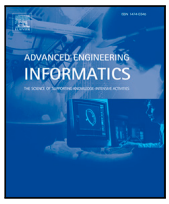
Advanced Engineering Informatics 64 (2025) 103036

Contents lists available at ScienceDirect

Advanced Engineering Informatics

journal homepage: www.elsevier.com/locate/aei



Full length article

CAMHighways: The Cambridge Highways dataset

Alix Marie d'Avigneau [a,b,*], Lilia Potseluyko [a,b], Nzebo Richard Anvo [a,b], Hussameldin M. Taha [a,b,c], Varun Kumar Reja [a], Diana Davletshina [a], Percy Lam [a], Lavindra de Silva [a], Abir Al-Tabbaa [a], Ioannis Brilakis [a]

[a] University of Cambridge, Department of Engineering, Civil Engineering Building, 7a JJ Thomson Ave, Cambridge, CB3 0FA, United Kingdom
[b] Costain PLC, Costain House, Vanwall Business Park, Maidenhead, SL6 4UB, United Kingdom
[c] University of Leeds, School of Civil Engineering, Leeds, LS2 9JT, United Kingdom

ARTICLE INFO

Dataset link: https://drf.eng.cam.ac.uk/research/camhighways-dataset

Keywords:
Datasets
Digital twins
Road maintenance
Point cloud labelling
Image labelling
Georeferencing

ABSTRACT

The CAMHighways dataset is presented, built from mobile mapping data that surveyed over 40 km of UK Highways. The dataset consists of textured meshes for road assets (including the pavement, traffic signs, and road furniture), segmented and classified point clouds, orthomosaics generated from pavement images, defect label annotations and shapefiles, and ground penetrating radar point clouds. All modalities are georeferenced and can be integrated into game engines and/or GIS software. The main aim of this work is to facilitate and automate the building of a Digital Twin (DT), a digital representation of the highway, in order to streamline inspection and maintenance through virtual reality, robotics simulation, and DT- and AI-driven data analysis. It also serves as a valuable source for other applications, such as training semantic scene understanding and defect detection algorithms. This paper introduces the dataset and outlines the data preparation process, including novel automation methods developed for this purpose, as well as integration guidelines and possible applications.

1. Introduction

Highways serve as vital arteries of transportation networks, crucial for economic activity and societal well-being [1,2]. In 2022, 86% of all passenger kilometres were travelled using on-road vehicles such as cars, vans, and taxis, 81% of freight was moved by road, and major roads made up 65% of all motor vehicle traffic [3]. However, ensuring their upkeep presents significant challenges, compounded by an ageing infrastructure and limited maintenance resources. Indeed, in 2022–23, only 13% of the UK public expenditure was spent on national roads, far behind railway expenditure [4]. Despite their criticality, the prevailing maintenance practices often fall short, plagued by inefficiencies stemming from outdated methodologies and resource allocation constraints. As an example, Norfolk's roads must deal with a backlog of £69 million worth of road repairs, and are actively looking for improved, long-term methods of sealing potholes [5].

In order to more effectively plan, maintain, and operate highways, it is important to make use of all the information and knowledge at our disposal. In the last few decades, knowledge and data-driven approaches that have been employed include modelling, simulation and ontologies to optimise maintenance planning [6,7] and highway construction [8], augmented reality to improve work zone safety [9], optimisation models and asset management principles to enhance decision-making [10–12], as well as a wide array of inspection tools and sensors, using data collected both on-site and remotely [13,14].

In recent years, Digital Twins (DTs) have become a valuable tool to improve real-life processes with the help of knowledge. A DT is a dynamic, real-time digital representation of a physical asset or system known as the Physical Twin (PT). The two communicate throughout the PT's life in order to streamline various processes [15]. DTs can be employed in several fields [16,17]. In this paper, we focus on the implementation of a DT in the highways sector. Road DTs have been explored in the past [18], and have many applications, including construction, monitoring, and traffic simulation and estimation [19,20]. In particular, a road DT can greatly improve the efficiency of operation

* Corresponding author at: University of Cambridge, Department of Engineering, Civil Engineering Building, 7a JJ Thomson Ave, Cambridge, CB3 0FA, United Kingdom.
E-mail addresses: agem2@cam.ac.uk (A. Marie d'Avigneau), lp625@cam.ac.uk (L. Potseluyko), nzra2@cam.ac.uk (N.R. Anvo), H.M.TahaAbdalgadir@leeds.ac.uk (H.M. Taha), varunreja7@gmail.com (V.K. Reja), dd593@cam.ac.uk (D. Davletshina), phl25@cam.ac.uk (P. Lam), lpd25@cam.ac.uk (L. de Silva), aa22@cam.ac.uk (A. Al-Tabbaa), ib340@cam.ac.uk (I. Brilakis).
URL: https://www.drf.eng.cam.ac.uk (A. Marie d'Avigneau).

https://doi.org/10.1016/j.aei.2024.103036
Received 19 July 2024; Received in revised form 3 December 2024; Accepted 4 December 2024
Available online 15 December 2024

and maintenance processes. For example, it can enable predictive maintenance, performance monitoring, and informed decision-making [21]. The focus of this dataset is to enable the building of a highway DT. DTs can take many forms, and in this paper, we consider the possibility of building one that can be viewed in the form of a 3D environment that has been *georeferenced*, such that all its components are linked to a coordinate system and can be placed on a map, and that can be integrated into game engines as well as Geographic Information System (GIS) software.

Mobile mapping technology [22,23] has been used for several decades for mapping and surveying purposes. In the context of highways, mobile mapping has applications such as pavement and drainage inspection [24,25], and 3D reconstruction [18]. A mobile mapping system generally consists of a mobile platform equipped with sensors such as cameras and, notably, Light Detection And Ranging (LiDAR) laser scanners, which can collect 3D point clouds of any environment. It is therefore capable of collecting invaluable data in the building of a DT [26–28].

Central to this paper is the introduction of the *CAMHighways* dataset, available by using the following link: https://drf.eng.cam.ac.uk/research/camhighways-dataset. It is based on 42.8 km of mobile mapping data collected on UK highways, generously provided by National Highways and KOREC.[1] CAMHighways comprises labelled RGB meshes generated from segmented point clouds, pavement imagery processed into orthomosaics, defect labels, and Ground Penetrating Radar (GPR) measurements. Every modality is also fully georeferenced. The dataset is tailored to build a DT and designed to enhance highway inspection and maintenance processes. It distinguishes itself from other datasets in several aspects. Firstly, existing datasets are primarily oriented towards either autonomous driving or defect detection (see Section 2), while CAMHighways can be used for both at the same time, in addition to providing a 3D virtual environment for detailed inspection, simulation, and testing. A second aspect is that while most existing datasets tend to focus on urban environments, CAMHighways exclusively provides data collected on highways. Finally, this is the first such dataset to comprise several modalities integrated together, namely imagery, point clouds, and GPR data, as well as corresponding defect and asset labels. This provides the academic and industry communities a solid footing for running simulations and training machine learning and AI algorithms, laying the groundwork for future infrastructure DTs.

As more and more data become available, it is important to streamline and automate the data preparation process to ensure the task of building large-scale DTs does not become prohibitively time-consuming. In this context, this paper aims, beyond presenting the CAMHighways dataset, to address the following questions:

1. How can we effectively make use of mobile mapping data to represent a highway DT?
2. What data and software are needed to build a high-fidelity highway 3D environment with authentic textures?
3. How can the process of building a highway DT be automated?

As we describe various aspects of mobile mapping data preparation, we also discuss state-of-the-art software tools used for processing mobile mapping data, investigate novel approaches specific to our dataset, and identify pathways for automating the building of a highway DT.

The main contributions of this paper can be summarised as follows:

- A new dataset is presented covering 42.8 km of UK highways, which comprises 190 3D meshes and point clouds, 579 orthomosaics built from over 68k pavement images, defect label shapefiles covering over 69k defects, and over 80 GPR data segments.
- A detailed breakdown of the process of building a DT to facilitate road inspection and maintenance from mobile mapping data.
- Investigation into automating several aspects of the DT generation process, including the creation of meshes, orthomosaics, GPR point clouds, and defect shapefiles for integration into both game engines and GIS software.

The paper is structured as follows. First, Section 2 reviews existing datasets created from mobile mapping or survey data and how CAMHighways aims to bridge gaps in the literature. Then, Section 3 formally presents the CAMHighways dataset, as well as state-of-the-art software investigated and employed to prepare the dataset. As part of describing the dataset's building process, Section 4 details segmenting and generating meshes for various highway assets from the available point clouds. Section 5 focuses on processing pavement images into orthomosaics georeferenced onto the mesh, as well as integrating GPR data. Labelling, both on the point cloud and pavement images, and the creation of georeferenced defect label shapefiles, are described in Section 6. Finally, Sections 7 and 8 respectively discuss integration possibilities and use cases for the CAMHighways dataset before Section 9 concludes. A table of abbreviations is also available at Table 10.

2. Related work

2.1. Existing datasets

The literature comprises several datasets with applications to transportation infrastructure and its maintenance. These can be divided into two main categories, namely real and synthetic datasets. In this section, we present an overview of existing datasets, gaps in the literature, and how the CAMHighways dataset aims to fill these gaps.

2.1.1. Real datasets

The first category consists of datasets collected from mobile scanning surveys whose end goal can be further split into two sub-categories, namely autonomous driving and defect detection.

Datasets whose main aim is to enable and improve autonomous driving are usually composed of images from cameras (e.g., CamVid [29,30], Cityscapes [31], BDD100K [32]), point clouds from LiDAR scanners (e.g., Paris-Lille-3D [33], Semantic USL [32]), or both (e.g., KITTI [34,35], KITTI-360 [36], Pandaset [37]). In general, the images and/or point clouds have been fully labelled, with the objective of scene understanding and semantic segmentation. Examples of labels include: vehicles (moving and static), pedestrians, structures, roadside assets, lanes, and vegetation. Table 1 summarises a selection of datasets with applications to autonomous driving.

The second sub-category includes datasets that instead focus on defect detection from images. The surveys carried out for these datasets do not include LiDAR scanning, so they do not consider point cloud data. They can focus on crack (AigleRN [38], CFD [39], CRACK500 [40]), pothole (Pothole-600 [41]) detection, or a mix of defects (GAPs [42–44], PID [45], RDD2022 [46], ISTD-PDS7 [47]). Table 2 summarises a selection of datasets with applications to defect detection. A more comprehensive review of datasets for visual inspection is provided in [48].

GPR-based datasets are less prominent than those relying on data collected above ground, but they are still used for similar purposes, as well as to collect data about materials making up the pavement. In [49–51], the authors collected GPR images for detecting distresses in asphalt pavement, while the dataset TU1208 [52] contains GPR images collected in a fully controlled environment with the aim of being used as a reliable reference for future GPR datasets.

2.1.2. Synthetic datasets

Synthetic datasets are created in a game engine, such as Unity or Unreal Engine, and have applications for autonomous driving and

[1] https://www.korecgroup.com/

multi-object tracking in a more controlled environment, where environmental conditions such as lightning and weather can be tweaked, and annotations are more accurate. They can be fully synthetic (e.g., SYNTHIA [53]) or quite literally be taken from a video game such as Grand Theft Auto V [54]. Alternatively, they can be based on real locations (e.g., ParallelEye [55]) or real datasets (e.g., Virtual KITTI 2 [56,57]).

2.2. CAMHighways

The existing datasets presented above are invaluable for advancements in their main applications (i.e., autonomous driving and defect detection). However, there are still some gaps in the literature. Firstly, no dataset combines data that have applications for both autonomous driving and defect detection in the form of georeferenced point clouds, images, and GPR data. Secondly, while 3D virtual environments have been built based on real-life datasets (e.g., Virtual KITTI 2), no dataset exists with the express aim of building a DT. While the authors in [58] outline a method for creating a highway DT from a mobile mapping survey, they only focus on the pavement and do not consider other roadside assets such as barriers, signs, and vegetation. Thirdly, most datasets focus on urban environments, and few consider highways. Both autonomous driving and inspection applications are affected by the type of road, namely highways, rural, and urban environments. Indeed, the authors in [59] contrast the nature of the fast, mostly uniform traffic flow of highways to that of urban environments, which is slower but more complex. As for defect detection, National Highways manuals [60] provide different guidance for categorising the severity of defects depending on the road type. Finally, the focus of existing defect detection datasets is for computer vision algorithms to recognise defects on images, but it is also important to be able to place these defects on a map, for example in the form of a georeferenced shapefile. This can be applied to pavement performance estimation and prediction, spatial modelling of defects and distresses, and predictive and preventative maintenance.

The CAMHighways dataset differs from the datasets listed above but combines elements from all of them. As will be presented in Section 3, it is built from mobile mapping data, which includes images, point clouds, and GPR, and it can be integrated into GIS software, as well as VR simulation environments and game engines such as Unreal Engine and Unity. Even though its end goal is to enable the building of a DT to facilitate road inspection and maintenance, the point cloud labelling (see Section 6.1) included in the preparation of our dataset can be applied to autonomous driving and the annotations (see Section 6.2) can be used to train defect detection algorithms. Additionally, as part of the textured meshes generated from the segmented point clouds (see Section 4), the dataset provides the first available 3D sign textured mesh library generated from real-world data.

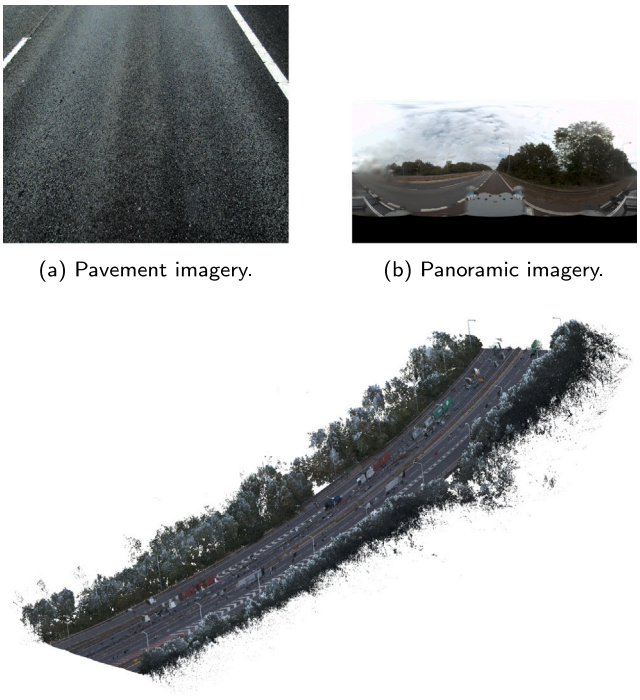
To the best of the authors' knowledge, no dataset exists so far that combines point cloud, image, and GPR sensing modalities with data collected on a highway rather than in an urban environment.

3. Overview of the dataset

In this section, we first present the mobile mapping data collected as part of the surveys carried out by KOREC, which served as the input for the data preparation process. Then, we formally introduce the CAMHighways dataset and a high-level overview of the data preparation process. We also present a review of all software tested and employed as part of the process.

3.1. Mobile mapping data collected and format

The areas surveyed extend over 42.8 km in total and span over three main roads on the SRN, namely the A11, A12 and A14. A map of the roads surveyed, including the name of each section and the length (in km) of highway captured during each survey, is provided in the Appendix (Fig. 34).



(a) Pavement imagery. (b) Panoramic imagery.

(c) RGB Point cloud.

Fig. 1. Outputs from Trimble MX9 survey.

Mobile mapping data were collected by KOREC using a Trimble MX9 in 2021. GPR data were also collected at a later time, recording information on the pavement layer material's properties. The Trimble MX9 mobile mapping system is equipped with various sensors, including two LiDAR, or laser scanners for collecting RGB point cloud data, a panoramic camera, and a pavement-facing planar camera. It was affixed on top of a vehicle (Kia Sportage), facing backwards. Once the mobile mapping runs were completed, the raw mobile mapping data were imported into Trimble Business Center (TBC) [76] for registering mobile mapping runs together, colourising point clouds, and carrying out initial automatic point cloud classification. Additional point cloud classification improvements were completed in Terrasolid software, before handing the dataset over from KOREC to National Highways and the University of Cambridge. The data extracted from both surveys are detailed in Table 3, and the Trimble MX9 output (namely point clouds, pavement images, and panoramic images) is illustrated in Fig. 1.

Each set of images and GPR measurement was georeferenced according to the British National Grid (EPSG:27700 or OSGB36). The latitude, longitude, ellipsoidal altitude, roll, pitch, heading and projected (x, y, z) locations (in metres) of the centre of the cameras were recorded every time an image was taken. A similar reference table is also available for the full trajectory of the vehicle itself. Each point in the point clouds contains projected (x, y, z) locations, RGB values, and has an intensity field which is mainly based on the reflectivity properties of the surfaces where a laser was beamed. The 3D GPR data collected by KOREC are in the form of GeoTIFF image format files, also known as rasters. Each GeoTIFF file name includes northing, easting coordinates, and the depth to which the electromagnetic reflection of the GPR corresponds. See Fig. 2 for an example of the GPR data received.

The two main purposes of the collected data are: (1) to build a DT that facilitates inspection and maintenance, consisting of a 3D environment of the roads surveyed in which every asset and defect is

Table 1
Selection of datasets with applications to autonomous driving. Guide to abbreviations: **SSU**: Semantic Scene Understanding; **AD**: Autonomous Driving; **MO**: Moving Objects (e.g. moving vehicles, pedestrians); **SO**: Static Objects (e.g. static vehicles, signs, etc.); **L**: Landscape (i.e., scenery, buildings, etc.).

Dataset	Area	Year	Location	Imagery	Point cloud	Labelling	Objectives	Ref.
CamVid	urban	2008	Cambridge, UK	✓	✗	2D, MO, SO, L	AD, SSU	[29,30]
KITTI	urban	2011	Karlsruhe, Germany	✓	✓	2D & 3D, MO	SSU, AD	[34,35]
Cityscapes	urban, off-road	2016	Germany	✓	✗	2D, MO, SO, L	SSU	[31]
Mapillary Vistas	urban, off-road	2017	6 continents	✓	✗	2D, MO, SO, L	SSU	[61]
Semantic3D	urban	2017	Central Europe	✓	✓	3D,MO, SO, L	SSU	[62]
Paris-Lille-3D	urban	2018	Paris, Lille, France	✗	✓	3D, MO, SO, L	SSU, AD	[33]
ApolloScape	urban	2018	China	✓	✓	2D & 3D, MO, SO, L, lanes	AD	[63]
nuScenes	urban	2019	USA, Singapore	✓	✓	2D & 3D, MO, SO, L	AD, SSU	[64]
A2D2	urban	2019	South Germany	✓	✓	3D, MO, SO, L, lanes	AD, SSU	[65]
Waymo Open Dataset	urban	2019	USA	✓	✓	2D & 3D, MO, SO, L	AD	[66]
KITTI-360	urban, rural	2020	Karlsruhe, Germany	✓	✓	2D & 3D, MO, SO, L	SSU, AD	[36]
Toronto-3D	urban	2020	Toronto, Canada	✓	✓	3D, MO, SO	SSU	[67]
SemanticPOSS	urban	2020	Peking University, China	✓	✓	3D, MO, SO, L	AD, SSU	[68]
HighwayDriving	highway	2020	Korea	✗	✓	2D, MO, SO, L	AD, SSU	[69]
BDD100K	urban	2020	USA	✓	✗	2D & 3D, MO, SO, L, lanes	AD	[70]
Semantic USL	campus, off-road	2021	Texas A&M, USA	✗	✓	3D, MO, SO, L	SSU	[32]
Pandaset	urban	2020	USA	✓	✓	3D, MO, SO, L	AD	[71]
K-Lane	urban, highway	2023	Korea	✓	✓	2D & 3D, lanes	AD	[37]

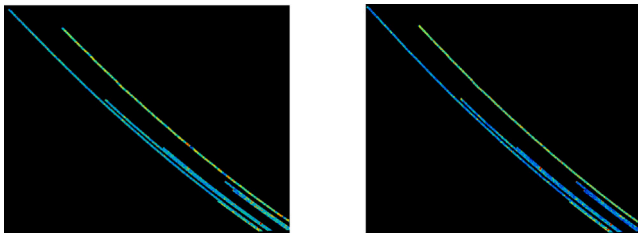
Table 2
Selection of datasets with applications to defect detection. These datasets are all fully image-based and do not contain point cloud data.

