IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received January 12, 2022, accepted March 7, 2022, date of publication March 14, 2022, date of current version March 22, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3159239

Collision Avoidance Route Planning for Autonomous Medical Devices Using Multiple Depth Cameras

MOHD MAHMEEN[1,2], RAUL DAVID DOMINGUEZ SANCHEZ[1], MICHAEL FRIEBE[3,4], (Senior Member, IEEE), MACIEJ PECH[2], AND SULTAN HAIDER[5]

[1]Innovation Think Tank, Siemens Healthcare GmbH, 95478 Kemnath, Germany
[2]Department of Radiology and Nuclear Medicine, Otto von Guericke University, 39120 Magdeburg, Germany
[3]Department of Measurement and Electronics, AGH University of Science and Technology (UST), 30-059 Kraków, Poland
[4]Faculty of Medicine, Otto von Guericke University, 39120 Magdeburg, Germany
[5]Innovation Think Tank, Siemens Healthcare GmbH, 91058 Erlangen, Germany

Corresponding author: Mohd Mahmeen (mohd.mahmeen@siemens-healthineers.com)

This work was supported by Innovation Think Tank, Siemens Healthcare GmbH, Germany.

ABSTRACT Radiography is one of the most widely used imaging techniques in the world. Since its inception, it has continued to evolve, leading to the development of intelligent and automated radiography systems that are able to perceive parts of their environment and respond accordingly. However, such systems do not provide a complete view of the examination space and are therefore unable to detect multiple objects and fully ensure the safety of patients, staff and equipment during the execution of the movement. In this paper, we present a system architecture based on ROS (Robot Operating System) to solve these challenges and integrate an autonomous X-ray device. The architecture retrieves point clouds from range sensors placed at specific locations in the examination room. By integrating different subsystems, the architecture merges the data from the different sensors to map the space. It also implements downsampling and clustering methods to identify objects and later distinguish obstacles. A subsystem generates bounding boxes based on the detected obstacles and feeds them to a motion planning framework (MoveIt!) to enable collision avoidance during motion execution. At the same time, a subsystem implements a deep neural network model (PointNet) to classify the detected obstacles. Finally, the developed system architecture provided promising results after being deployed in a Gazebo simulated examination space and on a use case test platform.

INDEX TERMS 3D depth camera, motion planning, obstacle detection, object recognition radiography systems, robot operating system.

I. INTRODUCTION

Diagnostic imaging devices play a central role in modern healthcare. Almost all patient pathways rely on an effective and efficient radiography system to improve patient experience and outcomes [1]. Moreover, the integration of robotic assistance in medicine has revolutionised the healthcare sector and enabled the transition from human-controlled to fully autonomous medical imaging systems [2]. The increasing number of autonomous X-ray systems and the rising demand for X-ray examinations have increased the workload of medical staff. In addition, we are currently dealing with a worldwide shortage of radiographers, specialists and diagnostic radiologists [1], [3]–[5]. Both situations previously described have led to high levels of stress and burnout among diagnostic imaging staff [6]. Technologies are needed to address these problems in healthcare.

Automation is not a trend followed only by imaging devices. According to current trends and challenges in healthcare, medical technology is moving towards automation due to the need to simplify overly complex services, procedures, devices, and equipment, and to prioritize safety in healthcare. Moreover, the recent availability of affordable and robust range sensors has led to a great leap in the implementation of this technology in autonomous devices [7], [8].

The associate editor coordinating the review of this manuscript and approving it for publication was Liang Hu.

Collisions are a major problem in shared human-robot workspaces [9], such as X-ray examination rooms. These rooms consist of static and moving elements (medical equipment, patients and medical staff). In such environments, the multidirectional movements of an X-ray machine pose the risk of collisions. In particular, safety is compromised when the device collides with a patient, as this can result in damage to the X-ray device, injury to the patient, and interruption of the workflow. The responsibility for general safety in X-ray examination rooms lies with the medical staff. Consequently, this additional workload (when there is a shortage of radiographers and radiographer assistants) results in a suboptimal clinical workflow that increases treatment times and costs [10].

