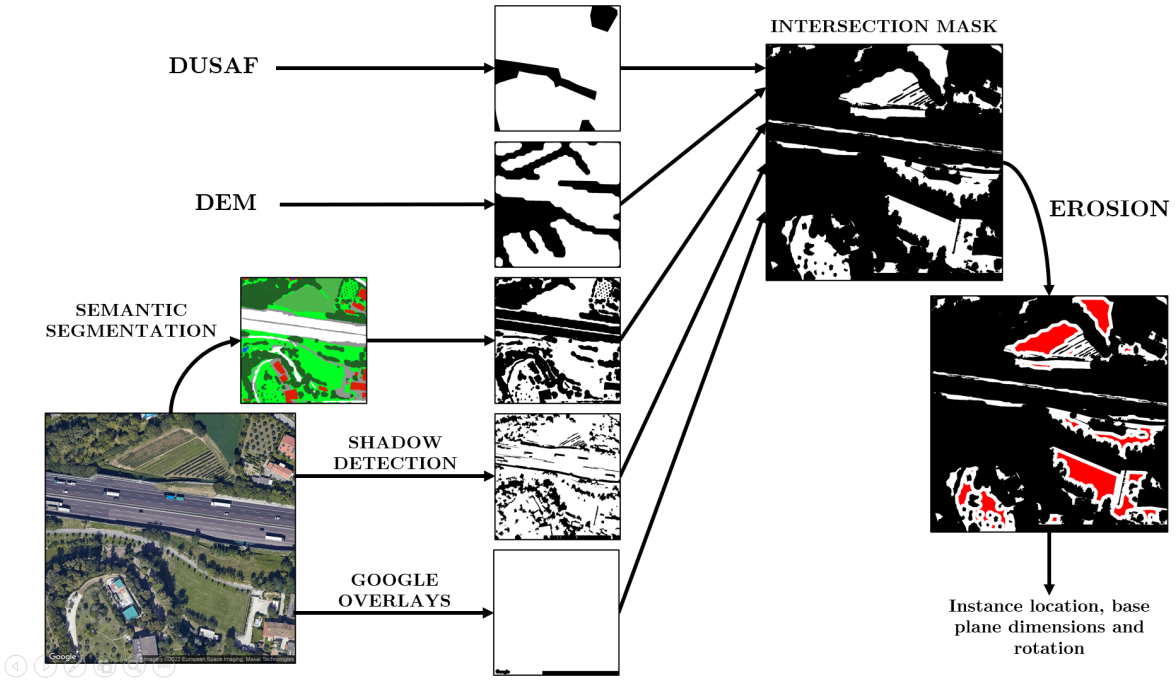



Figure 3.10: The *Instance placement* phase: input mask combination and effects of erosion on legal regions. The center of the output rectangle is found in the red areas.

Within this study, for each land cover category, we counted the number of positive images containing at least one polygon labelled with that category. Then, we performed the same count for the negative images and converted these absolute values into percentages, in order to compute a ratio between percentages for the positive and for the negative case. The results of these computations, which are visualized in the histograms in Figure 3.11, were analysed to determine which categories are most likely to host a waste instance. An interesting observation concerns category 2111, *Simple arable land*, which appears in a very high number of positive instances. However, this category is also present in a significant number of negatives and the percentage ratio is approximately 1. This result suggests that such category is simply very frequent within the dataset, thus preventing it from further analysis about its likelihood to contain waste instances.

Given this difficulty encountered in determining whether a category is likely to contain positives, the focus shifted towards the identification of categories which are most likely not to. These categories were assumed to be those that appear in a restricted number of positives and were specifically analysed via visual inspection with the aim of identifying some removable categories. Within this process, we selected all the categories appearing in less than 100 images and observed them in search of waste instances. If a category contained no waste instance or a very small percentage of them, it was deemed discardable. A list of discardable categories is provided in Table 3.3.

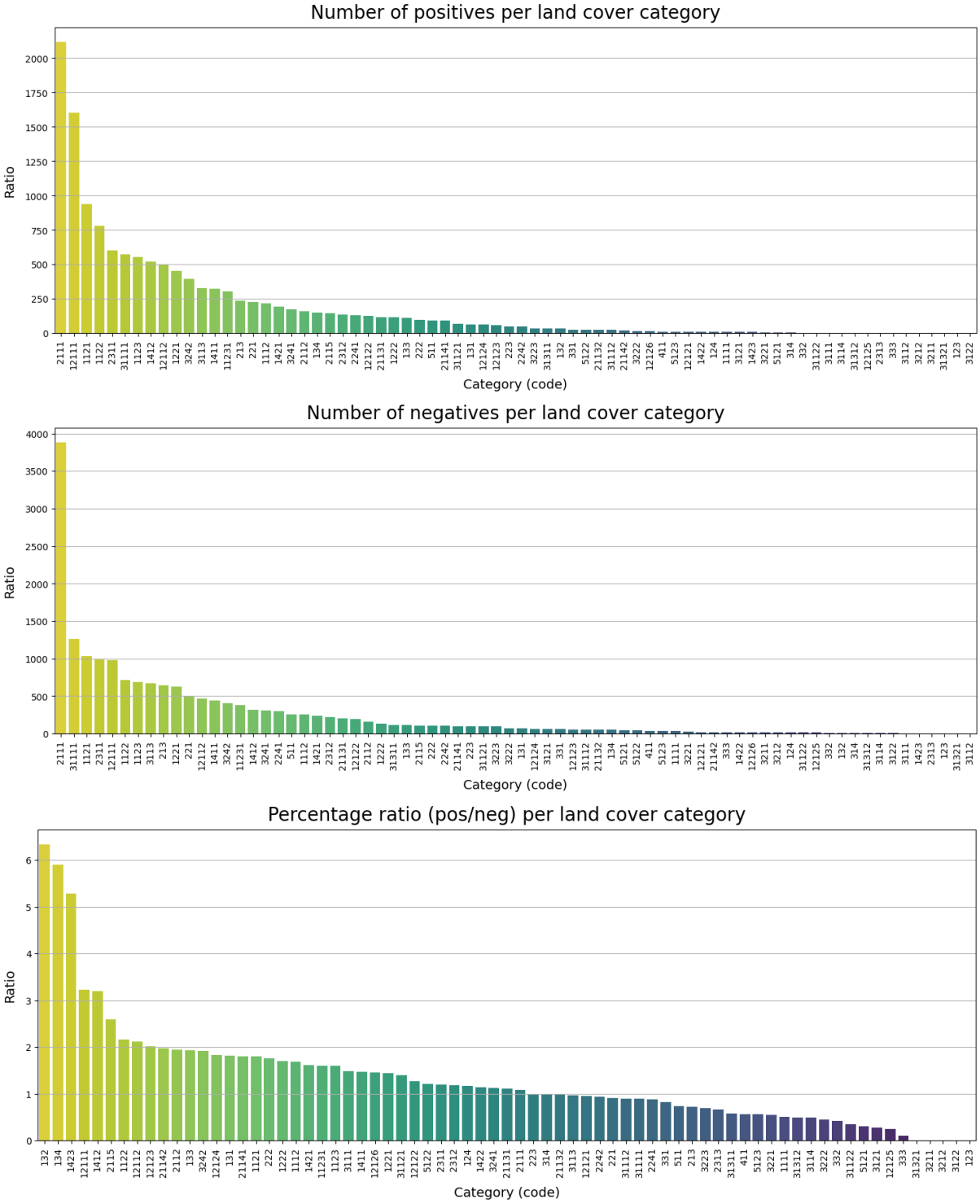



Figure 3.11: Land cover histograms. Category 3112 was excluded from percentage ratio because the ratio would have been infinite.

| ID | Name | Occurrences |
|---|---|---|
| **222** | Fruit trees and berry plantations | 96 |
| **511** | Water courses | 94 |
| **12123** | Technological units | 58 |
| **223** | Olive groves | 50 |
| **2242** | Other wood cultures | 49 |
| **31311** | High-density coppice mixed forest | 33 |
| **3223** | Bank vegetation | 33 |
| **5122** | Artificial water bodies | 27 |
| **331** | Beaches, dunes, sands | 27 |
| **31112** | High-density tall-tree broad-leaved forest | 25 |
| **3222** | Riverbed vegetation | 17 |
| **12121** | Hospital units | 11 |
| **411** | Inland marshes | 11 |
| **5123** | Water bodies from extractive aquifer activities | 11 |
| **3121** | Mid-high-density coniferous forest | 9 |
| **5121** | Natural water bodies | 7 |
| **314** | Recent reforestations | 4 |
| **31122** | Low-density tall-tree broad-leaved forest | 3 |
| **332** | Bare rocks | 3 |
| **3111** | High-density broad-leaved forest | 3 |
| **3114** | Chestnut groves | 2 |
| **31312** | High-density tall-tree mixed forest | 2 |
| **3112** | Low-density broad-leaved forest | 1 |
| **333** | Sparsely vegetated areas | 1 |
| **2313** | Water meadows | 1 |
| **3122** | Low-density coniferous forest | 0 |
| **31321** | Low-density coppice mixed forest | 0 |
| **3211** | Natural grasslands without arboreal or shrub-like species | 0 |
| **3212** | Natural grasslands with arboreal or shrub-like species | 0 |
| **123** | Port areas | 0 |

Table 3.3: Removable land cover categories with the number of positive images containing them (occurrences).

In summary, in the designed augmentation method, the land cover information has been extracted from the DUSAF7.0 dataset and adopted to determine which land use categories are highly unlikely to host a waste instance. These categories were deemed illegal for placement and, therefore, added to the black portion of the related mask, as shown in Figure 3.12.

(a) Image form AW dataset (b) Polygons from DUSAF7.0 (c) Land cover mask

Figure 3.12: Example for land cover.

### 3.4.2. DTM

Following the same guiding idea presented in Section 3.4.1, i.e., to aim at placing instances in location where they could realistically be, another situation to address is that of steep slopes, such as mountain sides. In these contexts, indeed, it is highly unlikely for landfills to be located, since the terrain conformation might lead waste to fall from its original place. However, since the AerialWaste dataset does not provide information about slopes or a terrain model, this must be introduced from an external source, such as the DTM5x5, presented in Section 3.1.4. Information from this dataset, that describes the altitude distribution over a region of interest, can be used to compute slope distribution over the same region. As a side effect, the introduction of such information also allows to exclude placement of instances across areas separated by a harsh discontinuity.

Therefore, slope information is used to create a specific mask to distinguish between high-slope illegal areas and low-slope legal ones. However, in order to obtain such information, it is necessary to extract it from the altitude levels presented in the reference raster data. For each point in the raster, slope can be computed as the ratio between the altitude difference and the distance with a neighboring point. In this case, we considered the slope of a point to be the highest value returned from this computation involving the point and each of its 8 surrounding neighbors. Then, the resulting value is converted into angular information, which is used to define illegal areas via the application of a threshold. In this case, such threshold was set to 20°, thus considering illegal areas with a higher slope.

Given the grid nature of the input dataset, masks obtained from this process usually result in displaying jagged borders between white and black regions. To avoid this unreal-

istic situation, we added a smoothing mechanism consisting of multiple subsequent erosion and expansion transformations to remove sharp corners from the represented polygons. The outputs of this process are exemplified in Figure 3.13.


(a) Image from AW dataset (b) Slope image (c) DTM mask

Figure 3.13: Example for DTM

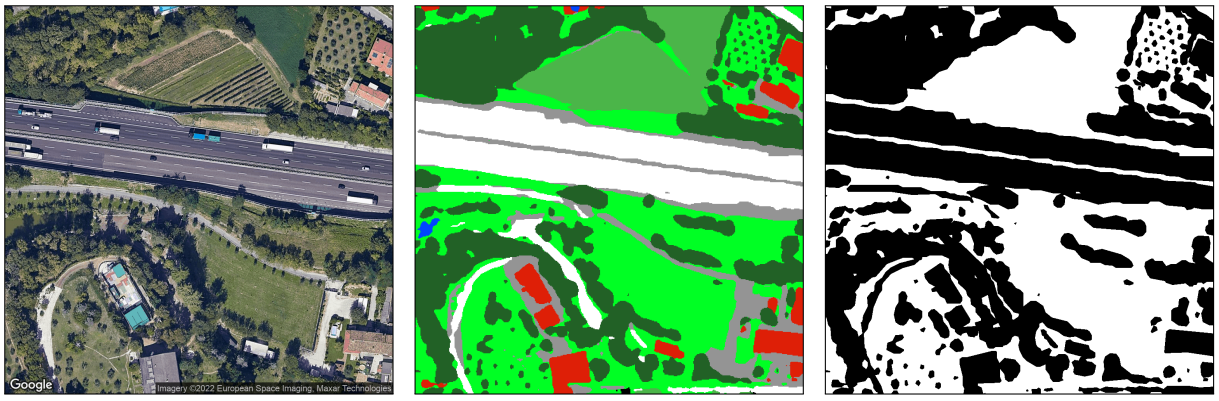
### 3.4.3. Semantic Segmentation

Some portions of the image where placement is unfeasible cannot be excluded only with the masks presented in the previous Sections sections 3.4.1 and 3.4.2. In this context, it is intuitively unlikely, if not impossible, to find instances over trees or on top of roofs, thus requiring the creation of a specific mask to consider such areas illegal. In order to obtain such mask, it is necessary to semantically interpret the image, practice which is nowadays feasible thanks to the modern DL methods, such as CNN. Therefore, we decided to train a network for Land Cover Mapping on the OEM dataset, and to use such network for segmenting images in the AerialWaste dataset.