Dataset	Area	Year	Location	Labelling	Objectives	Ref.
AigleRN	urban	2008	France	cracks	crack detection	[38]
CFD	urban	2016	Beijing, China	cracks	crack detection	[39]
GAPs	urban, highway	2017–21	Germany	defects	defect detection	[42–44]
CRACK500	campus	2018	Temple University, USA	cracks	crack detection	[40]
RDD2018	urban	2018	Japan	defects[a]	defect detection	[72]
Pavement Image Dataset (PID)	highways, interstate	2018	Midwest USA	defects	defect detection	[45]
CrackTree260, CrackLS315, CRKWH100	–	2019	China	cracks	crack detection	[73]
RDD2019[b]	urban	2019	Japan	defects	defect detection	[74]
Pothole-600	–	2020	unknown	potholes	pothole detection	[41]
RDD2020	urban	2020	Japan, India, Czechia	defects	defect detection	[75]
RDD2022	urban, highways	2022	Japan, India, Czechia, Norway, USA, China	defects	defect detection	[46]
ISTD-PDS7	–	2023	China	defects	defect detection	[47]

[a] Mostly cracks.
[b] Based on RDD2018, includes synthetically generated images via Generative Adversarial Networks (GANs).

Table 3
Mobile mapping data and formats collected as part of the survey.

Data	Format	Resolution
RGB point cloud	(x, y, z) coordinates (m), intensity, RGB values	1.6 million points/s
360° Panoramic images	RGB pixels	8 MP
Pavement images	RGB pixels	5 MP
GPR	depth (m) and amplitude of reflection (mV)	every 5 cm



(a) GPR at 0cm depth. (b) GPR at 5cm depth.

Fig. 2. GPR data representing different depths for the same road section.

labelled, and (2) to develop a framework for automating the building of DTs from large quantities of mobile mapping data.

3.2. Dataset overview and preparation

From the mobile mapping image, point cloud, and GPR data provided, the CAMHighways dataset has been prepared, which consists of the following:

- 579 orthomosaic segments;
- 190 point clouds classified via semantic segmentation, and the pavement meshes generated from them;
- 68,027 pavement images, of which 17,562 were manually inspected and annotated with defects;
- 12 defect shapefiles spanning all surveyed sections, containing 69,297 labelled defects spanning 10 and 12 categories;
- 292 semi-automatically 3D generated and textured signs;
- 1719 automatically generated 3D furniture meshes

Table 4
Summary of the contents of the CAMHighways dataset. For information on the length of each of the 12 surveyed road, see Fig. 34.

Data	Number	Format	Notes
Classified point clouds	190	.las	each segment spans approx. 200 m across the 12 surveyed roads (42.8km)
Pavement meshes	190	.obj, .fbx, .ply	corresponding to each point cloud
Textured 3D signs	292	.fbx	including sign images extracted from panoramic imagery and textures
3D furniture meshes	1719	.ply	including 967 barriers, 451 traffic signs, and 301 lamp posts.
Pavement orthomosaics	579	.tif	each segment spans 50 to 200 m across the 12 surveyed roads
Merged GPR point clouds	96	.ply	see Section 5.2 for information on which roads included GPR surveys
Pavement images	68,027	.jpg	including 17,562 inspected and annotated with defects
Location reference files	12	.csv	one per set of pavement images, across all 12 surveyed roads
Defect shapefiles	12	.shp	containing 69,297 labelled defects
Annotation files	2	.json	for the A12 Area 6 Mountnessing and A14 EB-WB J47A (Woolpit) surveys

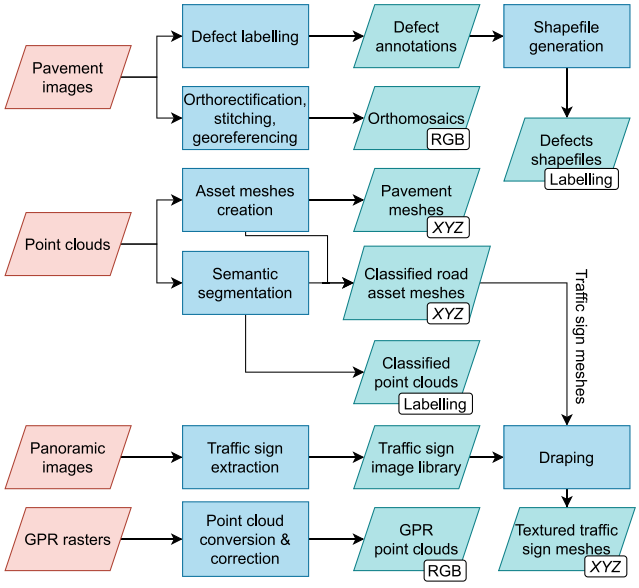


Fig. 3. General workflow for the data preparation process. Inputs from mobile mapping surveys are in red. Relevant outputs (in green) have an added tag indicating in which phase of the preparation process they were created, namely XYZ data preparation (Section 4), RGB data preparation (Section 5, and labelling (Section 6).

- Sets of 21 to 25 merged GPR point clouds for four surveyed road sections.

A summary of the contents of CAMHighways is also available in Table 4. In the preparation process, we distinguish two types of mobile mapping data. Anything generated from colourised point clouds, such as pavement and furniture meshes, is referred to as XYZ data, while orthomosaics and GPR point clouds, generated from pavement images and rasters, are referred to as RGB data. Fig. 3 provides a general overview of the dataset preparation workflow from the mobile mapping data received.

All outputs are georeferenced according to the British National Grid and can be imported into any GIS software. The prepared dataset can also be integrated into game engines such as Unreal Engine, as illustrated in Fig. 4, as a DT that enhances road inspection and maintenance but can also have other applications such as semantic scene understanding and autonomous driving. Section 8 provides a more comprehensive overview of use cases for the dataset.

3.3. Dataset structure

The CAMHighways dataset can be accessed by following this link: https://drf.eng.cam.ac.uk/research/camhighways-dataset, under a CC-BY 4.0 licence. It is made up of six folders, containing labelled point clouds, pavement orthomosaics, 3D meshes (including pavement, road furniture, and road sign meshes), defect labels, GPR data, and pavement images, respectively, and organised in the structure presented in Listings 1: and 2:.

Fig. 4. Integration of prepared CAMHighways data in UE5 with the Chaos Vehicle template. The pavement orthomosaic is draped onto the pavement mesh, and examples of road furniture, such as lamp posts and 3D-generated textured signs, are visible. The vegetation is included in the form of point clouds.

The processes for preparing the contents of the `03-Meshes` folder, i.e., pavement, roadside furniture and traffic sign 3D meshes, are summarised in Section 4. Similarly, the outputs from Section 5, namely the pavement orthomosaics, and GPR point clouds, are stored in the `02-Orthomosaics`, `05-GPR` folders, respectively. The outputs from Section 6 are available in the `01-PointClouds` folder, containing semantically segmented point clouds, and the `04-Labels` folder, containing defect shapefiles and image annotations. Additionally, pavement images for all surveyed sections are provided in `06-PavementImages`.

```
+---01-PointClouds
|   +---PointCloud-A11EBWBRedLodgetoBartonMills
|   [...]
|   \---PointCloud-
    A14EBWBJ47AWoolpittoHaugleyBridge
+---02-Orthomosaics
|   +---Orthomosaic-A11EBWBRedLodgetoBartonMills
|   [...]
|   \---Orthomosaic-
    A14EBWBJ47AWoolpittoHaugleyBridge
+---03-Meshes
|   +---01-RoadSurface
|   |   +---RoadMesh-
    A11EBWBRedLodgetoBartonMills
|   |   [...]
|   |   \---RoadMesh-
    A14EBWBJ47AWoolpittoHaugleyBridge
|   +---02-RoadFurniture
|   |   +---FurnitureMesh-
    A11EBWBRedLodgetoBartonMills
|   |   [...]
```

```
|   |   \---FurnitureMesh-
    A14EBWBJ47AWoolpittoHaugleyBridge
|   \---03-RoadSigns
|       +---SignMesh-
    A11EBWBRedLodgetoBartonMills
|       [...]
|       \---SignMesh-
    A14EBWBJ47AWoolpittoHaugleyBridge
```

Listing 1: Dataset structure for point clouds, orthomosaics, and meshes.

```
+---04-Labels
|   +---01-DefectShapefiles
|   |   +---DefectLabels-
    A11EBWBRedLodgetoBartonMills
|   |   [...]
|   |   \---DefectLabels-
    A14EBWBJ47AWoolpittoHaugleyBridge
|   \---02-PavementImageLabels
|       +---PavementImageLabels-
    A12Area6MountnessingHuttonBrentwood
|       |   \---annotations
|       \---PavementImageLabels-
    A14EBWBJ47AWoolpittoHaugleyBridge
|           \---annotations
+---05-GPR
|   +---GPR-A11EBWBSpoonerstoTuttles
|   +---GPR-A12EBWBMarksTeytoStanway
|   +---GPR-
    A12MountnessingandJ13toJ14Margarettingbypass
|   \---GPR-A14EBWBJ47AWoolpittoHaugleyBridge
\---06-PavementImages
    +---PavementImages-
    A11EBWBRedLodgetoBartonMills
    [...]
    \---PavementImages-
    A14EBWBJ47AWoolpittoHaugleyBridge
```

Listing 2: Dataset structure for defect labels and annotations, GPR data, and pavement images.

3.4. Software review

We highlight state-of-the-art software tested when identifying tools to process XYZ data, i.e., generating meshes from mobile mapping point cloud data (Section 4), as well as RGB data, i.e., building orthomosaics and GPR point clouds (Section 5), and labelling and classification (Section 6).

It is essential to consider various factors, such as vehicle speed, vibrations, weather conditions, and occlusions, as they can affect the performance of mobile mapping. Obstacles like vehicles, tall buildings, and dense vegetation can obstruct the LiDAR's line of sight, leading to reduced data quality in certain situations. Consequently, some advanced tools are better suited for photogrammetry and static LiDAR scanning rather than mobile mapping. Therefore, the review below concentrates on a curated list of software specifically investigated with the goal of meeting the requirements of this dataset. The results were put together following the assessment conducted during February–June 2023, and might not include software features added after those dates.

The complete list of software packages tested for dataset preparation includes: Trimble Business Centre (TBC) [76], TopoDOT [77], Bentley Orbit3DM [78], Agisoft Metashape [79,80], Autodesk Recap [81], Autodesk Navisworks [82], Esri ArcGIS [83], Esri CityEngine [84], QGIS [85], Unreal Engine 5.3 [86], Unity3D [87,88], 3DSurvey [89] and Cloud Compare [90].

The following paragraphs summarise the final software selection used for dataset preparation:

TBC was initially used for dataset pre-processing and serves as the tool for raw dataset assessment, planning, and investigation. State-of-the-art feature extraction tools in TopoDOT, Orbit3DM, and 3DSurvey were evaluated. However, TopoDOT and Orbit3DM require the original TBC mobile mapping project files to align images with the point cloud. Since these project files were not provided with the dataset, Agisoft Metashape was chosen as the most suitable tool for point cloud and image alignment and processing. Due to limitations with academic software licenses, TopoDOT and 3DSurvey were not explored further.

Navisworks was used to determine the precise geographical coordinates of traffic sign mesh insertion points. Its comprehensive point cloud navigation toolset enables bulk export of selected point coordinate tables. Before using Navisworks, point clouds are converted to the Autodesk interoperable format (`.rcp`) using Recap.

QGIS and ArcGIS were utilised throughout the process to visualise georeferenced data, such as images, point clouds and shapefiles. QGIS's Python integration through GDAL was employed to create orthomosaics from pavement images and generate labelled defect shapefiles. ArcGIS was used for georeferencing and correcting orthomosaics with ground control points selected from the road pavement and point cloud. CityEngine is employed to texture the pavement mesh using orthomosaics.

CloudCompare was used to manipulate point clouds, perform segmentation and classification, and generate point clouds from GPR data. While CloudCompare supports mesh generation, additional improvements using Python 3D libraries are needed for better results with the provided data. Thus, pavement mesh generation was carried out in Agisoft Metashape, while road furniture items were generated using Python libraries.

Due to sparse point cloud data for small objects, traffic sign textured meshes were produced using a semi-automated custom pipeline with Unreal Engine 5's geometry scripting. Finally, Unreal Engine and Unity3D were both used for visualising and simulating the prepared data.

4. Processing the XYZ data

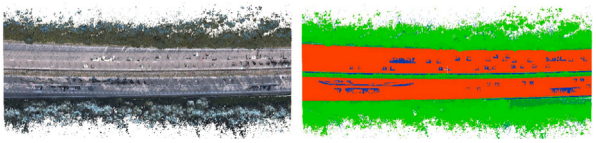
In this section, we outline the process of generating various 3D meshes, such as pavement, sign, median barrier, and lamp post meshes, from a segmented point cloud. These are referred to as XYZ data.

The point cloud segmentation (detailed in Section 6.1) enables the building of a high-resolution mesh of various elements of the point cloud, such as the road pavement, road signs, the median barrier, and lamp posts. As a result, different parts of the point cloud mesh are assembled to reconstruct the full environment. The advantage of building meshes of different parts of the point cloud is that it gives a clear distinction between different assets at a low computational cost.

4.1. The pavement mesh

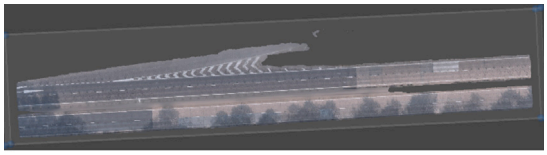
The existing classification of the point cloud done by KOREC, visible in Fig. 5, is used to segment the point cloud into different regions. This allows us to extract the road pavement point cloud for processing in Agisoft Metashape software to build a 3D mesh of the road pavement.

In order to build the polygonal mesh of the pavement based on the point cloud, we use the classification of the point cloud done by KOREC to segment the road pavement. Metashape is used to reconstruct a polygonal mesh model based on the point cloud information (dense point cloud). Using a dense point cloud results in a longer processing time but generates a high-quality output mesh. The maximum number of polygons in the final mesh can be processed with different values: High, Medium, and Low. In dense cloud-based reconstruction, these calculations are determined by the number of points within the source dense point cloud, with the ratios being 1/5, 1/15, and 1/45, respectively. Metashape allows users to specify the desired number of

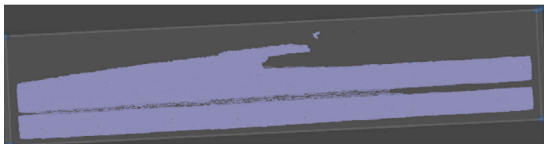


(a) Point cloud. (b) Existing segmentation (by KOREC).

Fig. 5. Examples of a point cloud and its existing segmentation. The road pavement point cloud (in red) is extracted to generate a pavement mesh.

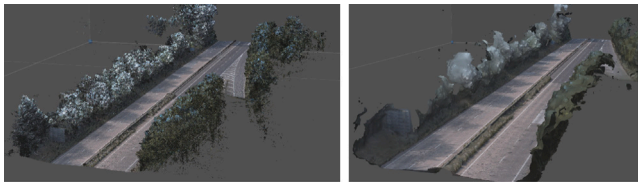


(a) Model shaded: 3D model with vertices coloured with interpolated colours



(b) Model wireframe

Fig. 6. Pavement mesh.



(a) (b)

Fig. 7. (a) Dense point cloud (b) Mesh of the point cloud.

polygons for the final mesh by adjusting the custom value of the face count parameter. Fig. 6 shows two models of the generated mesh.

Fig. 7(a) and (b) provide an example showing the full point cloud mesh constructed from a dense point cloud. The mesh built from the point cloud road pavement and road asset is saved in three different formats, namely `.obj`, `.fbx`, and `.ply` for various uses. In Section 5, we outline the process of improving the quality of the mesh's texture by employing pavement images.

The pavement meshes yield an average number of faces (i.e., triangles) of 2,869,210 per instance. Additionally, they are not watertight and have an average Euler-Poincaré (EP) characteristic [91], defined in topology as the sum of the number of vertices (V), the number of triangles (F) and the negative number of edges (E), i.e., $V + F - E$, approximately equal to 1, which is to be expected for a flat plane, which the pavement meshes resemble.

4.2. Sign meshes

We investigate three methods of generating sign meshes, both manual and automatic. We also explore how to texture these meshes using panoramic images.

4.2.1. Sign generation in Python using Open3D

We create an automatic pipeline for meshing the road furniture, including traffic signs, from point clouds. The output meshes are in

Fig. 8. Point clouds and corresponding mesh examples that capture geometry and colours from points (*xyz+rgb*) for traffic signs. Two rectangular signs and one circular sign.

Fig. 9. Failure cases for sign meshes.

`.ply` format. As the dataset does not have instance labels, we use DBSCAN (Density-Based Spatial Clustering of Applications with Noise) clustering to find instances within class labels as this method can robustly group points that are close to each other and can filter out outlier points laying separately in lower-density regions without the need for specifying the number of clusters (i.e., object instances in this case) in advance [92]. DBSCAN was run with the distance parameter *epsilon* equal to 0.7 and *minimum points* per cluster equal to 100, as found via grid search hyper-parameter tuning. Following the work of Davletshina et al. [93], we use Poisson surface reconstruction [94] for meshing because it can capture the geometry of the objects to the level of the point density by using each point in a point cloud as a node (a vertex) in triangle meshes, and it also does not introduce additional artefacts [95]. We employ the Python library implementation Open3D [95]. Due to the relatively noisy nature of the points clouds, the meshing is adjusted by removing the low support vertices, i.e., the vertices that were created in the low point density regions, since they are most likely to introduce inaccuracies to shapes due to insufficient points and noise; as well as by smoothing with average filtering [93]. We use a filtering threshold equal to 0.05 and smoothing iterations equal to 25, as found by hyper-parameter tuning with grid search. Examples can be seen in Fig. 8. As we can see, this meshing procedure can capture the geometry of the objects expressed via points. However, we note that while rectangular signs are more accurately captured via meshing, the parameters we chose are not the most optimal for traffic signs featuring circular shapes. Indeed, we can see how shapes were extrapolated near the pole and sign joint on the third example of in Fig. 8. Furthermore, the selected hyperparameters fail to capture geometry well in the areas of point clouds where the point density is lower (e.g., Fig. 9). Therefore, creating additional labels to separate these sign classes and densifying the point clouds should be considered to allow improved results.