An example of the previously described points is the GU60A fully automated X-ray U-arm developed by Samsung, which uses multiple sensors for automatic positioning and collision avoidance.) [11]. Other examples include camera-based collision avoidance systems for C-arm X-ray machines [12], [13] or patents proposing collision avoidance systems (e.g., floor-mounted) for radiography using soft limits, transducers, and proximity sensors [14], [15].

On the other hand, despite the research and implementation of collision avoidance in X-ray systems, the imaging device is not provided with the full extent of the examination space. In other words, the range and field of view of the sensors implemented in a device limit the possibilities of collision avoidance. In addition, as mentioned earlier, several groups have already begun to develop and implement automations for U-arm, C-arm, and floor-mounted radiography systems. Some of these approaches are seeking applications in various medical devices (e.g., radiation therapy, radiography, computed tomography) [16]. However, the development, testing, and implementation of a collision avoidance system for ceiling-mounted radiography equipment remains largely unexplored.

Moreover, the implementation of object recognition in autonomous radiography systems is still unexplored. In contrast, other fields have already taken steps to integrate object recognition into autonomous systems, such as implementing object recognition and obstacle avoidance for intelligent factory automation [17].

The integration of object detection and recognition into X-ray imaging devices allows the classification of the different objects in the space to react and interact accordingly when the X-ray device is near them. A use case for this technology in X-ray equipment would be to distinguish between patients and other obstacles. In the first case, the imaging device should get as close as possible to the region of interest (ROI). In contrast, in the second case, the system would avoid approaching it.

Currently, 3D cameras are becoming increasingly available and affordable [18]. Moreover, these range sensors are gradually becoming a relevant research area, as they provide reliable and accurate depth information for localising or characterising objects in 3D space [19]. Range sensing is crucial for the development of autonomous systems, as it provides the necessary primary range input to avoid collisions with obstacles [7]. Moreover, the output provided by this type of sensors can also be used for object recognition [20]–[22].

Based on the current trends in healthcare, the state of the art, the challenges and the pain points mentioned earlier. We envisioned the development of a system architecture for integrating fully autonomous ceiling-mounted X-ray devices that incorporates obstacle detection, object recognition, and path planning using three-dimensional data retrieved from depth sensors. The key contributions of our work are as follows:

- We have developed an obstacle detection, object recognition, and motion planning system for ceiling-mounted radiography devices to automate radiography procedures and reduce the workload of diagnostic imaging personnel.
- We implemented a sensor fusion based on an arrangement of depth sensors to provide an architecture with a close-to-full-view perception of the examination room
- We integrated a simulation and use case testing platform for the examination room to evaluate the performance of the architecture.

As mentioned above the system consisted of four subarchitectures, each responsible for a specific task: Merging sensors data, obstacle detection, object recognition and route planning. The result was a system capable of recognising objects in an environment based on multisensory data and understanding which objects they are, paving the way for the development of ceiling-mounted X-ray machines with a higher level of cognition.

The main framework for developing the architecture was the Robot Operating System (ROS) [23], as it provided us with a robust, multilingual, and open-source platform for developing complex robotic systems [24]. Furthermore, the integration of libraries and frameworks such as the Point Cloud Library (PCL) [25], the Open Motion Planning Library (OMPL) [26], TensorFlow (T.F.) ([27] and MoveIt! [28] were crucial for the elaboration of the subsystems. To test the integrated architecture and verify its functionality, we also used a simulation in Gazebo Simulator [29] and a technology use- case testing platform.

Our approach differs from existing approaches in 3 respects. First, as mentioned earlier, most of the work currently lies with the physician, who must either move the X-ray machine manually or ensure that there are no obstacles in the room in order to take advantage of automatic movement. Our approach, on the other hand, introduces perception into the radiography system to ensure that a target position can be reached automatically and without collisions, reducing the physician's workload. Second, we have shown some approaches that integrate collision avoidance into X-ray devices. However, previous works do not propose and execute alternative routes to reach a target position when an obstacle is encountered. Therefore, medical personnel must either remove the obstacle from the route or manually move the

X-ray device to the target position. In contrast, our architecture periodically updates the map of the room to detect when new objects have appeared or disappeared, and later identifies, executes, and updates routes to reach a target position. Third, our architecture provides a nearly complete view of the examination room, allowing path planning and obstacle detection to reach almost any location in the examination room. In contrast, recent works that have implemented obstacle detection usually provide only a limited view of the examination space.

