2025 IEEE International Conference on Artificial Intelligence and eXtended and Virtual Reality (AIxVR) | 979-8-3315-2157-8/25/$31.00 ©2025 IEEE | DOI: 10.1109/AIXVR63409.2025.00023

Cooperative Perception for Digital Twin Reconstruction*

1st Leonel Toledo
i2CAT Foundation, Spain
leonel.toledo@i2cat.net

2nd Ivan Huerta
i2CAT Foundation, Spain
ivan.huerta@i2cat.net

3rd Eric Casadella
i2CAT Foundation , Spain
eric.casadella@i2cat.net

4rd Ignasi Mas
i2CAT Foundation , Spain
ignasi.mas@i2cat.net

5th Maya Antoun
American University of Beirut (AUB) , Lebanon
mja27@mail.aub.edu

6th Wissam Tedros
AUB, Lebanon
wt14@aub.edu.lb

7th Daniel Asmar
AUB, Lebanon
da20@aub.edu.lb

8th Gerasimos Arvanitis
University of Patras (UPAT), Greece
arvanitis@ece.upatras.gr

9th Konstantinos Moustakas
UPAT, Greece
moustakas@upatras.gr

10th Efthymios Koukoulis
UPAT, Greece
koukoulis@ece.upatras.gr

11th Aleksandar Jevtic
Ficosa Automotive SLU, Spain
aleksandar.jevtic@ficosa.com

12th Romain Guesdon
Ficosa Automotive SLU, Spain
romain.guesdon@ficosa.com

13th Alberto Lara
Ficosa Automotive SLU, Spain
alberto.lara@ficosa.com

14th Sergi Fernandez
i2CAT Foundation, Spain
sergi.fernandez@i2cat.net

***Abstract*—This work introduces an automated system for generating digital twins of urban environments by integrating data from multiple sensors, including a mobile vehicle equipped with various cameras and a LiDAR sensor, as well as strategically placed stationary cameras. Through cooperative perception across these devices, the system improves the accuracy and precision of data capture. To achieve a detailed 3D reconstruction, we employ a hybrid SLAM and mapping approach that produces a dense point cloud. This preliminary point cloud is further refined with data from the stationary sensors using the FGICP point cloud registration method. The resulting digital twin produced by this pipeline are directly fine-tuned and have versatile applications across fields such as urban simulation and planning, cultural heritage preservation, traffic analysis, and more.**

***Index Terms*—Infrastructure, vehicle perception, sensor data, cooperation for 3D reconstruction, digital twin, SLAM, PCR, cooperative perception.**

I. Introduction

Digital twin is an emerging technology, driven by physical data and model, digital twins can perform virtual simulations of physical assets [1] . On top of that, this technology is relevant for making predictions, optimization and decision making. To do so, digital twins should provide a faithful recreation of the physical entity. Effective application of digital twins should be built from multiple modeling aspects such as geometrical structure, functionality, state, location, process and performance [2]. By integrating data collected from sensors, IoT devices, mobile phones, static cameras etc it is possible to constantly monitor and update the digital twin to accurately reflect changes in the physical entities.

Leveraging data from both static and dynamic sources can be helpful in gathering the information needed to build a large-scale digital twin, such as a city-scale digital twin. We use a recording car for this work. Cars can be easily equipped with different types of sensors, such as LIDARs and cameras, which can easily capture real-time information about the environment.

Digital twins that include a physical layer require the construction and update of a 3D geometric model that represents the twin. The traditional pipeline to do so is scan/mesh/render, where the scanning can be realized using either depth sensors (such as LIDARS or stereo cameras) or estimating it from multiple images using photogrammetry.

Pose estimation is another key element for digital twin creation and management. 3D reconstruction of a twin usually requires the knowledge of the pose of the mapping agent, estimated either independently of the mapping process (*e.g.* GPS) or simultaneously with the mapping in the spirit of SLAM (Simultaneous Localization and Mapping). When using GPS, mapping relies on predefined spatial coordinates to construct the 3D model, suitable for environments with a clear line of sight to satellites. SLAM, in contrast, enables real-time

This work has been mainly funded by by the European Union's Horizon Europe program under agreement nº 101092875 (DidymosXR project). This work has been partially funded by Agencia Estatal de Investigación (AEI), in the framework of Proyecto Estratégico Orientado a la Transición Ecológica y a la Transición Digital (TED) 2021, under agreement TED2021-131690B-C32 and TED2021-131690A-C33 (REVOLUTION project), in the framework of Proyectos Generación de Conocimiento 2022, under agreement PID2022-140749OBI0 (EVOLVE project). It has particularly funded by the Ministry for Digital Transformation and of Civil Service and by the "European Union NextGenerationEU/PRTR" within the call "UNICO-5G I+D: Programa de Universalización de Infraestructuras Digitales para la Cohesión – 2021" with Grant TSI-063000-2021-31 (6GTwinRoad). Likewise, this work has been partially funded by the European Union's Horizon Europe program under agreement nº 101070250 (XRECO project).

2771-7453/25/$31.00 ©2025 IEEE
DOI 10.1109/AIxVR63409.2025.00023

mapping by simultaneously localizing the agent and updating the digital twin to reflect changes in the physical environment, making it well-suited for complex spaces.

Point cloud registration involves aligning two or more point clouds within a shared coordinate system. This process is essential for tasks like SLAM, 3D reconstruction, and object recognition, among others. Despite its importance, point cloud registration presents significant challenges due to factors such as noise, clutter, and occlusion.

Digital twin technologies are relevant and widely used in industries such as manufacturing, healthcare, urban planning, etc. Providing important insights for optimizing operations, efficiency and assistance to the decision making process through simulations and predictive models. The contribution of this work is the description of an automatic pipeline that collects data from different sensors, namely a moving automobile and independent static cameras placed in different spots in a city to create a first version of the digital twin. To further refine the reconstruction, a Hybrid SLAM and mapping is performed, rendering a dense point cloud which using FGICP, a PCR method, is registered with the output of the static cameras. The technical details are further explained in section III and the results of the experiment are provided in section IV.