The sign meshes produced this way are not watertight, and have on average 63,838 faces per instance.

4.2.2. Sign generation in unreal engine

In the dataset, 292 permanent traffic signs have been identified. The exclusion of temporary signs from the dataset preparation stems from the intricate and labour-intensive nature of manually extracting point cloud clusters associated with them. Due to the mobile mapping data collection method, the density of the point cloud data and the overlap of the images are not sufficient to generate models with a high level of detail. The sign meshes produced in Metashape significantly lack detail and contain gaps in the mesh, and as a result in the texture as well. Therefore, a traditional 3D modelling approach with some automation is tested for texturing 3D signs. First, we needed to identify images that could be used for texturing 3D models. Then, the images are converted to textures, and finally they are applied to the mesh.

The full workflow for generating textured sign meshes in UE5 is available in Fig. 10. We describe the steps to follow next.

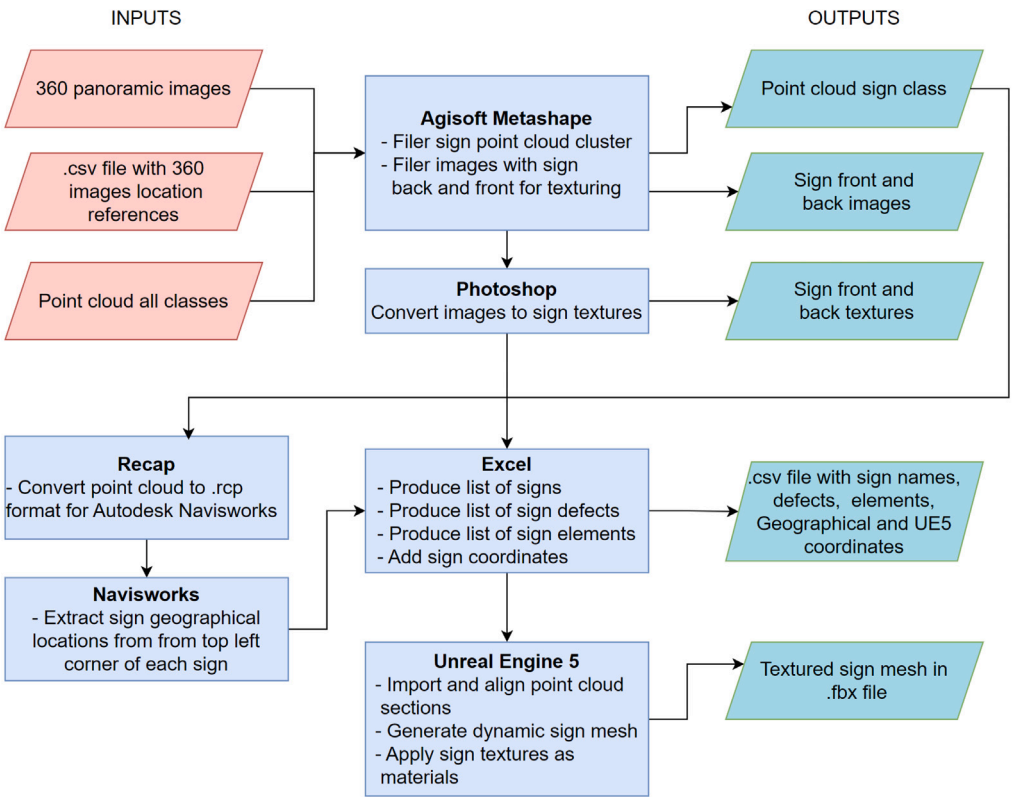


Fig. 10. Workflow for generating textured sign meshes.

Step 1. Extracting point cloud clusters and corresponding traffic sign images. Firstly, we use Agisoft Metashape Professional [79] to filter the relevant classifications of point clouds. Secondly, georeferenced panoramic images are used to extract sign images. Then, using Photoshop [96] tools, the images are cropped and converted to 3D model textures to fit in the predefined texture maps.

Step 2. Creating the table of sign features and defects from the collected images. Each of the nine road sections surveyed in the dataset contains a table of sign features and defects. The table contains (1) a sign names column with signs numbered from eastbound (EB) to westbound (WB) direction from the point cloud, excluding advertisement signs and temporary road signs; (2) sign features: number of posts in the sign, types of poles (rectangular, circular and triangular bases), number of fixings rows and clips; (3) defects visible from images, including vegetation covering the back or front of a sign, graffiti, dirt, missing sign board parts, posts rust, obvious tilt in signposts, and distortion in the sign board. Sign tables also contain geographical coordinates as well as relative coordinates.

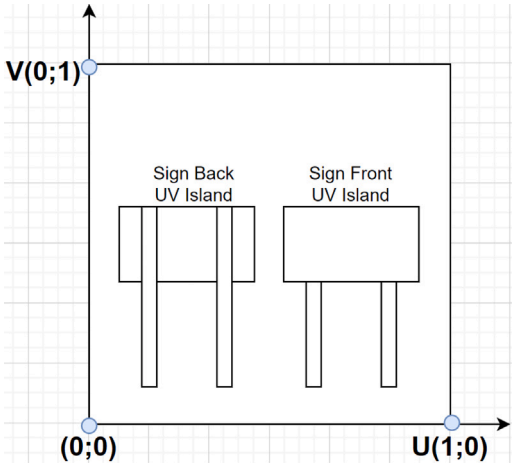


Fig. 11. Idealistic diagram presenting the concept of a texture map UV islands for a traffic sign.

4.2.3. Deciding on a method for sign texturing

The *UV islands* is a term commonly used to describe flattened elements of a 3D object mapped in UV space (texture space). In Fig. 11, the front and back of the sign are split into two UV islands. The texture map is generated randomly from the box projection method.

As each 3D object randomly produces its own UV map, this presents a significant challenge in automating the texturing process or reapplying texture with new data. Fig. 12 illustrates the above-described challenge.

The 18 signs generated through this method can be found in the A12 1 Mountnessing Section. The rest of the signs in the dataset are generated following a second method, described next.

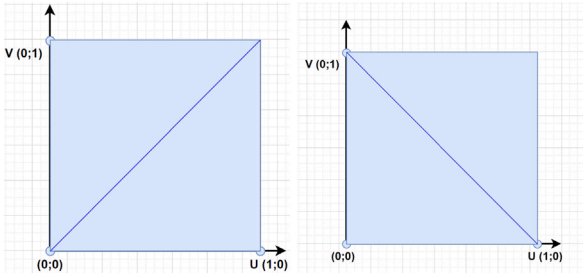
The second method uses a dynamic mesh in UE5 that has separate material channels for the front and back of a signboard, fixings, clips, posts and posts rust. The front and board signs have a predefined UV map, which maximises UV map space and has a 1×1 relationship, with recommended values 512×512, 1024×1024 pixels. The dynamic mesh is generated using dimensions from a sign point cloud cluster and its image features. Fig. 13 illustrates the second texturing method.

Step 3. Image to texture in Adobe Photoshop. In some cases, the quality of images is very low, and if defects are not found on signs, the textures are extracted from the National Highways sign database available in the traffic signs manual [97]. For each sign image, a square texture of 512×512 or 1024×1024 pixels is used. When signs have irregular shapes (e.g., circle, triangle) or some parts of the sign are missing, the background pixels are removed to create an alpha or transparency channel.

(a) Sign 1 UV map (b) Sign 1 Texture (c) Sign 1 Textured model

(d) Sign 2 UV map (e) Sign 2 Texture (f) Sign 2 Textured model

Fig. 12. Example process and the result of texturing a traffic sign mesh generated from point clouds following the first approach.



(a) Sign face UV map in material channel 1 (b) Sign back UV map in material channel 2

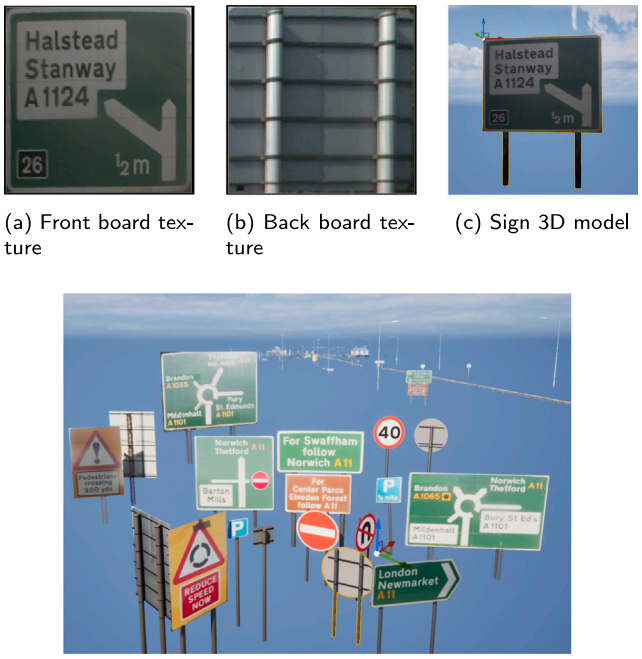
Fig. 13. Dynamic mesh UV map in UE5 for the second texturing approach.

Table 5
Material slots.

Slot number	Material
0	sign face material/texture
1	sign back material/texture
2	fixings material
3	posts material
4	clips material
5	posts rust material (if present)

Step 4. Generating a mesh in UE5 from the point cloud & sign data table using a dynamic mesh script. The dynamic script developed for sign mesh generation uses hybrid data taken from images, image data tables, and point clouds. The sign materials are added in the slots as described in Table 5. After this step, the dynamic mesh is converted into a static mesh with a pivot point (insertion point) at the top left front of the mesh. Finally, the mesh is exported in `.fbx` format. Fig. 14 illustrates the output of this method of sign generation.

Step 5. Extracting sign location coordinates. In the final step, the extracted point cloud clusters are exported to Autodesk Recap to be converted into `.rcp` and then taken to Navisworks Manage, and the coordinates of each sign point cloud cluster are extracted from the left top face point of each cluster the same way as generated sign mesh insertion points. The data table provided for each section shows information about the sign locations in both UE5 and British National Grid coordinates. This data table can be used to insert sign meshes



(a) Front board texture (b) Back board texture (c) Sign 3D model

(d) A11 Red Lodge section signs example

Fig. 14. Examples of signs generated using a dynamic mesh script.

into GIS software such as ArcGIS, iTWin or Navisworks Manage. For integration into UE5 or Unity, see Section 7.

4.2.4. Comparing methods for sign generation

The script was developed to automate mesh generation and texturing in this dataset. With the help of this script, several steps have been automated, and it took, on average, 10 min to generate each of the sign meshes, create, and apply textures. The second advantage is full automation feasibility. With further script development, the remaining steps of the mesh generation workflow will be automated, as the texturing process is uniform and repeatable. A third advantage is 2D detail preservation. No pixels are lost in the process of fitting images to the texture (UV) map, as the predefined UV map maximises the entire UV space, and each sign element group has its own material slot. The fourth advantage is a reduced triangle count. Only a relevant level of detail is created using a dynamic mesh script. Each sign generated directly from the point cloud results in 150,000 triangles on average –, which is approximately equal to the size of a medium game level made with optimised meshes –. In contrast, the dynamic script mesh is generated using fewer than 1000 triangles. Finally, as a part of the dynamic script, sign features and parameters are extracted, including board width, height, post height, and others, as well as parameters of basic defects, posts tilt or board tilt. These semantic data parameters could be utilised for decision-making in the future.

These two manual methods return higher-quality meshes but are slower and more labour-intensive than the fully automated method discussed in Section 4.2.1, in which a mesh is generated directly from the point cloud. Despite several gaps due to point cloud density, this automated method can provide additional information on object deformation and post-tilt.

Comparing meshes generated by each method can provide insights and help determine how the object has changed over time. Indeed, the pool of research work has focused on improving point cloud meshing through manually generated models as ground truths for better surface and edge detection, and gap elimination.

4.2.5. Sign defects information

The traffic sign library for each road section of the dataset contains a data table with physical defects for road signs. These include defects such as tilt, deformation, graffiti, missing elements, post-rust, and vegetation cover. While traffic sign reflectivity parameters were not assessed as part of the data preparation process, some point cloud data may have a reflectance attribute which provides details about the amount of laser light reflected back to the sensor from a target object. Additionally, some research has been done on assessing the reflectivity of traffic signs using LIDAR [98].

4.2.6. Traffic signs mesh quality

These dataset meshes are designed primarily for use with BIM applications, digital road inspections, computer vision algorithms, and synthetic dataset creation (see Section 8 for detailed use cases). Traffic sign meshes are also applicable for gaming, animation, and rendering, though further automated and semi-automated enhancements are required for high-detail close-up scenes or animation use cases.

Recalculated normals ensure consistent shading and lighting effects, critical for both computer vision processing and realistic rendering. Uniform UV mapping enables easy texture editing creating the ability to insert additional 2D defect information for synthetic datasets. UV channel 0 is reserved for texture mapping, additionally lightmap UVs could be autogenerated using standard scripts and methods in game engines or modelling applications. Mesh decimation techniques have been employed to achieve low polygon counts without compromising the geometric integrity of traffic signs. These methods ensure efficient rendering and processing, particularly for applications requiring real-time visualisation, such as VR or AR environments. In terms of tessellation, lightweight polygon counts provide a foundation for further tessellation to introduce deformations or simulate defects. All sub-elements are divided by five material channels (front, board, back board, fixings, poles and clippings). This provides an additional level of flexibility for editing tessellation by selecting a group of sub-elements by material channel through code or using the 3D modelling applications. The low-polygon count is also suitable for use as convex collision meshes in gaming and simulation contexts. Convex collision meshes provide accurate collision physics if other virtual objects are set to interact with them. Moreover, the models are designed to be compatible with BIM and IFC standards, offering a sufficient level of detail for road inspections. This includes features like the number of poles, fixings and clips as separate sub-elements. The metadata is separately provided in `.csv` files.

4.3. Barriers and lamp posts

The procedure for meshing barriers and lamp posts is analogous to that described for traffic signs in Section 4.2.1. However, the hyperparameters we found to be optimal for more accurate shape representation in the case of barriers are as follows: the low-support vertices filtering threshold is 0.3, and the number of smoothing iterations is 5. Examples of the resulting meshes can be seen in Fig. 15. We can see that meshes are highly affected by the density of the input point clouds, with the absent meshes in the areas with low density. Fig. 15 also demonstrates an exemplar surface area coverage compared to a corresponding point cloud for qualitative evaluation. Similarly, examples for lamp posts are visualised in Fig. 16, post-processed after meshing with the filtering threshold of 0.1 and smoothing iterations of 10. Failure cases in meshing these classes also occurred, which are linked to the varied density of the input point clouds. These meshes are not watertight and have on average 38,769 faces per instance, and lamp posts 60,723 faces.

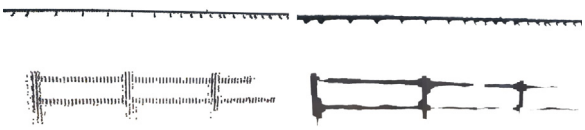


Fig. 15. Point clouds (left) and corresponding mesh examples (right) that capture geometry and colours from points (*xyz+rgb*) for barriers.

Fig. 16. Point clouds (every odd count) and corresponding mesh examples (every even count) that capture geometry and colours from points (*xyz+rgb*) for lamp posts.

4.4. Vegetation

Vegetation in point clouds was already automatically split using TBC into three classes based on the following height thresholds: Low vegetation (0-1 metr from ground points), medium vegetation (1–5 metres from ground points), and high vegetation (above 5 m). Exploring various proprietary software solutions, we attempted to determine a state-of-the-art method for creating a vegetation mesh. The study resulted in meshes that were over-detailed across all cases, posed computational challenges, and had unnecessary complexity.

Given it is possible to identify vegetation-related defects from a point cloud, generating a vegetation mesh is not deemed a priority for the purposes of this paper. However, reviewing existing literature on the subject reveals the application of mesh simplification algorithms to reduce the number of vertices while preserving essential features of the vegetation canopy [99–101]. Several studies [102] have addressed separating tree trunk points from the rest of the cluster. This investigation holds potential significance for the automation and robotics involved in roadside vegetation cutting. While our current study refrained from delving further into vegetation mesh generation, we aim to explore this task in future studies, considering its potential contributions to advancing automation and robotics in vegetation management.

5. Processing the RGB data

RGB data in this context consists of 2D data processed and added to the pavement mesh to visualise additional, high-resolution information. Here, the RGB data consists of pavement images and GPR data. In this section, we first detail the automated process of building orthomosaics from pavement images and georeferencing them onto the 3D pavement mesh. We then outline the process of integrating GPR data into the DT to display as a point cloud under the pavement mesh.

5.1. Building a pavement orthomosaic from pavement images

We wish to build a high-resolution road pavement texture from the images collected by the Trimble MX9 downwards-facing camera. The first step is to obtain bird's eye view, or *orthorectified* images of the road, and the second step is to georeference and stitch the

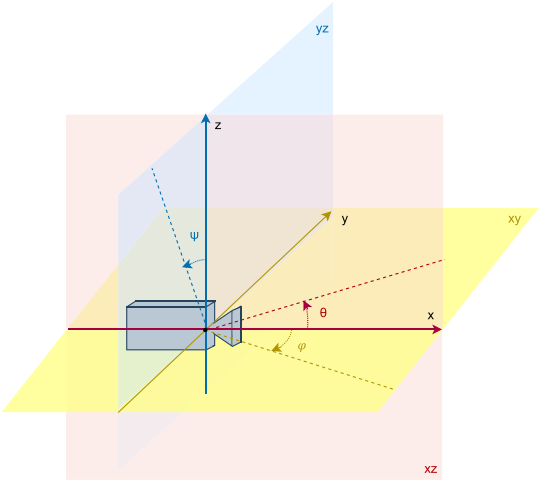


Fig. 17. Calibration of roll (ψ), pitch (θ) and heading (φ) angles of the pavement camera at its neutral position.

orthorectified images together into what is known as the *orthomosaic*. In the camera reference table, the projected easting (E_c) and northing (N_c) coordinates for the location of the camera are recorded in metres when each picture is taken, as well as its roll (ψ), pitch (θ) and heading (φ) angles, recorded in degrees and calibrated following Fig. 17.

5.1.1. Obtaining the relative pitch angle

To orthorectify an image of the road, it is important to know the *relative* pitch angle θ_r of the camera with respect to the vehicle, illustrated in Fig. 18. Let $i = 1, \dots, n$ represent the index of each picture taken by the camera, where n is the total number of pictures taken. Denote by the superscript $^{(i)}$ the parameter corresponding to each image, e.g., $(E_c^{(i)}, N_c^{(i)})$ were the easting and northing coordinates for the camera when the ith image was taken. This is as opposed to the recorded *absolute* pitch angle $\theta^{(i)}$ for each image i, which also takes into account the incline at which the vehicle is advancing. Formally, assume that the road surface lies on a plane and that the top of the vehicle (on which the camera is mounted) rests on a plane parallel to the road surface at all times. We can express the absolute pitch angle as follows,

$$\theta^{(i)} = \theta_r + \theta_v^{(i)}, \quad i = 1, \dots, n,$$

where $\theta_v^{(i)}$ is the pitch angle of the vehicle at image i. Note that by the assumption above, θ_r remains constant for every image.

