

Road Design Layer Detection in Point Cloud Data for Construction Progress Monitoring

Steven VICK¹, Ioannis BRILAKIS²

¹ PhD Candidate, Department of Engineering, University of Cambridge, Trumpington Street, Cambridge, UK, CB2 1PZ, Email: sv364@cam.ac.uk

² Laing O'Rourke Reader, Department of Engineering, University of Cambridge, Trumpington Street, Cambridge, UK, CB2 1PZ, Email: ib340@cam.ac.uk

Abstract

Poor performance in transportation construction is well-documented, with an estimated \$114.3 billion in global annual cost overrun. Studies aimed at identifying the causes highlighted traditional project management functions like progress monitoring as the most important contributing factors. Current methods for monitoring progress on road construction sites are not accurate, consistent, reliable, or timely enough to enable effective project control decisions. Automating this process can address these inefficiencies. The detection of layered design surfaces in digital as-built data is an essential step in this automation. A number of recent studies, mostly focused on structural building elements, aimed to accomplish similar detection but the methods proposed are either ill-suited for transportation projects or require labelled as-built data that can be costly and time consuming to produce. This paper proposes and experimentally validates a model-guided hierarchical space partitioning data structure for accomplishing this detection in discrete regions of 3D as-built data. The proposed solution achieved an F1 Score of 95.2% on real-world data confirming the suitability of this approach.

Keywords: Transportation Construction; Progress Monitoring; Drones; Automation

INTRODUCTION

A recent study estimated that \$3.2 trillion per year in infrastructure investment is required to keep pace with global GDP growth through 2030, with approximately 16% of that investment (\$512 billion) required for road construction (Dobbs et al., 2013). Poor performance in this field is well-documented, with more than forty independent studies and government audits over the past three decades identifying as many as 83 separate causes for transportation construction cost overrun (e.g. Siemiatycki, 2009; Cantarelli et al., 2010; Memon et al., 2012; Salling & Leleur, 2015). One study

analysed projects in Great Britain, Denmark, Sweden, and Norway, concluding that 77% experienced cost overruns with the average amount being 29% of original contract value (Salling & Leleur, 2015). Assuming these levels of performance continue, there is a potential for \$114.3 billion in annual cost overrun on road construction projects worldwide ($\$512B \times 77\% \times 29\%$). While the causes of these overruns are varied, traditional project management functions have been highlighted as the most important contributors (González et al., 2014). Project control is one such function that directly impacts project performance (Del Pico, 2013). It's a cyclical process consisting of progress monitoring and implementation of corrective actions. Automating the first part of the project control cycle, progress monitoring, can enable more timely and effective corrective actions aimed at keeping a project on time and within budget.

Progress monitoring is the cyclical collection, analysis, and reporting of work complete at a given point in time, compared to the planned work complete. The main processes of progress monitoring are: (1) collecting as-built data, (2) processing it into the form required for analysis, and (3) comparing it with the desired (or "as-planned") state of the project. Current transportation progress monitoring practice is largely "non-spatial" involving no or very sparse measurement of the physical progress on site. Project engineers, surveyors, crew foremen, and other on-site personnel collect as-built data using paper or electronic checklists and daily reports, verbal updates, site photographs, material delivery receipts, inventory reports, and subcontractor invoices. These manual data collection methods rely on the experience, training, and subjective assessment of the individuals performing the inspection. Additionally, collecting and parsing data from these disparate sources into meaningful activity-level inputs and compiling them into project-level performance metrics is a time consuming and arduous task that consumes as much as 30-50% of a project manager's time (Navon & Sacks, 2007). Progress determinations are made in increments of distance along the centreline of the road. For example, a project utilizing a 20m increment might report completion of a layer to station 01+20 (i.e. 120m from the beginning of the road). One drawback of this approach is that progress (or lack of progress) between the stations may be left unreported. Ultimately, non-spatial progress monitoring methods are too error-prone, inconsistent, and burdensome to enable timely and effective implementation of corrective project control actions.

The civil engineering community recently examined using a variety of commercially-available spatio-temporal sensors to support progress monitoring. Ultra-wide band (UWB) (e.g. Shahi et al., 2012), radio frequency identification (RFID) (e.g. Razavi & Moselhi, 2012), barcode (Cheng & Chen, 2002), global positioning system

(GPS), and various combinations of these sensors (e.g. Razavi & Haas, 2010) have been used to track materials, equipment, and other resources as indicators of progress. While these sensors can provide useful information, they don't directly measure the physical progress on the site. Additionally, their installation and maintenance can be burdensome for project management as tags and readers must be applied, monitored, and repositioned on a wide range of resources as the project evolves.

An alternative approach that directly measures the progress on site uses spatial data collection technologies. Spatial progress monitoring involves surveying the work performed, calculating the volumes, areas and/or distances that form the basis of the analysis, and comparing the results with planned progress to determine project status. The most common data collection technologies used are Robotic Total Stations, Global Navigation Satellite System (GNSS) receivers, Light Detection and Ranging (LiDAR) scanners, and Aerial Photogrammetry (NCHRP, 2013). However, the scale of most transportation construction projects dictates the use of methods capable of quickly surveying large areas to an acceptable level of accuracy and resolution, defined as 1-5 cm three-dimensional (3D) point accuracy and more than 150 points/m² (Olsen et al., 2013). Table 1 summarizes current technologies capable of achieving at or near this level of performance. Note that performance and cost can vary due to a variety of factors for each technology, so general ranges are provided based on a review of academic studies, technical reports, and manufacturer specifications. Unmanned aerial photogrammetry stands out as a cost-effective data collection solution capable of meeting the demands of spatial progress measurement on large transportation sites.

Although these data collection solutions exist, automatically recognizing progress in the as-built scene is a difficult task. The authors decompose automatic progress monitoring for road construction into the following steps: (1) aligning and registering the as-built data in a common coordinate system with the design model, (2) detecting 'if' and 'where' each road design layer occurs in the as-built data, (3) comparing this with the as-planned status, and (4) generating the progress report. Commercially-available solutions exist for the first step, typically using GPS and/or surveyed control targets to place the as-built data in the design model's geographic coordinate system (e.g. Trimble, 2017; Pix4d, 2017). The second step, however, remains an unsolved problem and is the focus of this study. Note that the authors define road layer detection as a process that identifies which, and exactly where, each design layer is present in the as-built point cloud data (PCD). While no methods exist for automatically detecting road design surfaces in as-built data, similar functions are performed in automated manufacturing quality control processes. Such systems

(e.g. GOM Inspect (2017)) compare as-built data from recently-manufactured components with 3D models to detect quality deviations beyond acceptable levels. However, these applications benefit from a variety of controlling factors that cannot feasibly be replicated on transportation construction sites. For example, they are designed to operate indoors with known and stable lighting conditions. Additionally, the components analysed are relatively small and can be carefully measured from all required perspectives.

Detecting road design layers in unlabelled spatial data requires a-priori information regarding the expected elevation of each layer throughout the corridor. Civil Information Modelling (CIM) is a mature technology capable of providing such information, and is gaining wider support from government agencies and contractors. The U.S. Army Corps of Engineers, for example, requires use of CIM tools in design and construction of all horizontal civil works projects (US Army, 2013). CIM models represent horizontal construction as combinations of triangulated surfaces defined by dynamic design objects such as alignments, profiles, and design cross sections. This enables extraction of individual model surfaces defining the finished grade of activity layers typically found in road construction. In current practice, CIMs are used mostly for design visualization and automated guidance of earthwork equipment (Schneider, 2013). Since CIM model surfaces can be very large and complex, an interesting problem to consider is how to best divide them into smaller sections to allow incremental progress detection.

The computer graphics, medical imaging, and robotics fields regularly deal with spatial subdivision of complex 3D spaces. Rasterization and voxelization are two popular methods for accomplishing this task. Rasterization uses a perspective-projection of 3D objects onto a 2D plane and then divides this projection into a regular grid. In computer graphics a grid of picture elements (pixels) is used. Voxelization divides 3D space into a network of regularly-spaced cuboid volume elements (voxels). A common voxel organizational structure known as the octree recursively divides a cube-bounded 3D space into eight sub-cubes down to a specified subdivision threshold (criteria are typically dimension-based or occupancy-based). Voxels are the smallest subdivision, or the ‘leaves’, of this hierarchical tree structure. The octree structure allows rapid searching and indexing of 3D space, and implicitly describes how various objects are distributed within complex 3D scenes (Watt, 2000). These benefits make octrees an attractive option for use in detecting 3D design surfaces in as-built point cloud data, but the trade-off between accuracy and efficiency must be kept in mind. For example, an octree of finer resolution (i.e. smaller leaf voxel size) more

accurately represents a 3D object but at the expense of increased memory usage and slower voxel traversal operations (Watt, 2000).

In summary, the basic building blocks for implementing an improved automated progress monitoring approach exist in current practice. LiDAR and photogrammetry are viable means for obtaining survey-quality as-built data on large construction sites, while CIM provides the basis for a digital comparison of the as-built data with the as-planned state. Detecting the CIM design layers in as-built data is the next unsolved problem in realizing this automation. This paper proposes and evaluates a novel model-guided and sparse hierarchical space partitioning approach for accomplishing road design layer detection in discrete regions of as-built point cloud data. The following section reviews recent research relevant to this task. The proposed solution is then described in detail and evaluated using both synthetic and real-world data from the construction of a residential road in Cambridge, UK. Finally, the researchers summarize the experimental results, draw conclusions, and discuss future research goals.

BACKGROUND

The specific technical problem addressed in this paper is detection of road design surfaces in as-built data to support automated progress monitoring. The following review evaluates related methods proposed in recent studies based on their level of automation and applicability for layered road design surfaces.

A group of related studies focused on detecting building construction design objects using as-built PCD collected by laser scanning (Bosché, 2010; Kim et al., 2013; Turkan et al., 2014; Kalasapudi et al., 2017), photogrammetry (Golparvar-Fard et al., 2015; Tuttas et al., 2014; Braun et al., 2015), or both (Han et al., 2018). Design object detection in the 3D data was accomplished in different ways. This study divides the automated design surface detection approaches into two main categories: (1) point-correspondence and (2) space partitioning. The following sub-sections review the literature in these two categories before examining the sparse research specifically related to transportation design surfaces.