By performing inference on such dataset, it is possible to obtain, for each image, segmentation labels where each pixel is assigned either to one of the 8 classes from the OEM dataset or to an additional background class, generally used for locations beyond the borders of sensed images. The final mask was created by considering illegal, beside the background, 4 of the 8 OEM categories: *Building*, *Tree*, *Water* and *Road*. Outputs of this process are exemplified in Figure 3.14.

### 3.4.4. Shadow Detection

In the scene described in Section 3.2, the implanted waste structures are the only objects casting shadows. Therefore, it is not possible for such implanted structures to be covered




(a) Image from AW dataset (b) SS output (c) SS mask

Figure 3.14: Example for Semantic Segmentation.

themselves, not even partially, by shadows. As a consequence, the artificial waste instances are always uniformly invested by light and placing them across light and shadow areas would result in illumination inconsistency. For this reason, the mask for shadow detection is introduced with the intent of defining shadow areas as illegal, thus allowing instance placement only in lit areas and avoiding any illumination artifact.

To detect shadows, we adopted an adjusted version of the automatic method expressed in [67] and centered around the computation of the *SSSI* index. In its original version, this method receives as input only an RGB image and starts with a pre-processing step motivated by a spectral analysis over the 3 channels. In this step, all the channels are applied a threshold both on the low end and on the high end, with the aim of classifying the smallest values as shadow and the highest as light. After computing the thresholds, the image pixels are normalized between such thresholds, thus creating a normalized image. The *SSSI* index can now be computed for each pixel with the following formula:

$$SSSI = \frac{PC1 + B + SENT}{R + G + 1} \tag{3.1}$$

where:

- **PC1** is the normalized image projection on the first component from *PCA*;
- **R**, **G** and **B** are the 3 channels from the original image;
- **SENT** is GLCM feature [68] computable from the greyscale version of the image.

The computed values consent to build an image where each pixel is associated its *SSSI*. In this image, the pixel values are then averaged on a cluster basis, with clusters computed

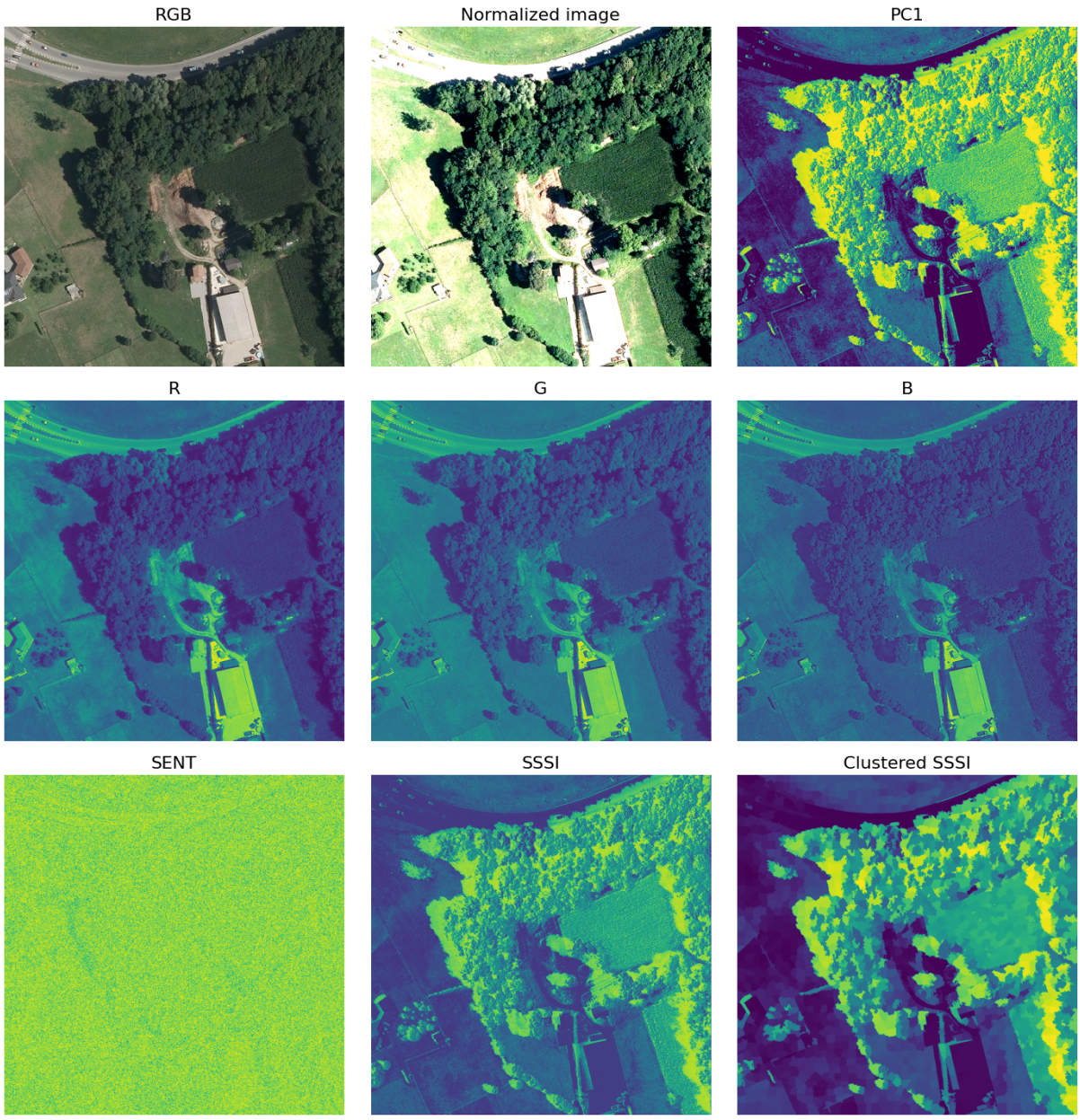
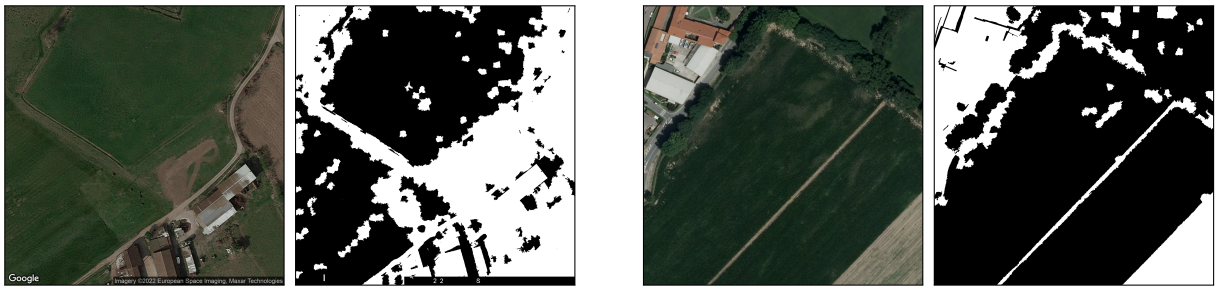



Figure 3.15: Components and outputs of the *SSSI* computation on an AW image. No adjustment applied to index computation.

following the *SNIC* algorithm [69]. At this point, a thresholding mechanism should be applied on the output averaged image with a dual approach, depending on the modality of the histogram of pixel values in the *SSSI* image. If the image is multimodal, then Otsu thresholding should be applied, otherwise, if the image is unimodal, another clustering-based method is proposed. After thresholding, pixels with values above the threshold should be considered in the shadow, whereas other pixels should be in the light.

In our adjusted version, the early stages have not been altered whereas some changes



(a) Unimodal (b) Bimodal

Figure 3.16: Thresholding flaws following the original approach from [67].

have been applied to the index computation and to the thresholding mechanism. With respect to thresholding, after visually inspecting the results, we noticed that the proposed approach was not effective, detecting in images a significantly higher quantity of shadow than what actually present (Figure 3.16). Therefore, we replaced the proposed approach with a relative threshold: this level would indicate a percentage in the range of *SSSI*s, allowing to classify as shadow pixels with an index above the threshold. Additional details about the choice of the threshold can be found in Section 4.1.2.

After updating the thresholding mechanism, we also adjusted the formula for computing the *SSSI* index. We noticed that the computation of the *SENT* factor was the most computationally expensive, without being accountable for significant changes in the index (Figure 3.17). For this reason, we decided to simplify the index formula into:

$$SSSI = \frac{PC1 + B}{R + G + 1} \tag{3.2}$$

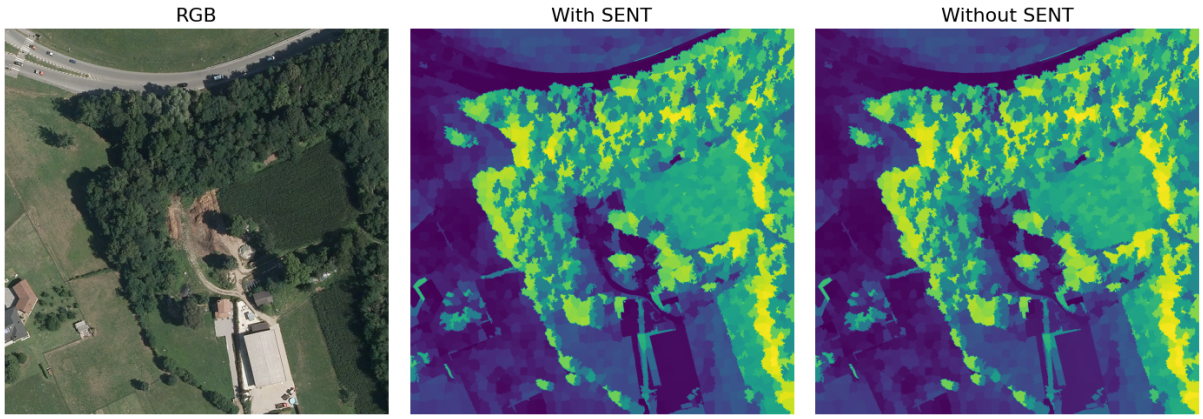



Figure 3.17: Comparison of *SSSI* image before thresholding after index computation with and without *SENT*.

thus significantly accelerating the shadow detection process. Figure 3.18 exemplifies outputs from this process.

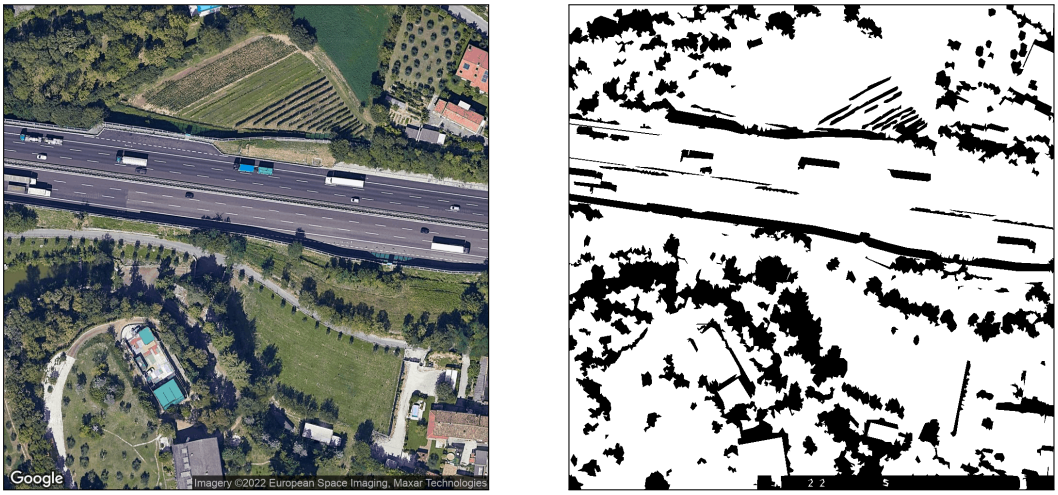



Figure 3.18: Exemplary image from the AW dataset and its related mask for Shadow Detection.

### 3.4.5. Google overlays

As described in Section 3.1.1, some images of the AW dataset have been obtained by means of a download from the GE platform. As a consequence, these images are all watermarked with 2 overlays: the Google logo in the bottom left corner and a box with copyright information in the bottom right one. These 2 overlays clearly represent regions of the image where it would be illegal to place an instance and, therefore, they should be excluded from placement by means of a specific mask.

The Google logo in the bottom left corner is always located in the same position and always has the same size across all the GE images in the AW dataset. Therefore, it was possible to manually create a polygon covering it and consider such area illegal.

