This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2024.3372064

# Instant Segmentation and Fitting of Excavations in Subsurface Utility Engineering

Marco Stranner, Philipp Fleck, Dieter Schmalstieg and Clemens Arth

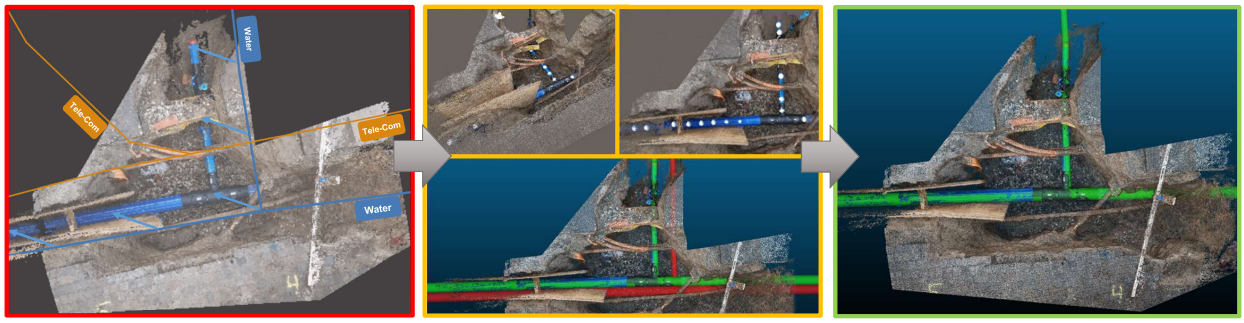



Figure 1: Left: top-down view of a 3D point cloud reconstruction with inaccurate SUE data visualized for different infrastructure types as simple line strings. Middle: Segmentation results from virtual cameras (top), highlighting inaccurate (red) and aligned (green) infrastructure data. Right: fitted 3D infrastructure model shown on a top-down view of a 3D point cloud reconstruction.

**Abstract**— Using augmented reality for subsurface utility engineering (SUE) has benefited from recent advances in sensing hardware, enabling the first practical and commercial applications. However, this progress has uncovered a latent problem – the insufficient quality of existing SUE data in terms of completeness and accuracy. In this work, we present a novel approach to automate the process of aligning existing SUE databases with measurements taken during excavation works, with the potential to correct the deviation from the as-planned to as-built documentation, which is still a big challenge for traditional workers at sight. Our segmentation algorithm performs infrastructure segmentation based on the live capture of an excavation on site. Our fitting approach correlates the inferred position and orientation with the existing digital plan and registers the as-planned model into the as-built state. Our approach is the first to circumvent tedious postprocessing, as it corrects data online and on-site. In our experiments, we show the results of our proposed method on both synthetic data and a set of real excavations.

**Index Terms**—Augmented Reality, Localization, 3D Models, Geometric Constraints, Segmentation, Infrastructure

## 1 Introduction

Most critical infrastructure today is buried underground, ranging from hard piping for water, waste water, thermal heating and gas, to soft cables for electricity and communication lines. Typically, such infrastructure is installed below paved roads and requires opening the surface for operations.

In *subsurface utility engineering* (SUE), the discrepancy between the as-planned and as-built is usually very noticeable. SUE generally lacks a standardized feedback loop, which would allow construction companies to report as-built information back into a common platform. Therefore, information on the exact placement and orientation of the infrastructure is mostly lost after the construction work is finished.

As a consequence, the completeness and accuracy of SUE data is very heterogeneous, limiting the value of SUE databases for both on-site and off-site applications.

Recent legislation in countries such as Denmark, the UK, or the Netherlands requires excavations related to subsurface infrastructure to be documented in a standardized way (*i.e.*, by images, videos, laser scans, and geo-referencing on site; see Fig. 3), providing a growing database of *corrected* SUE data. Unfortunately, even with the aid of state-of-the-art commercial visualization tools such as Trimble Sitevision[1], the data preparation process requires tedious data cleanup, which is usually performed off-site.

During excavation work, major tasks include the identification of existing piping and the retrieval, processing and mapping of these data to the real world. An important goal of these activities is to prevent damage to existing infrastructure, a frequent problem that costs billions annually [4, 32]. Previous work shows that AR can be effectively used to support SUE, for example, by using virtual spray paint marking or virtual daylighting [17]. Alas, AR applications are only as useful as the accuracy of the SUE data they use.

In this work, we propose a novel, instant approach that replaces the manual measuring and editing task on SUE data with an automated segmentation technique. This algorithm identifies pipes within the 3D model and correlates the resulting pipe models with existing noisy SUE data, such that the alignment to existing digital plans can be automated. The outcome is reprojected into excavation space using AR for a visual comparison between the SUE database and reality (Fig. 1). The main contributions of this work can thereby be summarized as follows:

- A novel and efficient segmentation approach for infrastructure in colorized 3D point cloud reconstructions on noisy, only partially visible data
- An efficient fitting approach to align the segmentation results to existing SUE data, existing as pure line strips only
- Mobile applicability of the approach providing real-time AR in the field;
- An evaluation of accuracy on synthetic data and real excavation data

• *M. Stranner, P. Fleck, C. Arth and D. Schmalstieg are with Graz University of Technology, Austria. E-mail: {m.stranner, philipp.fleck, arth, dieter}@icg.tugraz.at.*

*Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx*

[1] https://sitevision.trimble.com/

Our approach is the first to tackle the topic of subsurface infrastructure detection and correction in an instant, on-site and online fashion. Our experimental results show that commercial applications could derive significant benefit from further exploiting the proposed concept.

## 2 Related work

The advent of AR and easily accessible technologies for simultaneous localization and mapping (SLAM) has brought really useful applications in SUE to life. Hansen *et al*. [17] demonstrate how to combine high-precision GPS and SLAM to make subsurface infrastructure visually accessible. However, upon closer examination, two important limitations become apparent: (i) The application is crucially dependent on high-quality SUE data to be available. (ii) Relying on manual labor for the registration of SUE infrastructure defeats any hopes for scalability. The approach presented in this paper overcomes these two issues by aligning SUE datasets with a scanned excavation model, allowing the operator to immediately fix any deviations on site. In the remainder of this section, we will shed light on 3D segmentation and 3D fitting approaches and the current role of AR in SUE.

### 2.1 Segmentation of 3D structures

The segmentation of 3D structures, like point cloud data from photogrammetry or Lidar, usually relies on classification of input areas based on features such as normals, colors, or gradients. Since point cloud data tend to be noisy, sparse and unorganized, not all segmentation algorithms are equally suited for such a task. Nguyen *et al*. [35] present a survey of well-known algorithms, working on *edges* (regions are found by detecting boundaries), *regions* (neighborhood information is used to cluster similar regions), *attributes* (regions are clustered based on specific characteristics of the point cloud), *models* (primitive shapes are used for grouping) and *graphs* (a graph representation is used for efficient search in unstructured data, *e.g.*, to segment using surface normals). In the following, we discuss those choices, which are promising for infrastructure segmentation in SUE: region, model and graph algorithms.

Model-based segmentation Due to the repetitive pipe-like structure in SUE, a model-based approach seems most suitable. RANSAC-based algorithms such as the one by Fischer *et al*. [8] deliver good segmentation results and inspire algorithms for primitive shape detection [25, 46]. Another algorithm is the slippage analysis by Gelfand *et al*. [11], a model-based classification using so-called slippage shapes such as spheres, helix, planes, cylinders, and linear extensions. A Hough transform is commonly used for shape detection [36, 61] in 2D images, but was also extended to 3D. Planes, cylindrical shapes [40] and other primitives [41] can be used in a Hough transform. Kurdi *et al*. [55] found that RANSAC generally outperforms Hough-based alignment on 3D scans. Alas, in excavation works, it can be very hard to fit cylinders to partially visible pipe-like structures, since scans of pipes often miss the bottom half.

Region-based and graph-based segmentation Region-based methods are an established approach, usually relying on clustering of surface normals [7, 28, 58] or other geometric features. An octree-based approach was presented by Vo *et al*. [59] using region growing to compare similarity features like normals. KNN-trees over cylindrical regions were used by Lari *et al*. [29] to overcome inconsistent densities and to detect planar structures. Due to the known geometric properties of the pipes based on the planning data, the normals on a cylindrical surface could be used for segmentation. However, noise and reconstruction inaccuracies make such an approach impractical in SUE tasks.

Color-based segmentation, as proposed by Strom *et al*. [52], is more robust in such cases. Green *et al*. [14] as well as Xu *et al*. [63] presented approaches based on voxelization followed by spanning graph construction. Provided enough suitable training data is available, machine learning approaches are more accurate than classical algorithms. A good example is PointNet [42] and its extensions [15, 38, 39, 66]. However, due to the nature of how excavation work proceeds, it is very hard to come up with generic, yet representative data to train machine learning models. In particular, hand-labeling of the training data would be required, which is a very costly process.

Segmentation in 2D Some segmentation algorithms [34, 37] use deep learning to create segmentation masks, which are then back-projected [22] to the 3D point cloud. Such virtual views have been used by Kundus *et al*. [27] to segment 3D meshes. A rendering of a point cloud scene in 2D can serve as an input for this type of algorithms. Iannizzotto *et al*. [23] present such an algorithm, using an edge-based approach on virtual views, while Kaganami *et al*. [24] compare region-based and edge-detection approaches (virtual and real views), concluding that performance strongly depends on the problem domain. Furthermore, alternative color spaces such as *HSV* have been used [6, 54].

### 2.2 Fitting in 3D

Our approach includes a 3D fitting algorithm used to align 3D structures, referring to a segmented part of a 3D reconstruction and a synthetically generated 3D structure from SUE data. Due to the importance of the underlying problem across disciplines, many algorithms have emerged and can be categorized mainly into the following groups: Principal component analysis (PCA), singular value decomposition (SVD), iterative closest point (ICP) and deep learning (DL).