II. Related Work

The advent of ADAS and autonomous driving systems has led to a significant advancement in surround view perception systems. One of the most prevalent approaches is the utilization of RGB cameras, typically fisheye cameras [3], [4]. This enables the creation of a comprehensive 360° perception of the surrounding environment, including texture and color, while minimizing costs. However, this approach lacks the robust 3D information essential for the construction of digital twins. To address this limitation, several technical solutions can be employed, including stereo [5], time of flight [6], or LiDAR [7]. In this study, we have opted to integrate a fisheye surround view system with a top 360° LiDAR, thereby acquiring both color information and a 3D point cloud of the environment.

Visual SLAM is a method for constructing a 3D point cloud of a scene and localizing oneself within that scene. The process begins with feature extraction, where distinctive points, edges, or texture in the camera feed are identified. These features are then matched across frames to track movement, allowing for both mapping and localization. As the camera moves, Visual SLAM triangulates the position of features to reconstruct a 3D point cloud of the environment, representing the spatial arrangement of objects in the scene.

Visual SLAM systems [8] can be broadly categorized into direct, indirect, or a hybrid of both. Direct methods (e.g., DSO [9]) use raw pixel intensity values to estimate both camera motion and scene structure. By leveraging all available pixel information, direct SLAM methods are typically more robust in low-textured environments. Indirect methods (*e.g.*, ORB-SLAM3 [10]), on the other hand, rely on extracting and matching distinctive key points or features, which makes them more computationally efficient and suitable for real-time applications, though they may struggle in low-texture settings. Hybrid methods (*e.g.*, H-SLAM [11], [12]) combine aspects of both direct and indirect approaches, balancing accuracy and computational efficiency by using both feature-based and intensity-based tracking. This versatility enables hybrid SLAM to perform robustly across a range of environments and conditions, making it ideal for complex and dynamic scenes.

Point Cloud Registration (PCR) can be used for aligning multiple point clouds into a common coordinate system, the Iterative Closest Point (ICP) algorithm [13] is perhaps the most widely used method for PCR. Generalized ICP (GICP) [14] extends ICP, allowing additional information to be incorporated into the registration process, beyond the points' positions. Modern approaches often use soft correspondence methods, like the widely-known Coherent Point Drift (CPD) algorithm [15]. In CPD, one point cloud is treated as being generated by the Gaussian Mixture Model (GMM) of the other. The expectation-maximization algorithm is then used to solve this optimization problem. An extension of this method for registering multiple point sets is explored in [16]. While GMM-based techniques offer significant power and flexibility, they are also computationally intensive, presenting challenges for their integration into real-time applications.

III. Methodology

To achieve a pipeline for digital twin reconstruction, several components have to be considered and processed. With cooperative perception, a point cloud that represents the digital twin of a city is built using different data sources. Data is first acquired from multiple sources: a moving vehicle equipped with RGB cameras and a rotating LiDAR that provides a 360-degree view. This data is then combined with information from a stationary sensor system, which includes both GPS for location tracking and a stereo camera for capturing RGB data and estimating depth, supporting 3D reconstruction. Once all the real world information is collected, a SLAM and mapping process is applied, which renders a dense point cloud from the vehicle data sensors, which is used later for point cloud registration to fuse the extracted point clouds from the static and vehicles data sources. Finally, the resulting point cloud is visualized in real-time using a GPU-based module that allows to process millions of points at low processing times (less than 2ms).

A. Data Acquisition

1) Vehicle-centered acquisition: Urban areas are complex and dynamic, making them challenging to model. To tackle this, we used a vehicle-based capture system equipped with RGB cameras and a rotating LiDAR (Figure 1), therefore offering comprehensive 360º color and 3D data capture, supplemented by dGPS for precise positioning and CAN signals for additional vehicle data.

a) Sensors: The vehicle's surround-view system consists of four wide-angle cameras, each with a 1.3 Mpx resolution, providing a nearly complete RGB scene coverage around the

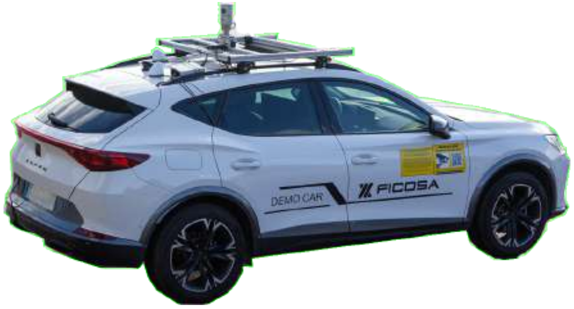


Fig. 1: FICOSA recording demo car

car (Figure 2). The LiDAR, a 32-bit HDL32E by Velodyne rotating at 10 Hz, mounted on the car's roof, offers high-definition 3D environment modeling with a detection range of up to 100 meters and minimal error. For precise navigation and positioning, the vehicle uses an XNAV650 v3 dGPS system, enhanced with an Inertial Measurement Unit (IMU) and capable of receiving signals from multiple GNSS constellations. Multi-sensor synchronization is achieved by setting the dGPS time as the master clock, ensuring all sensor data is accurately timestamped and synchronized, crucial for the integrity of multi-modal applications.

b) RGB point cloud generation: From the recorded 3D point cloud data and RGB images, we generate a colored point cloud. This data stream is useful for recognition algorithms and digital twin construction. Since the car's sensors are calibrated, the rigid transformations from each sensor's 3D space to the car are known. We therefore project the 3D point cloud onto the images using each camera's intrinsic parameters, allowing to match the 2D points with the projected 3D points. In the end, the color of a 3D point X seen by the camera n is defined by the function:

$$\text{RGB}_n(X) = I_n(K_n \cdot P_n \cdot X) \tag{1}$$

where K and P are the intrinsic and extrinsic parameters relative to the LiDAR, and I is the camera image. An example of point cloud is shown in Figure 3.