Unlike the Google logo, the copyright box can have different sizes across different images. With the aim of considering illegal only the area covered by the box, it became necessary to identify the exact position of the top right corner of such box. This information allows to create a rectangle covering the whole box, since its opposite corner is always the bottom right corner of the whole image. After visually inspecting some GE images from the AW dataset, we noticed that the top left corner of the box could be found via template matching, leveraging the observation that the copyright writing in the box always starts with the word "Imagery". Therefore, we extracted from an image the portion with such word and adopted it as template. Then, for each image, we performed matching on the


Figure 3.19: The "Imagery" filter used for template matching.

bottom 50 lines of pixels with the aforementioned filter and extracted the coordinates of the pixel with the highest correlation coefficient. With this method, only 5 points were found over more than 5,300 images: (47, 13), (275, 24), (391, 24), (532, 24) and (648, 24). We observed examples (Figure 3.20) of images for each of these points and noticed that the last 4 points actually corresponded to sensor information, whereas (47, 13) was the result in those images that do not contain a copyright box. Finally, the top-left corner of the rectangle was obtained with small manual adjustments, thus allowing the creation of such polygon and to considered it illegal.


Figure 3.20: Examples of Google overlays. Red cross indicates the output position from template matching. Cyan polygons highlight the final overlays to remove.


Figure 3.21: Exemplary image from the AW dataset and its related mask for Google overlays.

### 3.4.6. Online phase

All the masks described in the previous sections are created and combined offline, thus requiring, during augmentation, to use only information from the final overall mask. Within the legal regions of this mask, it is necessary to find a rectangular area aimed at hosting the base *Grid* or *Plane* meshes of the synthetic waste structures.

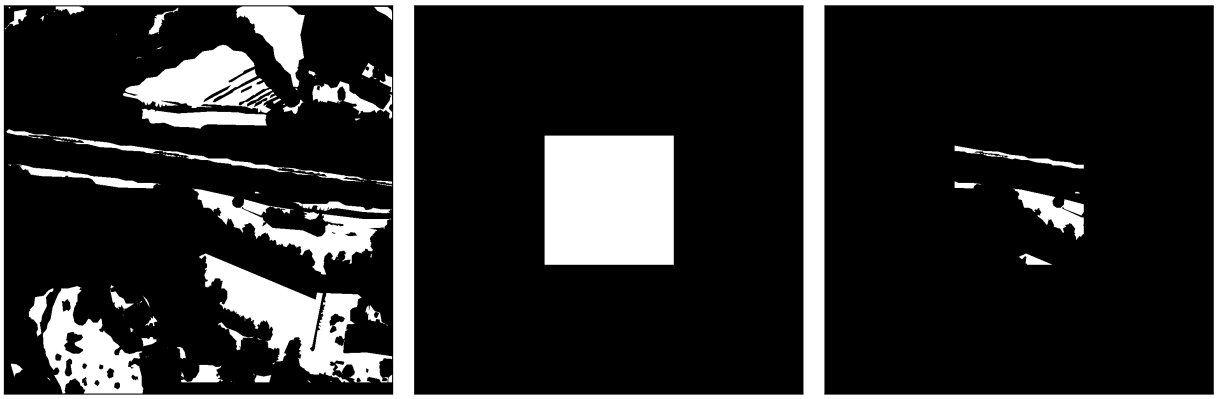
In order to allow eventual instances to entirely fit inside the legal regions, an erosion pre-processing step is introduced. Legal regions are eroded by the length value describing an instance minimum size, value which should be passed as parameter. This transformation ensures that any point selected inside the eroded region satisfies the condition of minimum distance from the border and, therefore, is a valid candidate to act as center of the searched-for rectangle. Among these candidates, one is randomly picked and we compute its actual distance from the border of the legal area before erosion. This distance defines the maximum value for the synthetic instance size. Then, a circle is created centered in the point found before and with a random radius between the minimum and maximum sizes defined above. The final rectangle is found inscribed in this circle.

Once the needed rectangle has been found, the augmentation process can proceed with the creation of the waste instance, described in Section 3.3.

# 4 | Experiments

This section describes the various experiments conducted within the work of this thesis, with a dedicated section for each of them. In these experiments, which all required the generation of synthetic images, waste instances were constrained by some general restrictions. Firstly, in order to ensure realism in the created instances, only a subset of all possible combinations of structure and waste types was admitted; therefore, in augmented images, *Tires* can be present in *Stacks* or *Heaps*, whereas *Rubble* can only appear in *Heaps* and *Bulky items* can only be *Scattered*. Secondly, the background aerial images were always chosen among the negatives of the AW training set.

Beside these general constraints, which were common to all experiments, a few other limitations, concerning dimensions or position, were applied to instances in a fraction of attempts. In terms of dimensions, instances could be divided into *standard-sized* and *greater-sized*, with the latter being generated in the same way as the first ones, but adopting a higher parameter for their minimum size. In terms of location, instead, instances could be *centered* or not; in the *non-centered* case, the overall mask for instance placement was used as is, whereas, in the *centered* case, it was combined with an additional total black mask with a white rectangle in the center (Figure 4.1), thus forcing instances to be placed in a central portion of the image.



(a) Initial overall mask (b) Center-constraining mask (c) Final overall mask

Figure 4.1: Mask combination for *centered* instances.

## 4.1. Computation environment

All the experiments presented in this chapter were conducted on a server at Politecnico di Milano equipped with 2 NVIDIA GeForce RTX 2080 Ti GPUs. For these experiments, all the synthetic images were rendered using only one of those GPUs and adopting the Cycles rendering engine, natively incorporated in Blender.

### 4.1.1. Semantic Segmentation network

In order to obtain the SS masks, a specific network was trained on the OpenEarthMap dataset [65], described in Section 3.1.5. On this dataset, we trained an FT-UNetFormer [70], a UNet-like network in which both the encoding and decoding parts are transformers. This choice was motivated by the results presented in the OEM reference paper, where the authors claimed such architecture to show the best performances. Therefore, with the aim obtaining similar results, we also tried to replicate the training environment by selecting the same number of epochs, 200, the same loss function, the multi-class cross-entropy, and the same optimizer, AdamW, initialized with the default PyTorch hyper-parameters apart from learning rate and weight decay. These latter values were defined via tuning, process which identified values significantly similar to those presented in the OEM paper. Therefore, we set the learning rate to $6 \times 10^{-5}$ and the optimizer weight decay to 0.01. Finally, the network backbone was initialized with the pre-train weights officially provided by the network authors and the batch size was chosen to be 8 in accordance with the OEM paper, even though memory limitations on our devices required us to train the model with both GPUs, instead of using a single one.

Given the eventual aim of our training, i.e., to obtain a model for inference on AW images, we introduced an augmentation phase to account for potential differences in image resolution. Since the OEM dataset contains images with a GSD in the range of 0.25-0.5 m/px, whereas AW images are in the range 0.2-0.3 m/px, we trained the network on a $512 \times 512$ crop of the original images, to allow AW ones to result in having a similar resolution after simple resize.

Finally, since the labels for the OEM test set are not released to the public, inference labels on test images were submitted to the official CodaLab competition [66], achieving the results presented in Table 4.1.

| Category | IoU |
|---|---|
| *Bareland* | 45.07% |
| *Rangeland* | 59.27% |
| *Developed space* | 55.89% |
| *Road* | 64.90% |
| *Tree* | 73.30% |
| *Water* | 86.58% |
| *Agriculture land* | 74.80% |
| *Building* | 80.70% |
| *mIoU* | 67.56% |

Table 4.1: Intersection over Union (IoU) metric for our model on the OEM test set.

### Qualitative evaluation on AW images

The AW dataset does not provide land cover mapping labels for its images. Therefore, considered the temporal cost of the annotation practice, which is allegedly 2.5 hours per image for the OEM dataset [65], it is unfeasible to quantitatively evaluate AW images, thus allowing only qualitative evaluation.

For this reason, Figure 4.2 shows some examples of SS labels obtained from inference on images from the AW dataset. Figures 4.2a and 4.2b display very accurate segmentation labels, mostly for trees, buildings, roads and water elements, in urban environments. A similar accuracy is achieved in Figures 4.2c and 4.2d, with good annotations in a forest context and a in rural setting. Instead, Figures 4.2e and 4.2f show two major limitations of our model: in the first figure, buildings in an industrial context are not segmented with the same accuracy as in the aforementioned ones, whereas in the latter figure, a portion of an agricultural field is misinterpreted as water. Nevertheless, despite these inaccuracies, a wide portion of the legal areas is still correctly segmented, thus allowing instances to be eventually placed in such regions. Finally, it can be noticed that the Google overlays are usually interpreted differently and rather randomly across the various images, as in Figures 4.2b, 4.2c, 4.2e and 4.2f.

## 4.1.2. Shadow detection threshold

Thresholding is a crucial phase in the creation of a SD mask. As described in Section 3.4.4, the original approach to this phase, proposed in [67], was replaced by one

(a) An urban context (b) Another urban context

(c) Forest environment (d) Rural context

(e) Industrial context (f) A field mistaken for water

Figure 4.2: Examples of good (4.2a to 4.2d) and bad (4.2e and 4.2f) segmentation labels on images from the AW dataset.

based on a relative threshold on the values of the *SSSI* index, thus introducing the need to define a unique threshold level. Similarly to the case of SS described in Section 4.1.1, the AW dataset does not provide shadow segmentation labels for its images, thus preventing the threshold definition via quantitative approaches. Therefore, it became necessary to determine the sought-for level via visual inspection.

Figure 4.3 shows SD masks obtained at different threshold levels, highlighting the issue of precision-recall trade-off in the choice of such level. Indeed, lowering the threshold percentage implies the correct classification of more shadow pixels, thus increasing recall. However, this process leads to considering as shadows also a high number of pixels which are in the light, thus lowering precision. Conversely, raising the threshold implies increasing precision, with almost all pixels classified as shadow being actually in the shadow,

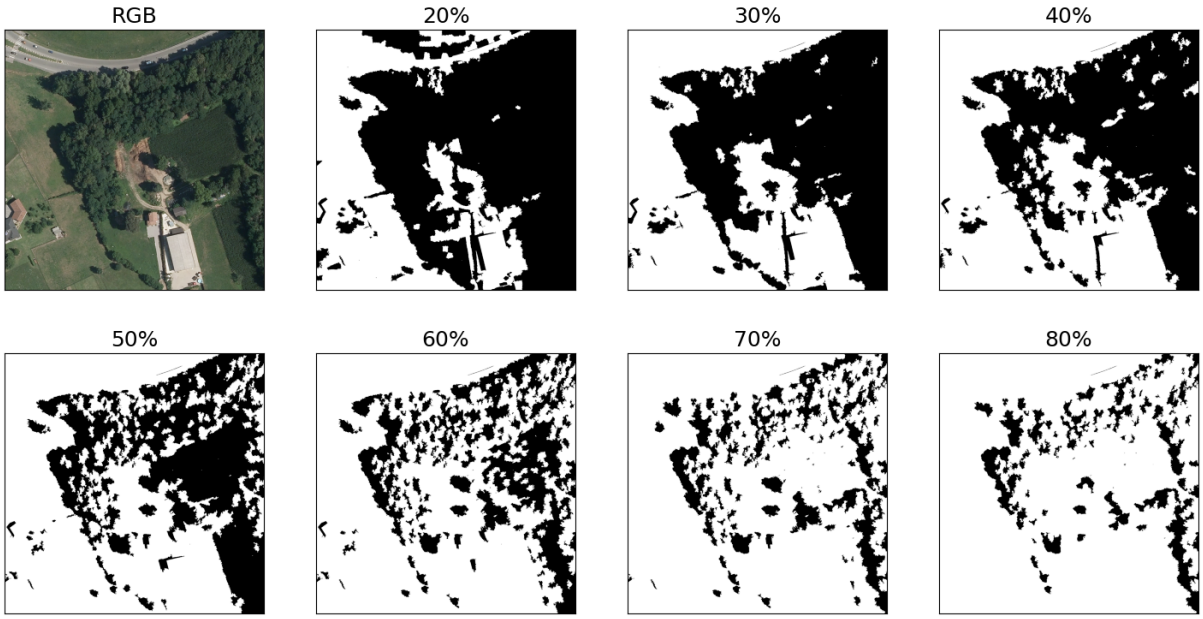



Figure 4.3: Shadow detection masks at various threshold levels.

though lowering recall, i.e., having a smaller percentage of all the shadow pixels classified as such. Given this trade-off, the threshold was set to 60%, leveraging the idea that for the eventual use of the shadow mask it would have been better to sacrifice precision in favor of recall, without lowering excessively any of the two metrics.

The choice of a threshold over 50% does not allow solving the major limitation encountered with this approach, i.e., misclassification of fields with certain cultures, such as those in the 50%-mask in Figure 4.3 and in Figure 4.4. The latter of these figures, highlights the persistence of this issue even with higher thresholds, with fields with dark-green plants still classified as shadow, whereas, to the human eye, they are clearly in the light.