PCA is traditionally used in compression and classification algorithms, but it can also be used for rough registration between two point clouds by aligning the eigenvectors of their covariance matrices [2]. Similar, SVD can be used to minimize Euclidean distances between points, if point correspondences are available. SVD for model alignment was shown in several early works [21, 47]. While SVD aims to directly solve the least-squares problem, ICP iteratively discards outliers to improve the result, as discussed by Besl *et al*. [3]. Approaches for various scenarios using ICP emerged over time, such as point-to-point [44], point-to-surface [31] or generalized ICP [48]. Some more modern variants are summarized by Rusinkiewicz *et al*. [43]. Usually, ICP requires a good initial alignment for accurate results and is severely affected by noise.

More recent methods rely on deep learning, such as the work of Li *et al*. [30] and the TEASER algorithm [64], which is highly robust against outliers, however, at the costs of requiring significant amounts of training data. We follow a different approach to balance complexity, computational cost, accuracy and robustness. We employ a lightweight algorithm to enable instant application on mobile hardware. Nevertheless, deep learning algorithms could potentially be used on mobile devices in the future.

### 2.3 SUE data and applications

SUE workers have raised concerns about the inadequate state of documentation in the field [18, 20]. Depth information on assets is often missing or unreliable, and companies generally do not thoroughly survey newly installed utilities. Incorrect documentation is typically only identified after damage has occurred, as companies are not legally obliged to report as-built information. Furthermore, documentation requirements only mandate the reporting of 2D data, and there is a lack of standardized digital formats, with many records in the form of scanned drawings that are difficult to integrate into SUE databases. Although several countries have implemented programs to address documentation challenges, data accuracy and reliability remain a concern.

An example of poor infrastructure documentation is shown in Fig. 1 (left), overlaying SUE data on geo-tagged 3D reconstructions. Survey data provided in accordance with current legislation is not always accurate enough to visualize infrastructure on excavation sites. As can be seen in Fig. 1 (center, bottom), there are significant deviations, especially in depth. Even already aligned pipes might show a displacement in height of half the radius of the pipes, suggesting that the raw survey measurement points on upper pipe surface are taken as pipe depth in the documentation.

Hansen *et al*. [19] suggest using construction and maintenance projects to improve documentation quality, and propose on-site documentation using photogrammetry or laser scanning, as well as utilizing

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2024.3372064

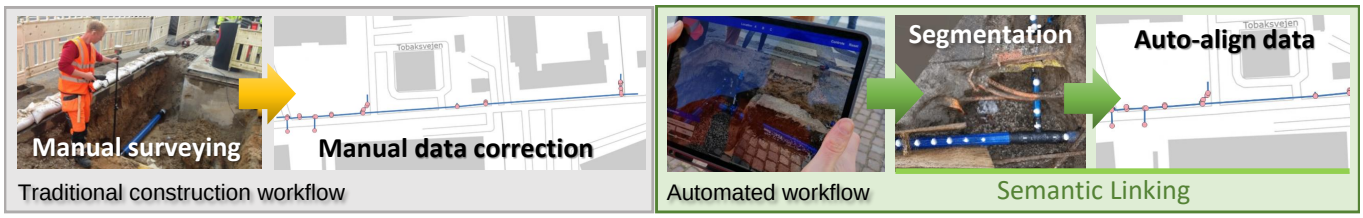



Fig. 2: A traditional (error prone) manual surveying task to (re-)map scans to SUE data (left) and our (semi-)automated semantic linking approach (right), using a Lidar enabled handheld device to automatically align SUE data with real infrastructure to make online information accessible.

Fig. 3: 3D reconstructions of excavation sites from our data set, depicting the predominant use of straight pipes and T-junctions. When hard pipes are used for water or gas, they show only sharp angles of 45°, 60°, or 90°, as shown in the bottom left image. Some reconstructions also show a small outgoing branch, as seen in the top image. Occasionally, pipes are partially covered by dirt, as shown in the bottom row, third image.

geo-tagged 3D reconstructions for accurate mapping. Mobile hardware for reconstructions and automated segmentation and alignment algorithms should be used for data verification, mapping, or correction.

Schall *et al*. were among the first to show the benefits of AR in SUE [12, 13, 45]. Hansen *et al*. [17] more recently described virtual spraymarkings and virtual daylighting approaches. However, precise localization of both the observer and the data has been a recurring challenge. Furthermore, SUE data [49, 53] have been used for localization. A combination of SUE, GPS and IMU data served the best results [13, 16, 45, 67]. The importance of localization and data accuracy is further underlined in Xu's review on AR in civil engineering [62]. However, all approaches fail to operate on inaccurate SUE data. Our work proposes a solution to this issue.

Various hardware and software products have been released in recent years: *AugView*[2] is a commercial AR solution for underground infrastructure; *vGIS*[3] uses SUE data for visualizations, and *AVUS*[4] focuses on *reality capture* and reconstructions. All these products rely on third party surveying equipment for localization, while Trimble is the only vendor offering an all-in-one solution, *SiteVision*. Compared to the method proposed by Hansen *et al*. [17], all these commercial solutions involve manual setup processes. Furthermore, they offer only very limited ways to overcome inconsistent SUE data. A comparison between the traditional workflow and the proposed approach is shown in Fig. 2.

## 3 System overview

In the following, we briefly describe our setup. Furthermore, we quickly describe the main problems of currently available SUE data.

### 3.1 Hardware

We utilize a setup similar to the one proposed by Hansen *et al*. [17], consisting of an external sensing cube for highly accurate global localization combined with an iPad Pro (second-gen or later) equipped with a Lidar sensor. Comparisons between 3D reconstructions from industrial Lidar scanners and the iPad Pro showed that the mobile device can match professional equipment in accuracy up to a range of 5 m [50, 60]. The external sensing cube consists of a GNSS RTK module for global position estimation (RTK float ¡ 20 cm and RTK fixed ¡ 2 cm), an IMU with compass for orientation estimation, and an altimeter to compensate for errors in GPS altitude estimation. Data are transferred via MQTT[5] using a mobile WIFI hotspot, while a semi-automated routine using the *kalibr*[6] tool [9, 10] is used for one-time calibration of the setup. This calibration is necessary for optimal accuracy; however, it is not mission-critical given the required accuracy level of around 20 cm employed in SUE.

### 3.2 Software

The Unity engine and its tracking API *ARFoundation* is utilized to ensure cross-platform compatibility and maintainability. A reference space is created in the Unity scene, allowing the fusion of global and local tracking information into a single coordinate space. This enables geo-referenced positioning of objects within Unity software. However, previous experiments have revealed limitations in compass-based north estimation and in local tracking drift [51].

In order to establish the relationship of as-planned and as-built data, these two points are crucial. Rotation errors need to be minimized to ensure accurate alignment of SUE data, while drift must be mitigated to prevent visualizations from wandering within the scene. We previously implemented a continuous drift correction that utilizes RTK-GPS as ground truth to re-initialize the reference coordinate system whenever the local tracking drifts too far. Furthermore, we developed a GPS-based north estimation algorithm, derived from frequently used approaches in robotics [5, 57, 65] using two antennae. North angles are continuously calculated inside UTM coordinate space between two positions in the virtual tracking space and the GPS space. The difference of these angles can be used to rotate the local coordinate space to the real-world north. Using this setup, we create colored point clouds, which are automatically globally registered during capture. Some examples can be seen in Fig. 3.

## 4 Semantic linking

We refer to the process of assigning correspondences from SUE data to real-world objects such as pipes as *semantic linking*. Once such

[2] https://www.augview.net/
[3] https://www.vgis.io/
[4] https://www.avus.tech/
[5] https://mqtt.org/
[6] https://github.com/ethz-asl/kalibr

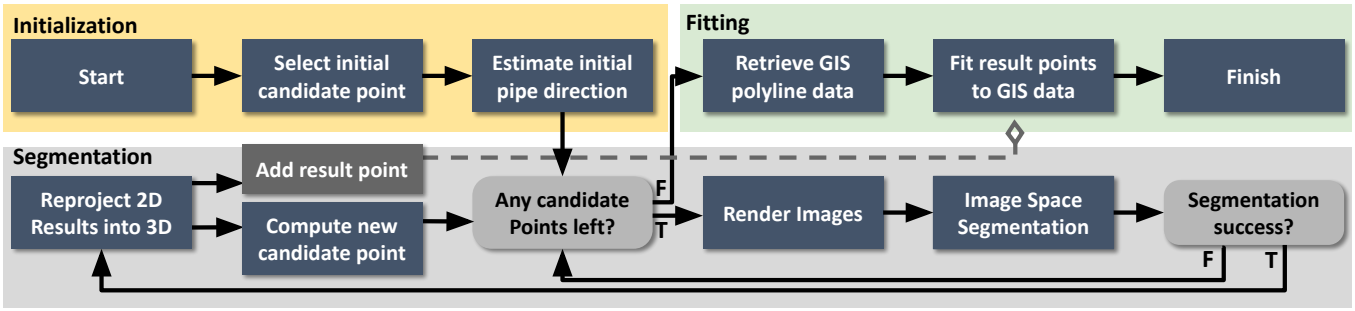
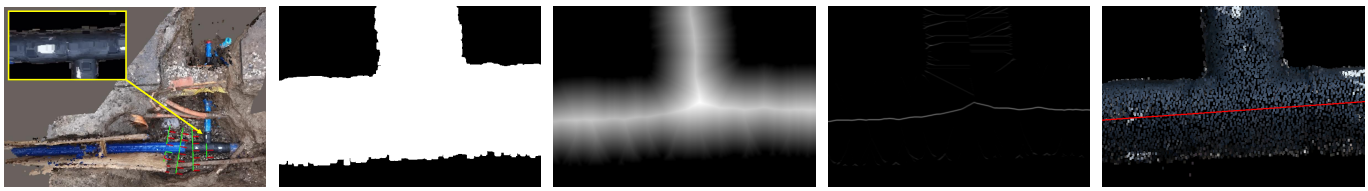



Fig. 4: *Semantic Linking* workflow: Initialization (yellow) begins with user-provided 3D-point input on the pipe, initializing a candidate point queue via Birds Eye-View Rendering for pipe heading estimation. Segmentation (gray) iteratively processes candidate points until no more remain. Each iteration dequeues a candidate point, renders images around it, and performs Image Space Segmentation within all rendered images. Segmentation success requires at least 4 out of 6 images to yield valid results. Successful segmentation projects 2D image space results into 3D space, producing a 3D-line (Result point, Pipe direction) for calculating subsequent candidates and saving the Result Point. Alignment (green) retrieves 3D polyline data from a GIS database and aligns it with result 3D points via a least-squares fit.