The relative pitch angle θ_r is not recorded when calibrating the Trimble MX9. While the vehicle pitch angle is recorded as part of the vehicle trajectory table, entries in the trajectory table do not correspond to those of the camera reference table. They are not recorded at the same time or with the same frequency (the trajectory recording begins long before the survey area is reached and the camera begins taking pictures), so individual vehicle pitch angles cannot be attributed to each image. As a result, $\theta_v^{(i)}$ remains unknown. However, it is not required to estimate θ_r via optimisation. This is done by finding the optimal θ_r at which lane lines visible in an orthorectified picture become parallel to each other. The process is detailed Appendix B.

5.1.2. Orthorectification

Each pavement image is orthorectified in MATLAB [103] via the Automated Driving Toolbox using *Inverse Perspective Mapping* (IPM) [104], in which every pixel from the image is projected via homography onto a new 2D coordinate system on the road surface plane, illustrated as the xy-plane in Fig. 20. This new coordinate system is known as the *vehicle coordinate system*. The parameters required to perform IPM are summarised in Table 6. These include *intrinsic* parameters, such as the camera specifications, and *extrinsic* parameters, such as its orientation and the height at which it is mounted on the vehicle. Among the camera extrinsics, the mounting of the camera on the vehicle was calibrated so that the relative roll angle ψ_r (around the x-axis in Fig. 17) of the camera with respect to the vehicle is zero. The *absolute* heading angle φ is more important, as it is required to rotate images to facilitate georeferencing and stitching in subsequent steps.

Table 6
Intrinsic and extrinsic camera parameters.

Parameter	Unit	Value
pixel height/width	mm	0.00345
focal length	mm	8.5
sensor width	pixels	2464
sensor height	pixels	2056
mounting height	m	1.96696
relative roll	degrees	0
relative pitch	degrees	$\hat{\theta}_r$ (Section 5.1.1)
heading	degrees	$\varphi^{(i)}$ for each image i (from camera reference table)

5.1.3. Stitching

Once an image is orthorectified (Fig. 19), it must be georeferenced to enable stitching. This is done by generating an accompanying *world file*, which describes each orthorectified image's location, scale and rotation on a map. The six parameters needed to make up a world file are summarised in (1).

$$\begin{aligned} A &= \cos(\phi)p_x, & D &= -\sin(\phi)p_x, \\ B &= -\sin(\phi)p_y, & E &= -\cos(\phi)p_y, \\ C &= E_{ul}, & F &= N_{ul}, \end{aligned} \tag{1}$$

where (p_x, p_y) are respectively the pixel width and height in metres in the vehicle coordinate system on the projected plane, and (E_{ul}, N_{ul}) are the easting and northing (in metres) locations of the centre of the upper left pixel on the map. Since rotation of images according to the absolute heading $\varphi^{(i)}$ of each image i has already been performed as part of orthorectification, the rotation angle $\phi = 0$ in (1), which yields $A = p_x$, $E = -p_y$ and $D = B = 0$. To obtain the location of the upper left pixel, as well as the height and width of a pixel on the projected plane, these must be projected onto the vehicle coordinate system, which is equivalent to obtaining their distance from the camera in metres along the x- and y-axis in Fig. 20. First, p_x and p_y are obtained by selecting any two pixels directly diagonal to each other on the orthorectified image, projecting them onto the vehicle coordinate system, then subtracting their x and y coordinates to obtain the width and height of a pixel, respectively. Subsequently, E_{ul} and N_{ul} are calculated by setting $E_{ul} = E_c - \delta_y$ and $N_{ul} = N_c - \delta_x$, where δ_x and δ_y are the distance in metres from the camera to the projected upper left pixel in the vehicle coordinate system.

Once a world file has been generated for every image, they are divided into 50 to 200 metre long segments, then the segments are stitched together into GeoTIFFs in GDAL [105], using the mosaicing option of the `gdalwarp` utility. The resulting GeoTIFFs, illustrated in Fig. 21, follow the British National Grid coordinate system and are known as the orthomosaic.

5.1.4. Georegistering the orthomosaic onto the mesh

Georegistration of the orthomosaic onto the mesh is a process of coordinate transformation to match the GeoTIFF files (also known as *rasters*) for each of the 50–200 m road segments that make up the orthomosaic to the pavement mesh before texturing the road pavement mesh with the orthomosaic. It is an important step in mapping the processed RGB data to the original point cloud with minimal distortion. The georeferencing work consists of matching the segmented road pavement RGB point cloud to the orthomosaic. The point cloud is used

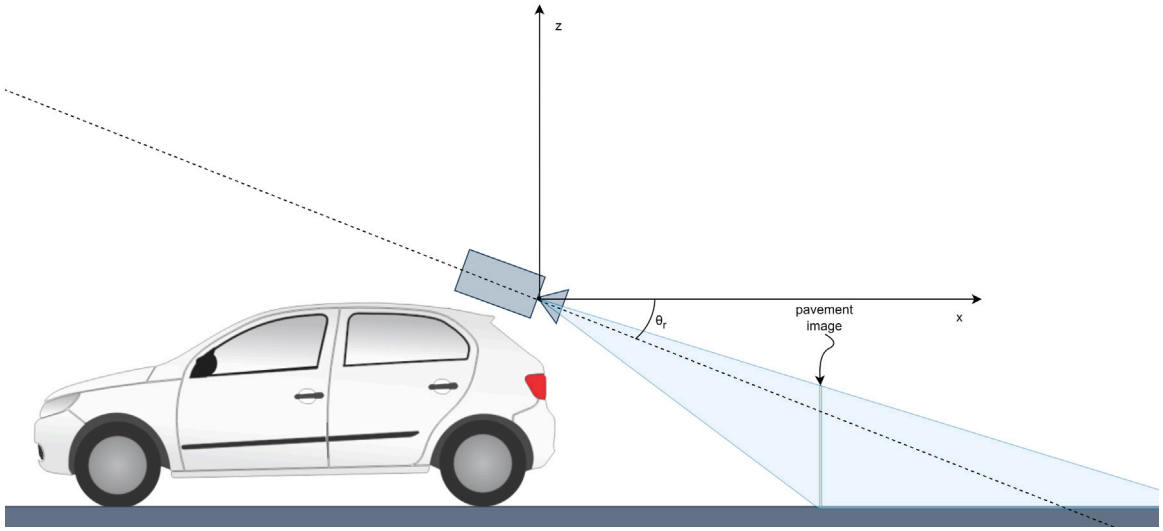


Fig. 18. Side view of the pavement camera positioning when the vehicle is horizontal, including the relative pitch angle θ_r.

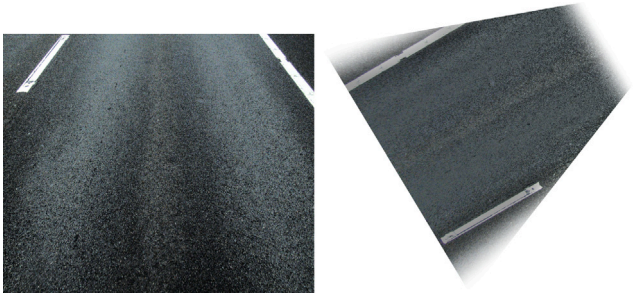


Fig. 19. Orthorectification of a pavement image. A gradient transparency layer is added to the edges to improve the stitching quality.

in order to better manually select a number of specific *ground control points* (GCPs) for reference. The features selected as GCPs from the orthomosaic GeoTIFF files and the pavement point cloud are the road studs, also known as *cat's eyes*. This is because from the data available, the cat's eyes are the most suitable feature to use in correcting a GeoTIFF file to map it onto the pavement mesh. While a Digital Terrain Model (DTM) raster of the point cloud can be built, the point cloud is favoured for GCP selection because the resolution of the DTM is not high enough for the cat's eyes to be visible on it. The GCPs are manually selected from the road pavement RGB point cloud as a reference and matched to the GeoTIFF. The number of selected GCPs depends on the distortion or complexity of the transformation that needs to be applied on each GeoTIFF and the length or size of the data. It is important to spread the GCPs over the entire raster for better registration rather than concentrating them in one area. It is also important to note that the accuracy of georeferencing data is as accurate as the data it is aligned with. This work uses the high-resolution road pavement point cloud to reference the GeoTIFF image, reducing misalignment errors. The correction is processed using Esri GIS software, including the georeferencing tool in ArcGIS Pro [83]. The software provides several polynomial transformations for data transformation. These options include 'polynomial', 'spline', 'adjust', and 'projective' to determine the correct map coordinate location for each cell in the raster. Various polynomials are used during the data processing due to the complexity of the data process, such as zero-, first-, and second-order polynomials, and sometimes adjustable.

A zero-order polynomial is used in segments where the GeoTIFF is least distorted, and up to 2–3 GCPs are used. A first-order polynomial

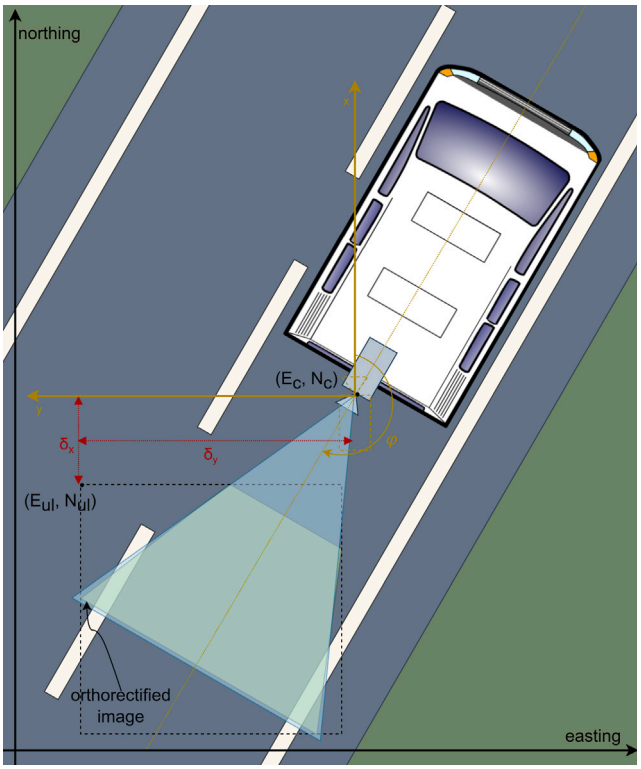


Fig. 20. Top-down view of the pavement camera positioning for a camera rotated at a heading angle φ. The x and y axes are defined when the camera is at its neutral position (i.e., before rotation), as previously illustrated in Fig. 17. The parameters δ_x and δ_y denote the distance in metres from the easting and northing of the centre of the camera (E_c, N_c) to that of the upper left pixel of the rotated orthorectified image (E_{ul}, N_{ul}) in the vehicle coordinate system.

is used when the GCPs from the GeoTIFF need to be stretched and rotated with the least distortion to exactly match each GCP to the target pavement point cloud. Generally, a minimum of 3 GCPs are needed due to the data size, and up to 8 GCPs spread over the segment can sometimes be required. Overlapping pavement point clouds and the corresponding GeoTIFF show high distortion alignment. In this case, the GCPs used to match the target of the GeoTIFF require bending and curving, and a second-order polynomial is applied to correct this high

Fig. 21. Segment of the orthomosaic in GeoTIFF format displayed in QGIS, generated through orthorectification, georeferencing, and stitching of pavement images.

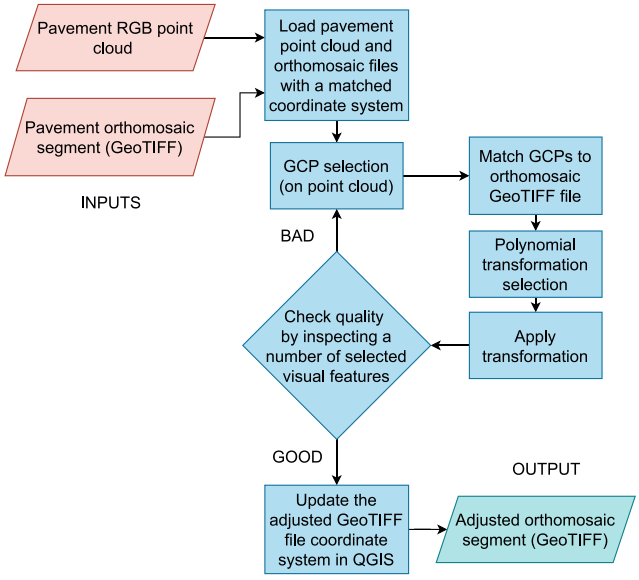


Fig. 22. Process of creating a georeferenced pavement GeoTIFF images.

distortion. This process takes computational resources depending on the data size and the number of GCPs, and it can take up to several minutes for the task to be completed. The 'adjust' transformation is used the least frequently during this data preparation because the overlap control points are less frequent in the dataset. The 'adjust' transform best preserves the original shape with a higher root mean square error (RMSE) than the polynomial transformations. It is important to note that the GCPs are picked manually and are, therefore, subject to error, so as many GCPs as possible must be selected to reduce the error. Also, we tend to use low-order polynomial transformation as much as possible to preserve the dataset's original shape. Fig. 22 shows the process for georeferencing a GeoTIFF (or raster).

The pavement point cloud and the GeoTIFF require the same coordinate system. When uploading the data into ArcGIS, the coordinate system of the point cloud needs to be adjusted for visibility on the map. The pavement point cloud from the elevation model is RGB colourised in order to use it as a reference for the correction. The features selected as GCPs are the cat's eyes, or studs, which can be clearly seen in both the point cloud and the GeoTIFF. Several GCPs are selected across the GeoTIFF orthomosaic segment to reduce the error (see Fig. 23), but it takes time to process the data. As mentioned previously, the selection of the number of GCPs and polynomial transform depends on how well the selected features match. ArcGIS provides real-time transformation which helps to make a quick decision on which polynomial transform to choose. Therefore, the quality of the correction depends on the user's

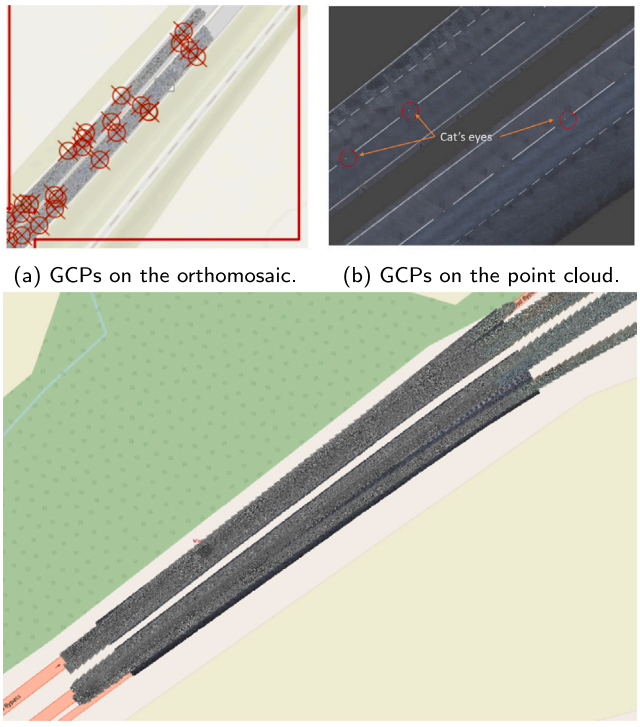


Fig. 23. Selection of cat's eyes (or studs) as GCPs to georeference the orthomosaic onto the point cloud.

needs. If the desired target is unmet, the user can increase or decrease the number of GCPs to meet their requirements. Once the results are satisfactory, the adjusted GeoTIFF coordinate system can be updated. It is an important step before using the adjusted orthomosaic to texture the pavement mesh; otherwise, the two will not align. This last step, performed in QGIS, consists of exporting the adjusted GeoTIFF file and resetting the original coordinate system.

5.1.5. Pavement mesh texture

The texture for the pavement mesh integrates georeferenced orthomosaic information into a textured mesh that includes elevation information. ArcGIS Pro is used to build the DTM from the pavement point cloud. The process outputs the DTM as meshes, which can be textured with the corresponding GeoTIFF images. The pavement mesh texture in `.obj` and `.ply` formats can be generated using 3Dsurvey [89] software. This software allows viewing the draped texture but not exporting it for further analysis. Software such as ArcGIS and CityEngine allow for texturing the DTM pavement mesh with the corresponding GeoTIFF image. The software also allows the work to be exported into different formats, including `.obj`, `.fbx`, and UE5 format for further use.

5.2. Ground penetrating radar

As part of the 3D GPR data received, a folder for each road section is named based on each surveyed road section, namely A11 EB-WB Spooners to Tuttles, A12 EB-WB Marks Tey to Stanway, A12 Moutnessing Hutton Brentwood, and A14 EB-WB J47A (Woolpit) to Haughley Bridge. Each folder includes GeoTIFF or raster images depicting GPR data for depths ranging from 0 to 120 cm at 5 cm depth intervals. The information embedded in each supplied `.tiff` image includes the coordinate referencing system, the geographical extent coordinates, the dimensions and origin coordinates of the image, and the pixel size. In addition, each GeoTIFF image contains four bands. The first three

bands represent RGB values, and the fourth represents the alpha or transparency layer. The GPR survey solely covers the pavement structure while excluding off-pavement areas; therefore, off-pavement pixels show a zero value for all four bands. The different pixel value distributions in each RGB band represent the change in the reflection intensity of the GPR signal collected along the road pavement, reflecting the differences in electromagnetic properties along the pavement, which, with further analysis, can be related to the pavement's mechanical and structural properties.

5.2.1. GPR data preparation

The objective of the GPR data preparation process is to enable the seamless integration of the GPR data with other received sensing data modalities, such as RGB images, point clouds, etc., presenting all these modalities on a unified platform. Considering this, the four main steps to achieve the stated objective are described next.

Step 1: Sorting GPR data based on depth. For each road section folder, the relevant GeoTIFF images with the same depths from the surface are sorted and copied into folders specifically created to contain images of only one specific depth. This process is automated by using a python script that identifies each GeoTIFF image's depth based on the file name. Thereafter, the script creates different folders for each depth, which contain the pertinent GeoTIFF files.

Step 2: Converting rasters to point clouds. In order to convert the GPR data to a compatible modality which can be visualised and integrated with other already available modalities, i.e., point cloud, mesh, and orthomosaic, it is necessary first to convert the GPR data to point cloud data. This is achieved by using CloudCompare through its Python wrapper `CloudComPy`.

CloudCompare converts the `.tiff` image to point cloud by initiating a point cloud grid with its x, y, z coordinates. The distribution of the grid is based on the boundary pixels centres. Therefore, the intensity of the GPR point cloud depends on the resolution of the original GeoTIFF image, assuming that all pixels have the same size. The grid size, i.e., the generated point cloud, is smaller than the actual GeoTIFF image size since the points generated are located at the centre of each pixel.