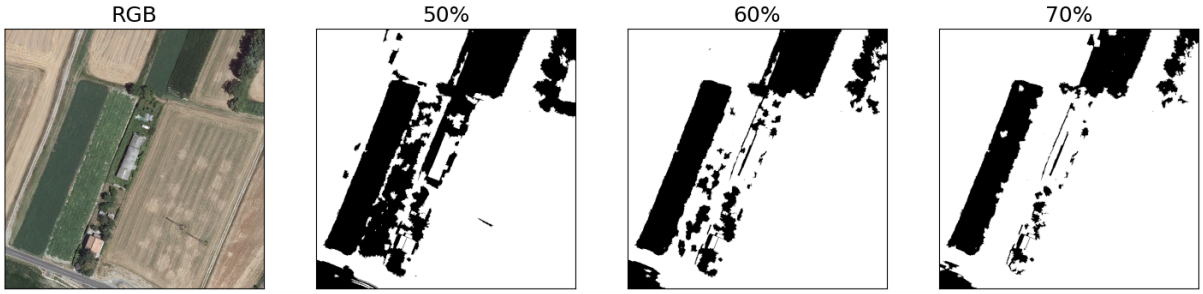



Figure 4.4: Field misclassification in SD masks with high thresholds.

## 4.2. Computation efficiency evaluation

Experiments in this section aim at evaluating the computational speed of the proposed method and the temporal impact of each type of waste on the augmentation process. For this evaluation, we generated 4 sets of 100 images each, with 3 instances of standard dimensions per image, and computed the average time for image generation. Each of these sets differs from the others solely for the waste type of implanted instances: 3 sets are built allowing introduction of instances of a single waste type, whereas the last one combines instances from all categories.

As shown in Table 4.2, the *Rubble* class accounts for most of the computational effort, with a temporal cost for augmentation which is almost 3 to 4 times higher than the cost for the other classes. This behavior can be attributed to the process for *Heap* generation, which contains procedures that have to be executed for all the vertices of the base *Grid*, vertices which might be particularly numerous, especially for bigger instances. This explains also why introducing *Tires* in an image requires more time than introducing *Bulky items*: as highlighted at the beginning of this chapter, *Tires* can appear in *Heaps*, whereas *Bulky items* cannot. Nevertheless, it can be noticed that the method requires less than 12 seconds per image to perform augmentation with instances from all categories, thus being suitable for augmentation on a large scale.

| Experiment | Time (s) | Time (min) | Time per image (s) | Time per instance (s) |
|---|---|---|---|---|
| Bulky only | 422 | 7.0 | 4.2 | 1.41 |
| Tire only | 577 | 9.6 | 5.8 | 1.92 |
| Rubble only | 1665 | 27.8 | 16.7 | 5.55 |
| All waste types | 1178 | 19.6 | 11.8 | 3.93 |

Table 4.2: Results from computation efficiency experiments. In each experiment 100 images with 3 standard-sized instances were generated.

## 4.3. Placement evaluation

*Instance placement* is a paramount phase in the proposed augmentation method. As described in Section 3.4, the designed pipeline combines several inputs, leveraging various datasets, with the aim of positioning instances in realistic areas of the background image. With the experiment described here, we aimed at evaluating the entity of such realism.

Similarly to the cases described in Sections 4.1.1 and 4.1.2, the AW dataset does not provide masks to compare with those obtained from the designed input combination. As a consequence, quantitative evaluation was once more unfeasible and the accuracy of instance placement could be evaluated only qualitatively. Therefore, for this experiment, we generated a set of 100 images, allowing placement of a random number of standard-sized instances in the range $[2, 5]$, and visually inspected them to classify each of the resulting 315 object structures according to the feasibility of their location. For this classification, we defined 3 categories:

- **Legal**: the instance is in a location that is physically feasible and potentially realistic;
- **Borderline**: the instance is located in an area that, according to the inserted inputs, is comprehensibly realistic, even though some detail suggests that it is not totally realistic: for example, this is the case of instances slightly crossing the border between light and shadow areas or lying over multiple objects with different heights;
- **Illegal**: the instance is in a location that should not be allowed by the mask combination, thus highlighting a fault in the masks themselves.

As visible in Table 4.3, the designed method places instances in totally feasible ares in 85% of the cases, leaving less than 4.5% of the waste structures in illegal regions.

| Images | Instances | Legal | Borderline | Illegal |
|---|---|---|---|---|
| 100 | 315 | 85.08% | 10.48% | 4.44% |

Table 4.3: Results from placement evaluation experiments.

(a) Illegal (b) Borderline (c) Legal

Figure 4.5: Samples from each of the categories for placement evaluation.

## 4.4. Inference on synthetic images

As an initial quantitative evaluation of the effectiveness of augmented images, we conducted a set of studies based on inference on such images with a pre-trained model. For these experiments, we adopted the public model trained by Torres et al. [12] on the AW dataset. Then, for each experiment we generated 100 images for each dimension-location constraint combination and input such images to the model. To evaluate the inference outputs, we computed the average of the confidence scores returned by the model and counted the number of recognized positives. Being all the 100 images generated to contain artificial waste instances, they are all inherently positive; therefore, the model is expected to return very high values both for average confidence and for the number of detected positives.

Results from these experiments, reported in Table 4.4, highlight instead significantly poor performances. Indeed, the model never exceeds an average confidence of 20% and detects at most 14 positives over 100 images. In addition, the network seems to be biased towards centered and, above all, greater instances. This latter type of instances, indeed, allows the model average confidence to double with respect to the cases with standard-sized instances.

| | **Not centered** | **Centered** |
|---|---|---|
| **Standard size** | 7.32% (3) | 9.67% (4) |
| **Greater size** | 13.75% (10) | 18.66% (14) |

Table 4.4: Average confidence from inference on synthetic images. In brackets, the number of detected positives.

## 4.5. Training with synthetic data

With the aim of evaluating the potential of synthetic data for the task of landfill detection, our final set of experiments focused on training a CNN for binary classification. The architecture chosen for these experiments is the one presented in [8], for which a model trained on an earlier version of the AW dataset is publicly available. In our experiments, we fine-tuned such public model by extending the training set with synthetic images, without altering the validation and test set, which were thus still composed of only real images.

Following the results from Section 4.4, which highlighted a network bias towards greater

and centered instances, we decided to attempt 4 sets of training experiments, with synthetic images differing across these sets for the constraints applied to the implanted instances. In particular, in the first set of attempts (Table 4.5), instances had their standard size and could be located all over the background image, in the legal regions defined by the placement masks. Then, for the second set of fine-tuning experiments (Table 4.6), instances were forced in a central region of the base image, while preserving their standard size. Finally, for the third and fourth sets of attempts (Table 4.8), the two experiments above were repeated with greater instances.

For each set of experiments, we evaluated the effect of extending the training set with a different number of synthetic images. In particular, the training set was extended by 250, 500, 750, 1000 and 1500 images. In all the attempted cases, we set to 0.5 the confidence threshold to discriminate between positives and negatives and fine-tuned the network with the same loss and hyper-parameters. Specifically, we trained our models on a single GPU, optimizing a binary cross-entropy loss with a linearly decreasing learning rate initialized at 0.005 and a batch size of 6. Furthermore, we introduced an early-stopping strategy based both on a patience of 10 epochs and on a 0.005 minimum delta on the loss value.

The resulting metrics, proposed in Tables 4.5 to 4.8, highlight a common pattern: the highest value in accuracy is always achieved by the base model, whereas the highest value in the F1 score is always obtained when extending the training set with exactly 500 synthetic positives. Because of this metric distribution, for which no case shows all the highest values, it is difficult to define whether the model improves or not. By considering only accuracy, indeed, adding synthetic images clearly leads to metric losses, thus suggesting a model regression. Instead, if considering only the F1 score, several fine-tuned models represent an improvement with respect to the baseline model. This dilemma might be solved

| Experiment | Accuracy | Precision | Recall | F1-Score |
|---|---|---|---|---|
| Base | **87.29%** | 81.89% | 79.54% | 80.70% |
| 250 | 86.60% | 77.92% | **83.56%** | 80.64% |
| 500 | 87.26% | 81.21% | 80.46% | **80.83%** |
| 750 | 86.68% | 82.16% | 76.78% | 79.38% |
| 1000 | 86.45% | **82.27%** | 75.75% | 78.87% |
| 1500 | 86.68% | 82.16% | 76.78% | 79.38% |

Table 4.5: Results from fine-tuning experiments with standard-sized instances located in any part of the image. In bold, highest value for each metric.

| Experiment | Accuracy | Precision | Recall | F1-Score |
|---|---|---|---|---|
| Base | **87.29%** | 81.89% | 79.54% | 80.70% |
| 250 | 86.72% | **82.91%** | 75.86% | 79.23% |
| 500 | 86.99% | 77.69% | **85.63%** | **81.47%** |
| 750 | 85.18% | 79.02% | 75.75% | 77.35% |
| 1000 | 86.99% | 79.47% | 82.30% | 80.86% |
| 1500 | 87.26% | 80.02% | 82.41% | 81.20% |

Table 4.6: Results from fine-tuning experiments with standard-sized instances located at the center of the image. In bold, highest value for each metric.

| Experiment | Accuracy | Precision | Recall | F1-Score |
|---|---|---|---|---|
| Base | **87.29%** | 81.89% | 79.54% | 80.70% |
| 250 | 86.18% | 79.51% | 78.97% | 79.24% |
| 500 | 87.10% | 78.40% | 84.71% | **81.44%** |
| 750 | 86.87% | **82.04%** | 77.70% | 79.81% |
| 1000 | 86.91% | 80.37% | 80.46% | 80.41% |
| 1500 | 86.30% | 74.83% | **88.85%** | 81.24% |

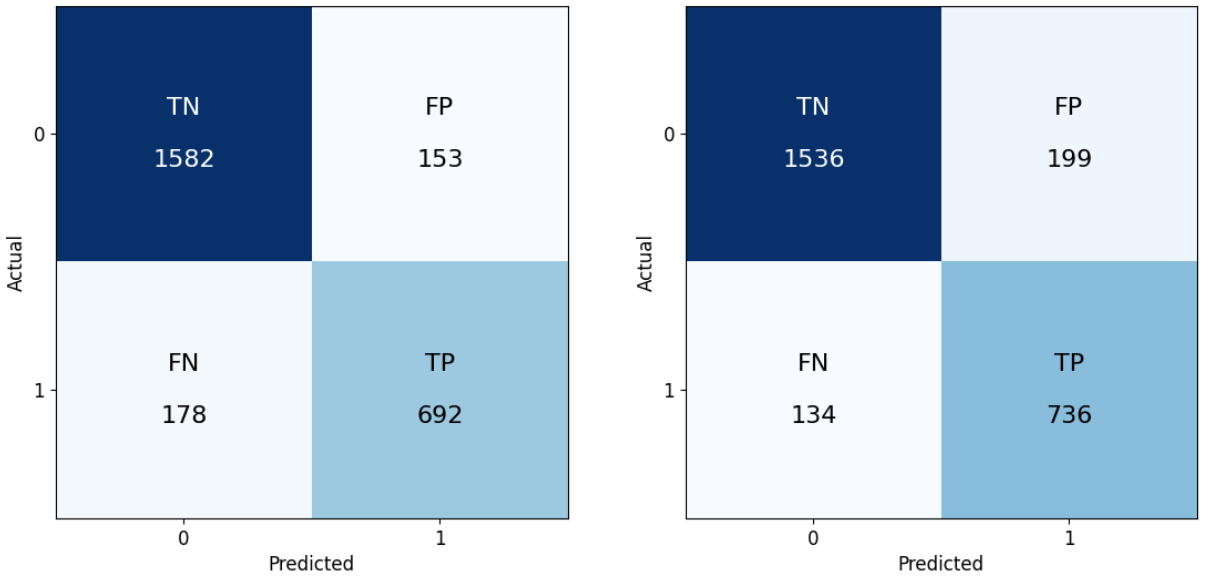
Table 4.7: Results from fine-tuning experiments with greater-sized images located in a any part of the image. In bold, highest value for each metric.

| Experiment | Accuracy | Precision | Recall | F1-Score |
|---|---|---|---|---|
| Base | **87.29%** | **81.89%** | 79.54% | 80.70% |
| 250 | 86.76% | 79.53% | 81.26% | 80.39% |
| 500 | 87.22% | 78.72% | **84.60%** | **81.55%** |
| 750 | 86.30% | 81.32% | 76.55% | 78.86% |
| 1000 | 87.02% | 78.60% | 84.02% | 81.22% |
| 1500 | 86.64% | 79.46% | 80.92% | 80.18% |

Table 4.8: Results from fine-tuning experiments with greater-sized images located in a central portion of the image. In bold, highest value for each metric.

by considering the eventual aim for deploying an application based on these models. As noticed by Torres et al. [8], these models might be useful to environmental inspectors for detecting dangerous landfills. In this context, a higher number of positives, coinciding with higher recall and lower precision, might be preferable than a better accuracy. Indeed, a higher number of false positives might simply translate in requiring either more human inspections of aerial images or more physical inspections on waste sites which, worst case scenario, will result in pointless visits. Under these assumptions, accuracy can be deemed as secondary, whereas F1 and recall can be considered more relevant. Therefore, given that the fine-tuned models improve F1 by significantly improving recall, these models might be considered an improvement with respect to the base one.