(a) rendered input (b) color segmentation (c) distance transform (d) filtered (e) estimated line

Fig. 5: Main steps of segmentation in image space: (a) render image around candidate point (b) color based segmentation to mark pipe area in the image, (c) distance transform to highlight the middle part of the tube, (d) individual filtering to detect horizontal lines, This used to eliminate disturbance of junction parts or other noise in the color segmentation, (e) visualization of the segmentation result in the given examples. Red lines are used for re-projection into 3D space.

correspondences are established, the operator can easily refer to items in the SUE database and visualize assets in situ. The linking procedure consists of three steps: First, an initialization phase obtains starting points. Second, a segmentation step identifies potential pipes, and, third, a fitting step determines their precise positions. Fig. 4 depicts the three stages and their main components, as well as the information flow and the iterative nature of our proposed algorithm.

### 4.1 Initialization

A common approach to get an initial pose and a starting point is to let the operator use a tablet to tap on any point on the pipe. This 3D point is used to extract the color of the pipe and initialize the segmentation (Fig. 4, yellow). First, we need to compute the orientation of the pipe, such that it can be traversed efficiently. We render a birds-eye view of the pipe at the given point using the gravity vector known from the sensors. The sensor ensures that the upward direction is aligned with gravity during reconstruction. Given the color of the initial point, we perform a straight-forward HSV color segmentation, resulting in a binary mask (Fig. 5b). We allow an empirically found color deviation of 15% of the hue component to ensure good segmentations. By fitting a bounding box to the mask, we find the rotation around the up-vector (sky), which indicates the longitudinal direction of the pipe. As we may assume that pipes are placed horizontally, only this rotation must be set. The selected point $\mathcal{C}_{init}$, and the corresponding direction $\mathfrak{D}_{init}$ are then added to the candidate points list $\mathcal{C}$ (Algorithm 1, L3).

### 4.2 Segmentation

The segmentation of pipe-like structures is the most important part of semantic linking. It has to overcome Lidar, photogrammetry and excavation limitations to enable on-site applications. Fig. 4 (Gray) and Algorithm 1 (L4-L15) depict the main steps of the segmentation procedure which combines $\langle \mathcal{C}_{init}, \mathfrak{D}_{init} \rangle$ from the initialization with a colored point-cloud/mesh to output a line based representation of the pipe in question. The segmentation processes candidate points $\mathcal{C}_i$ from the list $\mathcal{C}$ into result points $\mathcal{R}_i$ which are kept in a separate list $\mathcal{R}$. Every $\mathcal{C}_i$ with its directions $\mathfrak{D}_i$ represents a different point on the pipe. We use the estimated direction $\mathfrak{D}_i$ to iteratively process points along the pipe in the longitudinal direction, allowing us to resolve branching and curves. Therefore, the segmentation algorithm can be subdivided into six individual steps.

The exit condition **"Any candidate points left?"** checks if there are any $\mathcal{C}_i$ in $\mathcal{C}$ left. If so, the first is taken (in a FIFO manner, Algorithm 1, L5) and passed to the **"Render Images"** step. By orbiting a virtual camera ($\pm 30°$) around the longitudinal direction $\mathcal{D}_i$ of the pipe at the candidate point $\mathcal{C}_i$, we render an image of the 3D structure every twelve degrees, resulting in six image renderings $I$ (Fig. 5a and Fig. 6).

The rendered images are then passed on to the **"Image Space Segmentation"** step, where every $I_i$ is processed. For each $I_i$, we perform a color segmentation to create a binary mask of the pipe-like structures (Fig. 5b). Based on the binary mask, we perform a distance transform (highest distance from the boundaries) to obtain the center line of the pipe (Fig. 5c). A simple binary filter kernel is applied to remove any remaining outliers, resulting in an estimate of the center-line (Fig. 5d). A 2D line fitting algorithm yields the estimated center line (Fig. 5e). We thereby obtain up to six 2D estimates of the center-line $S$ of the pipe. However, we require the process to return at least four valid line estimates, otherwise, this candidate ($\mathcal{C}_i$) is terminated and we look for the next one (Fig. 4, **"Segmentation Success?"**, Algorithm 1 L8).

On success, $S$ is passed on to the **"Reproject 2D Results into 3D"** step, where the estimated center-lines are used to find the real center-line in 3D. Due to the noisy nature of the data, the line estimates might still be inaccurate. In order to arrive at a more reliable, but still computationally inexpensive approach running in real-time on mobile devices, we compute the pairwise intersection of all planes $\mathcal{P}$ given by the virtual camera center (Fig. 6) and the center-line on the image plane (Fig. 6). From the virtual camera orbiting, we have at most six virtual camera centers and corresponding planes, yielding up to 30 intersections. Using a second order Lagrange interpolation, we estimate the intersection line and represent it as the result point (center of the intersection line) and corresponding directional vector $\langle \mathcal{R}_i, \mathfrak{D}_i \rangle$ (Fig. 6,

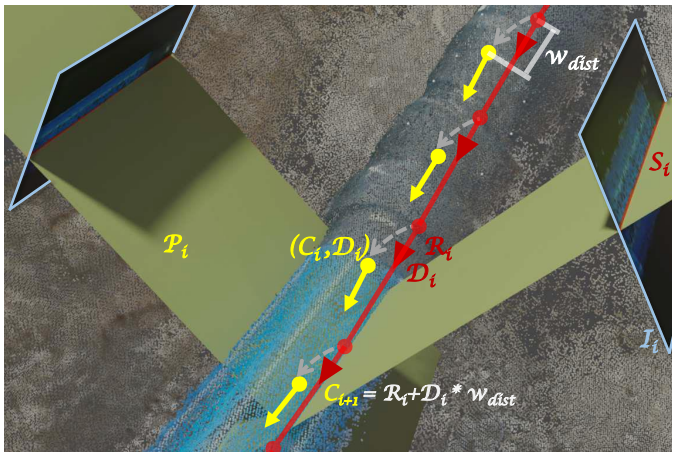



Fig. 6: Illustration of the segmentation approach. Planes ($P_i$) are projected through the segmented pipe in the image space ($S_i$) of rendered images ($I_i$) around the current candidate Point ($C_i$), resulting in their intersection at the center of the corresponding real-world pipe ($\mathcal{R}_i, \mathfrak{D}_i$). Using this result, the next candidate point ($C_{i+1}$) can be calculated.

**Algorithm 1** Segmentation

```
Input: C_init, colored point-cloud/mesh; Output: R
D_init ← EstimatePipeDirection( C_init )
C.Add(⟨C_init, D_init⟩)
while (C.length > 0) do
    ⟨C_j, D_j⟩ ← C.pop_front()
    I ← RenderImages(⟨C_j, D_j⟩)
    S ← SegmentImages(I)
    if S.length > 3 then
        P = CreatePlanes(S)
        ⟨R_i, D_i⟩ ← IntersectPlanes(P)
        R.add(R_i)
        C_{i+1} ← R_i + D_i * w_dist
        C.Add(⟨C_{i+1}, D_i⟩)
    end if
end while
```

| | | |
|---|---|---|
| Pipe direction | $\mathfrak{D}_i$ | The direction vector of a $\mathcal{C}_i$ or $\mathcal{R}_i$ |
| Candidate point | $\mathcal{C}_i$ | A guessed point on the pipe |
| Candidate points | $\mathcal{C}$ | $\mathcal{C} = [.., \langle \mathcal{C}_i, \mathfrak{D}_i \rangle, \langle \mathcal{C}_{i+1}, \mathfrak{D}_{i+1} \rangle, ..]$ |
| Result point | $\mathcal{R}_i$ | A verified point on the pipe |
| Result points | $\mathcal{R}$ | $\mathcal{R} = [.., \mathcal{R}_i, \mathcal{R}_{i+1}, ..]$ |
| Render Images | $I$ | List of rendered virtual views |
| Segmentation Results | $S$ | Estimated line per segmentation (Fig. 5) |
| Step size | $w_{dist}$ | Intermediate step size to walk along the pipe |

Table 1: Notation for Algorithm 1.

Algorithm 1, L10).

In contrast to a birds-eye direction, we consider a direction in 3D space, this allows step-wise progression in all directions, also up (sky) and down (bottom). This helps to sample around corners and junctions. However, rapid changes in direction, such as 90° curves or junctions, are not detected in this stage, but in a later step. The final result of this segmentation is obtained by averaging all intersection results.

Next, the $\mathcal{R}_i$ is added to $\mathcal{R}$ for later use (Fig. 4, **"Add result point"**). The tuple $\langle \mathcal{R}_i, \mathfrak{D}_i \rangle$ is passed to **"Compute new candidate point"** where it is used to compute a new candidate point $\mathcal{C}_{i+1} = \mathcal{R}_i + \mathfrak{D}_i \cdot w_{dist}$, where $w_{dist}$ refers to the step size when walking the pipe (Fig. 6). This allows use to follow the pipe structure and to accommodate for directional changes. Finally the loop is closed, by checking again the list of candidate points $\mathcal{C}$ (Fig. 4, **"Any candidate points left?"**). Once $\mathcal{C}$ becomes empty, we continue with the fitting step as described in Sect. 4.3.