Point Correspondence Approaches

Point-correspondence approaches (Bosché, 2010; Kim et al., 2013; Turkan et al., 2014) generate as-planned point clouds from 3D model elements and search for the nearest as-built correspondences within a threshold distance to determine if an as-planned point is present in the as-built scene. Design surface triangulation vertices are sparse, so

as-planned point clouds must be either subsampled on the design surface to closely match the density of the as-built cloud (e.g. Kim et al., 2013), or the as-built points must be projected onto the nearby surfaces before selecting the nearest projection as the correct as-planned point location (e.g. Bosché, 2010). The researchers mostly used thresholds on the number of confirmed point correspondences to determine if a model element was detected. These methods showed promise for automatically detecting planned structural building elements in 3D PCD, particularly when factoring in a-priori connectivity and construction sequencing information like Kim et al. (2013). However, they focused on all-or-nothing detection of the individual components. Such an approach works well for detecting relatively small components constructed within a progress monitoring cycle (e.g. a column), but fails to capture the kind of incremental progress needed to support effective project control on road construction sites with very large design surfaces.

Space Partitioning Approaches

Other researchers partitioned registered as-built/as-planned scenes and identified points falling into the partitioned regions to facilitate object detection. Some used the boundaries of each structural element model (e.g. a square column) to conduct this partitioning (e.g. Zhang & Arditi, 2013). More commonly used approaches partition the 3D space containing the as-built/as-planned scene using voxels. Some (Tuttas et al., 2014; Braun et al., 2015) built an octree over the as-built PCD and searched for points within orthogonal distance thresholds of rasterized planar segments (e.g. rectangles or triangles) extracted from each structural element's constituent surfaces. Although this approach is not "all-or-nothing," it does limit the detection decision to each planar surface segment that makes up the design object. Consider the effect of this on a road design surface that has very large planar regions (think of a large stretch of flat highway) as well as areas where the surface orientation changes in much smaller intervals (think of a banked highway around a curve). In this case, the surface detection would happen in uneven and inconsistent intervals along the length of the road, resulting in progress reports that are difficult to interpret and act upon.

An approach that more consistently partitioned larger and/or more complex scenes (Golparvar-Fard et al., 2015) voxelized the entire 3D space containing the registered as-is/as-built scene before traversing the voxels (twice: once each for as-planned and as-built data) from various unordered camera positions and labelling them based on the presence of design elements and as-built points. This method made use of ubiquitous daily job site images for as-built data collection, and even enforced visibility and visual consistency constraints to account for static and dynamic

occlusions in the scene. However, the uniform nature of the implemented voxelization is ill-suited for large and closely-layered road design surfaces. Consider the notional case illustrated in Figure 1, showing a longitudinal (y-z) 2D section of a small asphalt road design region. Realistic design layer thicknesses are represented to scale with the voxel size shown, approximating the design specifications for the road construction site evaluated later in this paper. With a notional 10 cm leaf voxel size, the thin upper design surfaces occupy the same voxel in multiple instances. As-built point and as-planned surface occupancy in this case would not be sufficient to identify which design surface is found in the as-built scene, presenting a problem for the detection approach used by Golparvar-Fard et al. (2015). Voxelizing the scene at 1/5 the previous resolution (2 cm shown in Figure 1) can address this concern, but results in a 125x increase in the total number of voxels that need to be traversed and analysed. Using a smarter space partitioning approach, like an octree that only subdivides branch cells containing as-planned data, can reduce this complexity but other issues still remain. First, use of regular cuboid voxels small enough to distinguish between thin asphalt layers requires extremely high-density as-built data. For example, a 2 cm leaf voxel size in Figure 1 would alleviate the problem of voxels occupied by multiple design layers, but would require as-built data with a spatial resolution of at least 2,500 pts/m² to ensure at least one point could be found in each voxel along the road corridor, not accounting for noise and measurement error. This limits the data collection options to the most expensive and time-consuming methods (Table 1) that are more susceptible to occlusions due to the ground-level perspective. Second, the cubic-grid structure of the voxelization produces voxels that do not uniformly fit the design surfaces. In some instances, a surface intersects only the corner of a voxel while in others it cuts directly through the middle. As a result, as-built points corresponding to a specific design surface are only sought above the surface in some cases, while in others only the region below is considered. This could results in further inconsistencies with the object detection decision, particularly when voxelizing at finer resolutions to ensure unique as-planned voxel labels.

Transportation-Specific Research

Just one study sought to compare road design surface layers to as-built point cloud data (Kivimaki & Heikkila, 2015). This study, however, stopped short of true design surface detection as the matching of surveyed points to the relevant design surface was accomplished by assigning a specific code linking each point to a surface. Such point codes are only possible on terrestrial survey equipment where the exact nature of each point can be coded into the machine prior to taking the measurement. The authors concede that manual matching of point clouds to surfaces is required for data that lack the point code information; a drawback on the efficiency and applicability of this approach.

Gaps in Knowledge, Objective, and Research Questions

The existing body of research is heavily-skewed towards structural building components in as-built PCD. Surfaces for such components are typically characterized by relatively-small orthogonally intersecting planar sections (e.g. the faces of a concrete foundation wall). Road design surfaces are larger and more complex, typically involving changes in slope along both the longitudinal and transverse axes of the road corridor. The closely-layered construction, particularly in the thin upper asphalt layers, also differentiates road design surfaces from building structural components. These differences limit the applicability of existing building-focused detection methods on as-built data for road projects. Considering the state of practice and body of research reviewed in the preceding sections, we identify the following gaps in knowledge: (1) no method exists for automatically measuring progress on road construction projects, (2) existing methods for automatically detecting design surfaces in as-built data are not well-suited for identifying and distinguishing individual layered road design surfaces, (3) none of the proposed methods tackle incremental progress detection, and (4) there is no formal understanding of the challenges and limitations associated with adopting an automated progress monitoring approach on linear transportation projects.

The objective of this study is to address gaps 2-4 by developing and testing a novel data structure for automatically and incrementally detecting layered road design surfaces in unlabelled as-built PCD. The authors examine the following research questions to accomplish this objective: (1) how can the 3D as-planned space be partitioned to allow for consistent and incremental road layer detection?, (2) how can as-built PCD be accurately classified within the partitioned regions, accounting for noise and uncertainty in the as-built scene?, and (3) what combination of design and/or input parameters will be required to produce the most desirable results?

PROPOSED SOLUTION

The authors propose a novel solution for detecting layered road design surfaces in discrete regions of as-built point cloud data. This solution implements a new model-guided and sparse hierarchical space partitioning data structure named *BrickTree*, a combination of the authors' names and a nod to the rectangular cuboid shape of the leaf voxels in this structure. Figure 2 illustrates the key processes in implementing this approach. The inputs to this solution are (1) an as-built point cloud, scaled in real-world units and registered in the CIM model's coordinate system, and (2) triangulated road layer surfaces extracted from the CIM model. The rest of this section describes in further detail the four processes of the proposed solution.

The BrickTree Data Structure

The first step in generating the *BrickTree* applies a rigid-body transformation to the layered design surfaces, positioning them just above the global origin with the longitudinal direction of the road corridor aligned with the y -axis and the transverse direction aligned with the x -axis. A 2D grid, named the *projectGrid*, is then created on the x - y plane below the registered and aligned design surface layers. The *projectGrid* serves as the basis of further partitioning as each square grid cell defines the x - y boundaries of a *rasterBranch* in the hierarchical tree structure (Figure 3). As such, each *rasterBranch* is a vertical projection of a *projectGrid* cell, and is defined within the hierarchy as a collection of *leafVoxels*. *LeafVoxels* are defined as 3D rectangular space partitioning elements that subdivide each *rasterBranch* region. The *gridCellSize* parameter determines the granularity of the incremental surface detection decision, which occurs at the *rasterBranch* level where only one design layer should be identified for a given dataset. Note that this could result in small classification errors in regions of transition between as-built layers, and selection of the *gridCellSize* parameter should consider this possible error in determining the acceptable level of granularity for the final progress report. The authors note that such errors (e.g. detecting a surface within a branch's boundaries when only half of it is actually present) are common for all spatial partitioning methods, and generally deemed acceptable if properly controlled considering the inaccuracy and subjectivity of current progress monitoring practice. The *gridCellSize* is set to 0.5m and controlled for in the following experiments based on conversations with local project management personnel that indicated this would be an acceptable level of granularity for the final progress determination, allowing more detailed reporting of progress than the centreline increment methods used in current practice.

Each *rasterBranch* is partitioned into *leafVoxels* at intersections with the layered design surfaces. The goal of this model-guided voxelization is to ensure each *leafVoxel* enables consistent searching for as-built points in the regions above and below their corresponding surface. This study constructs axes-aligned voxels, relying on the assumption that road surfaces can be reasonably approximated as flat within the regions defined by the *gridCellSize*. This simplifying assumption reduces the complexity of point-to-voxel assignment operations by limiting them to a series of simple indexing calculations in the x , y , and z -directions. The degree to which this assumption affects the performance of the proposed method and how well it generalizes to the possible combinations of road camber and grade is the focus of further research by the authors, and thus is not reported in this paper.

To construct the *leafVoxels*, the authors orthogonally project the *projectGrid* onto each surface, storing the centroid and normal direction of each projected grid cell for the following steps. The z -value of this centroid coupled with the *searchDistance* parameter defines the height of the constructed *leafVoxels*, while the horizontal limits of the grid cells define the voxel boundaries in the x and y directions. Selecting the optimal *searchDistance* value is a focus of experiments in the following section, but an important consideration is the thickness of the thinnest design layer as this sets an upper limit on the value in order to limit voxel overlap. The final step in constructing the *BrickTree* sets the upper z -limit of the outer 3D bounding box as the maximum point elevation of the voxels in the top surface layer. With the *BrickTree* constructed, the proposed solution moves on to the next step: populating the *leafVoxels* with as-built data.