Additional support to this thesis can derive from comparing the baseline model with the best one from fine-tuning. This best fine-tuned model is the one obtained by extending the training set with 500 images with greater and centered instances. By comparing its accuracy with that of the baseline, it is possible to notice that this metric decreases by 0.07%, an amount which is almost null and significantly lower than the increase in F1, 0.85%, which is more than 10 times higher. In addition, by extracting the confusion matrices for this 2 models (Figure 4.6), it can be computed that the accuracy decrease is due to an additional misclassification of 2 images over the 2605 in the test set. Furthermore, it is possible to notice that recall increases of more than 5%, whereas precision decreases of slightly more than 3%, with almost a 2% difference in favor of recall. Given the dis-




(a) Baseline model (b) Fine-tuned model

Figure 4.6: Confusion matrices for the models under comparison.

cussion above and this quantitative comparison, this fine-tuned model can rightfully be considered an improvement from the baseline.

Finally, it might be interesting to analyse why the best F1 for fine-tuned models is obtained with an extension of exactly 500 images. Reasons for this behavior might lie in the quantity of synthetic images with relation to the number of real positives already included in the dataset, i.e., 2301. 500 is approximately 20% of such number, a percentage which might be sufficiently high to positively impact on the learning capabilities of the network without preventing it from learning from the actual positives. Indeed, considering the limited variability of our synthetic images, raising too much their number might lead the network to focus more on our instances than on the real positives.

# 5 | Conclusions and future work

This thesis addressed the problem of illegal landfill detection as a binary classification task, aiming to solve one of its major issues, i.e., positive sample scarcity, by proposing a method for synthetic data augmentation. This method allows to artificially create positive samples with multiple synthetic landfill instances in averagely $\approx 10$ seconds per image.

Within this context, we designed a pipeline leveraging the Blender modelling tool to implant various waste object structures in Remote Sensing images from the AerialWaste dataset. This pipeline also combines geographic and semantic information from each image to define the most realistic location for implanting an instance, eventually positioning less than 4.5% of all instances in unfeasible areas. Furthermore, the pipeline was completely automated, thus eliminating any need for human intervention in the process, thanks to the *bpy* library, a Blender extension that allows to control this tool via Python scripts.

In order to evaluate the effectiveness of the generated images, some fine-tuning experiments were conducted on a Convolutional Neural Network for which a model trained on AerialWaste is publicly available. This network, which was originally proposed by Torres et al. [8], consists of a ResNet50 backbone extended with an FPN to better account for features at different scales. The aforementioned experiments highlight that none of the fine-tuned models improves the accuracy of classification, albeit providing, when extending the training set with 500 synthetic images, an increase in F1 Score motivated by a significant increase in recall. This result might be considered an improvement, with respect to the base model, under the assumption that recall might be privileged over precision given the eventual application of the trained models, i.e., to aid human inspectors in detecting seriously dangerous landfills.

These results highlight the potential of this approach, which might be even higher, considering the various limitations in the pipeline inputs and in the quality of output images. Therefore, future work can proceed in the following directions:

- **All-waste extension**: this method can be extended to create structures with objects from any of the categories from the AerialWaste dataset, thus allowing the

creation of synthetic images with a higher degree of variability.

- **Multi-label classification**: the designed method allows to know exactly, for each augmented image, the waste category of the artificially introduced objects, thus extremely simplifying the labelling process. This process might allow the introduction of any number of positive samples for each waste category, potentially resulting in class balance.
- **Waste segmentation**: Blender can also be used to render a scene segmentation mask. In this context, it is possible to render the base aerial image as background and the injected waste objects with a color depending on their categories. This would allow both binary and multi-label Semantic Segmentation.
- **Photo-realism improvement**: The augmented images still leave a significant space for improvement in photo-realism. Above all, work can be done to introduce shadow consistency between the implanted instances and the objects represented in background images.
- **Input improvement**: Some of the masks used for instance placement can be significantly improved to overcome their current limitations, for example in the cases of Semantic Segmentation and Shadow Detection.

# Bibliography

[1] P. P. Calow, *Handbook of environmental risk assessment and management*. John Wiley & Sons, 2009.

[2] C. P. Gerba, "Risk assessment," in *Environmental and pollution science*, pp. 541–563, Elsevier, 2019.

[3] S. Štrbac, N. Stojić, B. Lončar, L. Pezo, L. Ćuričić, D. Prokić, and M. Pucarević, "Heavy metal concentrations in the soil near illegal landfills in the vicinity of agricultural areas—artificial neural network approach," *Journal of Soils and Sediments*, pp. 1–17, 2023.

[4] E. Ociepa-Kicińska, "Socio-economic challenges of removing and disposing of illegal hazardous waste dumps: Poland case study," *European Research Studies Journal*, vol. 26, no. 3, pp. 570–583, 2023.

[5] L. Fazzo, V. Manno, I. Iavarone, G. Minelli, M. De Santis, E. Beccaloni, F. Scaini, E. Miotto, D. Airoma, and P. Comba, "The health impact of hazardous waste landfills and illegal dumps contaminated sites: An epidemiological study at ecological level in italian region," *Frontiers in Public Health*, vol. 11, 2023.

[6] "Introduction to PERIVALLON project." `https://perivallon-he.eu/introduction-to-perivallon-project-2/`. Accessed: 2023-10-30.

[7] B. Fang, J. Yu, Z. Chen, A. I. Osman, M. Farghali, I. Ihara, E. H. Hamza, D. W. Rooney, and P.-S. Yap, "Artificial intelligence for waste management in smart cities: a review," *Environmental Chemistry Letters*, pp. 1–31, 2023.

[8] R. N. Torres and P. Fraternali, "Learning to identify illegal landfills through scene classification in aerial images," *Remote Sensing*, vol. 13, no. 22, p. 4520, 2021.

[9] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2117–2125, 2017.

[10] X. Sun, D. Yin, F. Qin, H. Yu, W. Lu, F. Yao, Q. He, X. Huang, Z. Yan, P. Wang,

*et al.*, "Revealing influencing factors on global waste distribution via deep-learning based dumpsite detection from satellite imagery," *Nature Communications*, vol. 14, no. 1, p. 1444, 2023.

[11] X. Wang, S. Wang, C. Ning, and H. Zhou, "Enhanced feature pyramid network with deep semantic embedding for remote sensing scene classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 59, no. 9, pp. 7918–7932, 2021.

[12] R. N. Torres and P. Fraternali, "Aerialwaste dataset for landfill discovery in aerial and satellite images," *Scientific Data*, vol. 10, no. 1, p. 63, 2023.

[13] C. Fasana and S. Pasini, "Learning to detect illegal landfills in aerial images with scarce labeling data," 2022.

[14] C. Padubidri, A. Kamilaris, and S. Karatsiolis, "Accurate detection of illegal dumping sites using high resolution aerial photography and deep learning," in *2022 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, pp. 451–456, IEEE, 2022.

[15] "blender.org - Home of the Blender project - Free and Open 3D Creation Software." `https://www.blender.org`. Accessed: 2023-11-05.

[16] M. D. Vaverková, A. Maxianová, J. Winkler, D. Adamcová, and A. Podlasek, "Environmental consequences and the role of illegal waste dumps and their impact on land degradation," *Land Use Policy*, vol. 89, p. 104234, 2019.

[17] Y. Kim and J. Cho, "Aidm-strat: augmented illegal dumping monitoring strategy through deep neural network-based spatial separation attention of garbage," *Sensors*, vol. 22, no. 22, p. 8819, 2022.

[18] M. Kubásek and J. Hřebíček, "Involving citizens into mapping of illegal landfills and other civic issues in the czech republic," 2014.

[19] D. Garofalo and F. Wobber, "Solid waste and remote sensing," *Photogrammetric engineering*, vol. 40, no. 1, pp. 45–59, 1974.

[20] T. L. Erb, W. R. Philipson, W. L. Teng, and T. Liang, "Analysis of landfills with historic airphotos," *Photogrammetric Engineering and Remote Sensing*, vol. 47, no. 9, pp. 1363–1369, 1981.

[21] J. Lyon, "Use of maps, aerial photographs, and other remote sensor data for practical evaluations of hazardous waste sites," *Photogrammetric engineering and remote sensing*, vol. 53, no. 5, pp. 515–519, 1987.

[22] E. Zilioli, M. Gomarasca, and R. Tomasoni, "Application of terrestrial thermography to the detection of waste-disposal sites," *Remote Sensing of Environment*, vol. 40, no. 2, pp. 153–160, 1992.

[23] J. B. Salleh and M. Tsudagawa, "Classification of industrial disposal illegal dumping site images by using spatial and spectral information together," in *IMTC/2002. Proceedings of the 19th IEEE Instrumentation and Measurement Technology Conference (IEEE Cat. No. 00CH37276)*, vol. 1, pp. 559–563, IEEE, 2002.

[24] C. Notarnicola, M. Angiulli, and C. I. Giasi, "Southern italy illegal dumps detection based on spectral analysis of remotely sensed data and land-cover maps," in *Remote Sensing for Environmental Monitoring, GIS Applications, and Geology III*, vol. 5239, pp. 483–493, SPIE, 2004.

[25] S. Silvestri and M. Omri, "A method for the remote sensing identification of uncontrolled landfills: formulation and validation," *International Journal of Remote Sensing*, vol. 29, no. 4, pp. 975–989, 2008.

[26] G. Biotto, S. Silvestri, L. Gobbo, E. Furlan, S. Valenti, and R. Rosselli, "Gis, multicriteria and multi-factor spatial analysis for the probability assessment of the existence of illegal landfills," *International Journal of Geographical Information Science*, vol. 23, no. 10, pp. 1233–1244, 2009.

[27] Y. Chinatsu, "Possibility of monitoring of waste disposal site using satellite imagery," *JIFS*, vol. 6, pp. 23–28, 2009.

[28] K. Glanville and H.-C. Chang, "Remote sensing analysis techniques and sensor requirements to support the mapping of illegal domestic waste disposal sites in queensland, australia," *Remote Sensing*, vol. 7, no. 10, pp. 13053–13069, 2015.

[29] L. Selani, *Mapping illegal dumping using a high resolution remote sensing image case study: Soweto township in South Africa*. PhD thesis, University of the Witwatersrand, Faculty of Science, School of Geography ..., 2017.

[30] N. Cristianini and E. Ricci, "Support vector machines," in *Encyclopedia of Algorithms*, pp. 928–932, Springer-Verlag, 2008.

[31] L. Breiman, "Random forests," *Machine learning*, vol. 45, pp. 5–32, 2001.

[32] Y. Z. Ulloa-Torrealba, A. Schmitt, M. Wurm, and H. Taubenböck, "Litter on the streets-solid waste detection using vhr images," *European Journal of Remote Sensing*, vol. 56, no. 1, p. 2176006, 2023.

[33] S. Abdukhamet, "Landfill detection in satellite images using deep learning," *Shanghai Jiao Tong University Shanghai: Shanghai, China*, 2019.

[34] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988, 2017.

[35] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

[36] O. Youme, T. Bayet, J. M. Dembele, and C. Cambier, "Deep learning and remote sensing: detection of dumping waste using uav," *Procedia Computer Science*, vol. 185, pp. 361–369, 2021.

[37] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pp. 21–37, Springer, 2016.

[38] M. R. Devesa and A. V. Brust, "Mapping illegal waste dumping sites with neural-network classification of satellite imagery," *arXiv preprint arXiv:2110.08599*, 2021.

[39] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pp. 234–241, Springer, 2015.

[40] S. Shahab and M. Anjum, "Solid waste management scenario in india and illegal dump detection using deep learning: an ai approach towards the sustainable waste management," *Sustainability*, vol. 14, no. 23, p. 15896, 2022.

[41] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[42] 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*, pp. 770–778, 2016.

[43] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[44] J. Jordon, L. Szpruch, F. Houssiau, M. Bottarelli, G. Cherubin, C. Maple, S. N. Cohen, and A. Weller, "Synthetic data–what, why and how?," *arXiv preprint arXiv:2205.03257*, 2022.

[45] T. E. Raghunathan, "Synthetic data," *Annual review of statistics and its application*, vol. 8, pp. 129–140, 2021.

[46] S. P. Parker, *McGraw-Hill dictionary of scientific and technical terms*. McGraw-Hill, Inc., 1994.

[47] S. I. Nikolenko, "Synthetic data for deep learning," *arXiv preprint arXiv:1909.11512*, 2019.

[48] G. Paulin and M. Ivasic-Kos, "Review and analysis of synthetic dataset generation methods and techniques for application in computer vision," *Artificial Intelligence Review*, pp. 1–45, 2023.

[49] J. Song, H. Chen, and N. Yokoya, "Syntheworld: A large-scale synthetic dataset for land cover mapping and building change detection," *arXiv preprint arXiv:2309.01907*, 2023.

[50] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, "The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3234–3243, 2016.

[51] "Unity Real-Time Development Platform | 3D, 2D, VR & AR Engine." `https://unity.com`. Accessed: 2023-11-09.

[52] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig, "Virtual worlds as proxy for multi-object tracking analysis," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4340–4349, 2016.