#### 4.2.1 Implementation details

In this section, we give some additional details on the implementation of the *image rendering*, the *image-space segmentation*, the *optimal multi-plane intersection*, the *junction detection*, and the *abortion criteria*. In addition, we discuss the integration with Unity. The segmentation and alignment algorithms are implemented as a cross-platform C++ library. It functions as a standalone tool with background rendering through Open3D[7], or it is used in conjunction with a rendering engine (*e.g.*, Unity) through callbacks. When using Unity as a testbed, we have to account for different coordinate systems and handedness. Potree[8] was used to render 3D point cloud data. Synchronization between the main thread and the segmentation thread is necessary, as cameras and image rendering can only be served in the main thread. Image processing is done using OpenCV. The Ceres solver [1] was used to implement the optimization algorithm in C++ for real-time performance on mobile devices. An example of semantic linking running on the iPad within the Unity software is shown later in Fig. 16.

Why not normals? Naively, using surface normals would easily allow to estimate center points along pipe structures, but this approach fails on real-world noisy data. Fig. 8 depicts how challenging the reconstruction task can get when pipe-like structures are not fully excavated, deformed and only partially contained in the reconstruction built from images taken from a street-level viewpoint. Pipes fade into the surroundings and become hard to recognize from geometry alone. Works like Maurell *et al.* [33] apply the commonly used CGAL [56] library for shape detection of cylinders, but they only operate on isolated geometry where major parts of the cylindrical structures are visible. We further evaluated CGAL algorithms in Sect. 5.1. In contrast, for reconstructing subsurface utilities, usually only a (very noisy) portion of the structure is visible, and it can only be captured from the street level. Our approach addresses this problem through multi-image segmentation and projection into 3D space. The estimated model consists of 3D points along the center of the pipe-structure, which is used in the fitting step to precisely link to the corresponding SUE data.

Performing the segmentation procedure in the image domain overcomes the exhaustive 3D point matching employed in ICP-like algorithms, enabling our method to operate equally well on sparse point clouds by simply increasing the rendered point size. Image based edge detection in RGB-D images has the same advantage.

Image rendering The resolution of the rendered images is chosen for fast processing. Since we know approximate object sizes and can vary the distance accordingly, using lower resolutions is more efficient, while still delivering good segmentation results. Lower resolutions have the additional advantage that high-frequency noise gets suppressed. We empirically determined a resolution of $160 \times 120$ pixels to work best for our purposes. We set the distance of the virtual camera to $3-4\times$ the diameter of the pipe, typically around 60 cm, and we keep a horizontal field of view of 40°.

Longitudinal walks are sampled using steps of $10-20$ cm, depending on the field of view, to guarantee a slight overlap. Tests with adaptive sampling based on segmentation width yielded equivalent results. However, it is worth noting that optimal parameter choices still depend on the scenario and the type of excavation.

Our method to segment synthetic images effectively decouples the segmentation algorithm from the 3D input, as we only require the 3D structure to be rendered in color. Point-cloud density and point size thereby have a direct impact on the rendered image. We can use the near and far clipping plane to selectively suppress the background and foreground during rendering. As a result, only the pipe is rendered, helping the segmentation algorithm.

Image-space segmentation To ensure robust color segmentation, we apply Gaussian blur ($\sigma = 3$) to further reduce outliers and smooth the color. Additionally, on sparse or incomplete data, morphological operations can be applied to create a more homogeneous image. Dilate

[7] http://www.open3d.org/
[8] https://github.com/SFraissTU/BA_PointCloud

and erode operations help to preserve edges and close holes. Hue thresholding in HSV color space and pixel-based region growing can help to stabilize the segmentation results. However, the quality of segmentation was more than sufficient within our real-world dataset, even without these enhancements.

Optimal multi-plane intersection For every virtual camera, we compute the plane through the camera center and the segmented line, taking into account the rendered image properties. Planes are each represented as a point ($p_i$) and a normal vector ($n_i$). The normal vector ($n_i$) is obtained by calculating the cross product of two rays that originate from the camera position and pass through any point on the segmented line on the image plane (we use the leftmost and rightmost points). In addition to the line at the intersection of two planes, a point on the line near the cameras is required. Hence, a second step is necessary to find this point. These two steps can be combined into a single optimization step using Lagrange multipliers [26]. We find an intersection point ($p$) that has a minimum distance to the center of two cameras ($c = (p_1 + p_2)/2$) while satisfying the condition that the point lies on both planes. This optimization problem is characterized by an objective function

$$\arg\min_p \|p-c\|^2 = (p_x - c_x)^2 + (p_y - c_y)^2 + (p_z - c_z)^2 \tag{1}$$

subject to two constraints (Equation 2)

$$(p - p_1)\cdot n_1 = 0, \quad (p - p_2)\cdot n_2 = 0. \tag{2}$$

Five linear equations are represented in matrix form in Equation 3; $\lambda$ and $\mu$ are the Lagrange multipliers of the two constraints. Solving this system of equations yields the desired point $p$.

$$\begin{bmatrix} 2 & 0 & 0 & n_{1x} & n_{2x} \\ 0 & 2 & 0 & n_{1y} & n_{2y} \\ 0 & 0 & 2 & n_{1z} & n_{2z} \\ n_{1x} & n_{1y} & n_{1z} & 0 & 0 \\ n_{2x} & n_{2y} & n_{2z} & 0 & 0 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ \lambda \\ \mu \end{bmatrix} = \begin{bmatrix} 2c_x \\ 2c_{cy} \\ 2c_{cz} \\ p_1 \cdot n_1 \\ p_2 \cdot n_2 \end{bmatrix} \tag{3}$$

Subsequently, the direction of the intersection line can be obtained as the cross product of the normal vectors of the two planes. To calculate the intersection of all up to six planes, an iterative process is applied that involves estimating the intersection lines of pairs of planes and then averaging them while filtering out any outliers. This is necessary because the planes do not intersect perfectly in one line. Alternatively, a comprehensive optimization approach can perform the intersection of all planes in a single step. However, the integrated approach is slower. Since it must be executed in every single segmentation step, we prefer the sequential algorithm.

Junction detection T-junctions, cross junctions, and sharp curves are identified in an image by analyzing the contours of the color mask. Pixel-by-pixel analysis of the 2D border determines the type of junction currently observed. Based on the type, we take actions like adding a new point to the work queue for a T-junction or adjusting the walk direction in case of a steep curve.

### 4.3 Fitting

We refer to *fitting* (Fig. 4, green) as the process of establishing the link between the virtual data (*e.g.*, SUE representations) and the real-world observation. The segmentation step gives us a set of 3D points (result points, $\mathcal{R}$) representing the real pipe, and GPS lets us query SUE data (Fig. 4, **"Retrieve GIS polyline data"**). SUE data are commonly presented as polylines[9]. We take advantage of the polyline representation of the GIS data to break down the fitting problem to 3D lines and points instead of matching complex 3D structures. Polylines consist of points representing finite line segments; therefore, we can define a minimization problem to align result points with line segments of the polyline (Fig. 4, **"Fit result points to GIS data"**). The result

[9] https://docs.qgis.org/2.8/en/docs/gentle_gis_introduction/vector_data.html

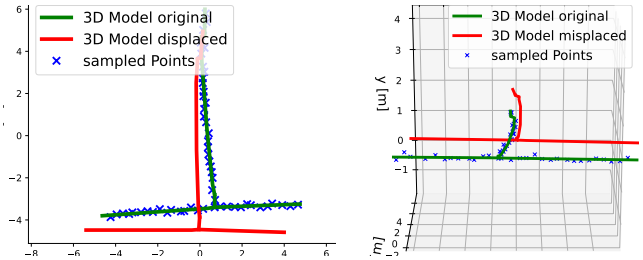



Fig. 7: Generation of synthetic data using Gaussian noise on top of SUE data for fitting evaluation.

of the minimization is a rigid 6DOF transform (Fig. 4, **"Finish"**) that aligns all line segments to the segmented points. In Equation 4, we compute the distance ($d$, Equation 5) between each segmented point $p_i$ and all transformed line segments $L$, where a single segment ($L_i$) is defined line between the startpoint ($S_{Ai}$) and the endpoint ($S_{Bi}$) as $L_i = \overline{S_{Ai}S_{Bi}}$, optimizing for rotation $R \in \mathbb{SO}(3)$ and translation T using least squares. Since we operate on real-world data, we operate on a fixed scale.

$$\begin{bmatrix} p_1 \\ \vdots \\ p_n \end{bmatrix} - \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \cdot [L_{1..i}] = 0 \tag{4}$$

$$d = \begin{cases} \|p_i \perp L_i\|^2, & \text{if base point } \in L_i \\ \min(\|p_i - S_{Ai}\|^2, \|p_i - S_{Bi}\|^2), & \text{otherwise} \end{cases} \tag{5}$$

To avoid expensive point-to-line pre-mapping steps, we just compute the distance of any line segment to the given point, and return the shortest distance – in other words, we match every $p_i$ against every $L_i \in L$ in a lightweight computation. Equation 5 decides how the distance is being computed, since we only look into finite line segments: If the current point $p_i$ lies within the line segment $L_i$, we compute the normal distance to the segment; if $p_i$ lies outside, we take the shorter distance to one of the line segments endpoints ($S_{Ai}$ or $S_{Bi}$).

### 4.4 Limitations

Segmentation of pipes that are half-covered with dirt or possess similar coloration is challenging and often leads to lower quality results. Steep pipes that are partially outside of the clipping space can cause problems during segmentation, leading to incorrect estimates or no valid segmentation at all. Additionally, proper initialization is crucial for a successful segmentation, as the algorithm may fail when processing points located at a junction or in other problematic positions. In terms of fitting, straight pipe sections present a challenge, as no unique 6DOF alignment is possible, leading to ambiguous solutions. However, solutions are usually sufficient to create a semantic link from the SUE data to the pipes. Finally, when only a small part of an outgoing branch is visible, achieving proper alignment becomes problematic.