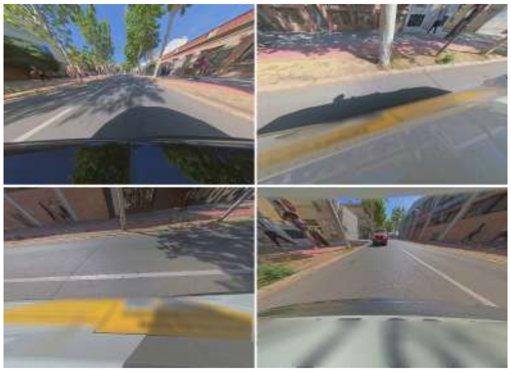


Fig. 2: Points of view from the four on-board cameras.

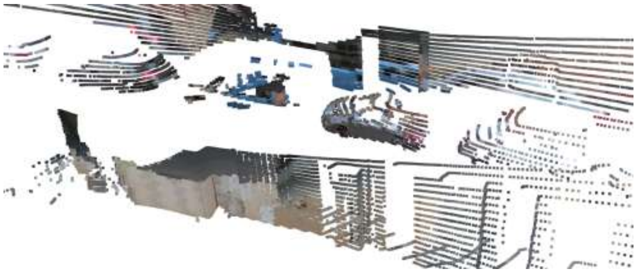


Fig. 3: Sample of 3D RGB point cloud

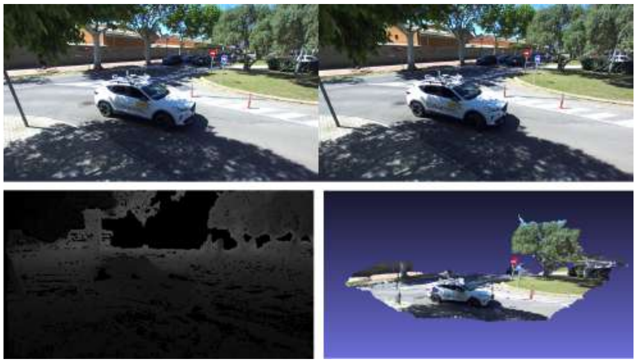


Fig. 4: An example of the frames and point clouds extracted from the static sensor.

2) Static Sensor Data acquisition: Collecting data from different sensors is important to enhance the quality and fidelity of digital twins. We can add more data to the digital twin using static sensors, we perform this process with two main components. A GNSS module and the camera feed. The GNSS module is a C100-F9K of U-BLOX. Which is used to drive the capture and allow for better synchronization with the on-board detection. It also allows the acquisition of high-quality positions of the capture. On the other hand, we use a Zed2 stereo camera that estimates the depth similarly as humans, in other words, it finds correspondences between two different views of the same scene, these views are separated by a known distance and the depth is inferred by their displacement. Stereo cameras are also able to estimate depth to a further distance than other devices, this is possible to achieve by increasing the distance between the sensors. The setup renders a capture range from 0.3m to 20m. To facilitate the synchronization with the on-board sensors, the static capturing system is guided by the timestamps received by the GNSS module. The timestamps are received periodically and the frequency can be adjusted. Every time a new timestamp is collected, frames are requested from the cameras and a point cloud is generated to update the digital twin.

Figure 4 shows an example of the frames obtained from the stereo static camera, on the top, both frames of the stereo devices are shown. On the bottom left, the resulting depth estimation is shown, which is estimated from the distance between both cameras. Finally, on the bottom right, the resulting point cloud is shown. Which is computed from the combination of both the depth estimation and the captured color information.

B. SLAM + MAPPING

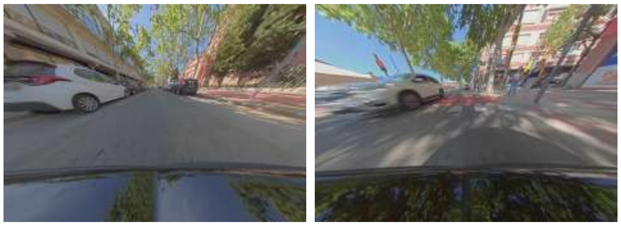
In a car setup of 4 cameras (front, rear, 2 sides) and a LIDAR, one would ideally hope to implement SLAM on each of the sensors and then register the resulting five points clouds into a dense 3D reconstruction of the scene. Unfortunately, while visual SLAM successfully processed the front camera stream, none of the visual SLAM system we tested were successful in processing the video streams from the two side cameras and the rear one. First, the four stock cameras of the mobile vehicle are fisheye cameras with significant radial and barrel distortion. Second, all cameras streams include regions that are static, as shown in Figure 5, which results in a significant area in each image that does not serve SLAM. Third, a large number of images are significantly blurred and as such have to be discarded. Finally, the orientation of the side cameras are not favorable to SLAM given the little overlap of adjacent frames and the predominant lateral motion with almost no forward motion.

As a result, producing a point cloud required us to rely on the front camera alone in which H-SLAM [11], [12] was successful in localizing, and then in propagating that motion estimate to the remaining 3 cameras. Once the motion was estimated for each camera, mapping is achieved using a spatial mapper to facilitate the construction of a comprehensive 3D representation of the environment based on the known camera poses (Figure 6)

H-SLAM is applied to the frames captured by the front camera to estimate the vehicle's trajectory, as shown in Figure 7.

With the frontal camera's trajectory and the extrinsics of the other three cameras, we can determine the trajectories of the remaining cameras, as they are all rigidly mounted to the vehicle and the transformation between any two cameras remains constant. Equation 2 illustrates how to obtain the rigid transformation between the frontal camera and any other arbitrary camera.

$$\mathbf{T}_{\text{front}\rightarrow\text{target}} = \begin{bmatrix} R_{\text{target}} & P_{\text{target}} \\ 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} R_{\text{front}} & P_{\text{front}} \\ 0 & 1 \end{bmatrix} \quad (2)$$



(a) static

(b) blurry

Fig. 5: Left: an example frame showing the hood of the vehicle in constant view of the camera; right: blurry image.