Step 3: Correction of GPR point cloud data elevation values. Fig. 24 displays the pavement surface GPR point cloud recorded at a depth of 0cm, also denoted zero-depth GPR point cloud, after being generated from its corresponding GeoTIFF image, together with a laser scanner (or LiDAR) point cloud. We note that the pavement surface GPR point cloud shows a higher elevation than the LiDAR point cloud. After inspecting the z coordinate values for the imported GPR point cloud, we observe that all the point clouds on the pavement record a value of 255, while the point clouds for the off-pavement, shown as blue-coloured point clouds in Fig. 24, show a value of 0. This latter observation indicates that, when converting the GPR GeoTIFF files to point clouds through the above-mentioned procedure, the alpha band, which represents transparency and therefore only takes values 0 and 255, is erroneously used to provide the z coordinate values for the produced GPR point cloud. Therefore, the following steps must be followed to correctly register the GPR point clouds with the laser scanner point cloud. First, we must filter out the off-pavement GPR point clouds. Once the GPR point cloud is imported to CloudCompare, the Cloth Simulation Filter (CSF) is used in the Relief scenes mode to obtain only the on-pavement GPR point cloud. This step is straightforwardly achievable given the elevation of all the off-pavement points is 0, whereas the on-pavement GPR point cloud elevations have values of 255. After this filtering step, the GPR point cloud contains just the on-pavement points.

The elevation values of the laser scanner point cloud are employed to correct the GPR point cloud elevation values for the on-pavement point cloud. A Python script running a K-D tree algorithm for nearest-neighbour search [106] is used to obtain the nearest distance between each GPR point cloud and the laser scanner point cloud. Then, the algorithm captures the elevation of the nearest laser scanner point cloud and encodes it as a new elevation value for the GPR point clouds. For this script, both the laser scanner and GPR point clouds are first converted to `.ply` files before they are imported into the Python script. Estimating the nearest distance is based on both point clouds' x and y coordinates. Fig. 25(a) shows the results of the correction process. Fig. 25(b) shows the nearest distance values on the y-axis against the point indices on the x-axis.

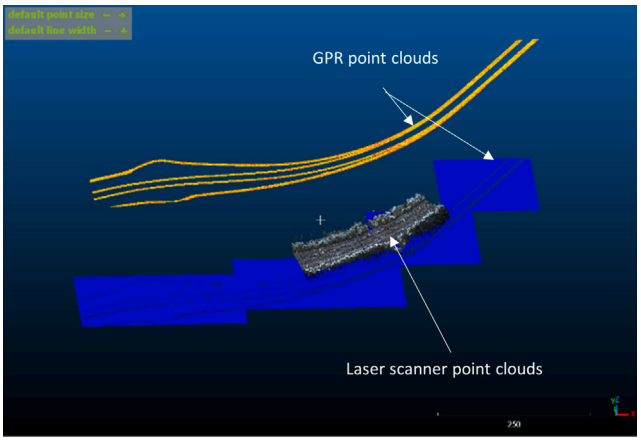


Fig. 24. GPR point clouds and laser scanner (LiDAR) point clouds for the same road section, both imported into CloudCompare. The elevation values for the GPR point cloud are encoded erroneously after conversion from GeoTIFFs into point clouds and must be corrected.

Step 4: Merging corrected GPR point clouds. Note that even though the nearest distance for the majority of the points has a value that is lower than 1 mm, some points –, especially those at higher index numbers, i.e., $> 10^6$ –, occasionally record a nearest distance of 100m. This is because, given the GPR survey was not performed at the same time as the Trimble MX9 survey, the GPR point clouds generated from their corresponding GeoTIFFs do not necessarily match the length of the laser scanner point clouds. Indeed, the laser scanner point cloud length can extend beyond that of the GPR point cloud section length, as is visible in Fig. 25(a). Therefore to avoid this limitation, all the laser scanner point clouds for the relevant road section are merged together before importing them into the K-D nearest point Python script, and similarly for the GPR point clouds.

After obtaining the surface elevation values for each zero-depth GPR point cloud, they are used to correct the GPR point clouds corresponding to other depths. This is achieved by deducting the relevant depth of each GPR point cloud from the surface elevation values. Fig. 26 shows a sample of the final corrected 3D GPR point cloud at three different depths, namely 0, 50, and 100cm, fully corrected and registered with their relevant laser scanner point cloud.

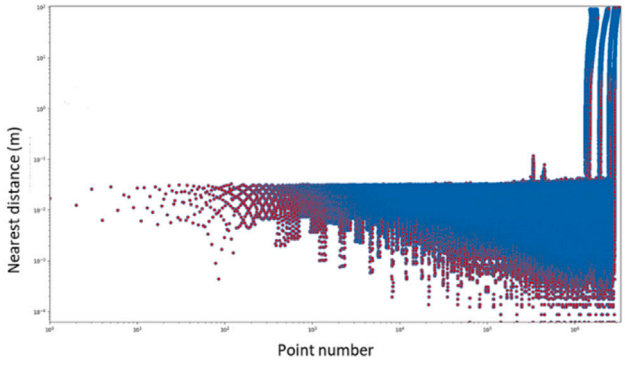
6. Labelling

In this section, we describe the manual and automated approaches employed to label the dataset. Two major types of labelling are considered: semantic labels on the point cloud and defect labels on pavement images.

6.1. Point cloud labelling

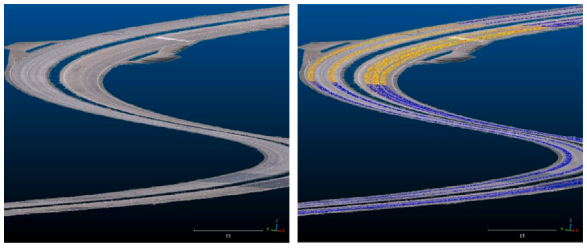
There are two stages to producing point cloud semantic labels. At first, KOREC labelled point clouds into seven classes (road furniture, vehicles on the road, non-pavement ground, low, medium and high

(a) GPR point cloud after elevation correction for the zero-depth GPR point cloud.

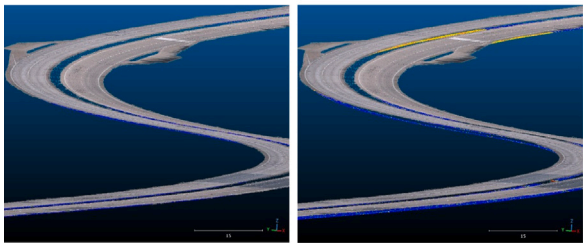


(b) K-D results for nearest points distance between the GPR and laser scanner point cloud.

Fig. 25. Illustration of the GPR point cloud post-correction. Some length mismatches between GPR and laser scanner point clouds stemming from the two different surveys are also visible in Fig. 25(a), and yield over-inflated distance values in Fig. 25(b).



(a) Laser scanner point cloud. (b) GPR point cloud at 0cm depth.

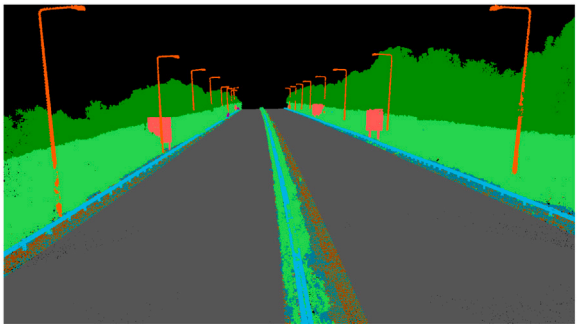


(c) GPR point cloud at 50cm depth. (d) GPR point cloud at 100cm depth.

Fig. 26. Laser scanner visualised with GPR point clouds at various depths.

vegetation, and pavement). As part of preparing this dataset, we enrich the labelling with five more classes separated from the existing labels. We provide a summary of labels in Table 7 and an example of a final labelled point cloud in Fig. 27, combining our labels with those provided by KOREC. We also provide the label IDs in Table 8.

(a) Final combined point cloud semantic segmentation on an entire point cloud viewed in CloudCompare.



(b) 3D view of a segmented point cloud in QGIS.

Fig. 27. Illustrations of point cloud segmentation viewed in various software.

Additional labels are added by manually labelling them in CloudCompare. As some labels are already provided by KOREC, we rely on them to reduce the input point clouds for labelling. We filter out the road furniture objects which are under one label and separate them into three different classes, namely barriers, lamps, and traffic signs by manually selecting the appropriate points. A similar procedure is followed in producing road markings and middle and side ground labels.

6.2. Defect labelling

The pavement images for the A12 Mountnessing Hutton Brentwood and A14 EB-WB Woolpit to Haughley Bridge surveys, depicted in Fig. 34, are selected for initial annotation. The two road sections were chosen because they were built of different materials and had different visual appearances. With a layer of overlain asphalt on the A12 Mountnessing road surface, the observed pavement distresses were more akin to asphalt roads, as opposed to the rigid concrete defects observed on the A14 Tothill. The pavement images were annotated at the level of instance segmentation to locate and classify defects of different categories, differentiate between instances and create masks for future visualisation and modelling. When establishing the annotation categories, we adopted the pavement diseases from Hadjidemetriou's defect network [107] and referred to defects outlined in the Catalogue of Road Defects [108]. The final categories included are shown in Table 9.

With the image dataset, level of details, and annotation categories confirmed, manual annotation commenced and was later supplemented by pseudo-labelling. The pavement images were first annotated manually by using the Computer Vision Annotation Tool [109]. They were then reviewed and used to train deep learning models, Mask-RCNN [110] (see Fig. 28 as an example) and YOLOv7 [111], for defect detection and segmentation. The detected instances from the deep learning models served as pseudo labels for annotators to carry out manual reviews. The reviewed annotations can be converted into a fully labelled, lightweight shapefile that matches the orthomosaic built-in Section 5.1. This method was applied to the 'A12 Area 6 Mountnessing Hutton Brentwood' and 'A14 EB-WB J47A (Woolpit) to Haughley Bridge' surveys. For the rest of the surveyed roads, pre-labelled shapefiles were provided by KOREC and manually adjusted in QGIS to fit the orthomosaics.

Table 7
Road sections and labelled classes in the point clouds. *Count* stands for the count of the point cloud files (approx. 200 m long road segments). Guide to abbreviations: **A11RLBM**: A11 EB-WB Red Lodge to Barton Mills; **The rest**: all surveyed sections of the A11, A12, and A14 except for A11 EB-WB Red Lodge to Barton Mills.

Section	Count	Labelled classes												
		barrier	car	middle ground	low veg.	medium veg.	high veg.	sign	sidewalk	road pavement	road marking	light	other	non-pavement ground
A11RLBM	20	+	+	+	+	+	+	+	+	+	+	+	+	–
The rest	190	+	+	–	+	+	+	+	–	+	–	+	+	+

Table 8
Semantic segmentation labels in the point clouds. Guide to abbreviations: **A11RLBM**: A11 EB-WB Red Lodge to Barton Mills; **The rest**: all surveyed sections of the A11, A12, and A14 except for A11 EB-WB Red Lodge to Barton Mills.

Label	Name	A11RLBM	The rest
0	Barriers	✓	✓
1	Cars on the road	✓	✓
2	Middle ground	✓	✗
2	Non-pavement ground	✗	✓
3	Low vegetation	✓	✓
4	Medium vegetation	✓	✓
5	High vegetation	✓	✓
6	Traffic signs	✓	✓
7	Sidewalk/side ground	✓	✗
8	Road pavement	✓	✗
9	Road marking	✓	✗
10	Other	✓	✓
11	Light/lamps	✓	✗
11	Road pavement (inc. label 9)	✗	✓
12	Light/lamps	✗	✓

Table 9
Annotation categories in the generated and KOREC supplied labels.

Generated label categories	KOREC label categories
- bleeding	- corner ckack
- crack_alligator	- diagonal crack
- crack_block	- failed repair
- crack_corner	- good condition repair
- crack_edge	- longitudinal crack
- crack_longitudinal	- pothole
- crack_transverse	- ravelling
- patch	- scaling
- pothole	- spalling
- ravelling	- transverse crack
- rutting	
- shoving	
- spalling	
- unknown	

6.2.1. Creating defect label shapefiles

The input consists of an image annotations `.json` file. These annotations are polygons segmenting every defect that has been classified in each image (see Fig. 28). First, the annotation files are split into one annotation file per defect category, and the following steps are repeated for each defect category. Binary mask images are generated from the annotations file, in which the polygons outlining the relevant defects take a value of 1, and everything else takes a value of 0. These masks are then orthorectified and stitched together in exactly the same way as their corresponding pavement images were in Section 5.1.2. To ensure a perfect match, the world files from the pavement images are copied over from the files required to build the pavement orthomosaic rather than generating new ones. This results in a defect mask orthomosaic, in raster (or GeoTIFF) format, consisting of segments of 50–200 m of road, much like before.

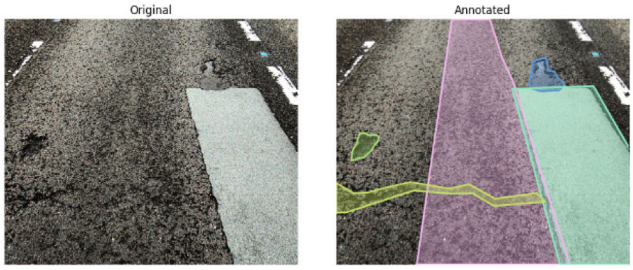


Fig. 28. Example of defect segmentation performed using Mask-RCNN on a pavement image.

While these defect mask rasters match the pavement orthomosaic, they are not a suitable format to display defect labels. Firstly, they are very heavy files, with an average of 2-5 GB per segment, and can easily be as heavy as 9 GB. Merging all segments together to create a single raster for the whole surveyed area will result in a prohibitively large raster. Secondly, merging defect mask rasters of different defect categories will result in data loss, as the rasters only contain RGB and location data and cannot discriminate between the different types of defects.

For these two reasons, the rasters are polygonised in GDAL using the `gdal_polygonize` utility, which outputs lightweight shapefiles consisting of polygons where each point corresponds to projected coordinates, outlining the shape of the various labelled defects visible on the pavement orthomosaic. In this format, each segment of the road surveyed takes up less than 1MB of storage on average, and they can be merged straightforwardly using GDAL. Since shapefiles are made to store labelled geometry, each type of defect can be labelled separately from the others. Therefore, the shapefiles can also be merged across the different defect categories. This results in a fully labelled, lightweight shapefile that covers the entire road surveyed. The workflow for creating defect label shapefiles is summarised in Fig. 29.

6.2.2. Georeferencing shapefiles onto the mesh

Once each orthomosaic segment has been georeferenced onto the mesh, as described in Section 5.1.4, the same must be applied to the corresponding shapefile. This is done for each segment in GDAL in much the same way as for the orthomosaic, and indeed using the same GCPs, previously selected from road features such as cat's eyes. This step is performed before the shapefiles are merged into a single shapefile in the final step of the workflow (Fig. 29).

7. Dataset integration

Integration of the prepared data has been tested for visualisation and DT prototyping. The integration engines explored include Unity3D, Unreal Engine 5, 3Dsurvey, ArcGIS, QGIS, and CityEngine. In this section, we outline steps for integration into some of these engines.

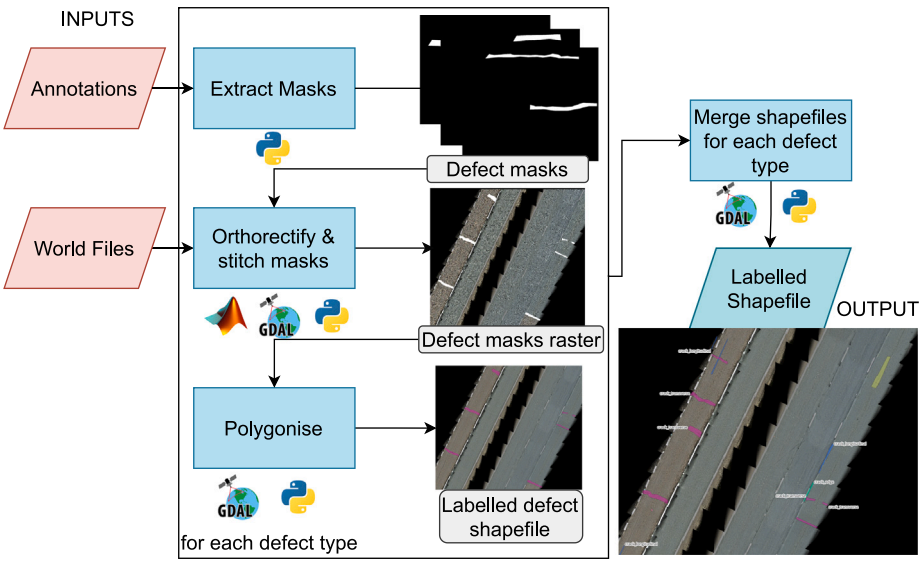
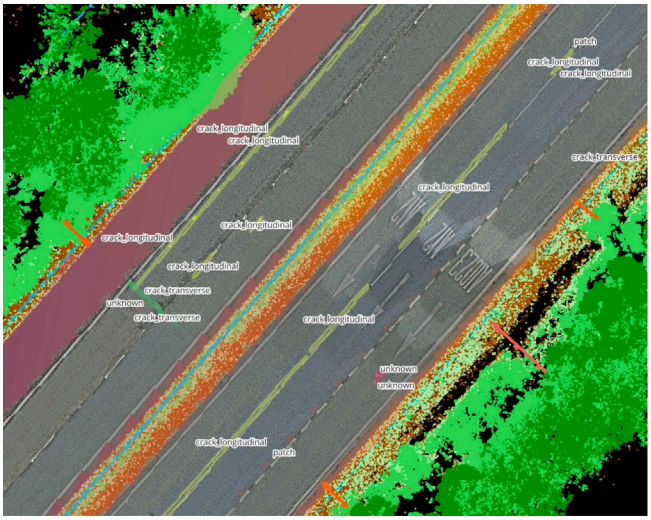
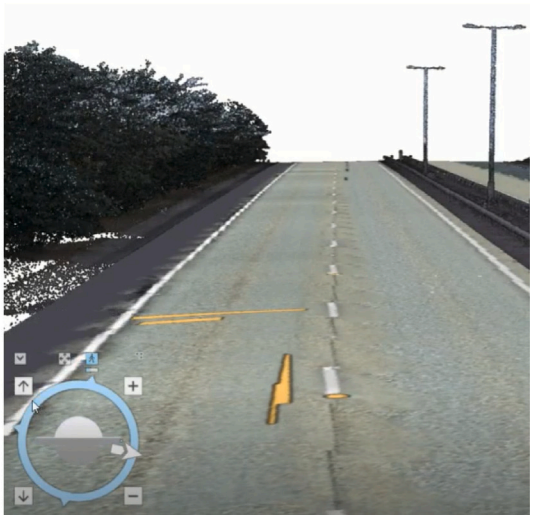


Fig. 29. Workflow for generating the label shapefiles.



(a) Orthomosaic, defect labels shapefile, pavement mesh (displaying elevation values in purple tones), and segmented point cloud integrated into QGIS. The point cloud labels include lamp posts (orange), barriers (light blue), signs (red), ground (brown), and vegetation (green).



(b) Pavement mesh (textured with the orthomosaic), defect labels shapefile, and RGB point cloud integrated into ArcGIS.