Populating the BrickTree with as-built points

There are two general strategies for assigning as-built points to a hierarchical space partitioning tree: (1) traverse the point cloud to determine where in the tree structure each point lies, or (2) traverse the tree structure to identify and assign points in the vicinity of each voxel. The axes-aligned *leafVoxel* structure allows points to be rapidly indexed and assigned to discrete 3D regions. For this reason, the authors chose to traverse the as-built cloud, directly assigning points to *leafVoxels*. Because the voxels are not evenly-spaced in the z -direction, rapid indexing only occurs at the branch level. For a given as-built point, p_b , the process calculates the branch index by comparing it to the origin (lower-left point) of the *projectGrid*, p_o , using the following equations where n_{i_cells} is the number of *projectGrid* cells in the x -direction. Note that the bracket notation used indicates floor-rounding to the nearest integer value.

$$i_{index} = \left\lfloor \frac{p_{bx} - p_{ox}}{gridCellSize} \right\rfloor \quad (1)$$

$$j_{index} = \left\lfloor \frac{p_{by} - p_{oy}}{gridCellSize} \right\rfloor \quad (2)$$

$$branch_{index} = i_{index} + j_{index} \times n_{i_cells} \quad (3)$$

With the branch identified, the method then compares the z -component of the built point, p_{bz} , to the z -limits of the *leafVoxels* to determine which, if any, to assign the point to. If an as-built point is detected inside a *leafVoxel*, the voxel is marked as 'detected'. Note that this initial labelling is reviewed and updated in the subsequent branch

classification step. After the algorithm processes all as-built points falling within the *BrickTree* bounding box and applies the *pointThreshold* to label the *leafVoxels*, the branch classification processes begins.

Branch classification

Noise and other errors could result in multiple *leafVoxels* being marked 'detected' in the same *rasterBranch*. A classification decision must be made to correct this conflict as it's only possible to detect one design surface per branch in an as-built scene. Sophisticated decision boundaries can be learned using supervised or unsupervised machine learning approaches, but these require a large number of training datasets from a variety of construction sites. While the authors aim to collect further data and explore machine learning based classification in future work, the current study implements a simple statistical threshold decision boundary as a proof of concept for the overall framework. Three possible rules are considered: maximum points (i.e. select the voxel with the most points), minimum distance (i.e. select the voxel with the minimum average orthogonal distance between its as-built points and the design surface), and minimum vertical variance (i.e. select the voxel whose points have the lowest variance in the z-direction). Regardless of the decision rule used, the algorithm labels each *rasterBranch* with the surface identified and updates the *leafVoxel* labels in the branch accordingly. The optimal classification rule out of the three considered is experimentally evaluated later in this paper.

Outlier correction

The final step in the surface detection process, outlier correction, aims to improve the final result's accuracy using a priori knowledge of road construction operations. Specifically, the method assumes that road surfaces are generally not constructed with gaps or missing areas in the middle. The authors propose a neighbourhood consensus algorithm to accomplish this task. Two input parameters, *neighbourDistance* and *consensusThreshold* are used. The *neighbourDistance* parameter defines the size of the neighbourhood in which a branch classification is compared to that of its neighbours. Figure 4 illustrates different construction cases for the branch comparison 'neighbourhood'. Implementing the different construction cases ensures the comparison region size remains constant no matter where the analysed branch lies on the *projectGrid*. The algorithm traverses the constructed neighbourhood, tallying branch classifications by surface identification. If the ratio of any surface's tallied count to the total number of branches in the neighbourhood exceeds the *consensusThreshold*, a consensus is declared in the neighbourhood. The algorithm then compares and updates (if required) the analysed branch's classification and *leafVoxel* labels. The optimal

consensusThreshold value and size of the outlier correction window, defined by the *neighborDistance* parameter, are explored in the experiments later in this paper.

Construction and as-built data quality

The proposed solution assumes an acceptable level of construction (ϵ_c) and as-built data measurement (ϵ_m) error. The authors divide these errors into the horizontal ($\epsilon_{c,h}$, $\epsilon_{m,h}$) and vertical ($\epsilon_{c,v}$, $\epsilon_{m,v}$) directions to describe their impact on the proposed method. Measurement errors also include noise, further divided into random noise, outliers, and occlusions. The *BrickTree* structure implicitly limits the influence of outliers and occlusions by only considering points within the *leafVoxel* regions. Points from low-level occlusions (e.g. equipment sitting on the road surface) could still trigger detection errors, but this type of noise is difficult to control on an active site. Additionally, the outlier correction step is designed to correct these errors as long as the occluded region is not much larger than the comparison neighbourhood. Consequently the following analysis of required input data quality only considers random noise (ϵ_n), which is assumed to be Gaussian and primarily affecting the vertical measurements.

Horizontal errors ($\epsilon_{c,h}$ and $\epsilon_{m,h}$) could result in detection errors, mostly along the boundaries of the road corridor. The *gridCellSize* parameter defines the proposed solution's sensitivity to horizontal error, with the following equation describing the relationship required to enable accurate detection:

$$\epsilon_{c,h} + \epsilon_{m,h} < \text{gridCellSize} \quad (4)$$

In practice, contract specifications dictate the acceptable level of construction error. The United Kingdom's Highways Agency sets the tolerance for $\epsilon_{c,h}$ at 25 mm (*Manual of Contract Documents for Highway Works, Volume 1, Series 0700*). Using the 0.5 m *gridCellSize* implemented in this study and the maximum acceptable $\epsilon_{c,h}$, as-built data with $\epsilon_{m,h}$ less than 47.5 cm is required, which is achievable using any of the technologies listed in Table 1.

Vertical errors ($\epsilon_{c,v}$, $\epsilon_{m,v}$, and ϵ_n) could result in detection errors within any branch of the *BrickTree*. The *searchDistance* parameter drives the proposed solution's vertical error sensitivity, and is limited to a maximum distance equal to the thickness of the thinnest road design layer to avoid overlapping *leafVoxels*. Equation 5 describes the relationship between this parameter and the input data error required to enable detection:

$$\epsilon_{c,v} + \epsilon_{m,v} + \epsilon_n < \text{searchDistance} \quad (5)$$

Again, contract specifications drive the acceptable $\varepsilon_{c,v}$ levels, with the UK's Highway's Agency defining the limits at +/- 6 mm for pavement, +/- 15 mm for base, and +/- 10 mm for subbase levels. Using the maximum allowable $\varepsilon_{c,v}$ of 15 mm and a notional 4 cm *searchDistance*, most points would have to be within a vertical measurement error ($\varepsilon_{m,v} + \varepsilon_n$) of 2.5 cm to facilitate detection. In this case, unmanned aerial photogrammetry and low-level LiDAR are the only feasible data collection methods per Table 1. The following section experimentally tests the viability of the proposed solution and determines the optimal values for the various parameters and classification rules mentioned in this section.

METHODOLOGY, EXPERIMENTS, AND RESULTS

The authors generated simulated point cloud data to test the feasibility of the proposed solution prior to conducting further verification experiments on real-world data. The simulated data allowed isolation and testing of the proposed solution's performance in the presence of typical construction and as-built measurement errors, while controlling for occlusions and clutter in the observed scene. Although occlusions are to be expected on a construction site, controlling for them in these initial experiments allowed the authors to focus on the performance of the proposed solution under the conditions in which it is designed to operate: when the measured surface is visible in the scene. Robustness to expected levels of occlusion, and strategies for addressing errors caused by such occlusions will be addressed in further research.

Unmanned aerial photogrammetry was selected as the data collection technology for this study, based on its low cost, fast mobilization, and ability to meet required data quality thresholds. The authors developed a data simulation module to produce the synthetic aerial photogrammetry point cloud, using a 3D surface model of an in-progress road site as the input (Figure 5). The model, developed using AutoCAD Civil 3D, was a 500-meter long undivided crowned road with -2% camber in the transverse direction to either side of the road's centreline, and 2% grade in the longitudinal direction. Random construction errors were added to the surface layers within the allowable limits discussed in the previous section.

Simulated Aerial Photogrammetry Data

The aerial data simulation module aims to produce point clouds that are qualitatively similar to real-world aerial photogrammetry data in measurement error profile and point density. Error in photogrammetric surveying products depends on a wide range of factors, to include sensor quality, image resolution, range (i.e. how close the sensor is to

the scene), focal length, angle of incidence, percent overlap between adjacent images, level of texture in the scene, lens distortion, and the quality of camera calibration (Dai et al., 2014). A recent study (Slocum & Parrish, 2017) proposed an in-depth computer graphics workflow for generating simulated aerial photogrammetry data; a method that shows promise for isolating and analysing the contributing factors in photogrammetric error. However, the complexity of this approach was deemed unnecessary for the purposes of this study, and the authors opted instead for a simpler approach that models photogrammetric error as a function of Ground Sample Distance (GSD); a descriptor that accounts for sensor quality, resolution, focal length, and range. Focusing on these factors is a reasonable simplification, as a number of other contributing factors can be controlled using effective data collection planning and established sensor calibration routines. GSD describes the size of an image pixel projected onto the observed ground surface, and is calculated as (Pix4d, 2017):

$$GSD = \frac{w_s \cdot h}{w_l \cdot f} \quad (6)$$

where w_s is the width of the sensor in meters, w_l is the width of the image in pixels, h is the data collection height in meters, and f is the focal length in meters. A recent white paper on a state-of-the-art aerial photogrammetry system reported horizontal root-mean-square errors (RMSE) in the 1 – 7 pixel range ($\mu = 2.7$, $\sigma = 1.5$) and vertical RMSE in the 1 – 4.3 pixel range ($\mu = 2.4$, $\sigma = 1.1$) after analysing data produced at various heights and under differing weather/lighting conditions (Pauly, 2016). Using this as a guide, the aerial data simulation module generates each noisy point ($p_{i,\varepsilon}$) in the following manner:

$$p_{i,\varepsilon} = p_i + \varepsilon_{xy} \hat{v}_{xy} + \varepsilon_z \hat{v}_z \quad (7)$$

where $p_i \in P$, $\varepsilon_{xy} \sim N(0, 2.7 \cdot GSD)$, and $\varepsilon_z \sim N(0, 2.4 \cdot GSD)$. Here, P is the set of points in the sampled cloud, \hat{v}_{xy} is a randomly-generated normalized vector in the xy -plane ($\langle x, y, 0 \rangle$) and \hat{v}_z is the vector $\langle 0, 0, 1 \rangle$.