By applying the transformation matrix from the frontal camera to each pose matrix of the target camera (right, left, and rear), as shown in Equation 3, we can generate the corresponding trajectories for the side and rear cameras.

$$\mathbf{T}_{\text{target}} = \mathbf{T}_{\text{front}\rightarrow\text{target}} \begin{bmatrix} R_{\text{front}} & P_{\text{front}} \\ 0 & 1 \end{bmatrix} \quad (3)$$

After transferring the estimated camera trajectory from the front to the side and rear cameras, a spatial mapping technique (e.g., Multi-View Stereo) is applied to generate 3D point clouds for each camera. Figure 8 displays the resulting 3D maps. Alongside the RGB camera-based point clouds, the output from the LIDAR sensor is utilized to create a fifth 3D map of the scene.

C. Point Cloud Registration

Point Cloud Registration (PCR) is the process of aligning multiple point clouds into a common coordinate system by finding the spatial transformations that relate them. This problem is essentially the dual problem of estimating the pose from which each point cloud was captured.

The distance between a point p and a set Q is defined as $d(p,Q) = \min_{q\in Q} d(p,q)$. We denote $\mathcal{C}(p,Q) = \arg\min_{q\in Q} d(p,q)$, the point in Q closest to p. In PCR, the objective is to find the transformation T that registers a point cloud P in the coordinate system of a point cloud Q. Typically, this transformation is obtained by minimizing an objective function, with the least squares being the most common:

$$T^{\star} = \arg\min_{T} \sum_{i=1}^{N} d^2(T(p_i), Q) \quad (4)$$

.

ICP [13] addresses the optimization problem in Eq. (4) in an iterative, expectation-maximization-like manner. Starting from an initial estimate T^0 for the transformation, ICP alternates between the correspondence and the optimization step, as shown in the following equations.

Correspondence Step:

$$s_i^k = \mathcal{C}(T^k(p_i), Q) \quad (5)$$

Optimization Step:

$$T^{k+1} = \arg\min_{T} \sum_{i=1}^{N} d^2(T(p_i), s_i^k) \quad (6)$$

During the correspondence step, ICP updates the correspondences based on the current estimate of the transformation. In the optimization step, the algorithm refines the transformation based on the newly established correspondences via optimization. This procedure is repeated until the algorithm converges. It has been demonstrated that the algorithm converges to the solution $T^{\star}$ of Eq. (4) provided that the initial estimate T^0 is sufficiently close to $T^{\star}$. It is important to highlight that, the correspondence-based problem occurring in the optimization step has a closed-form solution [17], [18]. This renders ICP a highly efficient algorithm.

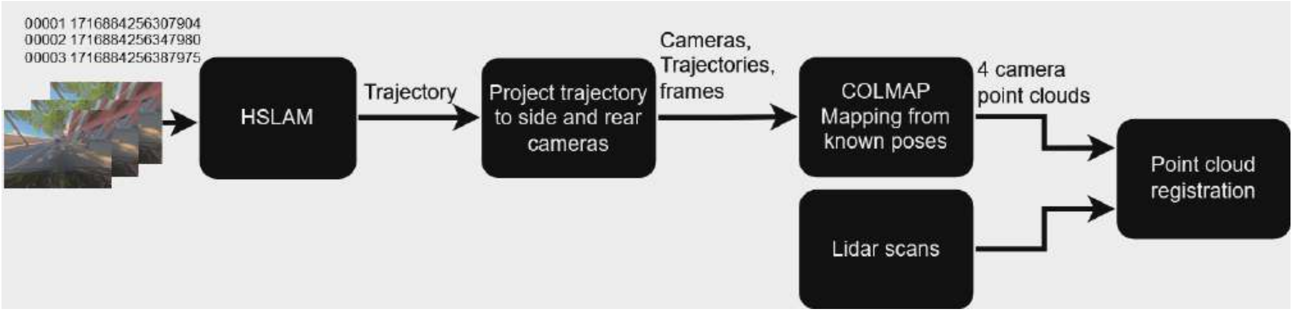


Fig. 6: 3D-scanning pipeline, include H-SLAM for localizing the front camera, and COLMAP for mapping from the side and rear cameras.

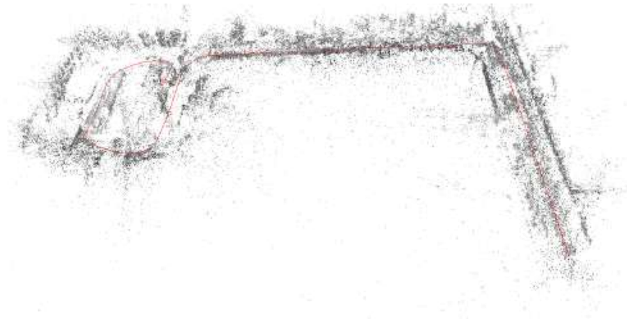


Fig. 7: Estimated car trajectory using H-SLAM on the front camera.

In GICP [19], each point is augmented with a covariance matrix. This covariance matrix aims to capture the properties of the underlying surface in the vicinity of the point. At each iteration, the standard correspondence-based optimization problem (6) is reformulated into the generalized form:

$$T^{k+1} = \arg\min_{T} \sum_{i=1}^{N} d^2_{\mathcal{M}}(T(p_i), s_i^k) \tag{7}$$

where $d_{\mathcal{M}}(T(p_i), s_i)$ represents the Mahalanobis distance between $T(p_i)$ and s_i. If $T(B_i)$ and D_i are the covariance matrices associated with $T(p_i)$ and s_i, respectively, the covariance matrix of their difference (the error) $e_i = T(p_i) - s_i$ is given by $A_i = T(B_i) + D_i$ and the Mahalanobis distance is defined as: $d^2_{\mathcal{M}}(T(p_i), s_i) = e_i^T A_i^{-1} e_i$

Our approach leverages the GICP framework to enhance the registration process by incorporating orientation information, specifically the normals, along with the points' positions, as detailed in Subsections III-C1, III-C2. In Subsection III-C3, the pipeline of our method for reconstructing a scene from a sequence of point clouds using PCR is presented.