Fig. 30. Integration into GIS software.

7.1. Integration into GIS software

All data generated as part of the preparation process are georeferenced according to the same coordinate system, namely the British National Grid, and can be viewed in GIS software. An open-source example of such software is QGIS. It suffices to drag and drop the various modalities onto the map to view them integrated together. Data sources with multiple attributes can be tweaked to display those attributes following user preference. For example, the segmented point clouds can be viewed as intensities in RGB or as separate classes, as illustrated in Fig. 30(a). Alternative software, such as ArcGIS, provides similar features and can be employed for the same purposes, as is visible in Fig. 30(b).

7.2. Integration into game engines

7.2.1. Integration into unreal engine

The integration framework as shown earlier in image Fig. 4 was developed in UE5, with the Collab Viewer template and the LiDAR Point Clouds plugin. To prepare the point clouds, they are initially segmented by class using CloudCompare before being imported into UE5. Upon importing into the engine, all point clouds are positioned at the $(0, 0, 0)$ coordinate point and then aligned to each other using tools in the LiDAR mode panel. For road mesh integration, a standard `.fbx` or `.obj` import workflow can be employed, followed by manual alignment with the point cloud. As for traffic sign models, they were originally crafted within the engine. Therefore, the provided coordinates and defects table associated with the models can be used to insert

(a) Asset visibility interface

(b) GPR data-enabled view

Fig. 31. UE5 Prototype with integrated dataset elements.

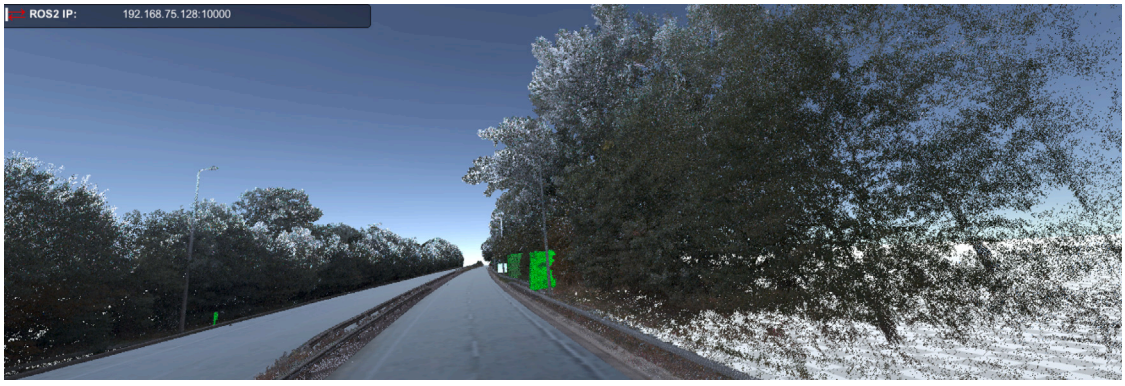


Fig. 32. Unity 3D road data integration. Road signs are highlighted in green as the camera progresses through the scene.

the models precisely at their designated coordinates. Fig. 31 illustrates some aspects of the integration of the dataset into UE5.

7.2.2. Integration into unity

The processed data road pavement, point cloud, and road sign have also been integrated into Unity 3D. This integration provides not only visual realism but also character shaping and environments. The platform offers various functionalities, including serving as an autonomous driving test ground, generating synthetic data from real-world sources, and performing machine learning for fault detection on road assets. Fig. 32 illustrates the integration of the dataset into Unity.

8. Use cases

The CAMHighways dataset is available at https://drf.eng.cam.ac.uk/research/camhighways-dataset. In this section, we provide examples of how its data and the knowledge than can be inferred from it may be used by researchers and practitioners.

8.1. DT construction

The first goal of CAMHighways is to facilitate the building of a highway DT. Such a DT ideally requires several key components, including accurate spatial data for asset representation, comprehensive condition assessment information, mechanisms for simulating scenarios, and a framework for real-time updates from sources such as weather data and pavement sensors to reflect changes over time. This paper and the CAMHighways dataset address these requirements by providing an integrated multimodal data collection that supports the development and continuous refinement of a DT. Specifically, a road DT would need high-fidelity 3D georeferenced models, defect annotations for assessing asset condition, integration of surface and subsurface data for deeper insights, and a dynamic virtual environment for simulating maintenance strategies. The CAMHighways dataset fulfils these needs through its accurate 3D meshes, annotated defect shapefiles, GPR data, and methodologies for creating interactive digital representations, which collectively enable the construction of a robust road DT framework.

The paper goes beyond simply stating compliance with DT requirements by offering pipelines and data formats essential for building a functioning DT, as well as guidance on how to automate many of the data preparation processes. The detailed asset condition data it provides supports scenario testing and real-time decision-making for maintenance. Additionally, the dataset's compatibility with GIS, BIM software, and game engines facilitates its integration into the user's preferred DT platform, making it adaptable for various DT applications such as performance monitoring, autonomous vehicle simulations, and virtual prototyping. Because of this, CAMHighways actively enables the creation of a complete digital representation that advances research and practical implementations in road infrastructure management.

8.2. Inspection and maintenance

CAMHighways is an invaluable source of data to improve road inspection and maintenance, especially for highways. First of all, all assets it provides are true to life, in that they are made from (or in some case fitted to) the original point cloud or image data, as opposed to using synthetic, stock textures or assets placed in the same geographic location, thus making it the first high-fidelity representation of the surveyed roads to date. Additionally, similarly to existing datasets, its vast collection of road pavement images, both annotated and not, can be used to train and improve defect detection algorithms. Another novelty it provides is the 3D library of road signs that can facilitate the detection of physical sign defects. The innovative high-fidelity 3D virtual environment can also be used for management, as well as repair testing and simulation for various highway assets, such as pavement,

markings, studs, signs, barriers, and vegetation. For example, Fig. 33 demonstrates a simple prototype interface for the comparison of traffic sign maintenance state data. The dynamic mesh is reusable, with textures coming from new data. Furthermore, the 3D environment created from this dataset can also assist inspectors in verifying whether the highways in question respect British Standard codes. The integration of GPR sensing with other sensing modalities for the first time can also provide insights into relationships between surface defects and GPR data. Finally, beyond offering defect annotations on pavement images, as is often the case for existing defect datasets, the defect shapefiles built from them are a new type form of defect data that can be employed to estimate pavement performance, as well as model the spatial distribution of defects for prediction and proactive maintenance purposes.

8.3. Virtualisation

The CAMHighways dataset offers many benefits related to virtualisation across various domains, such as maintenance simulation, synthetic data generation, autonomous vehicle development, virtual gaming environments, and VR prototyping. Firstly, this dataset consists of highly detailed and accurate 3D representations of road networks, built from the various novel components of CAMHighways, including high-quality pavement textures derived from orthomosaics, and true to life assets built from segmented point clouds, such as signage, road markings, and surrounding infrastructure. Therefore, it can be employed to perform in-depth inspection and provides an environment in which various repair methods can be tested via simulation. It also makes it a valuable resource for generating synthetic data used in training machine learning algorithms for object detection, classification, and path planning tasks. In the context of autonomous vehicle development and path planning, this dataset can enable developers to create realistic simulations of driving scenarios. The labelled point clouds provide a valuable source of training data for semantic segmentation algorithms. Additionally, by incorporating real-world road data, developers can test and validate their algorithms in diverse environments, improving the robustness and reliability of autonomous systems. Moreover, the ability to simulate various road conditions, traffic patterns, and signage configurations facilitates the development of safer and more efficient autonomous driving technologies. For virtual gaming environments, this dataset can provide game developers with a foundation for creating immersive and realistic gaming experiences. Accurate representations of roads and urban landscapes enhance the authenticity of virtual worlds, leading to more engaging gameplay and enhanced user experiences. Additionally, developers can leverage the dataset to dynamically generate procedurally generated environments, increasing their games' variety and replayability.

In VR prototyping, the fact that the dataset only contains assets made from the original mobile mapping data can serve as a valuable resource for creating realistic virtual environments for testing and showcasing VR applications and experiences. Developers can create immersive VR simulations for training, education, and entertainment by accurately modelling real-world road networks and infrastructure. Whether simulating driving scenarios, exploring urban landscapes, or designing interactive experiences, this dataset can provide a solid foundation for prototyping and iterating on VR concepts.

9. Conclusion

In this paper, we presented the CAMHighways dataset for building a DT that facilitates road inspection and maintenance. The paper encompassed various aspects of mobile mapping data preparation, including a comparison with existing datasets, highlighting their unique features and how we aimed to bridge gaps in the literature. Given the large quantity of survey data collected, we also developed novel methods to automate several aspects of the data preparation tasks.

When processing the XYZ data in Section 4, namely generating meshes from point clouds, we delved into the methods employed for creating meshes and textures for pavements, traffic signs, and median barriers. The pavement mesh was extracted following existing point cloud segmentation, then generated and exported using Metashape. We then detailed multiple possible approaches for generating textured sign meshes, including their respective efficiency and potential for full automation.

In Section 5, we tackled the challenge of integrating RGB data, such as pavement images and GPR measurements, into a DT. An orthomosaic was generated from pavement images automatically via orthorectification in MATLAB [103] and stitching using GDAL [105]. The orthomosaics were then manually georegistered onto the pavement mesh in ArcGIS [83] by selecting matching cat's eyes as GCPs and applying polynomial transforms on the orthomosaic segments.

We subsequently outlined in Section 6 the process of manually enhancing the existing point cloud segmentation and labels in CloudCompare before detailing the automated method employed for creating fully labelled defect shapefiles from pavement image annotations (previously performed using Mask-RCNN [110]). This was done by creating orthomosaics of defect masks in MATLAB, then poligonising them to generate fully labelled shapefiles in GDAL.

Finally, Sections 7 and 8 respectively guided how to integrate the dataset into GIS software, as well as the game engines UE5 and Unity, and showcased the versatility of the dataset and its potential to be harnessed in various domains beyond traditional mapping and surveying applications. We highlighted a list of possible use cases for the dataset, including applications to virtualisation, autonomous driving, inspection, and maintenance.

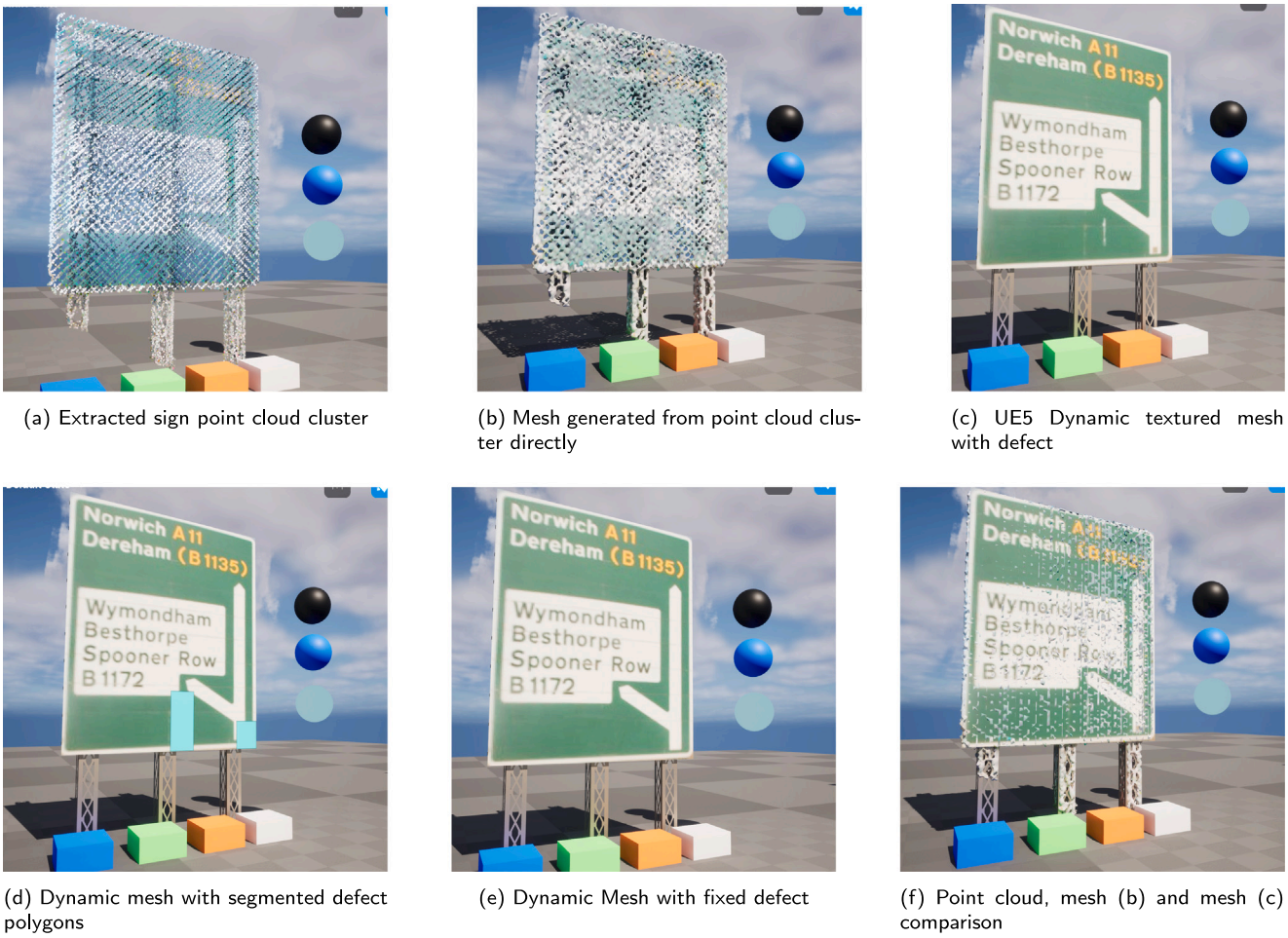
We believe the CAMHighways is a very versatile and comprehensive dataset and will be invaluable to both industry progress and academic research through its true-to-life 3D representations of the surveyed road networks and infrastructure. Indeed, the fact that CAMHighways contains data that preserve the real-world environment captured via point clouds and imagery makes it a valuable resource for advancing research in highway inspection, defect detection, and maintenance simulations. Its comprehensive collection of annotated and non-annotated pavement images, along with a novel 3D library of road signs, enables researchers to train and improve machine learning models for defect detection and condition assessment, especially in highway environments, which are under-represented in the literature in comparison to urban data. Additionally, the integration of GPR data with surface defect information offers new opportunities to understand relationships between surface and subsurface defects. This novel data combination in such a dataset allows for the development of new models that infer underlying pavement diseases from observed defect systems, and can be employed for pavement condition modelling with applications to forecasting and predictive maintenance.

Moreover, the dataset's virtualisation capabilities extend its utility across various domains such as advancing autonomous vehicle development, virtual gaming environments, and VR prototyping, and the high-fidelity 3D environment supports the simulation, testing, and development of new repair methods. CAMHighways is therefore an indispensable tool for advancing knowledge-driven solutions in transportation, urban planning, and virtual simulations.

9.1. Challenges, limitations, and future work

Extensive work was carried out to prepare the CAMHighways dataset. However, data preparation of this scale is not without its challenges and limitations. Many aspects of the process can be further improved, from quality improvements to task automation to incorporating new data sources.

First of all, the roads were surveyed as part of the National Highways Concrete Roads programme, and therefore most of them have a concrete surface, or an asphalt surface layer overlaid over a concrete



(a) Extracted sign point cloud cluster

(b) Mesh generated from point cloud cluster directly

(c) UE5 Dynamic textured mesh with defect

(d) Dynamic mesh with segmented defect polygons

(e) Dynamic Mesh with fixed defect

(f) Point cloud, mesh (b) and mesh (c) comparison

Fig. 33. A prototype interface for traffic sign maintenance state and data comparison.

base. This means that the defect and GPR data are mostly representative of concrete road-specific condition and damage.

Secondly, several tasks needed to be performed manually. For example, selecting GCPs to align the orthomosaics to the point clouds (Section 5.1.4). Additionally, some pavement defect label shapefiles were provided by KOREC rather than generated from annotations as explained in Section 6.2, and do not perfectly match our orthomosaic. Manual adjustment was performed for most, but not all of them, due to the labour-intensive nature of manual adjustment. In the context of creating an orthomosaic, training computer vision algorithms (such as Mask-RCNN, employed here for defect detection) to detect studs in both images and point clouds will significantly speed up the process of georegistering the orthomosaic onto the point cloud. Another challenge when building the orthomosaic was the lack of information recorded on the relative pitch angle of the pavement camera at the time the surveys took place, thus requiring it to be estimated in Section 5.1.1 and contributing to small inaccuracies in the alignment of some of the pavement images that make up the orthomosaics. Moreover, in the context of labelling for both point cloud and images, other assets such as the kerb, drainage covers (or gullies), and road markings have not or only partially been labelled, and future works aims to include them and assess their condition too.

Thirdly, generating meshes presented several challenges. Variations in the point cloud density unavoidably led to variations in the quality of some asset meshes, as highlighted in Section 4.2.1. Indeed, noisy point clouds can significantly worsen the mesh visual quality at times. As a result, the smoothing applied to mitigate this noise when generating meshes can alter the true shape of the some assets in small ways (see Fig. 9). Additionally, any mesh generated from an asset whose point cloud density is sparse is of low resolution. We also aim to improve upon existing methods for generating pavement meshes with textures. The current approaches have limitations in mesh quality, texture accuracy and resolution. The proposed workflow will divide the pavement orthomosaics into smaller, more manageable segments using GIS software before creating textures from these segments. These textures will be applied to the mesh. Then, the point cloud data will also be split into segments matching those of the orthomosaics. Finally, pavement mesh segments will be generated from the point cloud segments via the dynamic mesh method with pre-existing texture maps. These meshes will represent the pavement surface.

Another area of future work is vegetation mesh generation. The aim is to create detailed 3D vegetation models (such as roadside plants) using point cloud data. We tried to find the most advanced method for creating a vegetation mesh by exploring various proprietary software solutions when preparing the dataset. However, the meshes produced were excessively detailed in all cases, leading to computational challenges and unnecessary complexity. Examining existing literature on the subject, mesh simplification algorithms have been used to reduce the number of vertices while preserving essential features of vegetation canopies [99–101]. Some studies [102] have focused on separating tree trunk points from the rest of the cluster. This research is crucial for automating roadside vegetation management and prioritisation for taking control measures like cutting or pruning [112]. We aim to expand on existing ideas in the literature and integrate them with user

Table 10
Abbreviations.

AD	Autonomous Driving
CAD	Computer-Aided Design
CNN	Convolutional Neural Network
DEM	Digital Elevation Model
DT	Digital Twin
DTM	Digital Terrain Model
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
EB	East Bound
GAN	Generative Adversarial Network
GCP	Ground Control Points
GeoTIFF	Geographic Tagged Image File Format
GIS	Geographic Information System
GPR	Ground Penetrating Radar
IPM	Inverse Perspective Mapping
L	Landscape
LiDAR	Light Detection and Ranging
MO	Moving Objects
MP	Mega Pixel
NH	National Highways
RCNN	Region-based Convolutional Neural Network
RGB	Red, Green, Blue
SO	Static Objects
SRN	Strategic Road Network
SSU	Semantic Scene Understanding
TBC	Trimble Business Centre
UE5	Unreal Engine 5
VR	Virtual Reality
WB	West Bound

requirements and generative design frameworks to improve vegetation mesh generation.