To produce P , the algorithm randomly samples points on the input triangulated surface model (e.g. Figure 5) according to the user-specified density profile, defined by an input mean and standard deviation. This random sampling aims to mimic variations in point density attributable to changes in image texture while allowing the user to control the mode of these variations. This approach relies on the assumption that the as-built site contains sufficient texture for photogrammetric reconstruction, and that the density variations can be adequately described by a single normal distribution. Multimodal variations could be modelled by splitting the input surface into regions where

different materials are expected and assigning separate input parameters to each prior to simulation. Since this study is primarily focused on detecting asphalt road design layers, the uniform image texture within these regions should result in point densities that can be described by a unimodal normal distribution. The next section examines the veracity of this assumption. The simulated data produced for the purposes of this study was generated using a GSD of 2 cm with a density distribution of $\mu = 200$ pts/m² and $\sigma = 1.5$.

Real World Data

The authors collected as-planned data on two separate days during the construction of a residential road for a new development in Cambridge, UK. The asphalt road design included five distinct surfaces: (1) the formation (bottom of excavation), (2) a 520 mm thick sub-base layer, (3) a 125 mm thick base course, (4) a 65 mm thick asphalt binder course, and (5) a 40 mm thick asphalt wearing course. Autodesk's *Civil 3D* application was used to generate a 3D corridor model from the 2D design information provided by the contractor. The wearing course thickness limits the *searchDistance* parameter to a maximum of 4 cm, which requires most points to have a vertical measurement error of less than 2.5 cm to enable accurate detection.

Trimble's UX5 fixed wing UAS conducted the aerial surveys, the details of which are provided in Figure 6. The authors used the following steps to plan the data collection: (1) Select a desired GSD to achieve the required accuracy, considering typical error ranges discussed in the previous section, (2) Select a desired data collection height using Equation 6, taking into consideration local aviation authority regulations and on-site structures, equipment, and obstructions, (3) Define the desired degree of overlap between adjacent images, (4) Plan the flight route to achieve the parameters defined in the previous three steps, and (5) Establish ground control point (GCP) targets throughout the site to enable accurate geo-referencing during post-processing. The authors used Trimble's UX5 flight planning software to complete step 4, using a desired elevation of 70 m and target image overlap of 80% in both the longitudinal and transverse directions. Eight GNSS-surveyed GCP targets were distributed throughout the site. The authors performed photogrammetric post-processing of the collected images in Trimble's Business Center (TBC) aerial photogrammetry application. The process involved: (1) importing the images and synchronized on-board sensor data, (2) refining the sensor-defined camera positions using manual target observations in at least 3 images per GCP, and (3) conducting a dense photogrammetric reconstruction to produce the final survey products. One issue that can affect photogrammetric reconstruction quality on road sites is the potential for visual similarity between multiple regions

along the construction corridor, leading to errors in feature matching and calculation of external camera parameters. The employed method accounts for this by logging the UX5's GPS, altimeter, and inertial measurement unit (IMU) sensor readings at each image location, and then enforcing a feature-matching constraint that declares positive matches can only occur between images with overlapping frames measured from their sensor-defined positions. The quality of the collected data is analysed and compared to the synthetic data in the following section. Figure 7 depicts the point clouds and ground truth conditions at the time of data collection.

Comparing the Simulated and Real World Datasets

The authors verified the simulated data's suitability by comparing it to the real world data, specifically examining the point density and point-to-ground-truth-surface distance statistics. This was done by constructing histograms of the data for each metric; normal distributions were then fit to the histograms to describe and visualize the results (Figure 8). Isolating photogrammetric measurement error in this case would require accurately modelling the construction error throughout the road corridor, which would in-turn require measurement and modelling techniques that introduce their own errors. To account for this, and make the following analysis an 'apples-to-apples' comparison, the authors computed the simulated data's point-to-surface distances relative to the errorless design surfaces. Consequently, the point-to-surface distributions reported below are composite values that include both measurement and construction vertical errors along the length of the corridor. Points attributable to occlusions in the real-world data were manually cropped prior to performing the analysis in order to focus only on the quality of the road surface measurements. This comparison confirms that the simulated data reasonably approximates the real-world aerial photogrammetric data. The real-world vertical errors were within $\sigma_{Day1} = 1.5 \text{ px}$ and $\sigma_{Day2} = 1.3 \text{ px}$ respectively, both of which are within the expected ranges discussed above.

Development Platform and Performance Measures

The authors developed the proposed solution using an in-house coding platform named Gygax that allows for processing and visualization of both images and PCD, and incorporates the open-source *Emgu CV* and *Point Cloud Library* code libraries. The solution uses a combination of C++ and C# code written and compiled in Microsoft Visual Studio 2015. All experiments utilized a computer with 4.0 GHz Intel i7 processor, 32 GB RAM, a dedicated 1,280-core GPU with 2 GB memory, and Windows 10 64-bit operating system.

This study used average precision (p_μ), average recall (r_μ), and the F score (F_1) to measure experimental performance by comparing the binary *leafVoxel* detection labels to the ground truth. The ground truth was developed by overlaying the project grid with the as-built data and manually labelling each branch with the observed design layer name. Equations 6-8 define these metrics for a model with l design surfaces, where true positive (TP_c) is the number of *leafVoxels* correctly labelled as 'detected' in surface c , false positive (FP_c) is the number incorrectly labelled as 'detected', true negative (TN_c) is the number correctly labelled as 'not detected', and false negative (FN_c) is the number incorrectly labelled as 'not detected'.

$$p_\mu = \frac{\sum_{c=1}^l TP_c}{\sum_{c=1}^l TP_c + FP_c} \quad (8)$$

$$r_\mu = \frac{\sum_{c=1}^l TP_c}{\sum_{c=1}^l TP_c + FN_c} \quad (9)$$

$$F_1 = \frac{2 \times p_\mu \times r_\mu}{p_\mu + r_\mu} \quad (10)$$

p_μ measures the reliability of the positive-detection decision as the portion of *leafVoxels* labelled 'detected' that were detectable in the as-built scene. r_μ measures the proposed solution's positive-detection effectiveness as the portion of detectable *leafVoxels* that were labelled as 'detected'. Each of these metrics describes a different component of the proposed solution's overall effectiveness and optimal performance is achieved when each is maximized. In practice, this is difficult because there are often trade-offs between precision and recall. F_1 is the harmonic mean of p_μ and r_μ , and is used to measure overall performance in the following experiments.

Experiments on Synthetic Data

The first experiments conducted on the synthetic data aimed to determine which branch detection deconfliction rule produces the best results, and how variations in the voxel height (*searchDistance*) affect those results. The authors conducted 20 trials for each decision rule tested (60 in total), varying the *searchDistance* between 0.1 and 2.0 cm in 0.1 cm increments. As discussed previously, the wearing course's 40 mm design thickness drove selection of the *searchDistance* values tested. Figure 9 shows the F1 Score achieved in each of these trials. The maximum point decision rule resulted in the highest F1 Score in each case. Additionally, the authors concluded that the minimum distance rule is not reliable for deconflicting branch detection decisions, as it demonstrated no ability to improve detection performance. Finally, the results in Figure 9 show a preference for the maximum-possible voxel height,

using a search distance of 2 cm in this case. Using the optimal decision rule (maximum point) and search distance (2 cm), the authors recorded an average F_1 Score of 91.0% with the misclassifications occurring almost exclusively in the upper-two design layers (77.1% and 80.8% F_1 Scores respectively for the Binder and Asphalt Wearing Surface layers). This indicates that the normally-distributed measurement errors cause confusion in the classification decision for thinner layers. The following experiments aimed to correct these errors using the outlier correction step described in the previous section.

To examine the influence of the design parameters on the outlier correction step's performance, the authors conducted further trials iterating through *neighbourDistance* values from 1 to 10 while varying the *consensusThreshold* between 0.5 and 1.0 in increments of 0.05. The range of *neighbourDistance* values tested was determined experimentally to ensure an observable peak in F_1 Score. The authors chose to use 0.5 as the minimum *consensusThreshold* assuming that the most likely scenario would involve a neighbourhood with just two surface layers under consideration at a time. Figure 10 shows the results of these experiments. Peak performance was achieved using a neighbour distance of 2 and consensus threshold of 60%, which resulted in an average F_1 Score of 99.7%. The results confirm the utility of the proposed solution under ideal conditions, when the constructed surfaces are clearly visible. The following section examines the proposed solution's performance on the real-world datasets.

Experiments on Real-World Data

The data collection surface on Day 1 was subject to a number of occlusions caused by a parked vehicle, a roller compacter, and multiple safety barriers associated with work related to other schedule tasks. Additionally, the contractor intentionally built three regions of the binder material to a lower-than-designed elevation. These regions corresponded to areas where connecting roads were awaiting their asphalt binder installation, during which time the contractor would correct the intentionally-low regions to provide a smooth connection. The initial real-world-data experiments eliminated these regions and occluded areas, in both datasets, from the analysis in order to isolate the method's performance on regions where the ground truth surfaces are observable in the data. Strategies for dealing with these troubling regions are then discussed in subsequent paragraphs, and are the focus of continuing research. Figure 11 illustrates the location and type of some of the occlusions. The authors implemented the proposed solution on the Day 1 and Day 2 datasets using the optimal parameters from synthetic data testing. Table 2 summarizes the results of these experiments. The overall results show reasonably good performance, but further improvement is

needed. The reduced performance compared to the synthetic data is due to localized error regions in the ground-truth Asphalt Binder and Wearing Surface areas (Figure 12 highlights some typical examples). These localized errors could be due to unplanned deviations in surface elevation (i.e. construction errors) or systematic errors in the point cloud data. Regardless of the cause, a progress monitoring approach should be able to recognize which design layers these regions are supposed to belong to. This also holds true for the regions that are occluded or otherwise deviate from the design layer elevations. The following experiments examined how such regions could be accounted for using the proposed outlier correction method.