1) Construction of the Covariance Matrices: Principal Components Analysis (PCA) is the most fundamental technique for extracting information about the local structure of the surface in the vicinity of a point by analyzing its neighborhood. In practice, the ”neighborhood” of a point is defined as the set of points within a specified range from the point of its k nearest neighbors (kNN). The PCA of a set of points $\{p_i\}_{i=1}^{K}$ is the eigenanalysis of its covariance matrix:

$$C = \sum_{i=1}^{K} (p_i - \bar{p})(p_i - \bar{p})^T$$

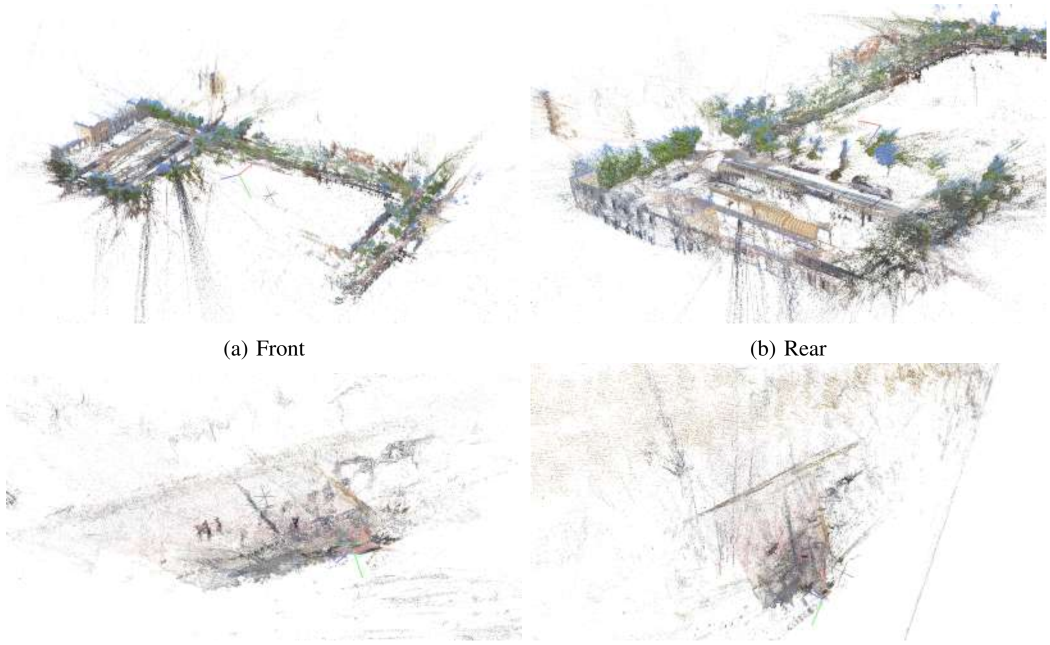
The eigenvectors of C are related to the surface orientation in space, with the eigenvector associated with the smallest eigenvalue indicating the direction of the normal vector. C can be written as $C = Q\Sigma Q^T$, where Q is an orthogonal matrix whose columns are the eigenvectors, and Σ is a diagonal matrix whose elements along the diagonal are the eigenvalues of C. This is known as the eigendecomposition of C. For our analysis, Σ and Q must be ordered such that the last diagonal element of Σ corresponds to the smallest eigenvalue and the last column of Q to its associated eigenvector. We reconstruct the covariance matrix as $\tilde{C} = Q\tilde{\Sigma}Q^T$, where $\tilde{\Sigma} = \text{diag}([1, 1, e])$ and e is a small positive constant. In that way, we impose a greater penalty for the error in the direction of the normal.

2) Optimization on the group of 3D Euclidean Transformations: An Euclidean or rigid transformation is a geometric transformation that can be expressed as $p' = Rp + t$, where R is a rotation matrix and t is a translation vector. In three-dimensional space, any rotation can be expressed as a composition of three rotations around the coordinate axes:

$$R(a_x, a_y, a_z) = R_z(a_z) \cdot R_y(a_y) \cdot R_x(a_x) \tag{8}$$

So any 3D Euclidean transformation can be encoded using a 6 parameters vector $\theta = [t_x, t_y, t_z, a_x, a_y, a_z]^T$, consisting of three translational and three angular components.

The FGICP method [20] demonstrates how the optimization of the transformation within the 6 degrees of freedom space of Euclidean transformations is achieved, therefore, we will not delve into the details here. Briefly, the fundamental step of the method is substituting the optimization function Eq. (7) at each step with its quadratic approximation, evaluated at the point under consideration in the parameter space. Once the objective function is replaced by its quadratic approximation, we minimize the latter instead. The advantage of this substitution is that, unlike the original function, we can determine the minimum of the quadratic approximation in closed form.



(a) Front

(b) Rear

(c) Left

(d) Right

Fig. 8: Visulization of the 3D point cloud output for the front, rear, left, and right cameras.