To solve the discussed challenge, we followed the Innovation Think Tank (ITT) methodology developed and first applied by Haider [30] at Siemens Healthineers in 2005. This methodology is suitable and supports innovation activities for product development considering the interdependencies of stakeholders throughout the product lifecycle.

The ITT methodology addresses acquiring project mandate, pain point and workflow analysis, stakeholder engagement, creation of decision proposition. To visualize and validate the technological solution in a simulated customer environment, we used the ITT Use Case Testing Platform (ITT-UCTP) for 1) use case creation including the customer environment and medical device, 2) proof of concept testing, and 3) validation with stakeholders. ITT-UCTP accelerates product development cycles and knowledge reuse.

In the following section (II), we elaborate on how we created each subsystem and the role that the previously mentioned frameworks, software tools, and libraries played in the development and testing of these systems. In addition, Section III presents the integrated architecture and evaluates the functionality of the architecture in scenarios that approximate an actual X-ray examination room.

Finally, section IV describes the conclusions, the scope of development of such a system, and the future steps and approaches needed to implement this technology in ceiling-mounted X-ray device.

II. MATERIALS AND METHODS

For the development of a system architecture four main components had to be considered leading to an automatic ceiling-mounted X-ray machine based on multisensory data and capable of understanding and responding to its environment:

A. Merging depth sensor data (Point clouds)
B. Detecting obstacles
C. Recognising objects
D. Motion planning

Based on these points, we divided the system into subsystems, each of which could perform one of these tasks. The subsystems consisted of a network of ROS nodes that received a specific input and provided a valuable output to the following subsystem(s). This section provides a detailed description of each subsystem.

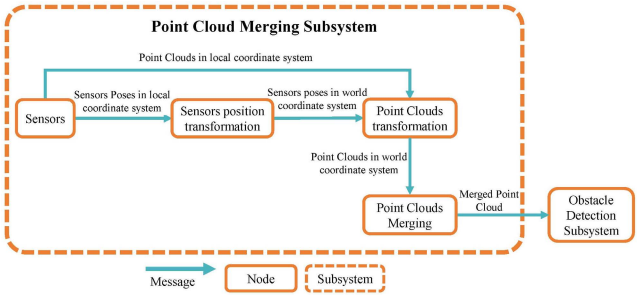

FIGURE 1. Architectural diagram of the point cloud merging subsystem. This subsystem continuously retrieves point clouds and local poses from four sensors to later position these point clouds in a world coordinate system and merge the four-point clouds into a single cloud. The merged point clouds are then retrieved by the obstacle detection subsystem.

A. POINT CLOUD MERGING SUBSYSTEM

As mentioned earlier, the proposed system would use multisensory information as its primary input. More specifically, it would use various point clouds (Figure 1). These provide three-dimensional data representing sampled surfaces in an environment [25] and enable the development of robust solutions for computer vision and automation tasks.

Depth sensors publish information with respect to their own coordinate system. Therefore, it is complicated to establish relationships and commonalities between data provided by different sensors is complicated without implementing a global coordinate system. Moreover, continuous individual processing of point clouds output from different sensors is computationally intensive and increases the latency of a system. Therefore, it was necessary to establish a world coordinate system in which the positions of the sensors are defined, and which allows correct alignment of the point cloud data from the different sensors. The successful alignment of the point cloud thus enabled a complete visualisation of the environment.

The ROS package ‘‘TF’’ allowed the creation of the global coordinate system and the definition of the position of the coordinate systems of the sensors in the global coordinate system by implementing position transformation nodes. These nodes require the parent frame (global coordinate system), child frames (coordinate system of depth sensors), and position and orientation of child frames in the global frame to perform the transformation.

To reduce the required computational power and latency of the proposed system, we merged the different point clouds, which resulted in processing only one point cloud containing the information from the different sensors.