Adjusting for error regions and occlusions

The authors experimented with increasing the outlier correction method's neighbour distance parameter to compensate for the error and occluded regions. The combinations of outlier correction parameters producing an F1 Score above 95% in the synthetic data (as shown in Figure 10), were implemented on the Day 1 and Day 2 datasets for search distances greater than 2. The best overall performance resulted from a search distance of 7 with a consensus threshold of 55%. Table 3 shows the results for this combination, both with and without the occlusions accounted for in the performance calculations. Figure 13 illustrates the results.

Processing Time

The proposed solution took approximately 11 seconds to generate the incremental surface detection decisions for the 200 m section of road analysed above, using a 5.5-million-point input cloud. Note that this does not include the *BrickTree* build time (~40 seconds), as this step only needs to be completed at project initialization or when the design is changed. The total cycle time (including *BrickTree* initialization, data collection, and post-processing) was approximately 3 hours, 35 minutes on Day 1, and 2 hours, 29 minutes on Day 2. The increased efficiency on Day 2 was due mostly to better familiarity with the post-processing software and field equipment. The authors tested the proposed solution on additional synthetic data in order to evaluate how changes in the size of the as-built and as-planned data affect its processing time, the results of which are summarized in Figure 14. Note that *BrickTree* initialization time (not shown) also increased linearly with projects size, with the 10-Lane, 500 m long road requiring 2 minutes, 8 seconds to initialize the data structure.

SUMMARY AND CONCLUSION

Effectively monitoring construction progress can enable timely and effective project control decisions aimed at keeping a project on track. Unfortunately, current transportation project monitoring practice is manual, error-prone, inefficient, and ultimately contributes to the annual \$114.3B in global cost overrun. Automating this process can improve overall project performance, and detecting design surfaces in digital as-built data is a key requirement for realizing such an automated approach. A number of recent studies proposed methods for conducting this detection on building structural components. Unfortunately, these methods are not well-suited for larger and more complex layered road design surfaces. The only related transportation-focused study required labelled as-built data, entirely skipping the automated surface detection process.

This paper marks the first study specifically aimed at automatically detecting layered road design surfaces in unlabelled as-built point cloud data. The authors proposed and evaluated a novel model-guided space partitioning hierarchical tree structure, termed the *BrickTree*, for accomplishing this task. A simulated aerial photogrammetry point cloud was generated and used to test the performance of the proposed solution under ideal conditions not subject to occlusions or construction errors beyond acceptable levels. These initial experiments culminated in an average F_1 Score of 99.7%, while providing perspective on the best decision rule and outlier correction parameters to use for further testing. The solution was then tested on two real-world datasets, ultimately resulting in an average F_1 Score of 95.2%.

It is important to note that the proposed solution does have limitations. This study focuses on progress monitoring, and as such does not aim to identify or classify construction errors. It requires the constructed surfaces to be mostly within established quality limits. The authors acknowledge the natural link between progress and quality, and note that large-scale quality measurement is inherent in the proposed *leafVoxel* populating and *rasterBranch* classifying steps, since as-built points lying outside the model-hugging voxel regions will not be considered. In this manner, road layers that are built with errors exceeding the acceptable thresholds in large areas will not be detected by the method, and this lack of detection could be used to highlight potential construction errors for further investigation by the project management team. Another limitation is that the proposed solution requires up-to-date design information that reflects field-level construction procedures. For example, consider the regions discussed earlier that were intentionally left lower than the design elevations to account for connection with intersecting roads

that had yet to be paved. Modelling this level of detail is possible, but requires close coordination between the design and construction teams. This study's main contributions are:

(1) The *BrickTree* space partitioning hierarchical data structure, which has numerous advantages. First, the sparse nature of the hierarchical tree structure, utilizing a fixed tree depth where voxels are only generated at known surface geometry locations, reduces the complexity of tree traversal and search operations while simultaneously reducing memory usage compared to equivalent non-sparse voxelization approaches (Laine & Karras, 2011). Second, the uniform fit of each voxel to its parent surface geometry, thanks to the model-guided construction topology, ensures a consistent detection framework across the entire surface. This improves the consistency of point-to-surface attribution and eliminates the need for an as-planned traversal of the tree structure (because it's already known which design surfaces belong to which voxels). Third, the ability to adjust voxel shape allows for point searches in tighter vertical tolerances while maintaining broader horizontal ranges that enable consideration of sparser as-built point clouds. Finally, the layered vertical branch orientation enables logical reasoning about the presence of a design surface based on the status of the other voxels within the branch and its' neighbours.

(2) Empirical evidence supporting the optimal *searchDistance* parameter and branch classification decision rule to use when implementing the proposed solution. The authors noted a preference in the results for use of the largest *searchDistance* possible based on the thinnest design surface layer. However, further research is needed to identify how well these conclusions generalize when applied to different as-built and as-planned datasets.

(3) A workflow for generating synthetic as-built road construction point cloud data that is qualitatively similar to real-world aerial photogrammetric data. The developed application allows the user to specify parameters controlling the synthetic point cloud's noise/error and density distribution profiles while also modelling construction errors. This can be used to test the viability of a proposed method and/or its sensitivity to fluctuations in data quality. It can also be used to determine flight parameters and equipment requirements (using ground sample distance) for achieving the level of data quality needed for a given method.

While the results reported here are promising, there remains room for improvement. The authors divide the remaining errors, as visible in the Day 1 results at the top of Figure 13, into two different categories. The first is confusion between the Asphalt Binder and Wearing Surface layers. These surfaces are separated by the thinnest margin, i.e. the thickness of the Wearing Surface layer, and are thus more susceptible to misclassification errors due

to noise in the PCD and construction errors near the acceptable limits. While more accurate and precise as-built data collection methods (e.g. laser scanning) could ameliorate this situation, UAS photogrammetry offers several advantages that warrant further investigation into methods for dealing with such misclassifications in the aerial PCD. The biggest advantages of UAS photogrammetry include the lower cost and ability to collect the data remotely. The latter advantage improves site safety by not exposing further personnel to the hazards inherent on a construction site, and ensures the data collection will not impede ongoing work.

The second error category is confusion at the boundary between adjacent layers. Note that this issue did not emerge while testing the synthetic data using a smaller neighbour distance during outlier correction. However, as the authors increased the neighbour distance to accommodate larger error regions than modelled in the synthetic data, the boundaries between adjacent layers became less well defined. For example, consider Figure 13 which shows that the staggered boundary between the Base Course and Asphalt Binder layers in the Day 1 data is lost using the larger neighbour distance. A more sophisticated outlier correction method could potentially be used to account for initial classification errors while maintaining the boundary conditions in the as-built data. Both of these error categories will be further examined in the authors' future research.

ACKNOWLEDGMENTS

This research is made possible through funding from the United States Air Force and the Cambridge Commonwealth and International Trust. The authors express gratitude to the Trimble Corporation for their support in lending equipment and expertise to the data collection operation. The views expressed in this paper are those of the authors and do not necessarily reflect the official policy or position of the Air Force, the Department of Defense or the U.S. Government. All supporting data is included in this paper and the provided references.

REFERENCES

- Bosché, F. (2010). "Automated recognition of 3D CAD model objects in laser scans and calculation of as-built dimensions for dimensional compliance control in construction." *Advanced Engineering Informatics*, 24 (1), 107–118.
- Braun, A., Tuttas, S., Borrmann, A. & Stilla, U. (2015). "Automated progress monitoring based on photogrammetric point clouds and precedence relationship graphs." *Proc., 32nd International Symposium on Automation and Robotics in Construction and Mining*, 274–280.
- Cantarelli, C.C., Flyvbjerg, B., Molin, E.J.E. & van Wee, B. (2010). "Cost Overruns in Large-Scale Transportation

584 Infrastructure Projects: Explanations and Their Theoretical Embeddedness." *European Journal of Transport and*
585 *Infrastructure Research*, 10 (1), 5–18.

586 Chan, D.W. & Kumaraswamy, M.M. (1997). "A comparative study of causes of time overruns in Hong Kong
587 construction projects." *International Journal of Project Management*, 15 (1), 55–63.

588 Cheng, M.Y. & Chen, J.C. (2002). "Integrating barcode and GIS for monitoring construction progress." *Automation*
589 *in Construction*, 11 (1), 23–33.

590 Dai, F., Feng, Y. & Hough, R. (2014). "Photogrammetric error sources and impacts on modeling and surveying in
591 construction engineering applications." *Visualization in Engineering*, 2 (1), 1–14.

592 Del Pico, W.J. (2013). *Project Control: Integrating Cost and Schedule in Construction*. Hoboken, NJ: John Wiley &
593 Sons.

594 DJI (2017). "DJI Site Survey Solution". <http://store.dji.com/product/site-survey-solutions-single-perpetual> (5
595 March 2017).

596 Dobbs, R., Pohl, H., Lin, D.-Y., Mischke, J., Garemo, N., Hexter, J., Matzinger, S., Palter, R. & Nanavatty, R.
597 (2013). *Infrastructure productivity: how to save \$1 trillion a year*. McKinsey Global Institute.

598 Faro (2017). "Laser Scanner Tech Sheets". <http://www.faro.com/en-gb/resources/tech-sheets/> (13 November
599 2017).

600 Golparvar-Fard, M., Peña-Mora, F. & Savarese, S. (2015). "Automated Progress Monitoring Using Unordered Daily
601 Construction Photographs and IFC-Based Building Information Models." *Journal of Computing in Civil*
602 *Engineering*, 10.1061/(ASCE)CP.1943-5487.0000205, 04014025.

603 GOM Inspect (2017). "GOM Inspect Software". <http://www.gom.com/3d-software/gom-inspect.html> (29 April
604 2017).

605 González, P., González, V., Molenaar, K. & Orozco, F. (2014). "Analysis of Causes of Delay and Time Performance
606 in Construction Projects." *Journal of Construction Engineering and Management*, 10.1061/(ASCE)CO.1943-
607 7862.0000721, 04013027.

608 Han, K.K., Degol, J. & Golparvar-Fard, M. (2018). "Geometry- and Appearance-Based Reasoning of Construction
609 Progress Monitoring." *Journal of Construction Engineering and Management*, 10.1061/(ASCE) CO.1943-
610 7862.0001428, 04017110.