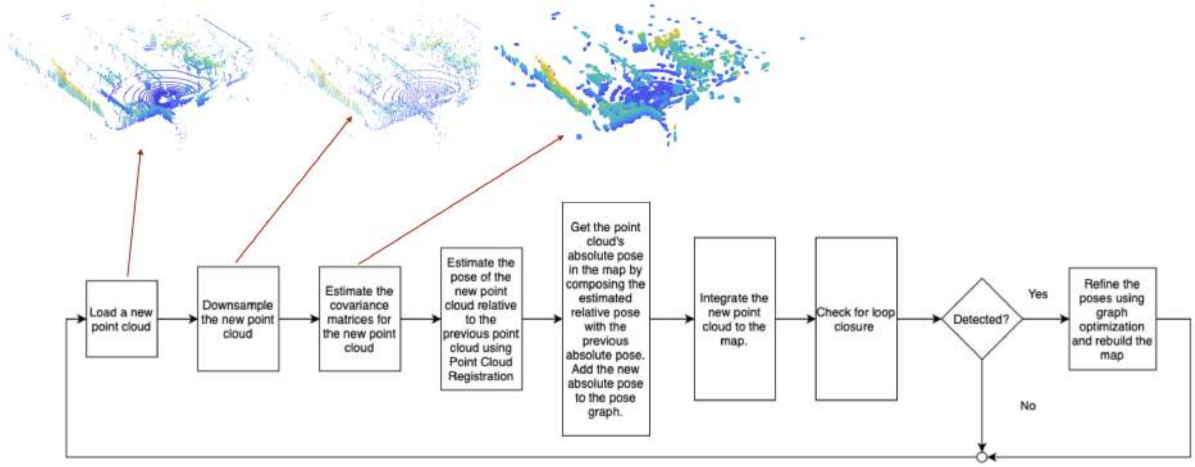

Load a new point cloud
Downsample the new point cloud
Estimate the covariance matrices for the new point cloud
Estimate the pose of the new point cloud relative to the previous point cloud using Point Cloud Registration
Get the point cloud's absolute pose in the map by composing the estimated relative pose with the previous absolute pose. Add the new absolute pose to the pose graph.
Integrate the new point cloud to the map.
Check for loop closure
Detected?
Yes
No
Refine the poses using graph optimization and rebuild the map

Fig. 9: Reconstruction Pipeline

3) Pipeline: We present the reconstruction pipeline in Figure 9. The output of the reconstruction if a high resolution point cloud model of the scene. The map is initialized with the first point cloud in the sequence. For each new point cloud to be integrated, Voxel Grid Downsampling is performed, in order to achieve a balanced representation of the all regions, and then the covariance matrices are computed. To predict the pose of the next point cloud in the map, we use the constant velocity state space model:

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \mathbf{v}_n + \mathbf{u}_n$$

where $\mathbf{x}_n$ is the pose at time n, $\mathbf{v}_n$ is the velocity, and $\mathbf{u}_n$ represents control input or noise. We refine the predicted pose via PCR. The pipeline concludes with a loop closure detector [21] and a graph optimization scheme [22].

D. Visualization

The process of constructing and rendering dense digital twins is computationally expensive and it poses several challenges, for instance, rendering hundreds of thousands or even millions of elements in real time is not trivial. A lot of memory and processing power is required. Point-based geometry has

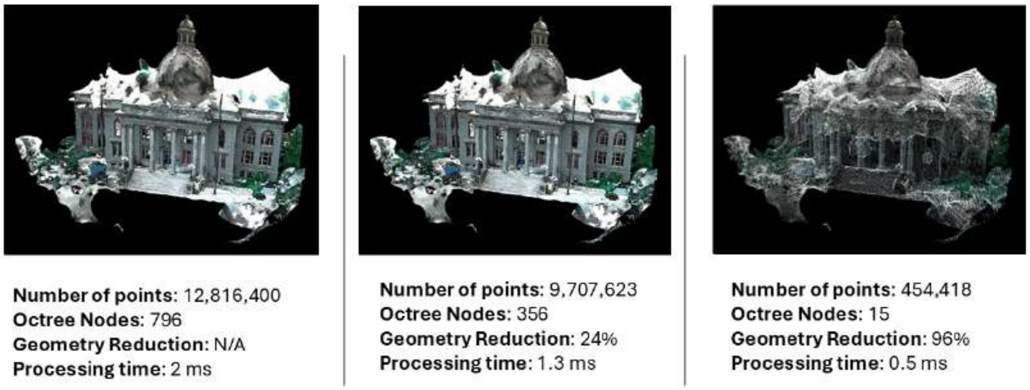


Fig. 10: Different levels of details using a 12 million voxels point cloud.

gained attention as an alternative for surface representation, due to the flexibility the format provides to render complex 3D models. Point clouds are one of the most common volumetric representation formats, due to its capability to describe geometry, colors, normals and transformations. Taking this format as an input, a real-time rendering system has been developed, using an octree-based representation to store the point cloud information in different levels. The geometry is calculated depending on the level of the octree, which is selected by traversing the different levels of the data structure. The fully detailed geometry is stored in the leaves and as the data structure goes to the upper levels, each node of the data structure estimates how the geometry should be rendered. This is calculated by sub-sampling and averaging the data in each of the nodes, this is true for all intermediate levels.In the same way as the work of Schutz et al. [23], the inner nodes are populated with lower resolution voxel models from bottom-up. If a new vertex is added to any node and it surpasses the geometry limit, the node is split averaging the colors and geometry of the involved nodes. Therefore, generating spatially constrained spaces, that can be processed in parallel.

After partitioning the input point cloud and populating the nodes, an additional list with a coarser, voxelized representation of the child nodes is considered too. That way when the representation is rendered using few nodes, it is possible to fill the gaps by using this voxelized representation, preserving the original shape of the point cloud in case the rendering device has processing bottlenecks. This discrete level of detail approach allows the system to render large point clouds (10 Million points or more) at interactive frame rates.

Figure 10 shows how the visualization is performed. On the left hand side, the whole model is shown without any decimation or level of detail reduction, for this level of detail, the full octree is visualized, which requires 796 nodes to display the full geometry, since the workload is offloaded to the GPU, it requires around 2 milliseconds to both process and render the digital Twin. On the center, the visualization is performed using some of the intermediate level nodes, in this case 356 nodes are considered, this provides a 24 per cent decrease of geometry, further reducing the processing time to 1.3 milliseconds. Finally, on the right hand side a visualization using information of few nodes is shown, in this case only 15 nodes are computed, and geometry is reduced to less than half a million points. In this case the processing time is less than 0.5 milliseconds and the complexity of the model is reduced to 4 percent of it's original size.