[53] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE conference on computer vision and pattern recognition*, pp. 3354–3361, IEEE, 2012.

[54] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, "Playing for data: Ground truth from computer games," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*, pp. 102–118, Springer, 2016.

[55] G. J. Brostow, J. Fauqueur, and R. Cipolla, "Semantic object classes in video: A

high-definition ground truth database," *Pattern Recognition Letters*, vol. 30, no. 2, pp. 88–97, 2009.

[56] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, "A naturalistic open source movie for optical flow evaluation," in *Computer Vision–ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part VI 12*, pp. 611–625, Springer, 2012.

[57] F. Kong, B. Huang, K. Bradbury, and J. Malof, "The Synthinel-1 dataset: A collection of high resolution synthetic overhead imagery for building segmentation," in *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pp. 1814–1823, 2020.

[58] "Procedural 3D City Generator | 3D City Design for Urban Environments." `https://www.esri.com/en-us/arcgis/products/arcgis-cityengine/overview`. Accessed: 2023-11-10.

[59] J. Shermeyer, T. Hossler, A. Van Etten, D. Hogan, R. Lewis, and D. Kim, "Rareplanes: Synthetic data takes flight," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 207–217, 2021.

[60] "The most powerful real-time 3D creation tool - Unreal Engine." `https://www.unrealengine.com/`. Accessed: 2023-11-10.

[61] Y. Xu, B. Huang, X. Luo, K. Bradbury, and J. M. Malof, "Simpl: Generating synthetic overhead imagery to address custom zero-shot and few-shot detection problems," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 15, pp. 4386–4396, 2022.

[62] "Sketchfab - The best 3D viewer on the web." `https://sketchfab.com/`. Accessed: 2023-11-16.

[63] "Home - Geoportale della Lombardia." `https://www.geoportale.regione.lombardia.it`. Accessed: 2023-11-09.

[64] "CORINE Land Cover — Copernicus Land Monitoring Service." `https://land.copernicus.eu/en/products/corine-land-cover`. Accessed: 2023-11-27.

[65] J. Xia, N. Yokoya, B. Adriano, and C. Broni-Bediako, "Openearthmap: A Benchmark Dataset for Global High-Resolution Land Cover Mapping," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, January 2023.

[66] "CodaLab - Competition." `https://codalab.lisn.upsaclay.fr/competitions/9121`. Accessed: 2023-11-16.

[67] M. Kakooei and Y. Baleghi, "Shadow detection in very high resolution RGB images using a special thresholding on a new spectral–spatial index," *Journal of Applied Remote Sensing*, vol. 14, no. 1, pp. 016503–016503, 2020.

[68] F. Albregtsen *et al.*, "Statistical texture measures computed from gray level coocurrence matrices," *Image processing laboratory, department of informatics, university of oslo*, vol. 5, no. 5, 2008.

[69] R. Achanta and S. Susstrunk, "Superpixels and polygons using simple non-iterative clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4651–4660, 2017.

[70] L. Wang, R. Li, C. Zhang, S. Fang, C. Duan, X. Meng, and P. M. Atkinson, "UNetFormer: A UNet-like transformer for efficient semantic segmentation of remote sensing urban scene imagery," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 190, pp. 196–214, 2022.

# A | Land cover categories

| ID | Name |
|---|---|
| **111** | Continuous urban fabric |
| **1111** | Dense residential fabric |
| **1112** | Averagely dense continuous residential fabric |
| **112** | Discontinuous urban fabric |
| **1121** | Discontinuous residential fabric |
| **1122** | Sparse and nucleiform residential fabric |
| **1123** | Scattered residential fabric |
| **11231** | Farmsteads |
| **121** | Industrial or commercial units of large factories for public and private services |
| **1211** | Industrial or commercial units and annexed spaces |
| **12111** | Industrial or commercial units |
| **12112** | Agricultural units |
| **1212** | Large factory units for public and private services |
| **12121** | Hospital units |
| **12122** | Public and private service units |
| **12123** | Technological units |
| **12124** | Cemeteries |
| **12125** | Military areas |
| **12126** | Photovoltaic installations |
| **122** | Road and rail networks and associated land |
| **1221** | Road networks and associated land |
| **1222** | Rail networks and associated land |
| **123** | Port areas |

| ID | Name |
|---|---|
| **124** | Airports |
| **131** | Mineral extraction sites |
| **132** | Dump sites |
| **133** | Construction sites |
| **134** | Unused and non-vegetated degradated areas |
| **141** | Green urban areas |
| **1411** | Parks and gardens |
| **1412** | Green uncultivated areas |
| **142** | Sport and leisure facilities |
| **1421** | Sport units |
| **1422** | Campings and receptive units |
| **1423** | Amusement parks |
| **1424** | Archeological areas |
| **211** | Non-irrigated arable land |
| **2111** | Simple arable land |
| **2112** | Wooded arable land |
| **2113** | Horticultural lands |
| **21131** | Full-field horticultural lands |
| **21132** | Protected horticultural lands |
| **2114** | Nursery cultures |
| **21141** | Full-field nursery cultures |
| **21142** | Protected nursery cultures |
| **2115** | Familiar cultures |
| **212** | Permanently irrigated land |
| **213** | Rice fields |
| **221** | Vineyards |
| **222** | Fruit trees and berry plantations |
| **223** | Olive groves |
| **224** | Firewood arboriculture |
| **2241** | Poplar woods |
| **2242** | Other wood cultures |
| **231** | Pastures |

| ID | Name |
|---|---|
| **2311** | Pastures without arboreal or shrub-like species |
| **2312** | Pastures with arboreal or shrub-like species |
| **2313** | Water meadows |
| **311** | Broad-leaved forest |
| **3111** | High-density broad-leaved forest |
| **31111** | High-density coppice broad-leaved forest |
| **31112** | High-density tall-tree broad-leaved forest |
| **3112** | Low-density broad-leaved forest |
| **31121** | Low-density coppice broad-leaved forest |
| **31122** | Low-density tall-tree broad-leaved forest |
| **3113** | Riparian formations |
| **3114** | Chestnut groves |
| **312** | Coniferous forest |
| **3121** | Mid-high-density coniferous forest |
| **3122** | Low-density coniferous forest |
| **313** | Mixed forest |
| **3131** | High-density mixed forest |
| **31311** | High-density coppice mixed forest |
| **31312** | High-density tall-tree mixed forest |
| **3132** | Low-density mixed forest |
| **31321** | Low-density coppice mixed forest |
| **31322** | Low-density tall-tree mixed forest |
| **314** | Recent reforestations |
| **321** | Natural grasslands |
| **3211** | Natural grasslands without arboreal or shrub-like species |
| **3212** | Natural grasslands with arboreal or shrub-like species |
| **322** | Moors and heathland |
| **3221** | Heathland |
| **3222** | Riverbed vegetation |
| **3223** | Bank vegetation |
| **324** | Transitional woodland-shrub |
| **3241** | Heathlands with arboreal or shrub-like species |
| **3241** | Heathlands with arboreal or shrub-like species |
| **3242** | Heathlands in abandoned agricultural areas |
| **331** | Beaches, dunes, sands |
| **332** | Bare rocks |
| **333** | Sparsely vegetated areas |
| **335** | Glaciers and perpetual snow |
| **411** | Inland marshes |
| **511** | Water courses |
| **512** | Water bodies |
| **5121** | Natural water bodies |
| **5122** | Artificial water bodies |
| **5123** | Water bodies from extractive aquifer activities |

# B | Geometry nodes

This section describes the structure and behavior of the *Geometry node* configurations used for generating the various object structures as discussed in Section 3.3.

Before delving into the details of each specific configuration, it might be worth noticing that each node is associated a type, which is described by a specific color. In particular, in our configurations, 4 types of nodes have been used:

- **Group input/output** (**black**): these nodes represent inputs and outputs of the whole processing flow and they are compulsory in every configuration.
- **Geometry** (**teal**): these nodes operate transformations on geometry elements.
- **Math** (**blue**): the nodes operate computations on numeric inputs and provide numeric outputs.
- **Input** (**wine**): these nodes can be used to provide additional inputs from the scene to the flow.

In addition, each node is featured by a set of input and output sockets, which have themselves a type depending on the handled data and associated to a specific color. The socket types used in our configurations are:

- **Integer**, associated to color **green**;
- **Float**, associated to color **grey**;
- **Geometry**, associated to color **aquamarine**;
- **Vector**, associated to color **purple**;
- **String**, associated to color **sky blue**;
- **Boolean**, associated to color **pink**.

## B.1. *Heaps*

The *Geometry node* configuration to create a *Heap* of objects is shown in Figure B.1 and, unlike the modelling step described in Section 3.3.2, is the simplest of the adopted configurations.

The core nodes for the creation of the *Heap* are the two Geometry nodes present in the configuration: the *Distribute Points on Faces* and the *Instance on Points*. The first of these nodes receives as input, on the *Mesh* socket, a geometry element which coincides with the mesh which the configuration is applied to as a modifier. As its name suggests, this node creates a random set of points lying on the input mesh. The geometric information of these points is passed to the other Geometry node, which accounts for placing object instances in the positions defined by the input points. The geometric output of this node is the final *Heap*.

However, no *Heap* of objects could be created if no object model was passed as input to the *Instance on Points* node, which is why a *Collection Info* node is introduced. This node inputs information about a collection of objects and defines the collection behavior in the flow. The *Original/Relative* attribute defines which position to consider for the collection

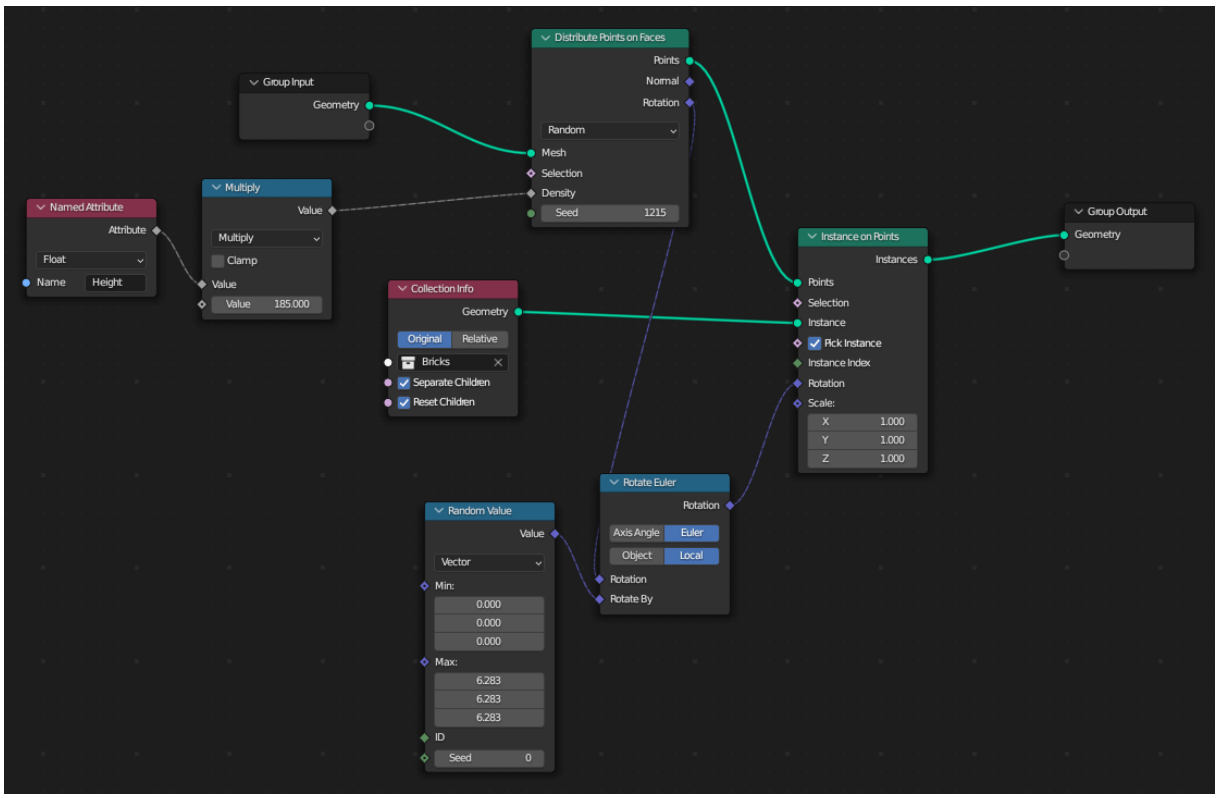



Figure B.1: Geometry node configuration to create *Heaps*.

objects: with *Original*, the considered position for each object ignores relative distances with other objects. The *Separate children* attribute allows to consider each object in the collection as a separate entity and it can be combined with the *Pick Instance* input on the *Instance on Points* node to replace each point with a single object, instead of using the whole collection. Finally, the *Reset children* input defines whether to reset all temporary transformations applied to the collection children.