In the future, we also plan to identify pathways for automating traffic sign inspection within DTs, including investigating computer vision algorithms to extract relevant features from point cloud and image data.

Finally, several researchers are now working on the collection, integration, and 3D reconstruction of thermal data in the dataset. The thermal data were collected as part of KOREC's survey, however time and software restrictions did now allow for this modality to be included in the final dataset. We aim in our future studies to enhance the dataset creation with the new source of data and further explore subjects like thermal data alignment with point clouds or increasing density of the point cloud data through thermal images, as well as pavement images.

CRediT authorship contribution statement

Alix Marie d'Avigneau: Writing – review & editing, Writing – original draft, Visualization, Software, Methodology, Investigation, Data curation, Conceptualization. **Lilia Potseluyko:** Writing – review & editing, Writing – original draft, Visualization, Software, Methodology, Investigation, Conceptualization. **Nzebo Richard Anvo:** Writing – original draft, Visualization, Software, Methodology. **Hussameldin M. Taha:** Writing – original draft, Visualization, Software, Methodology. **Varun Kumar Reja:** Writing – review & editing, Writing – original draft, Software. **Diana Davletshina:** Writing – original draft, Software. **Percy Lam:** Writing – original draft, Software. **Lavindra de Silva:** Visualization, Supervision, Software, Methodology, Conceptualization. **Abir Al-Tabbaa:** Supervision. **Ioannis Brilakis:** Supervision, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Alix Marie d'Avigneau reports financial support was provided by Costain Group. Lilia Potseluyko reports financial support was provided by Costain Group. Nzebo Richard Anvo reports financial support was provided by Costain Group. Hussameldin M. Taha reports financial support was provided by Costain Group. Prof Ioannis Brilakis serves on the Editorial Board for Advanced Engineering Informatics. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

We sincerely thank Mike Ambrose for providing access to mobile mapping dataset, and the members of KOREC, Mark Reid, Mark Bowers, and Zvonimir Blazic for carrying out the survey and providing advice throughout the data preparation process. We also thank all who were involved in various aspects of dataset preparation tasks: Xiaofang Wen, Zhening Huang, Wei Liang, Jingyu Ma, Runqi Chen, Shi Liu, and Shengqi Zhou.

This work is part of the Digital Roads Prosperity Partnership, supported by the Engineering and Physical Sciences Research Council (EPSRC) [EP/V056441/1]. Dr Reja is supported by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 101034337. Ms Davletshina is supported by the UK EPSRC Industrial CASE in partnership with National Highways and Costain. Mr Lam is funded by the UK EPSRC Centre for Doctoral Training in Future Infrastructure and Built Environment: Resilience in a Changing World (FIBE2) [EP/S02302X/1] and sponsored by National Highways, Costain, and Trimble Solutions.

Appendix A. Map of roads surveyed

See Fig. 34.

Appendix B. Finding the optimal pitch angle

B.1. Lane line detection

Consider a pavement image of size $(height, width)$, with visible lane lines (e.g., Figs. 1(a) and 35(a)). Lane lines are obtained by employing OpenCV [113] in Python 3.9. First, the images are pre-processed, and the white lines are extracted by selecting all the pixels with RGB values within a user-defined range of whites corresponding to the lane lines. The outlines of the lane lines are then obtained via Canny edge detection [114]. Once this is done, straight lines are detected from these edges via probabilistic Hough line transform [115]. The output of this algorithm is the initial and final points of L line segments, i.e., $(x_1^{(l)}, y_1^{(l)})$ and $(x_2^{(l)}, y_2^{(l)})$, respectively, where $l = 1, \dots, L$ is the segment index. From these points, the slope $\beta^{(l)}$, intercept $\beta_0^{(l)}$ and length $w^{(l)}$ of each detected segment l can straightforwardly be computed as follows:

$$\beta^{(l)} = \frac{y_2^{(l)} - y_1^{(l)}}{x_2^{(l)} - x_1^{(l)}}, \qquad \beta_0^{(l)} = y_1^{(l)} - \beta_1^{(l)} x_1^{(l)},$$

$$w^{(l)} = \sqrt{\left(x_2^{(l)} - x_1^{(l)}\right)^2 + \left(y_2^{(l)} - y_1^{(l)}\right)^2}.$$

The detected lines must then be divided into 'left' and 'right' lanes. This is done by vertically dividing the image in two down the middle and separating slopes and intercepts depending on the horizontal position of the point at which each segment begins. Formally, the slope is divided into left and right lanes as follows:

$$\beta^{(l)} := \begin{cases} \beta_{left}^{(l)}, & \text{if } x_1^{(l)} < \frac{width}{2}, \\ \beta_{right}^{(l)}, & \text{if } x_1^{(l)} \geq \frac{width}{2}, \end{cases} \qquad l = 1 \dots L,$$

and similarly for the intercept $\beta_0^{(l)}$ and segment length $w^{(l)}$. Finally, the respective slope and intercept $(\beta_{left}, \beta_{0,left})$ and $(\beta_{right}, \beta_{0,right})$ for

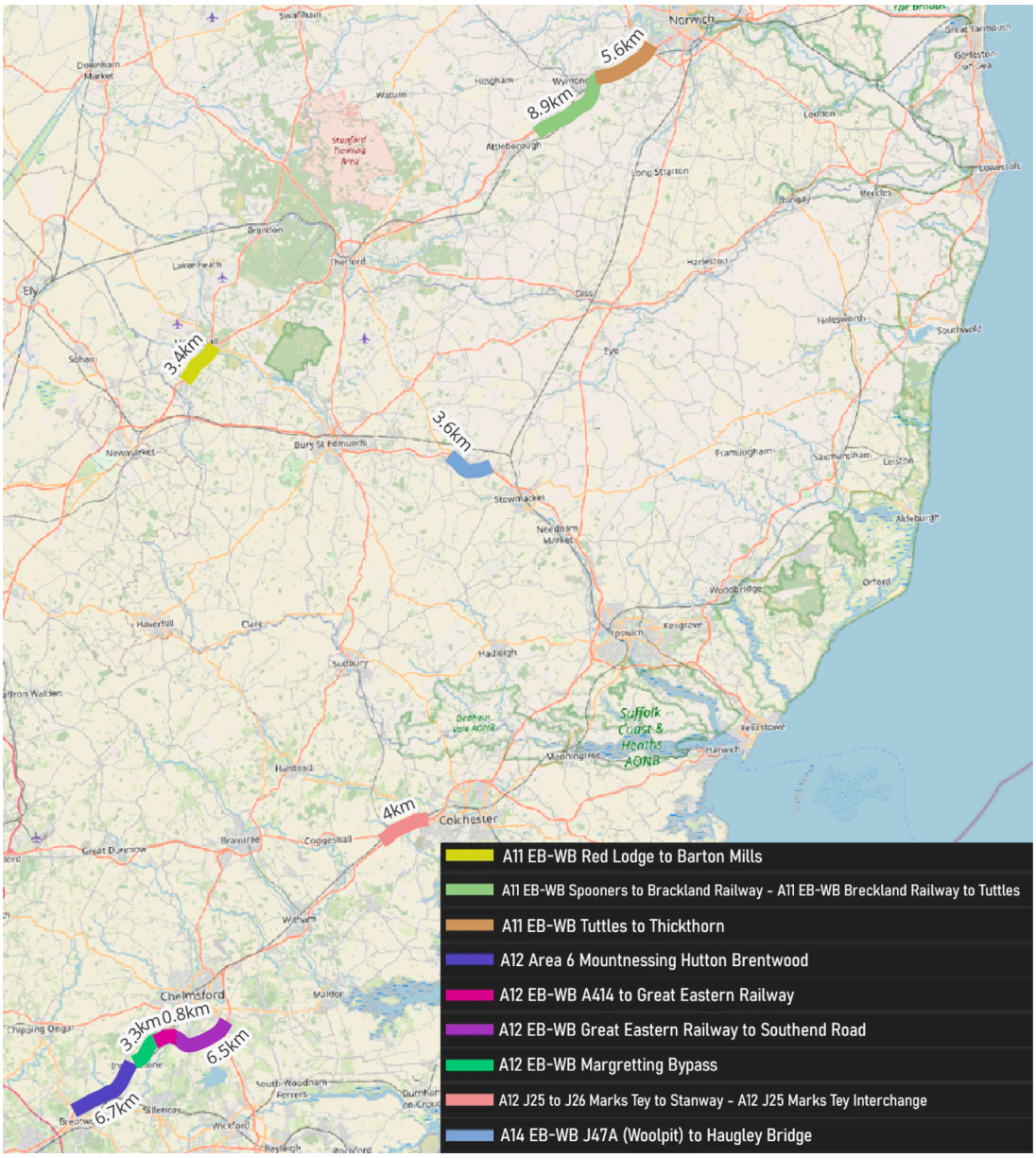


Fig. 34. Sections of the A11, A12 and A14 surveyed and their lengths (km).

the left and right lanes are estimated via weighted average, where the weight is proportional to the length of each detected segment, i.e.,

$$\beta_{left} = \frac{\sum_{l=1}^{L_{left}} w_{left}^{(l)} \beta_{left}^{(l)}}{\sum_{l=1}^{L_{left}} w_{left}^{(l)}},$$

where L_{left} is the number of segments corresponding to the left lane, and similarly for $\beta_{0,left}$, β_{right}, and $\beta_{0,right}$. The workflow for lane line detection is summarised in Fig. 35.

B.2. Optimising the relative pitch angle

Given the slopes for the left and right lanes can obtained via lane line detection, the idea is to find the optimal relative pitch angle, denoted $\hat{\theta}_r$, at which the left and right lanes visible in an orthorectified picture become parallel to each other. It is estimated by minimising the following expression:

$$\hat{\theta}_r = \arg\min_{\theta_r} \left| \beta_{left}(\theta_r) - \beta_{right}(\theta_r) \right|, \tag{2}$$

where $\beta_{left}(\theta_r)$ is the slope of the left lane at relative pitch angle θ_r, and similarly for the right lane. The reasoning is to find θ_r at which the two lane lines are parallel, which means their slopes are equal. This is achieved in practice as follows: a sample of 6 to 10 pavement images in which lane lines are clearly visible and minimal noise is present is (manually) selected. For a fine grid of relative pitch angle candidates ranging from $-50°$ to $-30°$, the images are each orthorectified (see Section 5.1.2) using candidate pitch angle θ_r. Lane line detection is performed, and the slopes $\beta_{left}(\theta_r)$ and $\beta_{right}(\theta_r)$ of the left and right lane, respectively, are obtained. The cost function $\left| \beta_{left}(\theta_r) - \beta_{right}(\theta_r) \right|$ in (2) is evaluated for each candidate angle in this fine grid. The optimal relative pitch angle is set to be the one in the grid of candidates that minimises the cost function. This process is repeated for all images in the sample, and the final optimal pitch angle $\hat{\theta}_r$ obtained is averaged over all images. Note that this method of optimisation is not exact, but it was found that the resulting orthomosaics are robust to the uncertainty in the estimation of θ_r (see Fig. 36).

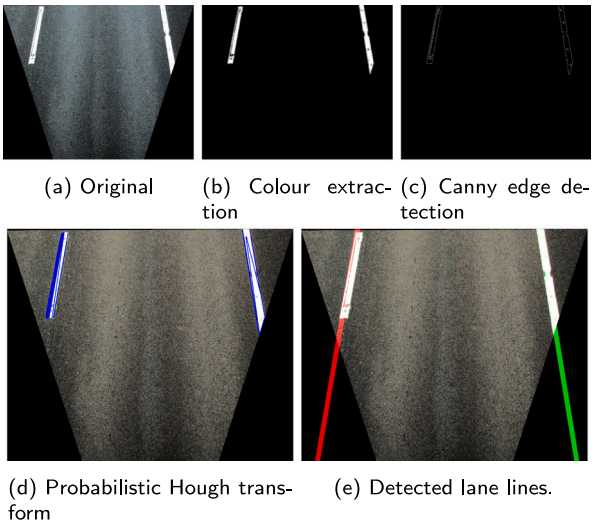


Fig. 35. Workflow for lane line detection.

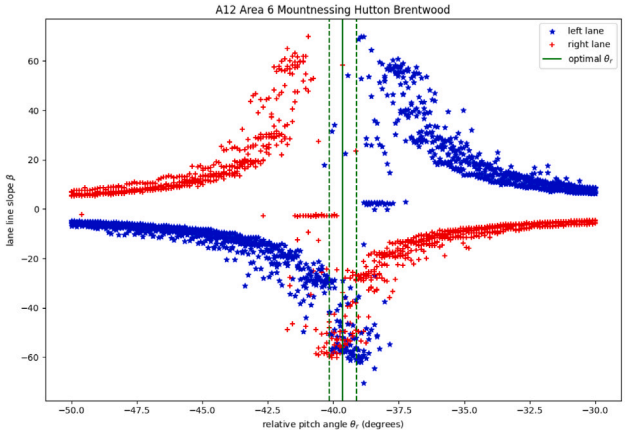


Fig. 36. Scatter plot of left and right lane line slopes for relative pitch angles θ_r ranging from $-50°$ to $-30°$. The green line represents the estimated optimal pitch angle $\hat{\theta}_r$ at which the lane lines are parallel, averaged over 6 to 10 images. The dashed green lines represent the accuracy.

Data availability

The CAMHighways dataset can be downloaded at the following link: https://drf.eng.cam.ac.uk/research/camhighways-dataset.

References

[1] E. Ivanova, J. Masarova, Importance of road infrastructure in the economic development and competitiveness, Econ. Manag. 18 (2) (2013) 263–274.
[2] C.Y. Chan, B. Huang, X. Yan, S. Richards, Investigating effects of asphalt pavement conditions on traffic accidents in Tennessee based on the pavement management system (PMS), J. Adv. Transp. 44 (3) (2010) 150–161.
[3] Department for Transport, Transport statistics great britain: 2022, 2023, https://www.gov.uk/government/statistics/transport-statistics-great-britain-2023/transport-statistics-great-britain-2022-summary.
[4] GOVUK, HM Treasury, Public sector expenditure on transport in the United Kingdom in 2022/23, by category, 2023, https://www.statista.com/statistics/298675/united-kingdom-uk-public-sector-expenditure-transport-by-category/.
[5] D. Grimmer, Norfolk Pothole Repairs Soar Amid Concerns Over Roads, Eastern Daily Press, 2024, URL https://www.edp24.co.uk/news/24198974.norfolk-pothole-repairs-soar-amid-concerns-roads/.
[6] S. Fallah-Fini, K. Triantis, H. Rahmandad, J.M. de la Garza, Measuring dynamic efficiency of highway maintenance operations, Omega 50 (2015) 18–28.
[7] J. France-Mensah, W.J. O'Brien, A shared ontology for integrated highway planning, Adv. Eng. Inform. 41 (2019) 100929.
[8] T.E. El-Diraby, K. Kashif, Distributed ontology architecture for knowledge management in highway construction, J. Construct. Eng. Manag. 131 (5) (2005) 591–603.
[9] S. Sabeti, O. Shoghli, M. Baharani, H. Tabkhi, Toward AI-enabled augmented reality to enhance the safety of highway work zones: Feasibility, requirements, and challenges, Adv. Eng. Inform. 50 (2021) 101429.
[10] R. Shah, O. McMann, F. Borthwick, Challenges and prospects of applying asset management principles to highway maintenance: A case study of the UK, Transp. Res. Part A: Policy Pract. 97 (2017) 231–243.
[11] J. Li, Z. Zhang, X. Wang, W. Yan, Intelligent decision-making model in preventive maintenance of asphalt pavement based on PSO-gru neural network, Adv. Eng. Inform. 51 (2022) 101525.
[12] J.M. de la Garza, S. Akyildiz, D.R. Bish, D.A. Krueger, Network-level optimization of pavement maintenance renewal strategies, Adv. Eng. Inform. 25 (4) (2011) 699–712.
[13] C. Koch, K. Georgieva, V. Kasireddy, B. Akinci, P. Fieguth, A review on computer vision based defect detection and condition assessment of concrete and asphalt civil infrastructure, Adv. Eng. Inform. 29 (2) (2015) 196–210.
[14] N. Dinh Le, D. Tran, R. Sturgill, C. Harper, Exploring remote sensing and monitoring technology for highway infrastructure inspection, in: Construction Research Congress 2024, 2024, pp. 416–425.
[15] Y. Pan, Z. Hu, I. Brilakis, Digital twins and their roles in building deep renovation life cycle, in: Disrupting Buildings: Digitalisation and the Transformation of Deep Renovation, Springer International Publishing Cham, 2023, pp. 83–96.
[16] X. Liu, D. Jiang, B. Tao, F. Xiang, G. Jiang, Y. Sun, J. Kong, G. Li, A systematic review of digital twin about physical entities, virtual models, twin data, and applications, Adv. Eng. Inform. 55 (2023) 101876.
[17] Y. Liu, J. Feng, J. Lu, S. Zhou, A review of digital twin capabilities, technologies, and applications based on the maturity model, Adv. Eng. Inform. 62 (2024) 102592.
[18] F. Jiang, L. Ma, T. Broyd, W. Chen, H. Luo, Building digital twins of existing highways using map data based on engineering expertise, Autom. Constr. 134 (2022) 104081.
[19] K. Kušić, R. Schumann, E. Ivanjko, A digital twin in transportation: Real-time synergy of traffic data streams and simulation for virtualizing motorway dynamics, Adv. Eng. Inform. 55 (2023) 101858.
[20] J. Azimjonov, A. Özmen, A real-time vehicle detection and a novel vehicle tracking systems for estimating and monitoring traffic flow on highways, Adv. Eng. Inform. 50 (2021) 101393.
[21] W. Chen, I. Brilakis, Developing digital twin data structure and integrated cloud digital twin architecture for roads, in: Computing in Civil Engineering 2023, 2023, pp. 424–432.
[22] R. Li, Mobile mapping: An emerging technology for spatial data acquisition, Photogramm. Eng. Remote Sens. 63 (9) (1997) 1085–1092.
[23] I. Puente, H. González-Jorge, J. Martínez-Sánchez, P. Arias, Review of mobile mapping and surveying technologies, Measurement 46 (7) (2013) 2127–2145.
[24] H. Wu, L. Yao, Z. Xu, Y. Li, X. Ao, Q. Chen, Z. Li, B. Meng, Road pothole extraction and safety evaluation by integration of point cloud and images derived from mobile mapping sensors, Adv. Eng. Inform. 42 (2019) 100936.
[25] Y.C. Lin, D. Bullock, A. Habib, Mobile LiDAR Mapping of Roadside Ditches for Drainage Analysis, Technical Report, 2021.
[26] M. Trzeciak, K. Pluta, Y. Fathy, L. Alcalde, S. Chee, A. Bromley, I. Brilakis, P. Alliez, ConSLAM: Periodically collected real-world construction dataset for SLAM and progress monitoring, in: European Conference on Computer Vision, Springer, 2022, pp. 317–331.
[27] H. Sofia, E. Anas, O. Faïz, Mobile mapping, machine learning and digital twin for road infrastructure monitoring and maintenance: Case study of mohammed VI bridge in Morocco, in: 2020 IEEE International Conference of Moroccan Geomatics (Morgeo), IEEE, 2020, pp. 1–6.
[28] F. Alberti, A. Alessandrini, D. Bubboloni, C. Catalano, M. Fanfani, M. Loda, A. Marino, A. Masiero, M. Meocci, P. Nesi, et al., Mobile mapping to support an integrated transport-territory modelling approach, Int. Arch. Photogramm. Rem. Sens. Spatial Inform. Sci. 48 (2023) 1–7.
[29] G.J. Brostow, J. Shotton, J. Fauqueur, R. Cipolla, Segmentation and recognition using structure from motion point clouds, in: Computer Vision–ECCV 2008: 10th European Conference on Computer Vision, Marseille, France, October 12-18, 2008, Proceedings, Part I 10, Springer, 2008, pp. 44–57.
[30] G.J. Brostow, J. Fauqueur, R. Cipolla, Semantic object classes in video: A high-definition ground truth database, Pattern Recognit. Lett. 30 (2) (2009) 88–97.
[31] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, B. Schiele, The cityscapes dataset for semantic urban scene understanding, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 3213–3223.
[32] P. Jiang, S. Saripalli, LiDARNet: A boundary-aware domain adaptation model for point cloud semantic segmentation, in: 2021 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2021, pp. 2457–2464.
[33] X. Roynard, J.-E. Deschaud, F. Goulette, Paris-Lille-3D: A large and high-quality ground-truth urban point cloud dataset for automatic segmentation and classification, Int. J. Robot. Res. 37 (6) (2018) 545–557.