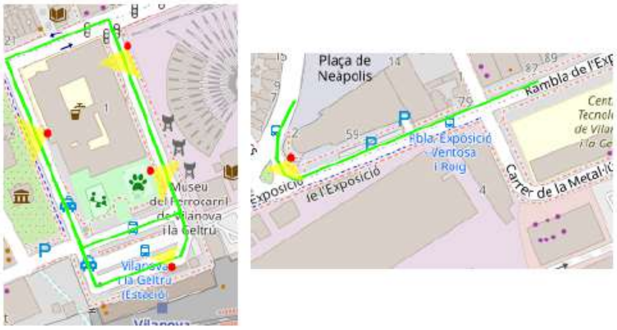

Museu del Ferrocarril de Vilanova i la Geltrú
Vilanova i la Geltrú (Estació)
Vilanova
Plaça de Neàpolis
Rambla de l'Exp
Rbla. Exposició Ventosa i Roig
Exposició
ie l'Exposició
Carrer de la Metal·lú
Cent. Tecnol. de Vilan i la Ge

Fig. 11: Map of the two recording sites in Vilanova: The train station (left) and Neapolis (right). The car routes are shown in green, while the static cameras are represented in red and yellow.

IV. Experiment Results and Discussion

The recordings for this experiment are made in the city center of Vilanova. The routes are divided into two main test sites: the train station and the technological center of Neapolis (Figure 11). Short recording sessions were performed there, on 100-300m segments at low speed (< 20km/h). Each segment passes in front of a static camera installed on one of the city's facilities, allowing to capture both the car and the infrastructure viewpoints.

As discussed previously, this work presents a semi-automatic digital twin creation pipeline where data comes from different devices in order to describe reality more accurately, the captured data is put together using SLAM and mapping algorithm, and PCR is performed to further refine the resulting digital twin. After generating the 3D maps for the 4 cameras (Figure 8) and the LIDAR (Figure 12), these maps are aligned together to create a dense point cloud for the digital twin.

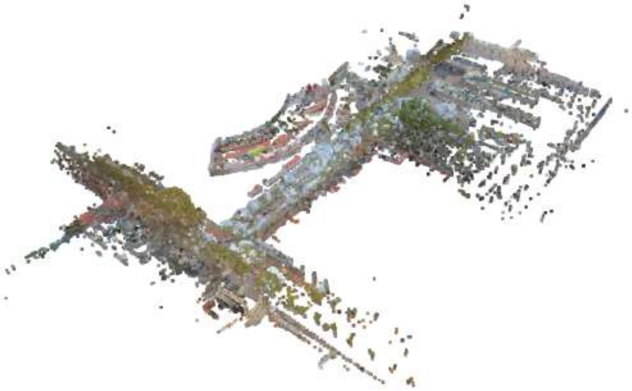


Fig. 12: 3D point cloud generated by the LIDAR.

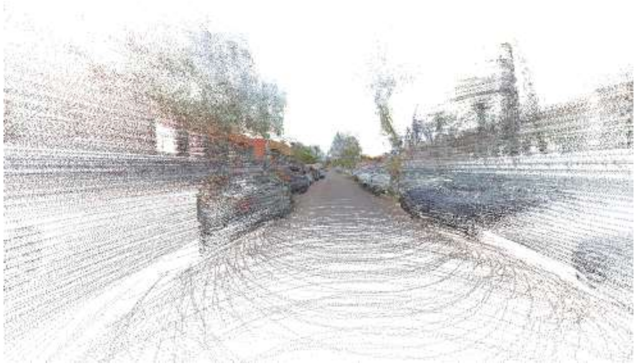


Fig. 13: View of a reconstructed scene of the dataset

We can see a view of a reconstructed scene of the dataset used for the point cloud registration in Figure 13. For performance evaluation of the point cloud registration, we measure the mean computation time per point cloud and the mean translation and rotation error. The results are presented in Table I. We use the GPS measurements as ground truth values. The errors between the estimated R, t and the ground truth $R^\star, t^\star$ are calculated as:

$$\mathrm{errt} = \|\mathbf{t} - \mathbf{t}^\star\|$$

$$\mathrm{erra} = \cos^{-1}\left(\left(\mathrm{trace}(RR^{\star -1}) - 1\right)/2\right)$$

Metric	Value
Mean Computation Time	30 ms
Mean Translation Error	0.01 m
Mean Rotation Error	0.075°

TABLE I: Performance Evaluation Metrics

Applying PCR from section III-C over the vehicle and infrastructure point clouds is challenging for two reasons. First, it is difficult to register both point clouds since they are very different in nature, for instance, the city point cloud is very sparse and overall has less points per region. Second, the point clouds collected from the sensors are significantly denser and noisier, making it difficult to find common features to use for alignment. Having these problems results in the no convergence of both point clouds making it not suitable for automatic registration. For that reason it is necessary to perform additional tool assisted adjustments to fully demonstrate the capabilities of cooperative perception between different data sources such as vehicles and infrastructure In Figure 14 can be seen the results of the process showing how cooperative perception can help with the enhancement of the data and coverage for the digital twin.