611 Highways Agency, UK (2014). "Manual of Contract Documents for Highway Works, Volume 1, Series 0700",
612 London, UK.

613 Kalasapudi, V.S., Tang, P. & Turkan, Y. (2017). "Computationally efficient change analysis of piece-wise
614 cylindrical building elements for proactive project control." *Automation in Construction*, 81, 300–312.

615 Kim, C., Son, H. & Kim, C. (2013). "Automated construction progress measurement using a 4D building
616 information model and 3D data." *Automation in Construction*, 31, 75–82.

617 Kivimäki, T. & Heikkilä, R. (2015). "Infra BIM based Real-time Quality Control of Infrastructure Construction
618 Projects." *Proc., 32nd International Symposium on Automation and Robotics in Construction and Mining*, 877–882.

619 Laine, S. & Karras, T. (2011). "Efficient sparse voxel octrees." *IEEE Transactions on Visualization and Computer
620 Graphics*, 17 (8), 1048–1059.

621 Leica (2017). "Leica Pegasus:Two Mobile Reality Capture." <http://www.leica-geosystems.co.uk> (13 November
622 2017).

623 Memon, A.H., Abdul Rahman, I. & Abdul Aziz, A.A. (2012). "The cause factors of large project's cost overrun: a
624 survey in the southern part of Peninsular Malaysia." *International Journal of Real Estate Studies (INTREST)*, 7 (2),
625 1–15.

626 Navon, R. & Sacks, R. (2007). Assessing research issues in Automated Project Performance Control (APPC).
627 *Automation in Construction*, 16 (4), 474–484.

628 NCHRP (2013). "Synthesis 446: Use of Geospatial Data, Tools, Technologies, and Information in Department of
629 Transportation Projects." Washington, D.C.

630 Olsen, M.J., Roe, G. V., Glennie, C., Persi, F., Reedy, M., Hurwitz, D., Williams, K., Tuss, H., Squellati, A. &
631 Knodler, M. (2013). "NCHRP Report 748: Guidelines for the Use of Mobile LIDAR in Transportation
632 Applications." Washington, D.C.

633 Pauly, K. (2016). "White Paper: Trimble UX5 - Increasing Your Productivity." Trimble Corporation, Belgium.

634 Pix4d (2017). "Pix4D Drone Mapping." <https://pix4d.com/> (7 July 2015).

635 Razavi, S.N. & Haas, C.T. (2010). "Multisensor data fusion for on-site materials tracking in construction."
636 *Automation in Construction*, 19 (8), 1037–1046.

637 Razavi, S.N. & Moselhi, O. (2012). "GPS-less indoor construction location sensing." *Automation in Construction*,
638 28, 128–136.

639 Salling, K.B. & Leleur, S. (2015). "Accounting for the inaccuracies in demand forecasts and construction cost
640 estimations in transport project evaluation." *Transport Policy*, 38, 8–18.

641 Schneider, C. (2013). "Techbrief: 3D, 4D, and 5D Engineered Models for Construction, an Executive Summary."
642 US Department of Transportation, Federal Highway Administration.

643 Shahi, A., Aryan, A., West, J.S., Haas, C.T. & Haas, R.C.G. (2012). "Deterioration of UWB positioning during
644 construction." *Automation in Construction*, 24, 72–80.

- Siemiatycki, M. (2009). "Academics and Auditors: Comparing Perspectives on Transportation Project Cost Overruns." *Journal of Planning Education and Research*, 29 (2), 142–156.
- Slocum, R. & Parrish, C. (2017). "Simulated Imagery Rendering Workflow for UAS-Based Photogrammetric 3D Reconstruction Accuracy Assessments." *Remote Sensing*, 9 (4), 396.
- The Survey Association (2015). "Client Guide to Aerial LiDAR Surveys." Newark-on-Trent, UK.
- The Survey Association (2016). "Client Guide to Small Unmanned Aircraft Surveys." Newark-on-Trent, UK.
- TransMagic (2017). "MagicCheck – CAD Model Comparison." <https://transmagic.com/magiccheck-cad-model-comparison/> (29 April 2017).
- Trimble (2015). "Trimble TX8 Laser Scanner Datasheet." [Online]. <http://www.trimble.com/3d-laser-scanning/index.aspx> (16 July 2015).
- Trimble (2017). "Trimble Business Center Aerial Photogrammetry Module." [Online]. <http://uas.trimble.com/tbc-am> (29 April 2017).
- Turkan, Y., Bosché, F., Haas, C.T. & Haas, R. (2014). "Tracking of secondary and temporary objects in structural concrete work." *Construction Innovation*, 14 (2), 145–167.
- Tuttas, S., Braun, A., Borrmann, A. & Stilla, U. (2014). "Comparison of Photogrammetric Point Clouds with BIM Building Elements for Construction Progress Monitoring." *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, XL-3, 341–345.
- U.S. Army Corps of Engineers (2013). *Engineering and Construction Bulletin 2013-18*. (ECB 2013-18). Washington, D.C.
- Watt, A. (2000). *3D Computer Graphics*. Third. Essex, UK: Pearson Education.
- Zhang, C. & Ardit, D. (2013). "Automated progress control using laser scanning technology." *Automation in Construction*, 36, 108–116.

TABLES

Table 1. Summary of accurate and dense spatial data collection technologies.

Technology	Cost	Accuracy	Density	Strengths	Limitations
Manned aerial LiDAR ^{a,c,d}	\$\$	cm	10s - 100s	Lighting invariance, large coverage area, non-intrusive	Cost, mobilization time
Unmanned aerial LiDAR ^{a,c,d}	\$	mm - cm	100s	Lighting invariance, non-intrusive	Battery life
Unmanned aerial photogrammetry ^{a,b,e}	< \$	cm	100s	Cost, non-intrusive	Battery life, daylight collection only
Mobile LiDAR ^{a,c,f}	\$ - \$\$	cm	100s - 1,000s	Lighting invariance, coverage area	Cost, intrusive, ground view more sensitive to occlusions
Terrestrial LiDAR ^{a,g,h}	\$	mm	100s - 10,000s	Accuracy, density, Lighting invariance	Coverage, intrusive, ground view more sensitive to occlusions

Note: Cost ranges are up-front, \$ indicates costs in the \$10,000s and \$\$ indicates costs in the \$100,000s. Density ranges are in pts/m². The intrusive/non-intrusive assessment considers whether or not the technology requires on-site collection in the proximity of ongoing work, which has implications for safety and work productivity.

Sources: ^aNCHRP (2013), ^bThe Survey Association (2016), ^cOlsen et al. (2013), ^dThe Survey Association (2015b), ^eDJI (2017), ^fLeica (2017), ^gTrimble (2015), ^hFaro (2017)

Table 2. Summary of initial results on the Day 1 and Day 2 datasets

Dataset	p_{μ}	r_{μ}	FI
Day 1	83.6%	82.8%	83.2%
Day 2	84.7%	84.7%	84.7%
Overall	84.2%	83.8%	84.0%

Table 3. Summary of final results on the Day 1 and Day 2 datasets without (and *with*) occlusions considered

Dataset	p_{μ}	r_{μ}	FI
Day 1	90.2% (83.6%)	89.8% (83.2%)	90.0% (83.4%)
Day 2	99.9% (99.9%)	99.9% (99.8%)	99.9% (99.9%)
Overall	95.3% (91.7%)	95.1% (91.5%)	95.2% (91.6%)

Figure Captions

Figure 1. Longitudinal road design section showing 2D projection of 10 cm voxel grid

Figure 2. Proposed Surface Detection Process

Figure 3. The BrickTree Structure

Figure 4. Neighbourhood construction example for *neighbourDistance* = 1

Figure 5. Simulated In-Progress Road Construction Site

Figure 6. UAS Flight Details

Figure 7. As built point clouds and ground truth conditions

Figure 8. Comparison of the real world and simulated datasets

Figure 9. Performance for the tested search distances and decision rules

Figure 10. Performance for each combination of outlier correction parameters tested

Figure 11. Occlusions and deviations in the Day 1 data (ground truth surfaces shown)

Figure 12. Localized error regions (circled) in the Day 1 dataset

Figure 13. Day 1 and Day 2 surface detection results after increasing the *searchDistance* parameter

Figure 14. Complexity of the proposed solution

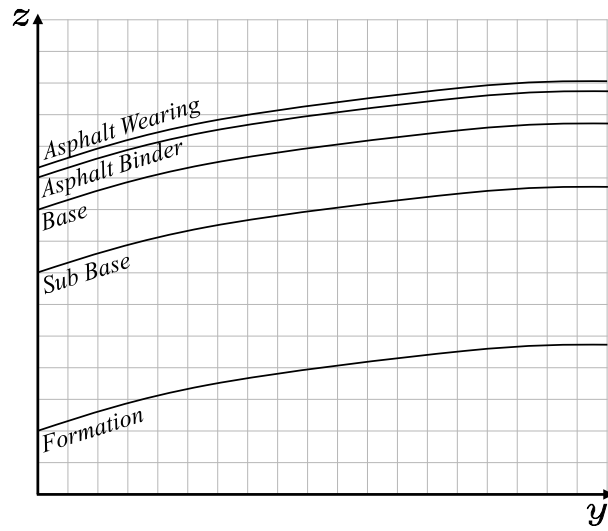
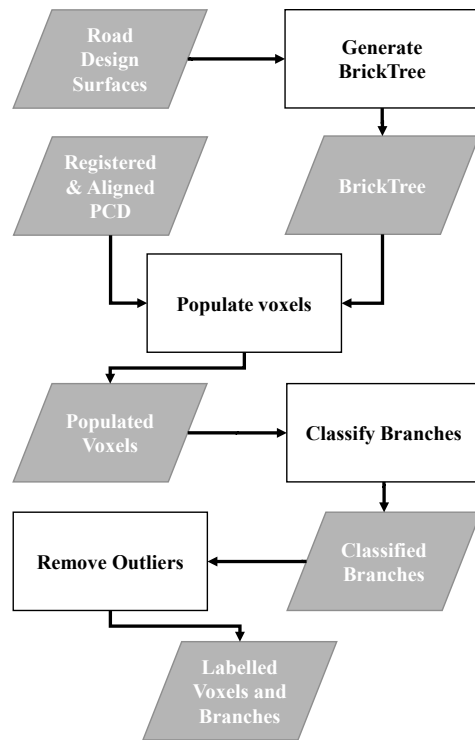