[34] A. Geiger, P. Lenz, R. Urtasun, Are we ready for autonomous driving? The KITTI vision benchmark suite, in: Conference on Computer Vision and Pattern Recognition, CVPR, 2012.

[35] A. Geiger, P. Lenz, C. Stiller, R. Urtasun, Vision meets robotics: The KITTI dataset, Int. J. Robot. Res. 32 (11) (2013) 1231–1237.

[36] Y. Liao, J. Xie, A. Geiger, KITTI-360: A novel dataset and benchmarks for urban scene understanding in 2D and 3D, IEEE Trans. Pattern Anal. Mach. Intell. 45 (3) (2022) 3292–3310.

[37] D.-H. Paek, S.-H. Kong, K.T. Wijaya, K-lane: Lidar lane dataset and benchmark for urban roads and highways, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 4450–4459.

[38] S. Chambon, J.-M. Moliard, Automatic road pavement assessment with image processing: Review and comparison, Int. J. Geophys. 2011 (2011).

[39] Y. Shi, L. Cui, Z. Qi, F. Meng, Z. Chen, Automatic road crack detection using random structured forests, IEEE Trans. Intell. Transp. Syst. 17 (12) (2016) 3434–3445.

[40] F. Yang, L. Zhang, S. Yu, D. Prokhorov, X. Mei, H. Ling, Feature pyramid and hierarchical boosting network for pavement crack detection, IEEE Trans. Intell. Transp. Syst. 21 (4) (2019) 1525–1535.

[41] R. Fan, H. Wang, M.J. Bocus, M. Liu, We learn better road pothole detection: from attention aggregation to adversarial domain adaptation, in: Computer Vision–ECCV 2020 Workshops: Glasgow, UK, August 23–28, 2020, Proceedings, Part IV 16, Springer, 2020, pp. 285–300.

[42] M. Eisenbach, R. Stricker, D. Seichter, K. Amende, K. Debes, M. Sesselmann, D. Ebersbach, U. Stoeckert, H.-M. Gross, How to get pavement distress detection ready for deep learning? A systematic approach, in: International Joint Conference on Neural Networks, IJCNN, 2017, pp. 2039–2047.

[43] R. Stricker, M. Eisenbach, M. Sesselmann, K. Debes, H.-M. Gross, Improving visual road condition assessment by extensive experiments on the etended GAPs dataset, in: International Joint Conference on Neural Networks, IJCNN, 2019, pp. 1–8.

[44] R. Stricker, D. Aganian, M. Sesselmann, D. Seichter, M. Engelhardt, R. Spielhofer, M. Hahn, A. Hautz, K. Debes, H.-M. Gross, Road surface segmentation - pixel-perfect distress and object detection for road assessment, in: International Conference on Automation Science and Engineering, CASE, 2021, pp. 1–8.

[45] H. Majidifard, P. Jin, Y. Adu-Gyamfi, W.G. Buttlar, Pavement image datasets: A new benchmark dataset to classify and densify pavement distresses, Transp. Res. Rec. 2674 (2) (2020) 328–339.

[46] D. Arya, H. Maeda, S.K. Ghosh, D. Toshniwal, Y. Sekimoto, RDD2022: A multi-national image dataset for automatic road damage detection, 2022, arXiv preprint arXiv:2209.08538.

[47] W. Song, Z. Zhang, B. Zhang, G. Jia, H. Zhu, J. Zhang, ISTD-PDS7: A benchmark dataset for multi-type pavement distress segmentation from CCD images in complex scenarios, Remote Sens. 15 (7) (2023) 1750.

[48] G. Yang, K. Liu, J. Zhang, B. Zhao, Z. Zhao, X. Chen, B.M. Chen, Datasets and processing methods for boosting visual inspection of civil infrastructure: A comprehensive review and algorithm comparison for crack classification, segmentation, and detection, Constr. Build. Mater. 356 (2022) 129226.

[49] Z. Tong, J. Gao, H. Zhang, Recognition, location, measurement, and 3D reconstruction of concealed cracks using convolutional neural networks, Constr. Build. Mater. 146 (2017) 775–787.

[50] J. Yang, K. Ruan, J. Gao, S. Yang, L. Zhang, Pavement distress detection using three-dimension ground penetrating radar and deep learning, Appl. Sci. 12 (11) (2022) 5738.

[51] Z. Liu, X. Gu, W. Wu, X. Zou, Q. Dong, L. Wang, GPR-based detection of internal cracks in asphalt pavement: A combination method of DeepAugment data and object detection, Measurement 197 (2022) 111281.

[52] X. Dérobert, L. Pajewski, TU1208 open database of radargrams: The dataset of the IFSTTAR geophysical test site, Remote Sens. 10 (4) (2018) 530.

[53] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, 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, 2016, pp. 3234–3243.

[54] S.R. Richter, V. Vineet, S. Roth, 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, Springer, 2016, pp. 102–118.

[55] X. Li, K. Wang, Y. Tian, L. Yan, F. Deng, F.-Y. Wang, The ParallelEye dataset: A large collection of virtual images for traffic vision research, IEEE Trans. Intell. Transp. Syst. 20 (6) (2018) 2072–2084.

[56] Y. Cabon, N. Murray, M. Humenberger, Virtual KITTI 2, 2020, arXiv:2001.10773.

[57] A. Gaidon, Q. Wang, Y. Cabon, E. Vig, Virtual worlds as proxy for multi-object tracking analysis, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 4340–4349.

[58] R. Fox-Ivey, J. Laurent, B. Petitclerc, Using 3D pavement surveys to create a digital twin of your runway or highway, in: Airfield and Highway Pavements 2021, 2021, pp. 180–192, http://dx.doi.org/10.1061/9780784483527.016, arXiv:https://ascelibrary.org/doi/pdf/10.1061/9780784483527.016.

[59] M. Campbell, M. Egerstedt, J.P. How, R.M. Murray, Autonomous driving in urban environments: approaches, lessons and challenges, Phil. Trans. R. Soc. A 368 (1928) (2010) 4649–4672.

[60] National Highways, CS 230 Pavement maintenance assessment procedure, National Highways, 2022, URL https://www.standardsforhighways.co.uk/search/html/5c21c19f-4292-4764-86f6-bbb51df313e0?standard=DMRB.

[61] G. Neuhold, T. Ollmann, S. Rota Bulo, P. Kontschieder, The Mapillary vistas dataset for semantic understanding of street scenes, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 4990–4999.

[62] T. Hackel, N. Savinov, L. Ladicky, J.D. Wegner, K. Schindler, M. Pollefeys, SEMANTIC3D.NET: A new large-scale point cloud classification benchmark, in: ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, vol. IV-1-W1, 2017, pp. 91–98.

[63] X. Huang, P. Wang, X. Cheng, D. Zhou, Q. Geng, R. Yang, The ApolloScape open dataset for autonomous driving and its application, IEEE Trans. Pattern Anal. Mach. Intell. 42 (10) (2019) 2702–2719.

[64] H. Caesar, V. Bankiti, A.H. Lang, S. Vora, V.E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, O. Beijbom, nuScenes: A multimodal dataset for autonomous driving, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 11621–11631.

[65] J. Geyer, Y. Kassahun, M. Mahmudi, X. Ricou, R. Durgesh, A.S. Chung, L. Hauswald, V.H. Pham, M. M¨uhlegg, S. Dorn, T. Fernandez, M. J¨anicke, S. Mirashi, C. Savani, M. Sturm, O. Vorobiov, M. Oelker, S. Garreis, P. Schuberth, A2D2: Audi autonomous driving dataset, 2020, arXiv:2004.06320.

[66] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, et al., Scalability in perception for autonomous driving: Waymo open dataset, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 2446–2454.

[67] W. Tan, N. Qin, L. Ma, Y. Li, J. Du, G. Cai, K. Yang, J. Li, Toronto-3D: A large-scale mobile LiDAR dataset for semantic segmentation of urban roadways, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2020, pp. 202–203.

[68] Y. Pan, B. Gao, J. Mei, S. Geng, C. Li, H. Zhao, SemanticPOSS: A point cloud dataset with large quantity of dynamic instances, in: 2020 IEEE Intelligent Vehicles Symposium, IV, IEEE, 2020, pp. 687–693.

[69] B. Kim, J. Yim, J. Kim, Highway driving dataset for semantic video segmentation, 2020, arXiv preprint arXiv:2011.00674.

[70] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, T. Darrell, BDD100k: A diverse driving dataset for heterogeneous multitask learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 2636–2645.

[71] P. Xiao, Z. Shao, S. Hao, Z. Zhang, X. Chai, J. Jiao, Z. Li, J. Wu, K. Sun, K. Jiang, et al., Pandaset: Advanced sensor suite dataset for autonomous driving, in: 2021 IEEE International Intelligent Transportation Systems Conference, ITSC, IEEE, 2021, pp. 3095–3101.

[72] H. Maeda, Y. Sekimoto, T. Seto, T. Kashiyama, H. Omata, Road damage detection and classification using deep neural networks with smartphone images, Comput.-Aided Civ. Infrastruct. Eng. 33 (2018) 1127–1141, http://dx.doi.org/10.1111/mice.12387.

[73] Q. Zou, Z. Zhang, Q. Li, X. Qi, Q. Wang, S. Wang, DeepCrack: Learning hierarchical convolutional features for crack detection, IEEE Trans. Image Process. 28 (3) (2018) 1498–1512.

[74] H. Maeda, T. Kashiyama, Y. Sekimoto, T. Seto, H. Omata, Generative adversarial network for road damage detection, Comput.-Aided Civ. Infrastruct. Eng. 36 (2021) 47–60, http://dx.doi.org/10.1111/mice.12561.

[75] D. Arya, H. Maeda, S.K. Ghosh, D. Toshniwal, Y. Sekimoto, RDD2020: An annotated image dataset for automatic road damage detection using deep learning, Data brief 36 (2021) 107133.

[76] Trimble Geospatial, Trimble Business Centre 5.81, Trimble Inc, 2024, https://geospatial.trimble.com/en/products/software/trimble-business-center.

[77] Certainty 3D, TopoDOT Documentation, Certainty 3D, 2023, URL.

[78] N. Moerman, Getting started with Orbit 3DM Desktop, Bentley Systems, 2023, URL https://communities.bentley.com/products/3d_imaging_and_point_cloud_software/w/wiki/51746/getting-started-with-orbit-3dm-desktop.

[79] Agisoft, Agisoft metashape, Agisoft, 2023, https://www.agisoft.com/downloads/installer/.

[80] N.R. Anvo, T.G. Thuruthel, H.M. Taha, L. de Silva, A. Al-Tabbaa, I. Brilakis, F. Iida, Automated 3D mapping, localization and pavement inspection with low cost RGB-D cameras and IMUs, in: Annual Conference Towards Autonomous Robotic Systems, Springer, 2023, pp. 279–291.

[81] Autodesk Staff, ReCap Learning Support, Autodesk, 2023, URL https://help.autodesk.com/view/RECAP/ENU/.

[82] Autodesk, Documentation, point clouds, 2023, https://help.autodesk.com/view/FDU/2022/ENU/?guid=FDU_Navisworks_Factory_Help_Working_With_Point_Clouds_Attaching_Point_Clouds_html.

[83] Esri, ArcGIS pro 3.1.2, 2023, https://pro.arcgis.com/en/pro-app/latest/get-started/get-started.htm.

[84] Esri, ArcGIS CityEngine, 2023, https://www.esri.com/en-us/arcgis/products/arcgis-cityengine/overview.

[85] QGIS Development Team, QGIS Geographic Information System, QGIS Association, 2024, https://www.qgis.org.
[86] Epic Games, Unreal Engine Documentation, Epic Games, 2023, URL.
[87] J.K. Haas, A history of the unity game engine, Diss. Worcester Polytech. Inst. 483 (2014) (2014) 484.
[88] A. Juliani, V.-P. Berges, E. Teng, A. Cohen, J. Harper, C. Elion, C. Goy, Y. Gao, H. Henry, M. Mattar, et al., Unity: A general platform for intelligent agents, 2018, arXiv preprint arXiv:1809.02627.
[89] 3Dsurvey, 3Dsurvey, 2023, URL https://3dsurvey.si/.
[90] CloudCompare, CloudCompare, GPL software, 2023, https://www.cloudcompare.org/.
[91] E. Akleman, J. Chen, Regular meshes, in: Proceedings of the 2005 ACM Symposium on Solid and Physical Modeling, SPM '05, Association for Computing Machinery, New York, NY, USA, 2005, pp. 213–219, http://dx.doi.org/10.1145/1060244.1060268.
[92] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al., A density-based algorithm for discovering clusters in large spatial databases with noise, in: Kdd, vol. 96, (no. 34) 1996, pp. 226–231.
[93] D. Davletshina, V.K. Reja, I. Brilakis, Automating construction of road digital twin geometry using context and location aware segmentation, Autom. Constr. 168 (2024) 105795, http://dx.doi.org/10.1016/j.autcon.2024.105795, URL https://www.sciencedirect.com/science/article/pii/S0926580524005314.
[94] M. Kazhdan, M. Bolitho, H. Hoppe, Poisson surface reconstruction, in: Proceedings of the Fourth Eurographics Symposium on Geometry Processing, vol. 7, 2006.
[95] Open3D, Surface reconstruction, 2023, http://www.open3d.org/docs/latest/tutorial/Advanced/surface_reconstruction.html.
[96] Adobe Photoshop Documentation, How to crop and straighten in photoshop, 2023, https://helpx.adobe.com/photoshop/using/crop-straighten-photos.html.
[97] G.U. contributors, Road Traffic Sign Images for Reproduction, Government UK, 2023, URL https://www.gov.uk/guidance/traffic-sign-images.
[98] C. Ai, Y.J. Tsai, An automated sign retroreflectivity condition evaluation methodology using mobile LIDAR and computer vision, Transp. Res. C 63 (2016) 96–113.
[99] T. Jiang, Y. Wang, S. Liu, Q. Zhang, L. Zhao, J. Sun, Instance recognition of street trees from urban point clouds using a three-stage neural network, ISPRS J. Photogramm. Remote Sens. 199 (2023) 305–334, http://dx.doi.org/10.1016/j.isprsjprs.2023.04.010, URL https://www.sciencedirect.com/science/article/pii/S0924271623000990.
[100] F. Aiteanu, R. Klein, Exploring shape spaces of 3D tree point clouds, Comput. Graph. 100 (2021) 21–31, http://dx.doi.org/10.1016/j.cag.2021.07.013, URL https://www.sciencedirect.com/science/article/pii/S0097849321001448.
[101] Y. Xu, X. Tong, U. Stilla, Voxel-based representation of 3D point clouds: Methods, applications, and its potential use in the construction industry, Autom. Constr. 126 (2021) 103675, http://dx.doi.org/10.1016/j.autcon.2021.103675, URL https://www.sciencedirect.com/science/article/pii/S0926580521001266.
[102] W. Wang, Y. Li, H. Huang, L. Hong, S. Du, L. Xie, X. Li, R. Guo, S. Tang, Branching the limits: Robust 3D tree reconstruction from incomplete laser point clouds, Int. J. Appl. Earth Obs. Geoinf. 125 (2023) 103557, http://dx.doi.org/10.1016/j.jag.2023.103557, URL https://www.sciencedirect.com/science/article/pii/S1569843223003813.
[103] MATLAB, version 9.13.0 (R2022b), The MathWorks Inc, Natick, Massachusetts, 2023.
[104] R. Hartley, A. Zisserman, Multiple view geometry in computer vision, Cambridge University Press, 2003.
[105] GDAL/OGR contributors, GDAL/OGR Geospatial Data Abstraction software Library, Open Source Geospatial Foundation, 2023, http://dx.doi.org/10.5281/zenodo.5884351, URL https://gdal.org.
[106] J.L. Bentley, Multidimensional binary search trees used for associative searching, Commun. ACM 18 (9) (1975) 509–517.
[107] G.M. Hadjidemetriou, J. Masino, S.E. Christodoulou, F. Gauterin, I. Brilakis, Comprehensive decision support system for managing asphalt pavements, J. Transp. Eng. Part B: Pavem. 146 (3) (2020) 06020001, http://dx.doi.org/10.1061/jpeodx.0000189.
[108] Highways Department, RD/GN/015B Catalogue of Road Defects (CORD), Highways Department, Hong Kong SAR, Hong Kong, 2013.
[109] CVAT.ai Corporation, Computer Vision Annotation Tool (CVAT), GitHub, 2022, URL https://github.com/opencv/cvat?tab=readme-ov-file.
[110] K. He, G. Gkioxari, P. Dollár, R. Girshick, Mask R-CNN, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 2961–2969.
[111] C.-Y. Wang, A. Bochkovskiy, H.-Y.M. Liao, YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, 2023.
[112] V.K. Reja, D. Davletshina, M. Yin, R. Wei, Q.F. Adam, I. Brilakis, F. Perrotta, A digital twin-based approach to control overgrowth of roadside vegetation, in: Proceedings of the 41st International Symposium on Automation and Robotics in Construction, 2024, http://dx.doi.org/10.22260/ISARC2024/0086, URL https://www.iaarc.org/publications/2024_proceedings_of_the_41st_isarc_lille_france/a_digital_twin_based_approach_to_control_overgrowth_of_roadside_vegetation.html.
[113] G. Bradski, The OpenCV library, Dr. Dobb's J. Softw. Tools (2000).
[114] J. Canny, A computational approach to edge detection, IEEE Trans. Pattern Anal. Mach. Intell. (6) (1986) 679–698.
[115] J. Matas, C. Galambos, J. Kittler, Robust detection of lines using the progressive probabilistic hough transform, Comput. Vis. Image Underst. 78 (1) (2000) 119–137.