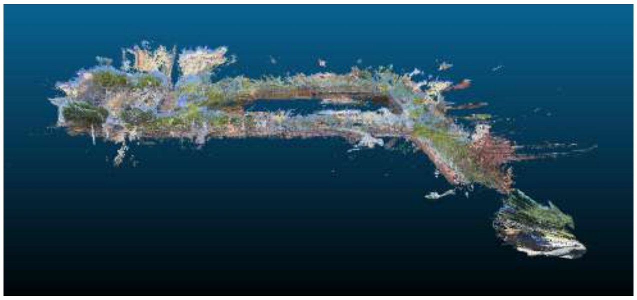


Fig. 14: Final point cloud reconstruction from both the vehicle and the infrastructure point clouds

V. Conclusions

This paper describes an automated process for creating an initial digital twin of an urban area by combining data from various sensors and employing cooperative perception to improve data precision and range. To achieve an accurate 3D model, we utilize a hybrid SLAM and mapping technique to generate a dense point cloud, which is then precisely aligned with static camera data using the FGICP method. The resulting point cloud undergoes additional tool-assisted adjustments to finalize a high-quality 3D reconstruction, illustrating the effectiveness of cooperative perception in developing detailed digital twins. Future research will need to address challenges related to data quality in point cloud alignment, as inconsistencies can impact the final model's fidelity. Tackling these data quality issues will be essential for strengthening the accuracy and scalability of digital twin reconstructions, particularly in complex settings. Future work needs to review the problem of the data quality that affects the point cloud registration. Future work should address the challenge of data quality in point cloud registration, as the data can impact the final accuracy of the digital twin. Addressing these issues will be crucial for enhancing the robustness and scalability of digital twin reconstruction in complex environments.

References

[1] F. Tao, B. Xiao, Q. Qi, J. Cheng, and P. Ji, "Digital twin modeling," *Journal of Manufacturing Systems*, vol. 64, pp. 372–389, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0278612522001108

[2] V. V. Tuhaise, J. H. M. Tah, and F. H. Abanda, “Technologies for digital twin applications in construction,” *Automation in Construction*, vol. 152, p. 104931, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0926580523001917

[3] C. Eising, J. Horgan, and S. Yogamani, “Near-field perception for low-speed vehicle automation using surround-view fisheye cameras,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 9, pp. 13 976–13 993, 2021.

[4] V. R. Kumar, C. Eising, C. Witt, and S. K. Yogamani, “Surround-view fisheye camera perception for automated driving: Overview, survey & challenges,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 4, pp. 3638–3659, 2023.

[5] M. Derome, A. Plyer, M. Sanfourche, and G. L. Besnerais, “Moving object detection in real-time using stereo from a mobile platform,” *Unmanned Systems*, vol. 3, no. 04, pp. 253–266, 2015.

[6] C. Nahler, C. Steger, and N. Druml, “Quantitative and qualitative evaluation methods of automotive time of flight based sensors,” in *2020 23rd Euromicro Conference on Digital System Design (DSD)*, 2020, pp. 651–659.

[7] G. A. Kumar, J. H. Lee, J. Hwang, J. Park, S. H. Youn, and S. Kwon, “Lidar and camera fusion approach for object distance estimation in self-driving vehicles,” *Symmetry*, vol. 12, no. 2, p. 324, 2020.

[8] G. Younes, D. Asmar, E. Shammas, and J. Zelek, “Keyframe-based monocular slam: design, survey, and future directions,” *Robotics and Autonomous Systems*, vol. 96, pp. 67–88, 2017.

[9] J. Engel, V. Koltun, and D. Cremers, “Direct sparse odometry,” *EEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 3, pp. 611–625, 2017.

[10] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. M. Montiel, and J. D. Tardós, “Orb-slam3: An accurate open-source library for visual, visual-inertial and multi-map slam,” *EEE transactions on robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.

[11] G. Younes, D. Khalil, J. Zelek, and D. Asmar, “H-slam: Hybrid direct-indirect visual slam,” *Robotics and Autonomous Systems*, 2024.

[12] N. Abboud, M. Sayour, I. H. Elhajj, J. Zelek, and D. Asmar, “In-line photometrically calibrated hybrid visual slam,” *arXiv preprint arXiv:2409.16810*, 2024.

[13] P. Besl and N. D. McKay, “A method for registration of 3-d shapes,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[14] A. Segal, D. Hähnel, and S. Thrun, “Generalized-icp.” in *Robotics: Science and Systems*, J. Trinkle, Y. Matsuoka, and J. A. Castellanos, Eds. The MIT Press, 2009. [Online]. Available: http://dblp.uni-trier.de/db/conf/rss/rss2009.htmlSegalHT09

[15] A. Myronenko and X. Song, “Point set registration: Coherent point drift,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 12, pp. 2262–2275, 2010.

[16] G. D. Evangelidis, D. Kounades-Bastian, R. Horaud, and E. Z. Psarakis, “A generative model for the joint registration of multiple point sets,” in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 109–122.

[17] K. S. Arun, T. S. Huang, and S. D. Blostein, “Least-squares fitting of two 3-d point sets,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-9, no. 5, pp. 698–700, 1987.

[18] B. K. P. Horn, “Closed-form solution of absolute orientation using unit quaternions,” *J. Opt. Soc. Am. A*, vol. 4, no. 4, pp. 629–642, Apr 1987. [Online]. Available: https://opg.optica.org/josaa/abstract.cfm?URI=josaa-4-4-629

[19] A. Segal, D. Hähnel, and S. Thrun, “Generalized-icp.” in *Robotics: Science and Systems*, J. Trinkle, Y. Matsuoka, and J. A. Castellanos, Eds. The MIT Press, 2009. [Online]. Available: http://dblp.uni-trier.de/db/conf/rss/rss2009.html/SegalHT09

[20] E. Koukoulis, G. Arvanitis, and K. Moustakas, “Unleashing the power of generalized iterative closest point for swift and effective point cloud registration,” in *2024 IEEE International Conference on Image Processing (ICIP)*, 2024, pp. 3403–3409.

[21] G. Kim and A. Kim, “Scan context: Egocentric spatial descriptor for place recognition within 3d point cloud map,” in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE Press, 2018, p. 4802–4809. [Online]. Available: https://doi.org/10.1109/IROS.2018.8593953

[22] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard, “A tutorial on graph-based slam,” *IEEE Intelligent Transportation Systems Magazine*, vol. 2, no. 4, pp. 31–43, 2010.

[23] M. Schütz, B. Kerbl, P. Klaus, and M. Wimmer, “Gpu-accelerated lod generation for point clouds,” 2023. [Online]. Available: https://arxiv.org/abs/2302.14801