Although the implementation of the previously described nodes aligned the different point clouds by positioning the sensor frames relative to a common parent frame, the point clouds were still independent of each other. It is important to note that each depth sensor frame defines the coordinates of the corresponding point cloud. Therefore, it was necessary to integrate nodes that convert the point clouds to the global frame.

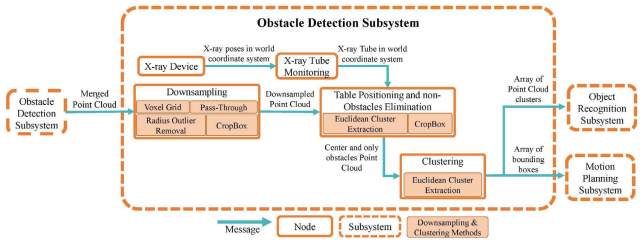

FIGURE 2. Architecture diagram of the obstacle detection subsystem. The subsystem receives point clouds from the previous subsystem and the position of the X-ray tube from the X-ray machine. Then, the point clouds are downsampled, the position of the examination table in the cloud is identified, and the cloud is repositioned to place the table at the origin of the world coordinate system. Then, the subsystem uses the position of the table and the position of the X-ray tube to remove objects that are not obstacles from the cloud. Then, obstacle clusters in the form of bounding boxes and obstacle point clouds are extracted from the obstacle clouds. Finally, the object recognition subsystem and the motion planning subsystem retrieve these clusters.

We then developed ROS nodes to subscribe to the point cloud and listen to the ''TF'' transformations between the parent and child frames. We also applied these position transformations to the respective point clouds and then published the transformed point clouds.

Once we created nodes that transformed the point clouds into the global frame, it was possible to implement a point cloud merging algorithm using the PCL function to concatenate point clouds. This node was subscribed to the translated point clouds and used the previously mentioned function to concatenate them into a point cloud required by the following subsystems of the architecture.

The integration of the different nodes mentioned before resulted in the first subsystem architecture (Fig. 1). This first subsystem received depth information as input from the 3D sensors and provided a composite point cloud as output, which contained the aligned data from each of the implemented depth sensors in the system.

B. OBSTACLE DETECTION SUBSYSTEM

Point clouds containing data from multiple sensors contain redundant data, noise, irrelevant points, and are computationally expensive to process Figure 2. Therefore, in order to achieve low latency, correct object identification in the point cloud, and a reduction in the cost of processing concatenated point clouds, it was necessary to implement the filters for the retrieved point clouds from the previous subsystem. More specifically, the goal of the filters was to eliminate noise, reduce the number of points in the cloud, and delete redundant information to leave only the relevant points in the cloud. According to [31], the most suitable filters to accomplish most of the previously mentioned tasks were the Voxel Grid filter, the Pass-Through filter, and the Radius Outlier Removal filter.

To implement the filters described above, we created a ROS node. The node was subscribed to the output of the previous subsystem and applied the Voxel Grid filter to the cloud using a class from the PCL. This filter created a 3D grid over the retrieved cloud to later downsample the resulting voxels based on their centroids. Subsequently, to remove irrelevant points from the cloud, i.e., points from the walls and floor, we used the Pass-Through filter to eliminate the points outside certain boundary conditions, such as an interval in one of the coordinate axes. We implemented this filter three times using a class from PCL to set constraints in the three coordinate axes and eliminate points that fall outside a range of interest. Finally, we used the Radius Outlier Removal class from the Point Cloud Library to remove noise in the cloud. This filter works by specifying multiple adjacent points that a given point must have within a predefined radius.

The result was a point cloud that contained only the objects of the room. It was important to transform the cloud to set the patient table to the origin of the global coordinate system, since X-ray machines usually consider the position of this object as the origin. The aforementioned implementation was also necessary to avoid performing position transformations between the coordinate system of the system architecture and the coordinate system of the X-ray machine. To address this issue, we created a new node. This node implemented Euclidean Cluster Extraction, which allows the identification of clusters in a point cloud. After extracting the clusters from the down-sampled point cloud, we compared them to the known dimensions of the table. If a cluster had similar dimensions, we identified its coordinates and used them to center the point cloud using a position transformation.