Figure 1. Longitudinal road design section showing 2D projection of 10 cm voxel grid





Legend:



 Process Inputs/Outputs  Process

Figure 2. Proposed Surface Detection Process



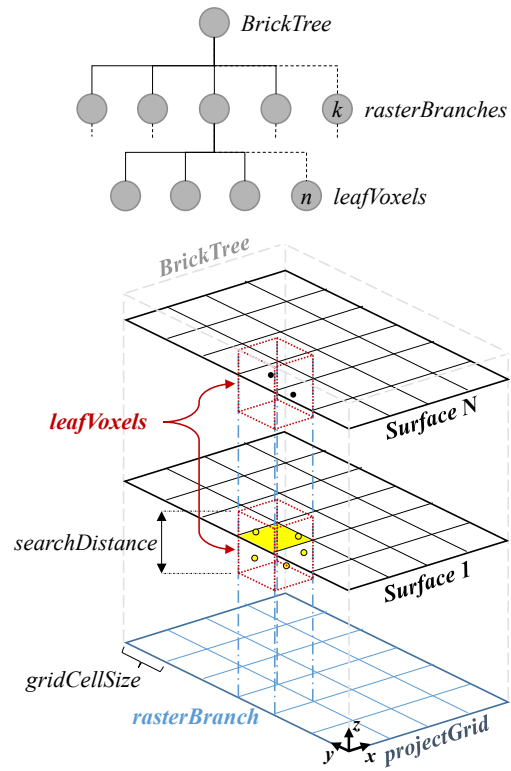


Figure 3. The BrickTree Structure



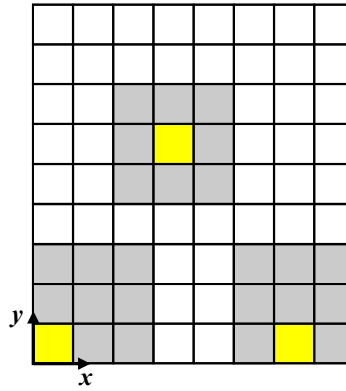


Figure 4. Neighbourhood construction example for $\text{neighbourDistance} = 1$



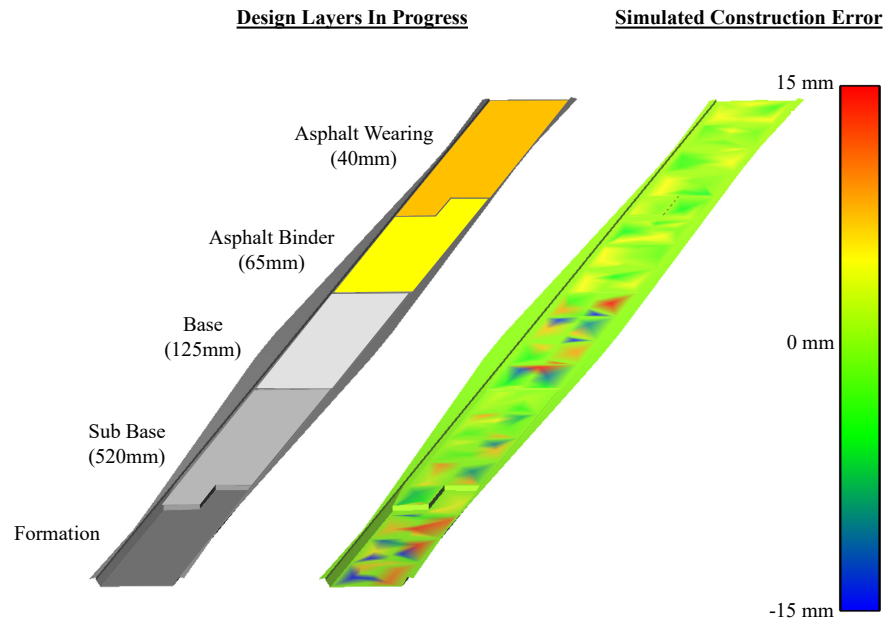


Figure 5. Simulated In-Progress Road Construction Site





<u>Flight Information</u>	<u>Day 1</u>	<u>Day 2</u>
Avg. Flying Height AGL (m):	70.3	70.7
No. of Images:	224	251
Avg. Photo Scale:	1 : 4799	1 : 4883
Avg. GSD (cm):	2.29	2.33
Pre-Flight Setup (min)	45	27
Flight Duration (min)	11	12
Post Processing (min)	158	110