## 5 Evaluation

We evaluate the proposed semantic linking approach on real and synthetic data. The real-world dataset consists of 30 reconstructed excavations together with the corresponding SUE data. Fig. 3 depicts a few examples of the dataset. Overall, it consists of ten straight segmented, two curved segments, and 18 T-junctions, where 17/30 also contain fire hydrants. On the downside, the real-world dataset contains errors regarding SUE data and reconstruction artifacts.

Moreover, we generate 1.000 random samples to evaluate our point-to-segment fitting approach. Synthetic data is generated based on the real SUE data (depicted in Fig. 2). We take a random point on an existing SUE position and extract all pipe information within a radius of 6 m, matching the size of real-world excavations in our dataset. Furthermore, we sample points along the pipe and add Gaussian noise to simulate real input to the fitting. Lastly, we randomly displace the SUE data to simulate a significant error, as depicted in Fig. 7.

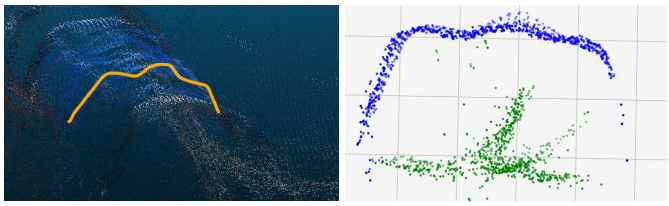


Fig. 8: Observed noise level on Lidar and image-based reconstructions. Narrow viewing angles in top-down perspective cause artefacts and impair reconstructing tube-like structures. *Orange* highlights the deformed contour of a round pipe. Using normals for segmentation or center estimation leads to severe difficulties.

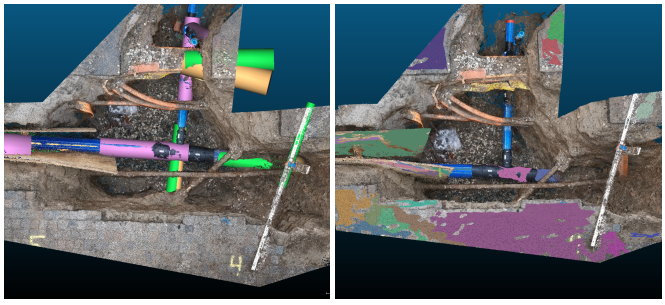


Fig. 9: CGAL results. Left image shows filtered efficient RANSAC results using different normal thresholds (green, orange, purple). Due to poor normal quality, different parts of the pipe fall into the segmentation with variance of this parameter. Nevertheless, it is possible to obtain reasonably accurate segmentations of parts of the pipe. The right Image shows region growing result with every region colored in a different color. Region growing does not yield usable results.

We want to evaluate segmentation and fitting both individually and together. Therefore, we run the segmentation on the real data and compare it to ground truth data, *i.e.*, the aligned SUE data from the dataset. Moreover, we use synthetic data to evaluate the fitting on synthetic data with real-world characteristics and compare it with the same ground truth. Lastly, we evaluate semantic linking and compare it against the same ground truth.

Although real-world examples are limited, our evaluation provides insight into the value of our method. While the fitting approach can be evaluated using synthetic data in a satisfactory manner, generating good synthetic data sets to evaluate the segmentation poses a challenge. In general, the results achieve the targeted accuracy of $< 20$ cm. However, problematic cases can arise where segmentation fails or produces poor estimates, and the presence of a few poorly segmented points at pipe ends can significantly impact the alignment process. Translational results, as the more critical factor, are usually sufficient, but rotational alignment may not be adequate for some of the real-world examples.

### 5.1 Comparison to other algortihms

The *Computational Geometry Algorithms Library* (CGAL) implements highly sophisticated algorithms in the area of computer vision and graphics. Algorithms for primitive fitting (*e.g.*, cylinders) or segmentation are well supported and find a broad use among the computer science community. As already mentioned in Sect. 2, our use case (daylighted subsurface infrastructure) is not well suited for this kind of algorithm, which is designed to work on "more clean" data. Within CGAL, we identify two promising algorithms[10]: (a) a RANSAC based primitive fitting to detect cylidrical shapes in 3D point-cloud data, and (b) a region growing implementation to segment 3D point-clouds. We tuned the parameters of both versions to ensure the best performance on our datasets. Both implementations require a point cloud with unoriented normals. This requires an additional pre-processing step to compute the normals taking approx. 30 seconds with tools like CloudCompare[11] for metric point-cloud reconstruction spanning around $5 \times 5$ meters similar to Fig. 3.

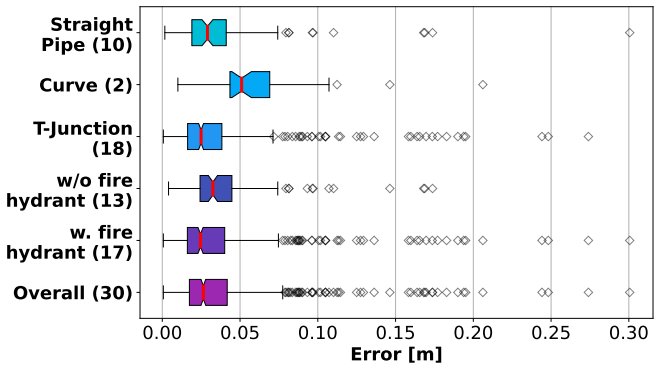



Fig. 10: Different pipe structures yield similar errors, which shows the segmentation performance is less affected by the structure itself. Due to the single occurrence within the curve category, it is hard to draw conclusions from the higher error. 75% of the segmentation yield an error below 5 cm ($<$ 10 cm including curve), and the overall error is smaller than 30 cm.

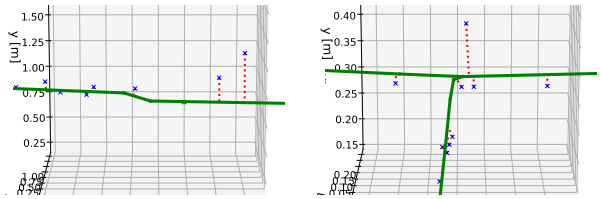


Fig. 11: Representative segmentation results of real-world reconstructions where *blue* depicts the segmented 3D points; *green*, the ground truth, and *red*, the point-to-model distance used for evaluation.

A good depiction of the data our algorithm has to handle is presented in Fig. 8. Through the narrow angles, *e.g.*, when looking into a hole, we get a deformed surface most of the time. Additionally, pipes are not fully visible, and, due to the deformation, the surface normals point into a variety of directions, making it especially hard for normal-based algorithms.

The point clouds shown in Fig. 1 consist of $4 \times 10^6$ points, and take 3.9 seconds with our implementation on a 2015 laptop computer (i7-7700HQ, GTX 1050, 16GB). We use this scene to illustrate differences among the different algorithms.

CGAL: efficient RANSAC (Primitive Fitting) CGAL's efficient RANSAC implementation is based on Schnabel *et al.* [46]. When running with default parameters on our test point cloud, the algorithm fails to detect any cylinders and takes 31 seconds to complete, ten times slower than ours. Accounting for the pre-processing step (normals estimation), we can add another 30 seconds to the runtime. Nevertheless, adjusting the parameters does lead to partial pipe segmentations, as depicted in Fig. 9, left. Three different segmentation runs are visualized (green, orange, purple), where only the `normal_threshold` parameter varies, leading to different parts of the pipe being detected. All runs yield more than 50 cylinders, but only a few exhibit reasonable radii to match the pipe. After filtering cylinders with inadequate radii, we obtain estimates corresponding to small sections of the pipe. Wrong estimates are also present, especially seen in green here. The CGAL RANSAC implementation falls short of detecting pipes over bigger distances. More importantly, a runtime of 30 seconds renders it unusable for instant use in AR applications. Tuning parameters for better results even increases runtimes to 268, 456, and 652 seconds for three variations shown in Fig. 9, left. The runtimes are compared in Table 2.

CGAL: Region Growing Lafrage *et al.* [28] is used as a reference implementation. The algorithm primarily utilizes distances between neighbors and angles between normal vectors for accurate primitive

[10] `https://www.cgal.org/index.html`

[11] `https://www.danielgm.net/cc/`

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2024.3372064

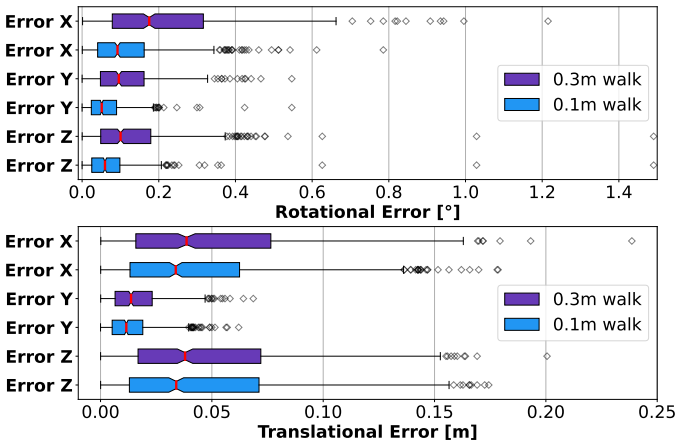



Fig. 12: The direct comparison between results of higher (blue) and smaller (purple) sample rate is plotted. Rotational error is shown on the left, and translational error, on the right. A higher sample rate of segmentation points is observed to reduce the error of the alignment process, particularly in the rotational case. The overall results, with a sample distance of 0.3 m, already demonstrate sufficient accuracy, with 95% of rotational alignment error being within 0.6°. The translational alignment is similar for both sample sizes.