A crucial aspect of the *Heap* structure is the effect of the *Weight paint*, described in Section 3.3.2, and introduced to allow instances to be positioned only on the mesh hill-like protrusions. This effect is achieved thanks to the combination of the *Named attribute* node with a *Math* node for multiplication. The first of these two nodes introduces the *Weight paint* information from the attribute called *Height* and passes it to the other node. This node scales it according to a density value, which should be passed as parameter, and outputs the scaled value to the *Distribute Points on Faces* node. In this way, points are distributed only where the scaled density value is different from 0, i.e., on the mesh protrusions.

Finally, the remaining *Math* nodes are used improve the graphical aspect of the output geometry by forcing a randomized rotation on the instances of the covering objects. The *Random value* node is used to generate a vector of 3 random floats in the range $[0, 2\pi]$. This output vector is used by the *Rotate Euler* node to overwrite default rotations for points from the *Distribute Points on Faces*, passing the output rotation values to the *Instance on Points* node.

## B.2. *Scattered objects*

The configuration for *Scattered objects* is significantly similar to that of *Heaps*, described in Section B.1, differing only in 2 aspects related to the left part of the diagram showed in Figure B.2, which shows the node configuration for this case. The right part of the diagram, involving the Geometry nodes, the *Collection Info* and the *Rotate Euler* nodes is unaltered with respect to the configuration for *Heaps*.

The main difference with the *Heap* node configuration, lies in the approach to create the rotation vector. In this case, the core idea is to restrict rotation on the $x$ and $y$ axes to multiples of 90°. This choice leveraged the assumption that, being this structure intended for objects scattered on a plane, it would be unrealistic, for such objects, to be assigned a random rotation on the aforementioned axes, since this would allow the objects to detach from the ground plane. Therefore, the only allowed random rotation was considered to happen on the $z$ axis, which is orthogonal to the ground plane and, thus, cannot detach

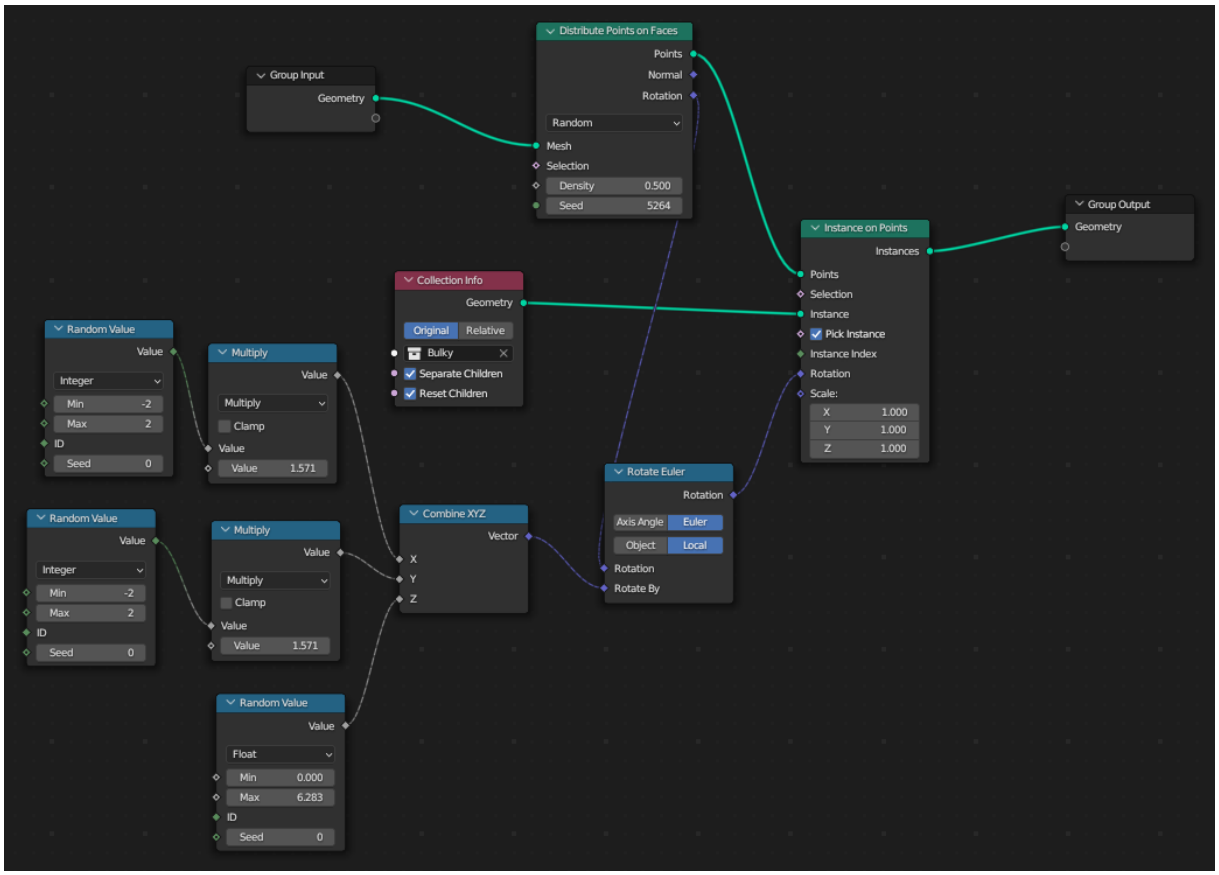



Figure B.2: Geometry node configuration for *Scattered objects*.

objects from it. This complex rotation mechanism is handled with the set of *Math* nodes in the bottom left corner of Figure B.2. The *Random Value*-*Multiply* combinations return a multiple of $\pi/2$ in the range $[-\pi, \pi]$, whereas the bottom *Random Value* determines the angle for rotation around the $z$ axis. The three numbers obtained from this process are provided as input to the *Combine XYZ* node, which combines the three inputs into the vector to pass to the *Rotate Euler* node.

The other difference from the *Heap* case lies in the absence of the two nodes for the *Weight paint* information. Given that such mechanism was not adopted to create this structure, the related node flow could be removed.

## B.3. *Stacks*

An almost totally different configuration allows to create *Stacks* of objects. With respect to the configurations described in the previous Sections B.1 and B.2, only 2 nodes are still present: the *Collection Info* and the *Instance on Points*, which are used to replace the detected points with object instances.

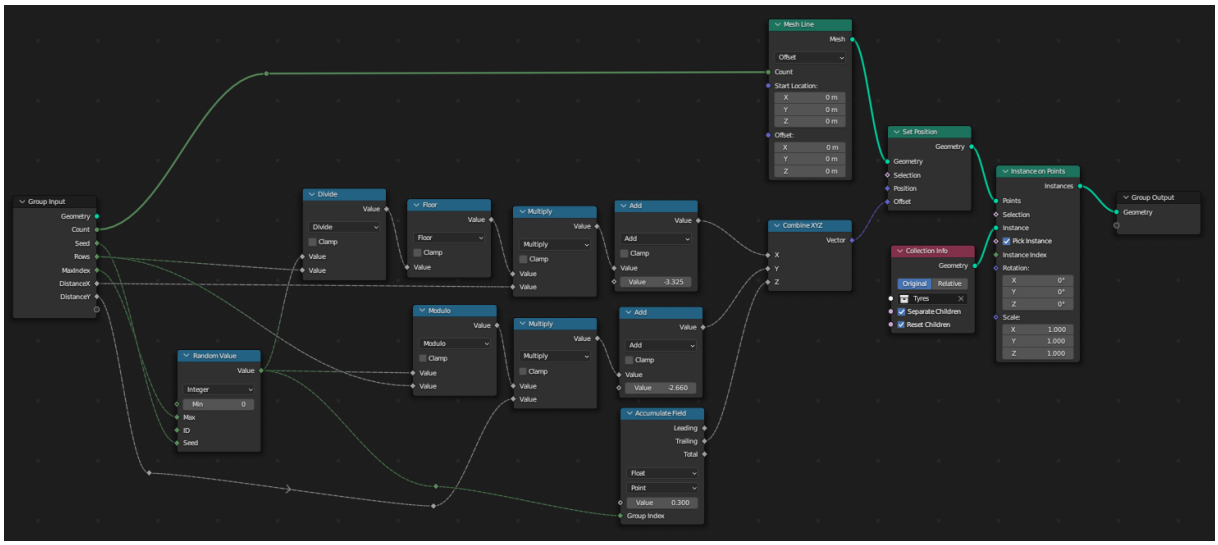



Figure B.3: Geometry node configuration to create *Stacks*.

A representation of the node configuration for *Stack* creation is provided in Figure B.3. In this representation, all the necessary parameters extend the *Group Input*. Among these parameters:

- ***Count*** is the number of objects to include in the *Stacks*;
- ***Seed*** is a random integer to control the *Random Value* node;
- ***Rows*** is the number of rows of *Stacks* to create;
- ***MaxIndex*** is an integer determining the number of *Stacks* to create
- ***DistanceX*** and ***DistanceY*** are 2 float values to use as offsets for distributing *Stacks* on a grid structure.

The key Geometry node in this configuration is the *Mesh line*, which generates vertices on a line, connected by edges and distant a given offset from each other. In this case, since position is assigned to the created points by means of another node, the *Set Position*, the *Offset* input in the *Mesh Line* has been set to a constant 0 vector. This node controls also the number of objects to stack up by means of the *Count* input, which defines the number of vertices the node should create.

As already suggested, the position for each node is defined by the *Offset* input of the *Set Position* node, which creates the mesh to pass to the *Instance on Points*. The aforementioned input is a vector obtained from the *Math* node section of the flow, which produces values for the 3 spatial coordinates and eventually combines them with a *Combine XYZ* node. For the $x$ and $y$ coordinates, the core idea is to distribute *Stacks* on a grid: the

*Random Value* node produces an index for each point which allows to determine, via modulo and division operation by the *Row* input, indices for its positioning on the grid. These indices are combined with the general inputs *DistanceX* and *DistanceY* to determine the actual $x$ and $y$ coordinates. For the $z$ coordinate, instead, the accountable node is the *Accumulate Field*, which returns an increasing value for each of the points with a specific *Group Index*. In our case, this index is the number obtained from the *Random Value* node, number which inherently identifies a specific stack on the aforementioned grid. Therefore, with the *Accumulate Field*, all the points in a specific stack are assigned an increasing value, equivalent to the $z$ of the previous point processed in the same group plus a given offset, i.e., the *Value* input of this node.

Finally, the most careful readers might have noticed that the configuration in Figure B.3 contains two *Add* nodes with fixed values. These values can be computed at run-time from the collection objects and were introduced for the sole purpose of placing the overall *Stack* structure at the center of the scene.

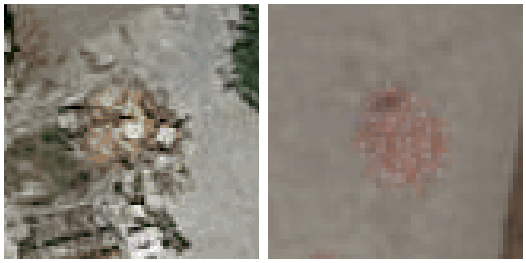
# C | Rubble collection creation

Unlike the categories of *Tires* and *Bulky items*, whose Blender collections are composed of models downloaded from the Internet, collections of shards for the *Rubble* class were procedurally generated. In particular, this generation process produced 10 types of *Rubble*, which were created by taking as reference just as many real images from AerialWaste. Examples of synthetic instances of all these types, along with their real references, are provided in Figure C.1.

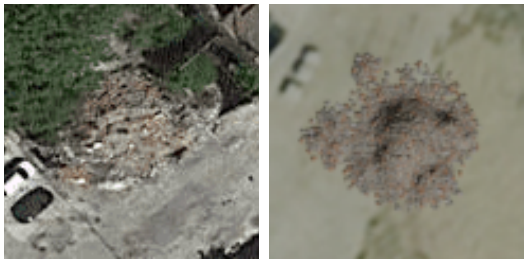
The modelling procedure started by introducing in the scene multiple copies of a Blender mesh which is deemed suitable to represent shards of a specific *Rubble* type. In this context, the chosen meshes were either the *Cube* or the *UV Sphere*. However, these meshes per se are too regular to represent shards obtained from destructive activities such as building demolition. Therefore, we added a randomization phase in which all the vertices of the base objects are shifted of a random constrained amount in any of the 3 spatial direction. This process makes the initial mesh significantly more irregular, thus resembling much more an actual *Rubble* shard.

Once the shape of the shards is defined, these must be assigned a material, which is accountable for their color and other rendering properties. All the *Rubble* types are assigned a material with shading properties depending on a node configuration, similar to those described in Appendix B. The default structure of the shading node configuration, composed of a *Principled BSDF* and a *Material Output* node, is extended with a *ColorRamp* and an *Object Info* node, as shown in Figure C.2. The *ColorRamp* allows multiple objects with the same material to acquire a different color within a specified gradient of tonalities, after receiving a certain index number, which is provided by the *Random* output of the *Object Info* node.

After applying this material to all the shards in the scene, the shards are grouped in the same collection and the scene is exported as a *blend* file. This will allow the collection to be appended to the scene when necessary during augmentation.