The result of the previous implementation was a mapped point cloud that contained all objects located in the examination room, including the patient table and the X-ray machine (since these are the most important components in the examination room). However, the X-ray system knows the position and dimensions of the patient table to avoid collisions with it, since it is part of the X-ray system. On the other hand, the X-ray machine is not an obstacle, but a machine that must avoid collisions with obstacles in a room. Therefore, the elimination of the points in the cloud that represented these objects was necessary.

The Crop Box filter defines a box (dimension, position, and orientation) and then uses that box to filter the points included in a defined point cloud. Although it is very similar to the Pass-Through filter, the latter is better suited for real-time data processing than the Crop Box filter. Because it is so easy to define only one Crop Box filter to eliminate an object, and because the previous filters would have already reduced the point cloud information, implementing this filter would not have any relevant impact on the performance of the system. Therefore, we decided to implement the Crop Box filter to eliminate the X-ray machine and the patient table. In other words: After correctly positioning the point cloud, the subsystem eliminated the points that contained the table, based on its known dimensions and position (origin of the coordinate system).

Only the telescope mechanism, the X-ray tube, and the collimator of the X-ray device appeared in the point cloud. By monitoring certain joints of the X-ray device (e.g., the X-ray tube joint), it was possible to eliminate the X-ray device because the information obtained from this joint was the position and orientation of the X-ray tube. This information was useful in creating a box whose height was equal to the extended height of the telescope mechanism and whose width and length were equal to the diameter of the most protruding tube of the telescope mechanism. We then positioned the bottom of the box at the determined coordinates of the X-ray tube joint, which meant that whenever these coordinates changed, the position of the box also changed. The crop box filter was then applied with the box previously described, eliminating the telescope mechanism.

Elimination of the remaining components of the X-ray device followed a similar procedure. Based on the dimensions of the X-ray tube and the attached collimator, a new box was created to enclose both. The position of this box depended on the coordinates of the monitored joints and was rotated based on the angles retrieved from one of these joints (X-ray tube joint). The last addition was necessary because the X-ray tube can rotate. Once the box was defined, the Crop Box filter used this box to eliminate the X-ray tube and collimator. Also, it is important to mention that the results of implementing such downsampling and clustering methods can be seen in Figure 7a to Figure 7c and in Figure 9a to Figure 9d.

Eliminating the table and the X-ray equipment resulted in a point cloud that contained only the obstacles in the room. These obstacles were needed for both the object recognition subsystem and the motion planning subsystem. Therefore, we created a ROS node to publish the information from the filtered and cantered point cloud in the format needed by each of the following subsystems. This node subscribed to the topic that published the point cloud, which contained only the obstacles, and then implemented Euclidean Cluster Extraction. Once the clusters were present, the node analysed each of them to retrieve their dimensions and position and generate green bounding boxes, which were simplified representations of the obstacles to facilitate motion planning and collision avoidance tasks. These bounding boxes were stored in an array, published, and then retrieved by the motion planning subsystem. On the other hand, we needed to send the points that make up each obstacle as individual obstacle point clouds to the object recognition subsystem to classify the obstacles. To accomplish this, we created a custom ROS message, which we defined as an array that would contain multiple point clouds. More specifically, these obstacles point clouds in the array were the previously retrieved clusters.

C. OBJECT RECOGNITION SUBSYSTEM

3D object recognition is a young field, and even today it is challenging to recognise 3D objects based on their complex shapes. According to [32], the most commonly used methods

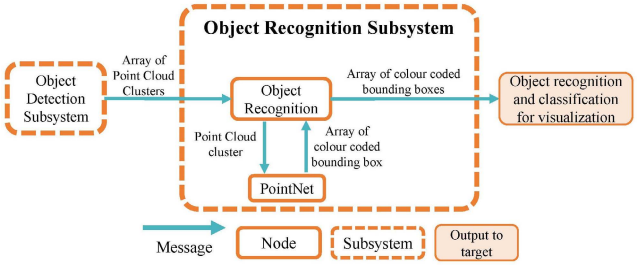

FIGURE 3. Architectural diagram of the object recognition subsystem. The subsystem receives a set of point cloud clusters, which are then sent individually to PointNet to recognize the obstacle. Based on the cluster and the detected class, a color-coded bounding box is created. The clusters are attached to an array of bounding boxes and later published for visualization purposes.

for 3D object classification include neural networks and Deep Learning. Moreover, it is foreseen that in the coming years, the use of Deep Learning-based models for 3D object recognition will increase significantly.