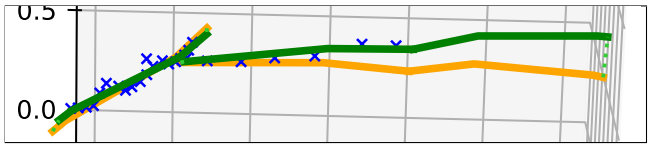



Fig. 13: A higher rotational error along the main axis of the T-junction was received due to insufficient sampling along the branch. *Orange* shows the ground truth; *green* depicts the aligned model, and *blue* shows the sampled points. The point-to-model distance was $< 3$ cm and the rotational error was at 7°. All errors are below the constraints of a real excavation use-case.

detection. Fig. 9, right, illustrates an example segmentation where the obtained results are entirely unusable. Neither pipes nor background can be identified for further processing. Despite attempting various parameter adjustments, no improvements were achieved. Even if a reasonable segmentation was achievable, the runtime remains a significant drawback with this implementation. The region growing algorithm takes 376 seconds, approx. 50 times longer than ours, not including the additional 30 seconds spent on the pre-processing to estimate the normals.

### 5.2 Segmentation evaluation

We run the presented segmentation algorithm over the full real-world dataset and compare it with the ground truth, which is the manually aligned GPS data. In 75% of the cases, we stay below a total error of 10 cm and, for all cases, below 30 cm. On average, we observe around 5 cm total error, where the *curved* scene is slightly higher at 8 cm. Since we only have one scene of this kind, it is hard to draw conclusions from this observation. Fig. 10 shows a good overall segmentation performance across the dataset, including outliers. However, for a real-world application, we perform within the bounds of a *shovel width* or even better.

We observed issues when fire hydrants face towards the camera, as they can interfere with the correct detection of branches and can also be segmented as part of the pipe. Limiting the culling space to reduce noise can prevent the detection of T-junctions with branches that do not extend to the ground and cause problems with very steep pipes.

However, better reconstructed pipes with a slope of about 45° could be perfectly segmented in another scene. To improve the segmentation of fire hydrants, points with steep slopes relative to the previous point are treated as outliers using a gradient threshold greater than 60°. Adjustable parameters allow us to overcome segmentation shortcomings

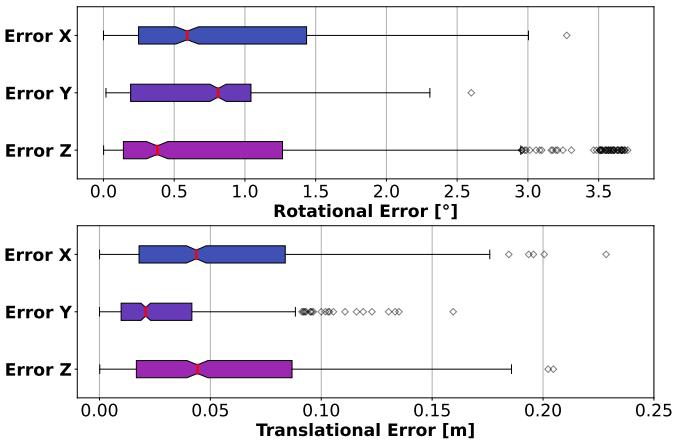



Fig. 14: Results on real segmentation data show suitable median values. The distribution of translational alignment errors is similar to that of synthetic data. A median error of less than 5 cm and maximum error of less than 20 cm is considered adequate for practical applications. However, a few bad segmentation points at the end of pipes and a significant number of real-world scenarios with only small exposed parts of the outgoing branches at junctions lead to higher rotational errors. Despite this problem, a majority of the rotation results are still within one degree offset, which is considered accurate enough.

at sight, but especially fire hydrants can give the algorithm a hard time.

We compute the point-to-model (Fig. 11) distance for each segmented 3D point using the distance function from Equation 5. The default parameters worked for 24/30 of the real-world datasets. Applying heuristics related to the diameter of the pipe to control camera distance and step-size, the rest of the scenes worked flawlessly.

### 5.3 Fitting evaluation

The 3D points in the synthetic data set were offset from the real data by adding Gaussian noise ($\sigma = 0.1$ m). Additionally, the pipe structures were randomly moved ($\pm 2$ m, $\pm 5°$ sky, $\pm 2°$ north/east) to simulate inaccurate and misaligned SUE data. Once the rigid transform is computed, we transform the polylines and measure a pointwise distance from the ground truth. Usually, the error is smaller than the applied Gaussian noise, indicating good fitting results. Within the dataset, we also find scenes with short branches (*e.g.*, at T-junctions) that lead to a small number of sampled points. This can cause a larger rotational error around the long side of the junction, caused by too few samples.

The median translation error is $< 5$ cm, and the median rotational error $< 0.3°$. The estimation error for rotation can be determined with an accuracy of $< 0.6°$, and, for translation, within 15 cm. Higher sampling rates in the segmentation improve the accuracy of the fitting process. Outliers produce a maximum error of 5° along the pipe, due to the horizontal nature of SUE infrastructure. We achieved three times higher rotational accuracy with an increased sampling rate ($30 - 10$ cm), but only marginal improvements for translation (Fig. 12).

### 5.4 Semantic linking evaluation

A start-to-end evaluation of the segmentation and the fitting was performed on the real dataset without the straight segments, due to the rotational ambiguity along the longitudinal direction, resulting in 20 test scenes. We again randomly displaced the ground truth SUE data ($\pm 2$ m, $\pm 5°$ sky, $\pm 2°$ north/east) to simulate the real-world misalignment of SUE data and the real-world. We found that a particular scene caused a consistent rotational error of 7° along the x-axis, no matter in which direction the ground truth model was shifted. The average alignment error was 3 cm, while the ground truth displacement was 4 cm, indicating a better fit with the segmentation as with the SUE data. Fig. 13 depicts this behavior, where inaccurate segmentation along the short branch causes the error. We observed this behavior in a number of other scenes as well. On large scenes (10 m length), such an error can cause a displacement of a border region of up to 5 m. Rotational

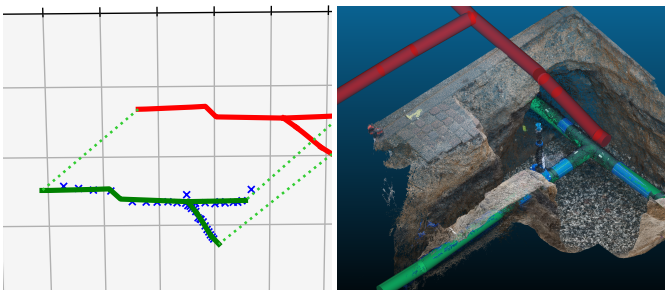


Fig. 15: A successfully semantically linked *T-junction* on real data. Left: SUE data as a red line strip, segmentation result in green. Right: original SUE model in red, the corrected model in green.

errors of $< 1^\circ$ have negligible effect on the quality of visualizations. In addition to the rotation component, we found that the transnational errors remain low, even with bad segmentations – which allows on-site use (*e.g.*, accessing related data).

On average, the translation error across the entire dataset is $< 10$ cm, which is within the range of a *shovel width*. Fig. 14 (right) shows the translation error where the y-axis (sky) remains $< 5$ cm on average for the dataset. Taking into account the outliers in edge cases, the error remains below 25 cm. Fig. 14 (left) shows the rotational error $< 1.5^\circ$ among the dataset. The y-axis (sky) has a smaller error ($< 1^\circ$) due to the overall good orientation of SUE data and the correct north direction estimate. On average, the rotational error stays around $1^\circ$, which is suitable for the application scenario, but can be higher on segmentations of the short sides of junctions. Especially problematic are the areas where the pipe smoothly vanishes into the ground. A *T-junction* example with vanishing pipes is shown in Fig. 15.

### 5.5 Timings

We found, that our prototypical implementation of the proposed semantic linking algorithm is well-suited for on-site applications. Using an Apple iPad (second generation, with Lidar), we achieved an average virtual view rendering performance $< 20$ ms and a center estimation across all views of single step $< 150$ ms. On average, it takes $< 180$ ms for a complete step. Using an adaptive sampling size $10 - 30$ cm gives us an approximate runtime of 1 m/s. The fitting step, due to the low number of well-selected points and the overall cost of the segmentation, is negligible, with runtimes $< 1$ s. Table 2 shows details of runtime on a laptop (i7-7700HQ, GTX 1050, 16GB) and an iPad Pro.

## 6 Discussion and conclusions

In this work, we propose a semantic linking algorithm using segmentation to identify pipes at open excavation sites, together with an approach to align existing (and often displaced) as-built data precisely to the created results. The successful link between the real and virtual data gives access to additional information (*e.g.*, in information retrieval systems) and can further be used to improve workflow efficiency in SUE tasks. Our platform-independent library allows for the use of any rendering engine and, therefore, easy integration. This is a key goal for a wide range of applications, including real-time on-site applications or the development of plugins employing background renderings for processing software like QGIS.

| [ms]<br>Device | 1×<br>Render | 1×<br>Seg. | MultiView<br>Compute | Total<br>Time | per<br>Step |
|---|---|---|---|---|---|
| **Ours** | | | | | |
| PC-*pre* | 8 | 11 | 187 | 6001 | 313 |
| PC | 26 | 10 | 258 | 8155 | 429 |
| IPad | 19 | 1 | 142 | 2971 | 177 |
| **CGAL** (PC) | | | | | |
| Default [46] | | | | $3 \times 10^4$ | |
| Tuned [46] | | | | $4 \times 10^5$ | |
| RG [28] | | | | $4 \times 10^5$ | |

Table 2: Average runtime performance over a subset 5 from our 30 real-data scenes. *pre* indicates the headless mode where pre-rendered images where used. *Render* indicates time taken for one virtual view; *Seg.*, the time for a single segmentation; *MultiView Compute*, the time to estimate the center of the pipe over all virtual view (includes checking the failed segmentations); *Total Time* is the complete time until convergence or abortion, and *perStep* is the average time for a single segmentation. Bottom half of the table shows average runtime of **CGAL** algorithms on this subset.