Figure 6. UAS Flight Details



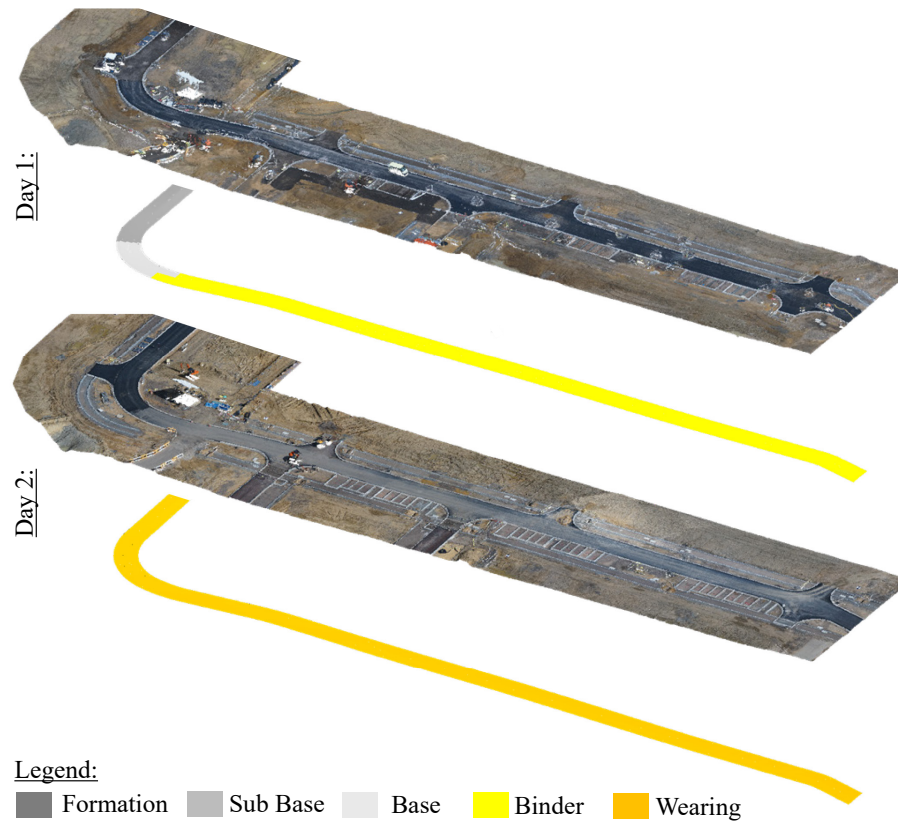


Figure 7. As built point clouds and ground truth conditions



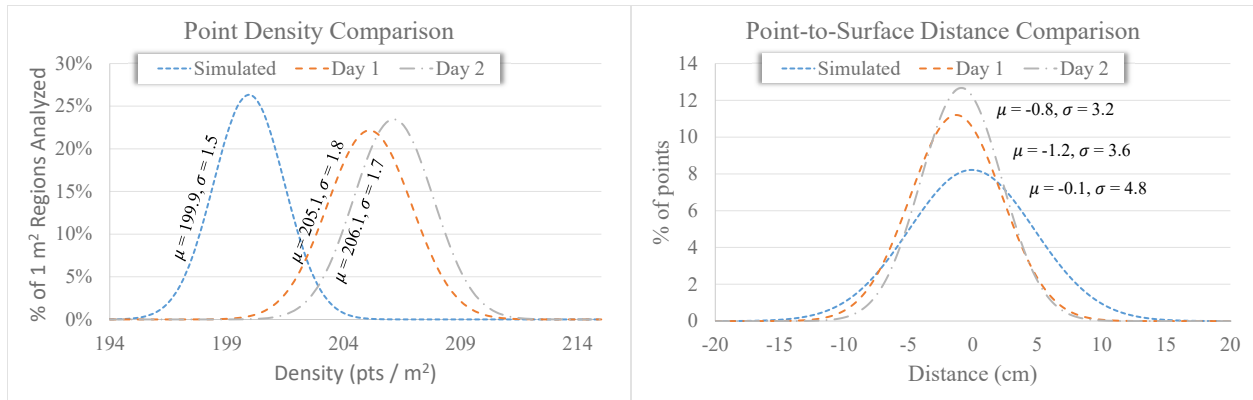


Figure 8. Comparison of the real world and simulated datasets



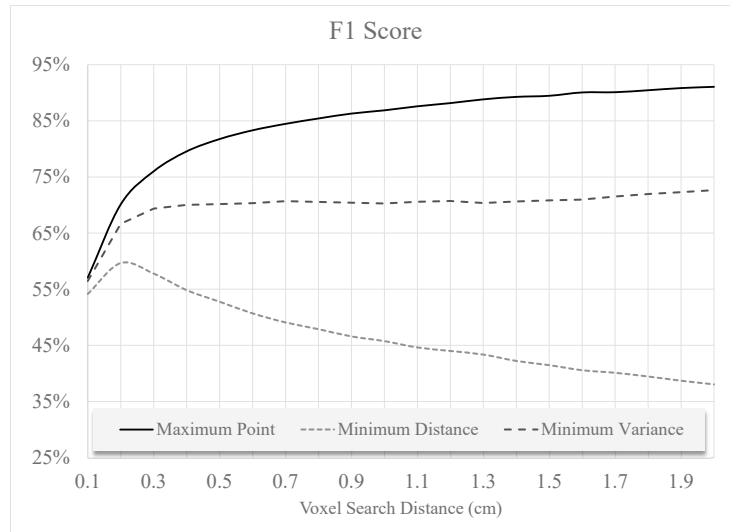


Figure 9. Performance for the tested search distances and decision rules



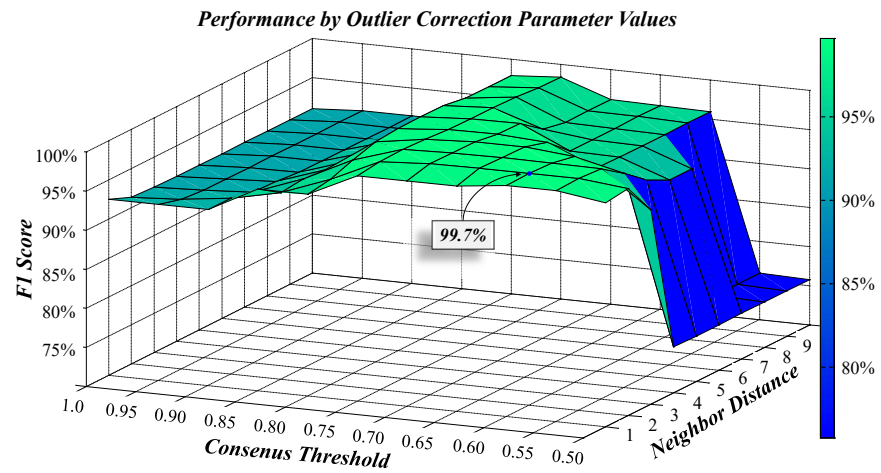


Figure 10. Performance for each combination of outlier correction parameters tested



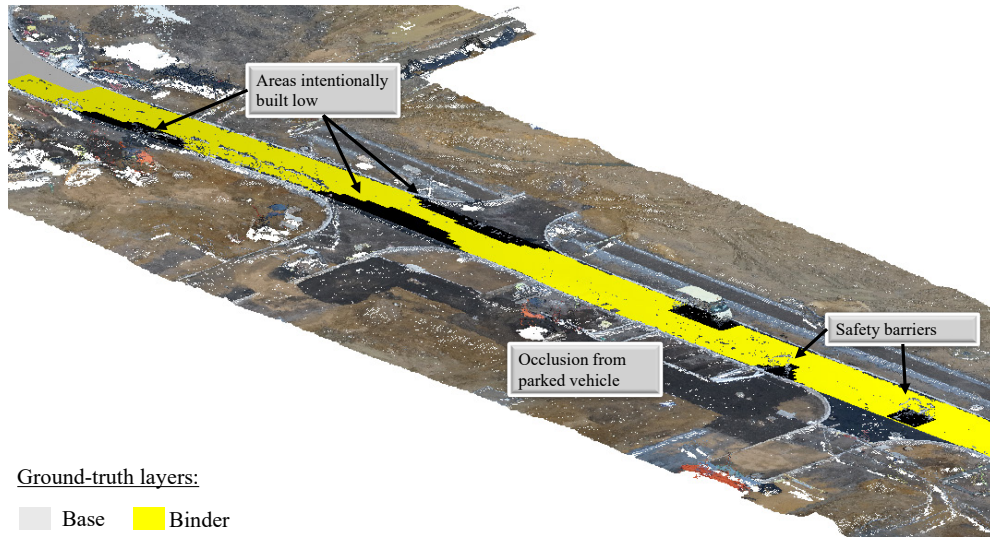


Figure 11. Occlusions and deviations in the Day 1 data (ground truth surfaces shown)



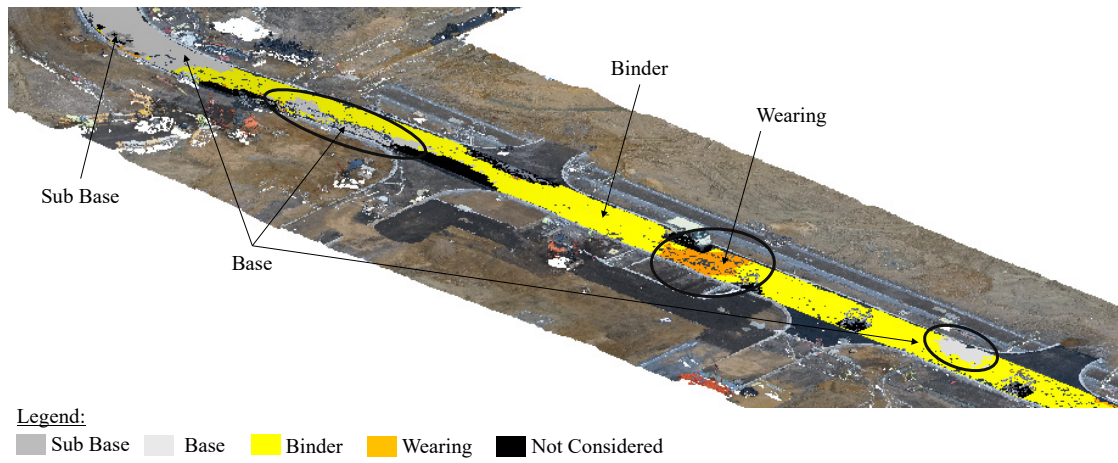
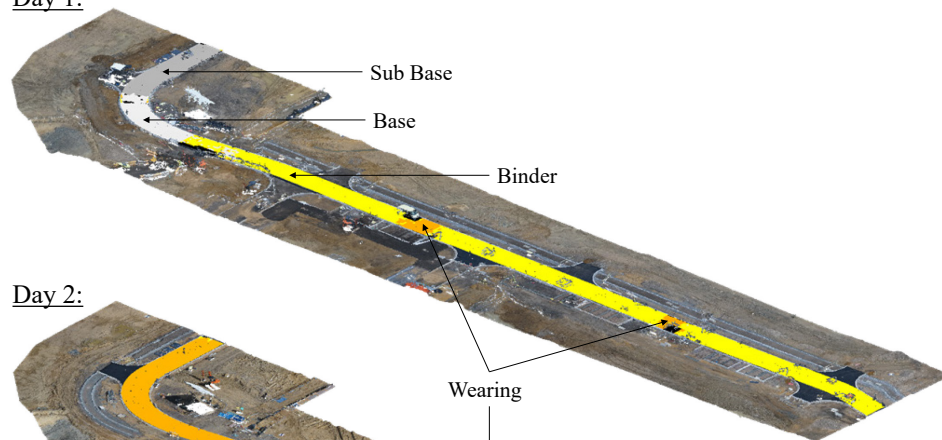


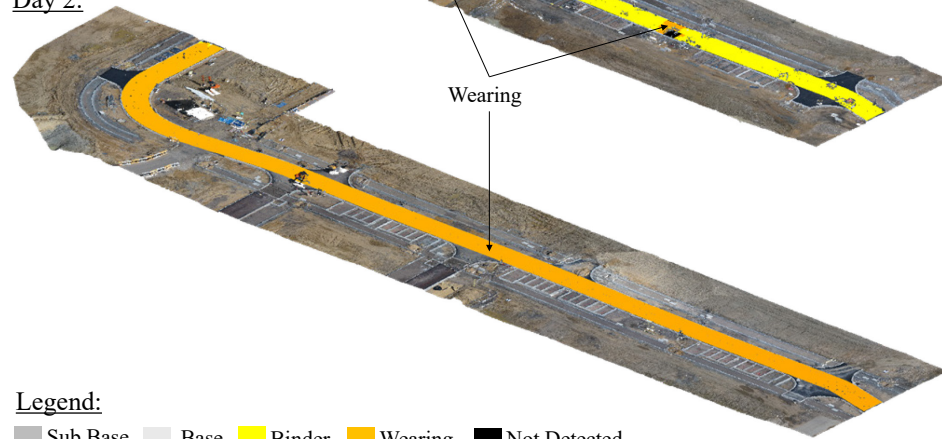
Figure 12. Localized error regions (circled) in the Day 1 dataset



Day 1:



Day 2:



Legend:

Sub Base Base Binder Wearing Not Detected

Figure 13. Day 1 and Day 2 surface detection results after increasing the searchDistance parameter



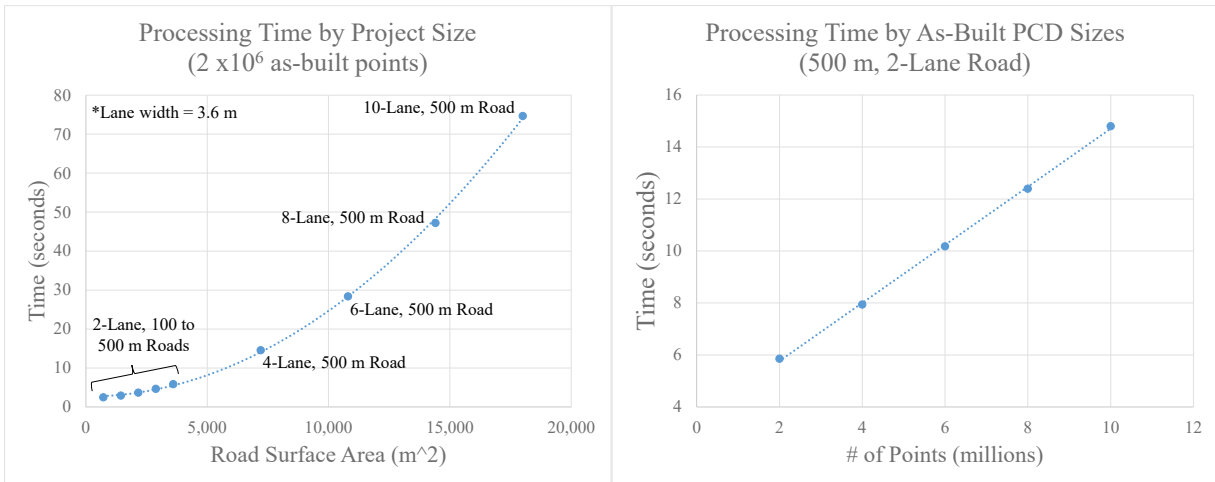


Figure 14. Complexity of the proposed solution