(a) Bricks



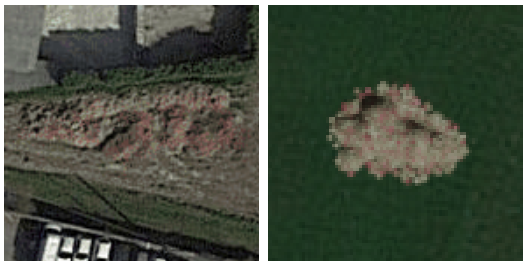
(b) Concrete

(c) Scrap

(d) Metal

(e) Woodlike 0

(f) Woodlike 1



(g) Bricks and grass

(h) Sand

(i) Earth sand

(j) Grey sand

Figure C.1: Types of *Rubble*. On the right of each pair, a synthetic instance of the specified type; on the left its real reference.

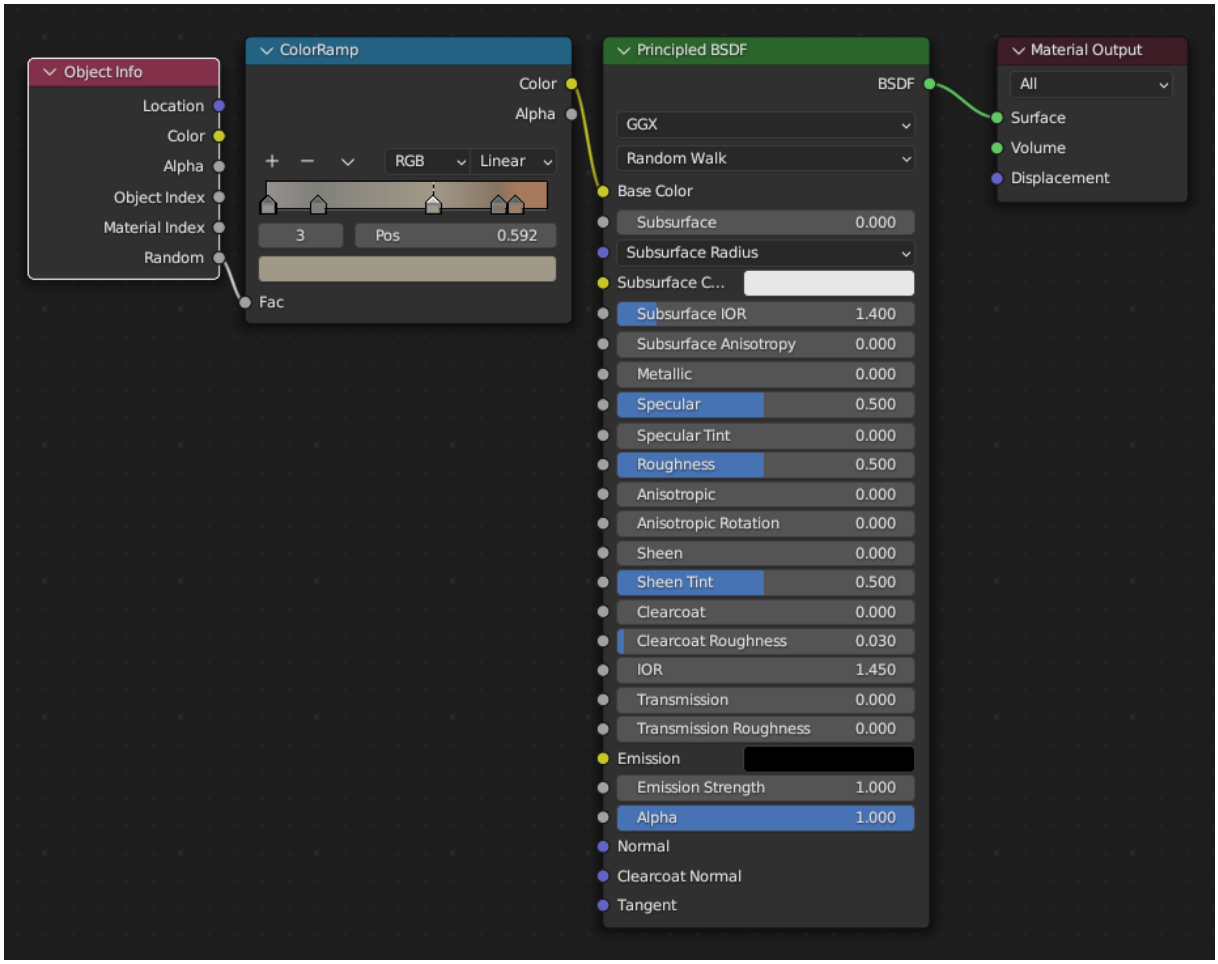



Figure C.2: Shading nodes for the *Concrete* type of *Rubble*.

# Acronyms

**AI** Artificial Intelligence. 1, 6

**AW** AerialWaste. 8, 16–19, 21–23, 26, 30, 34–37, 39, 40, 43–46, 49, 50, 55, 73, 79, 80

**CNN** Convolutional Neural Network. 1, 2, 6–9, 35, 50, 55

**CV** Computer Vision. 1, 9, 10

**DL** Deep Learning. 1, 6, 8, 35

**FPN** Feature Pyramid Network. 2, 8, 55

**GE** Google Earth. 18, 19, 39

**GIS** Geographic Information System. 6, 14

**GSD** Ground Sampling Distance. 7, 18, 22, 24, 44

**ML** Machine Learning. 6, 9

**MTS** <Mesh-Texture-Shader> combination. 11, 12

**OEM** OpenEarthMap. 24, 25, 35, 44, 45, 79, 81

**RF** Random Forest. 6

**RS** Remote Sensing. 1, 2, 6, 13, 17, 18, 24, 55

**SD** Shadow Detection. 19, 39, 45–47, 56, 80

**SDA** Synthetic Data Augmentation. 2

**SS** Semantic Segmentation. 9, 10, 19, 24, 25, 36, 44–46, 56, 80

# List of Figures

2.1 Example from the SYNTHIA dataset. From [50]. . . . . 10
2.2 Example from the Virtual KITTI dataset. Original image, variations and segmentation label. Adapted from [52]. . . . . 11
2.3 Examples from the Grand Theft Auto V dataset. From [54]. . . . . 12
2.4 Examples of frames and related motion images from the MPI-Sintel dataset. Adapted from [56]. . . . . 13
2.5 Selected styles for Synthinel-1 aerial synthetic images. From [57]. . . . . 14
2.6 Examples of real (top row) and synthetic (bottom row) images from the RarePlanes dataset. Adapted from [59] . . . . . 14
2.7 The SIMPL procedure for synthetic aerial images generation. (a) is a real image for background, (b) is a 3D object model, (c) is the set of parameters to build the scene. From [61] . . . . . 15
2.8 Examples from SyntheWorld. Adapted from [49]. . . . . 16

3.1 Object structures and waste categories. . . . . 17
3.2 Examples of AerialWaste images from different sources. . . . . 19
3.3 Examples of manual segmentations of waste instances in aerial images of the AW dataset. TO = Type of Object. SM = Storage Mode. From [12]. . 21
3.4 Blender *collections* for the different waste types. . . . . 22
3.5 Examples of images from the AW dataset with overlaid polygons from DUSAF7.0. . . . . 23
3.6 Examples of images and labels from the OEM dataset. . . . . 24
3.7 The pipeline and its phases, with focus on *Scene setup* and *Rendering*. . . 25
3.8 Representation of the *Instance creation* phase. Top line: the procedure to create *Heaps*. Bottom line: the procedure for *Stacks* and *Scattered objects*. 27
3.9 Different approaches for lifting vertices . . . . . 29
3.10 The *Instance placement* phase: input mask combination and effects of erosion on legal regions. The center of the output rectangle is found in the red areas. . . . . 31

3.11 Land cover histograms. Category 3112 was excluded from percentage ratio because the ratio would have been infinite. . . . . 32
3.12 Example for land cover. . . . . 34
3.13 Example for DTM . . . . . 35
3.14 Example for Semantic Segmentation. . . . . 36
3.15 Components and outputs of the *SSSI* computation on an AW image. No adjustment applied to index computation. . . . . 37
3.16 Thresholding flaws following the original approach from [67]. . . . . 38
3.17 Comparison of *SSSI* image before thresholding after index computation with and without *SENT*. . . . . 38
3.18 Exemplary image from the AW dataset and its related mask for Shadow Detection. . . . . 39
3.19 The "Imagery" filter used for template matching. . . . . 40
3.20 Examples of Google overlays. Red cross indicates the output position from template matching. Cyan polygons highlight the final overlays to remove. . 40
3.21 Exemplary image from the AW dataset and its related mask for Google overlays. . . . . 40

4.1 Mask combination for *centered* instances. . . . . 43
4.2 Examples of good (4.2a to 4.2d) and bad (4.2e and 4.2f) segmentation labels on images from the AW dataset. . . . . 46
4.3 Shadow detection masks at various threshold levels. . . . . 47
4.4 Field misclassification in SD masks with high thresholds. . . . . 47
4.5 Samples from each of the categories for placement evaluation. . . . . 49
4.6 Confusion matrices for the models under comparison. . . . . 53

B.1 Geometry node configuration to create *Heaps*. . . . . 68
B.2 Geometry node configuration for *Scattered objects*. . . . . 70
B.3 Geometry node configuration to create *Stacks*. . . . . 71

C.1 Types of *Rubble*. On the right of each pair, a synthetic instance of the specified type; on the left its real reference. . . . . 74
C.2 Shading nodes for the *Concrete* type of *Rubble*. . . . . 75

# List of Tables

3.1 Distribution of positives and negatives for binary classification in the AerialWaste dataset. . . . . 19
3.2 Distribution of categories for multi-label classification in the AerialWaste dataset. . . . . 20
3.3 Removable land cover categories with the number of positive images containing them (occurrences). . . . . 33

4.1 Intersection over Union (IoU) metric for our model on the OEM test set. . 45
4.2 Results from computation efficiency experiments. In each experiment 100 images with 3 standard-sized instances were generated. . . . . 48
4.3 Results from placement evaluation experiments. . . . . 49
4.4 Average confidence from inference on synthetic images. In brackets, the number of detected positives. . . . . 50
4.5 Results from fine-tuning experiments with standard-sized instances located in any part of the image. In bold, highest value for each metric. . . . . 51
4.6 Results from fine-tuning experiments with standard-sized instances located at the center of the image. In bold, highest value for each metric. . . . . 52
4.7 Results from fine-tuning experiments with greater-sized images located in a any part of the image. In bold, highest value for each metric. . . . . 52
4.8 Results from fine-tuning experiments with greater-sized images located in a central portion of the image. In bold, highest value for each metric. . . . 52

# Acknowledgements

Special thanks to Athanasios Petsanis, who contributed with suggestions for quality improvement in the process of *Heap* generation, and to all the Sketchfab artists who created the models included in our waste collections and released them under the CC-BY license. These artists are presented in the following table.

| **Author** | **Model** |
|---|---|
| Tahir | Old Dirty Tire |
| Joel Wood | Old Tire |
| SusanKing | Used tire / tyre |
| ROY | Tire |
| Rookster | Tyre |
| ittoKubashi7 | Big plastic water tank |
| Isaack - Tacko The 3D Guy | Painted I-Beam |
| Isaack - Tacko The 3D Guy | I-Beam |
| JCarpenter_48 | Metal Beam - Low Poly |
| MaX3Dd | Wooden Boxes |
| paethon | Pipe Straight |
| Modelling in Progress | Rusty Pipe |
| Thunder | Pipe Melee Weapon |
| NotAnotherApocalypticCo. | Set of Cardboard Boxes |
| Artem Goyko | Box |
| SIgn_man | Brick low poly |
| Gato_Ze | Tijolo Low Poly |

# Ringraziamenti

La conclusione di questo lavoro di tesi è destinata a diventare una pietra miliare della mia carriera da studente, e non solo. Pertanto, credo sia arrivato il momento di spendere qualche parola per manifestare la mia riconoscenza verso tutti coloro che hanno contribuito a rendere questi ultimi anni indimenticabili. Anche se, probabilmente, basterebbe una parola sola: grazie.

Grazie, in primis, al mio relatore, il Professor Piero Fraternali, per avermi offerto questa opportunità di tesi ed essere stato un valido riferimento nel corso dell'ultimo anno.

Grazie a tutti i membri del progetto PERIVALLON, al Professor Giacomo Boracchi e ai dottorandi, Federico, Sergio e Andrea, per avermi guidato nel percorso di tesi, e agli altri tesisti, Elena e Simone, con i quali ho avuto il piacere di condividere parte di questo lavoro.

Grazie a tutti i colleghi e a tutti gli amici conosciuti sui banchi del Politecnico e, in particolare, a coloro con cui ho collaborato per i vari progetti didattici: è anche merito vostro se, oggi, sono qui a scrivere queste parole.

Kiitos, a tutti i membri della Aalto Community e a tutte le persone conosciute durante il mio Erasmus in Finlandia, per aver contribuito a rendere tale esperienza straordinaria, permettendomi di creare ricordi indelebili.

Grazie infine, ma soprattutto, alla mia famiglia e, nello specifico, ai miei genitori, per avermi sempre supportato in ogni modo e per aver sempre creduto in me.

*Federico*