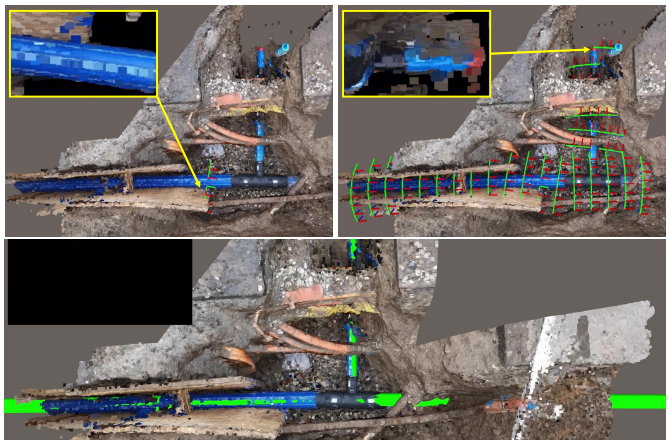


Fig. 16: Semantic Linking on the iPad. (Top) The algorithm iteratively adds new points to the work queue, until it fails in one direction and continues in another direction, ultimately terminating when running out of points. (Bottom) shows the final result in green.

We leverage virtual views to perform image-based segmentation and estimate the center of pipe-like structures over multiple views. The fitting algorithm takes GIS data and aligns them to the sampled pipe structure using a least-squares optimization. The evaluations show an accuracy of $< 10$ cm for the proposed segmentation approach, being within the SUE constraints of $< 20$ cm. Rotation errors remain mostly $< 1^\circ$, but contain inconsistencies in a few cases. Although rotation errors can result from bad segmentations at pipe ends, our results are clearly usable for the translational component. Our work should also be generalizable to other utilities. In addition to water pipes, there are several underground utilities present, including gas pipes, television or communication cables, sewer lines, electric lines, and occasionally metal and concrete pipes that serve various purposes. All these utilities use a pipe form-factor, which should make the algorithm applicable as well. Gas pipes resemble water pipes in terms of configuration, with the only distinction being the use of different colors. Therefore, the approach utilized for water pipes can also be applied to gas pipes. Similar circumstances apply to sewer lines and other metal or concrete pipes. Larger main sewer lines, which may not be completely exposed, can be handled, provided the color of the pipes is clearly distinguishable from the surrounding soil.

Challenges may arise when dealing with soft telecommunication and electricity cables, which are not as rigid and therefore tend to have slight bends and curves. Since these cables also have a small diameter, the segmentation process must account for shorter camera-to-cable distances. However, the segmentation of curved cables should generally be feasible. A primary difficulty with cables lies not within the segmentation process, but rather in the lack of available GIS data. No reporting obligations exist, and the displacement of cables during backfilling can cause discrepancies in recorded data. In addition, excavations that uncover multiple utilities, which are laid on top of each other, can pose challenges. Sporadic cross-over of utilities is not a problem, but more complex scenarios may require a different approach. GIS data could already be used during segmentation to skip non-segmentable parts that are hidden by other pipes.

Possible improvements and extensions to the segmentation algorithm and potential applications are worth exploring. These improvements include investigating the cause of height estimation problems at excavation borders. Moreover, we can optimize geo-referenced 3D recon-

structions captured by excavation workers using our semantic linking method as an offline approach, as we found that even accurate and recently registered infrastructure data is still off, in most cases by half the diameter of a pipe, due to the misinterpretation of geodesic measurements. Our work can also support the creation of a large dataset for a deep learning-based algorithm, which has the potential to produce fast, accurate and domain specific results, using the automatic segmentation as a basis for labeling training data.

To the best of our knowledge, no previous approach takes inaccurate SUE data into account and attempts to tackle this problem. The procedures currently in place in the SUE industry tend to be very static and rely on as-planned data even if it is of insufficient quality. We believe that our work can help modernize typical SUE work at excavation sites.

## Acknowledgements

Funded by the European Union under Grant Agreement No. 101092861. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the granting authority can be held responsible for them. Funded by the German Research Foundation (DFG) project 495135767 and the Austria Science Fund (FWF) project I 5912-N (joint Weave project). The project is associated with and further supported by the DFG Excellence Cluster EXC 2120/1 – 390831618.

## References

[1] S. Agarwal, K. Mierle, and T. C. S. Team. Ceres Solver, 3 2022. 5

[2] B. Bellekens, V. Spruyt, R. Berkvens, and M. Weyn. A survey of rigid 3d pointcloud registration algorithms. In *AMBIENT 2014 : The Fourth International Conference on Ambient Computing, Applications, Services and Technologies*, 08 2014. 2

[3] P. Besl and N. D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992. doi: 10.1109/34.121791 2

[4] R. Broome, P. Cornforth, S. Crossland, N. Metje, and A. Rhoades. Utility strike damage report, 2019. Visited online 3 Mar 2021, https://www.utilitystrikeavoidancegroup.org/uploads/1/3/6/6/13667105/usag_2019_strike_damages_report.pdf. 1

[5] D. Covaciu, I. Preda, D. Dima, and C. Anghel. Study on the possibility to estimate the vehicle side slip using two independent gps receivers. *Applied Mechanics and Materials*, Vol. 822:321–330, 01 2016. doi: 10.4028/www.scientific.net/AMM.822.321 3

[6] D. Crevier. Hue-based segmentation of color images. In *Proceedings of Canadian Conference on Electrical and Computer Engineering*, pp. 1250–1253 vol.2, 1993. doi: 10.1109/CCECE.1993.332483 2

[7] P. Dorninger and C. Nothegger. 3d segmentation of unstructured point clouds for building modelling. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 36, 01 2007. 2

[8] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, jun 1981. doi: 10.1145/358669.358692 2

[9] P. Furgale, T. D. Barfoot, and G. Sibley. Continuous-time batch estimation using temporal basis functions. In *2012 IEEE International Conference on Robotics and Automation*, pp. 2088–2095, 2012. doi: 10.1109/ICRA.2012.6225005 3

[10] P. Furgale, J. Rehder, and R. Siegwart. Unified temporal and spatial calibration for multi-sensor systems. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1280–1286, 2013. doi: 10.1109/IROS.2013.6696514 3

[11] N. Gelfand and L. Guibas. Shape segmentation using local slippage analysis. In *Proc. Symp. Geometry Processing*, vol. 2004, pp. 219–228, 01 2004. doi: 10.1145/1057432.1057461 2

[12] Gerhard Schall, Daniel Wagner, Gerhard Reitmayr, Elise Taichmann, Manfred Wieser, Dieter Schmalstieg and Bernhard Hofmann-Wellenhof. Global Pose Estimation using Multi-Sensor Fusion for Outdoor Augmented Reality. *International Symposium on Mixed and Augmented Reality*, pp. 153–162, 2009. 3

[13] Gerhard Schall, Stefanie Zollmann and Gerhard Reitmayr. Smart vidente: advances in mobile augmented reality for interactive visualization of underground infrastructure. *Personal and Ubiquitous Computing*, p. 1533–1549, 2012. 3

[14] W. R. Green and H. Grobler. Normal distribution transform graph-based point cloud segmentation. In *2015 Pattern Recognition Association of South Africa and Robotics and Mechatronics International Conference (PRASA-RobMech)*, pp. 54–59, 2015. doi: 10.1109/RoboMech.2015.7359498 2

[15] P. Guerrero, Y. Kleiman, M. Ovsjanikov, and N. J. Mitra. PCPNET: learning local shape properties from raw point clouds. *CoRR*, abs/1710.04954, 2017. 2

[16] L. Hansen. *Augmented Reality for Subsurface Utility Engineering: Exploring and developing 3D capture and AR visualization methods for subsurface utilities*. PhD thesis, Aalborg Universitet, 2021. PhD supervisor: Assoc. Prof. Erik Kjems, Aalborg University. doi: 10.54337/aau466407995 3

[17] L. Hansen, P. Fleck, M. Stranner, D. Schmalstieg, and C. Arth. Augmented reality for subsurface utility engineering, revisited. *IEEE Transactions on Visualization and Computer Graphics*, 27(11):4119–4128, Nov. 2021. Publisher Copyright: © 1995-2012 IEEE. doi: 10.1109/TVCG.2021.3106479 1, 2, 3

[18] L. Hansen, E. Kjems, and S. Wyke. Towards ar-enabled informed decision-making in subsurface utility projects through visualising 3d capture data of as-built utilities. Workingpaper, Aalborg Universitet, 2021. 2

[19] L. Hansen, T. Pedersen, E. Kjems, and S. Wyke. Smartphone-based reality capture for subsurface utilities: Experiences from water utility companies in denmark. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLVI-4/W4-2021:25–31, 10 2021. doi: 10.5194/isprs-archives-XLVI-4-W4-2021-25-2021 2

[20] L. Hansen, R. van Son, A. Wieser, and E. Kjems. Addressing the elephant in the underground: An argument for the integration of heterogeneous data sources for reconciliation of subsurface utility data. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLVI-4/W4-2021:43–48, 10 2021. doi: 10.5194/isprs-archives-XLVI-4-W4-2021-43-2021 2

[21] R. Haralick, H. Joo, C. LEE, X. ZHUANG, V. Vaidya, and M. Kim. Pose estimation from corresponding point data. *Systems, Man and Cybernetics, IEEE Transactions on*, 19:1426 – 1446, 12 1989. doi: 10.1109/21.44063 2

[22] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, New York, NY, USA, 2 ed., 2003. 2

[23] G. Iannizzotto and L. Vita. Fast and accurate edge-based segmentation with no contour smoothing in 2-d real images. *IEEE Transactions on Image Processing*, 9(7):1232–1237, 2000. doi: 10.1109/83.847835 2

[24] H. G. Kaganami and Z. Beiji. Region-based segmentation versus edge detection. In *2009 Fifth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pp. 1217–1221, 2009. doi: 10.1109/IIH-MSP.2009.13 2

[25] Z. Kang and Z. Li. Primitive fitting based on the efficient multibaysac algorithm. *PloS one*, 10:e0117341, 03 2015. doi: 10.1371/journal.pone.0117341 2

[26] J. Krumm. Intersection of two planes, May 2000. Microsoft Research. 6

[27] A. Kundu, X. Yin, A. Fathi, D. Ross, B. Brewington, T. Funkhouser, and C. Pantofaru. Virtual multi-view fusion for 3d semantic segmentation. In *ECCV*, pp. 518–535, 11 2020. doi: 10.1007/978-3-030-58586-0_31 2

[28] F. Lafarge and C. Mallet. Creating large-scale city models from 3d-point clouds: A robust approach with hybrid representation. *International Journal of Computer Vision*, 99, 08 2012. doi: 10.1007/s11263-012-0517-8 2, 7, 9

[29] Z. Lari and A. F. Habib. An adaptive approach for the segmentation and extraction of planar and linear/cylindrical features from laser scanning data. *Isprs Journal of Photogrammetry and Remote Sensing*, 93:192–212, 2014. 2

[30] L. Li, M. Sung, A. Dubrovina, L. Yi, and L. J. Guibas. Supervised fitting of geometric primitives to 3d point clouds. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2647–2655, 2019. doi: 10.1109/CVPR.2019.00276 2

[31] K.-L. Low. Linear least-squares optimization for point-to-plane icp surface registration. Technical report, University of Singapore, 01 2004. 2

[32] L. O. Makana, N. Metje, I. Jefferson, M. Sackey, and C. D. Rogers. Cost Estimation of Utility Strikes: Towards Proactive Management of Street Works. *Infrastructure Asset Management*, pp. 1–34, 2018. doi: 10.1680/jinam.17.00033 1

[33] I. P. Maurell, C. Ferreira, C. A. Eguti, and P. Drews-Jr. Geometric primitive fitting in large structure 3d point clouds acquired by drones. In *2021 Latin*

*American Robotics Symposium (LARS), 2021 Brazilian Symposium on Robotics (SBR), and 2021 Workshop on Robotics in Education (WRE)*, pp. 108–113. IEEE, 2021. 5

[34] A. Murtiyoso, C. Lhenry, T. Landes, P. Grussenmeyer, and E. Alby. Semantic segmentation for building faÇade 3d point cloud from 2d orthophoto images using transfer learning. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLIII-B2-2021:201–206, 06 2021. doi: 10.5194/isprs-archives-XLIII-B2-2021-201-2021 2

[35] A. Nguyen and B. Le. 3d point cloud segmentation: A survey. In *IEEE Conference on Robotics, Automation and Mechatronics, RAM - Proceedings*, pp. 225–230, 11 2013. doi: 10.1109/RAM.2013.6758588 2

[36] K. ODA, T. TAKANO, P. WANG, T. OHGA, T. DOIHARA, and R. Shibasaki. Automatic 3-d city modeling based on fusion of laser scanner data and aerial photo images. *Journal of the Japan society of photogrammetry and remote sensing*, 43:16–23, 01 2004. doi: 10.4287/jsprs.43.5_16 2

[37] E. Pellis, A. Murtiyoso, A. Masiero, G. Tucci, M. Betti, and P. Grussenmeyer. An image-based deep learning workflow for 3d heritage point cloud semantic segmentation. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLVI-2/W1-2022:429–434, 02 2022. doi: 10.5194/isprs-archives-XLVI-2-W1-2022-429-2022 2

[38] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas. Frustum pointnets for 3d object detection from RGB-D data. *CoRR*, abs/1711.08488, 2017. 2

[39] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *CoRR*, abs/1706.02413, 2017. 2

[40] T. Rabbani and F. Heuvel. Efficient hough transform for automatic detection of cylinders in point clouds. *Proc ISPRS Workshop Laser Scan 2005, ISPRS Arch*, 36, 01 2005. 2

[41] C. Romanengo, A. Raffo, S. Biasotti, and B. Falcidieno. Recognizing geometric primitives in 3d point clouds of mechanical cad objects. *Computer-Aided Design*, 157:103479, 2023. doi: 10.1016/j.cad.2023.103479 2

[42] C. Ruizhongtai Qi, H. Su, K. Mo, and L. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *CVPR*, 12 2016. 2

[43] S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. In *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*, pp. 145–152, 2001. doi: 10.1109/IM.2001.924423 2

[44] R. Rusu. Semantic 3d object maps for everyday manipulation in human living environments. *KI - Künstliche Intelligenz*, 24, 11 2010. doi: 10.1007/s13218-010-0059-6 2

[45] G. Schall, E. Mendez, and D. Schmalstieg. Virtual redlining for civil engineering in real environments. In *ISMAR*, pp. 95–98, 09 2008. doi: 10.1109/ISMAR.2008.4637332 3

[46] R. Schnabel, R. Wahl, and R. Klein. Efficient ransac for point-cloud shape detection. *Computer Graphics Forum*, 26(2):214–226, 2007. doi: 10.1111/j.1467-8659.2007.01016.x 2, 7, 9

[47] J. Schwartz and M. Sharir. Identification of partially obscured objects in two and three dimensions by matching noisy characteristic curves. *International Journal of Robotic Research - IJRR*, 6:29–44, 06 1987. doi: 10.1177/027836498700600203 2

[48] A. Segal, D. Hähnel, and S. Thrun. Generalized-icp. In *Proc. of Robotics: Science and Systems*, 06 2009. doi: 10.15607/RSS.2009.V.021 2

[49] G. Soria, L. Ortega, and F. Feito. Augmented and virtual reality for underground facilities management. *Journal of Computing and Information Science in Engineering*, 18:041008, 07 2018. doi: 10.1115/1.4040460 3

[50] A. Spreafico, F. Chiabrando, L. Teppati Losè, and F. Giulio Tonolo. The ipad pro built-in lidar sensor: 3d rapid mapping tests and quality assessment. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLIII-B1-2021:63–69, 06 2021. doi: 10.5194/isprs-archives-XLIII-B1-2021-63-2021 3

[51] M. Stranner. Subsurface infrastructure localization for gis data alignment using semantic segmentation. Master's thesis, Graz University of Technology, 03 2023. 3

[52] J. Strom, A. Richardson, and E. Olson. Graph-based segmentation for colored 3d laser point clouds. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2131 – 2136, 11 2010. doi: 10.1109/IROS.2010.5650459 2

[53] M. Sun, L. Lei, W. Huang, and X. Ren. Interactive registration for augmented reality gis. In *Proceedings of International Conference on Computer Vision in Remote Sensing, CVRS 2012*, pp. 246–251, 12 2012. doi: 10.1109/CVRS.2012.6421269 3

[54] J. Tang. A color image segmentation algorithm based on region growing. In *2010 2nd International Conference on Computer Engineering and Technology*, vol. 6, pp. V6–634–V6–637, 2010. doi: 10.1109/ICCET.2010.5486012 2

[55] F. Tarsha-Kurdi, T. Landes, and P. Grussenmeyer. Hough-Transform and Extended RANSAC Algorithms for Automatic Detection of 3D Building Roof Planes from Lidar Data. In *ISPRS Workshop on Laser Scanning 2007 and SilviLaser 2007*, vol. XXXVI, pp. 407–412. Espoo, Finland, Sept. 2007. 2

[56] The CGAL Project. *CGAL User and Reference Manual*. CGAL Editorial Board, 5.5.2 ed., 2023. 5

[57] G. Trigo, D. Donas-Boto, C. Silva, and J. E. Sanguino. Vehicle heading estimation using a two low-cost gps receiver configuration. In *2011 IEEE 73rd Vehicular Technology Conference (VTC Spring)*, pp. 1–5, 2011. doi: 10.1109/VETECS.2011.5956715 3

[58] D. Tóvári and N. Pfeifer. Segmentation based robust interpolation- a new approach to laser data filtering. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 36, 05 2012. 2

[59] A.-V. Vo, L. Truong-Hong, D. F. Laefer, and M. Bertolotto. Octree-based region growing for point cloud segmentation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 104:88–100, 2015. doi: 10.1016/j.isprsjprs.2015.01.011 2

[60] M. Vogt, A. Rips, and C. Emmelmann. Comparison of ipad pro®'s lidar and truedepth capabilities with an industrial 3d scanning solution. *Technologies*, 9:25, 04 2021. doi: 10.3390/technologies9020025 3

[61] G. Vosselman and S. Dijkman. 3d building model reconstruction from point clouds and ground plans. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXIV, 01 2001. 2

[62] J. Xu and F. Moreu. A review of augmented reality applications in civil infrastructure during the 4th industrial revolution. *Frontiers in Built Environment*, 7:640732, 2021. 3

[63] Y. Xu, L. Hoegner, S. Tuttas, and U. Stilla. Voxel- and graph-based point cloud segmentation of 3d scenes using perceptual grouping laws. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-1/W1:43–50, 05 2017. doi: 10.5194/isprs-annals-IV-1-W1-43-2017 2

[64] H. Yang, J. Shi, and L. Carlone. Teaser: Fast and certifiable point cloud registration. *IEEE Transactions on Robotics*, 37(2):314–333, 2021. doi: 10.1109/TRO.2020.3033695 2

[65] J.-H. Yoon and H. Peng. Vehicle sideslip angle estimation using two single-antenna gps receivers. In *ASME Dynamic Systems and Control Conference*, vol. 2, 09 2010. doi: 10.1115/DSCC2010-4249 3

[66] Y. Zhou and O. Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection, 2017. doi: 10.48550/ARXIV.1711.06396 2

[67] S. Zollmann, R. Grasset, G. Reitmayr, and T. Langlotz. Image-based x-ray visualization techniques for spatial understanding in outdoor augmented reality. In *Proc. of ACM OzCHI*, 12 2014. doi: 10.1145/2686612.2686642 3