3D object classification using Deep Learning is divided into three main categories:

A. Multi-view-based methods
B. Volumetric-based methods
C. Point-based methods

The first uses multiple images from different views of an object to learn how to recognise it from different angles. The second converts the detected 3D objects into meshes or voxels to create a simpler view of the object that can be used to classify a model more easily. Finally, the last method uses the raw data from point clouds and passes it to an object classification model Figure 3.

Even though the previous methods correctly classify 3D objects, the first two approaches drastically simplify the information about 3D objects. For example, to implement the second method, we would have had to apply voxelization to the already downsampled obstacle point clouds, which would have resulted in a critical loss of data. However, the third method provides an approach that uses raw data from sensors (point clouds) to detect objects. Therefore, we implemented the third method to develop an accurate object recognition subsystem that can work directly with raw data from sensors.

PointNet [21] is a Deep Learning architecture for 3D object classification and segmentation that uses point clouds and outputs scores for all possible classes, where the index with the highest score is the predicted class. The PointNet structure includes a classification network and a segmentation network. More specifically, the architecture works by first taking a certain number of points as input to the classification network. Then, the network applies input and feature transformations and then aggregates the point features by max-pooling. The output is classification results for a previously defined number of classes. Moreover, the segmentation network is an extension of the classification network that combines global and local features to later output point values later.

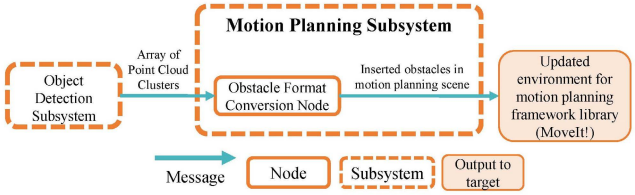

FIGURE 4. Architecture diagram of the motion planning subsystem. The subsystem receives the array of bounding boxes from the object detection subsystem, reformats the bounding boxes into the format specified by the motion planning framework, and inserts the obstacles into the motion planning scene. The scene is later used by the motion planning framework to plan and execute routes.

This architecture can provide equivalent or better results than modern neural networks based on meshes, images, or volumes. In addition, PointNet laid the foundation for point-based Deep Learning methods. Today, several architectures are based on it, which means that it is still a reliable benchmark for 3D shape classification tasks. For this reason, PointNet has been defined as the architecture for classifying obstacles in the examination room. We implemented the architecture using TensorFlow and trained it with a customised version of the ModelNet40 dataset [33], which contained only objects commonly found in an X-ray examination room. The training result was a neural network with an accuracy of 92.02 and an average class accuracy of 92.94. Once we had a trained model, we created a script to initialise the model and run it after obtaining a point cloud.

We also developed a bridge node to connect PointNet to the ROS architecture. This ROS node subscribed to the array of obstacle point clouds and initialised the PointNet model. The node then converted the obstacle point clouds to NumPy arrays, normalised them, reordered the x, y, and z columns, and checked the number of points of each obstacle to provide the input expected from the PointNet implementation. We needed to check the number of points that make up an obstacle because the PointNet implementation had a defined number of input points (1024). The previous statement means that if the objects had less than 1024 points, classification for these obstacles was not possible. On the other hand, if the obstacles had more points than required, the node applied random point deletion until the point clouds of these obstacles reached the required number of points.

Once we had correctly formatted point clouds, we called the PointNet implementation function and passed it the previously mentioned point clouds (one at a time). The function then returned the predicted classification for that point cloud. Once there was an object prediction, a box was generated whose colour depended on the returned classification. We also defined the dimensions and position of the box based on the classified obstacle. The boxes generated after recognising an obstacle point cloud were inserted into an array. After analysing all obstacles from a received array that contained the obstacle point clouds, the new array was published to adjust the behaviour of the X-ray system based on the objects in a room in the future work.

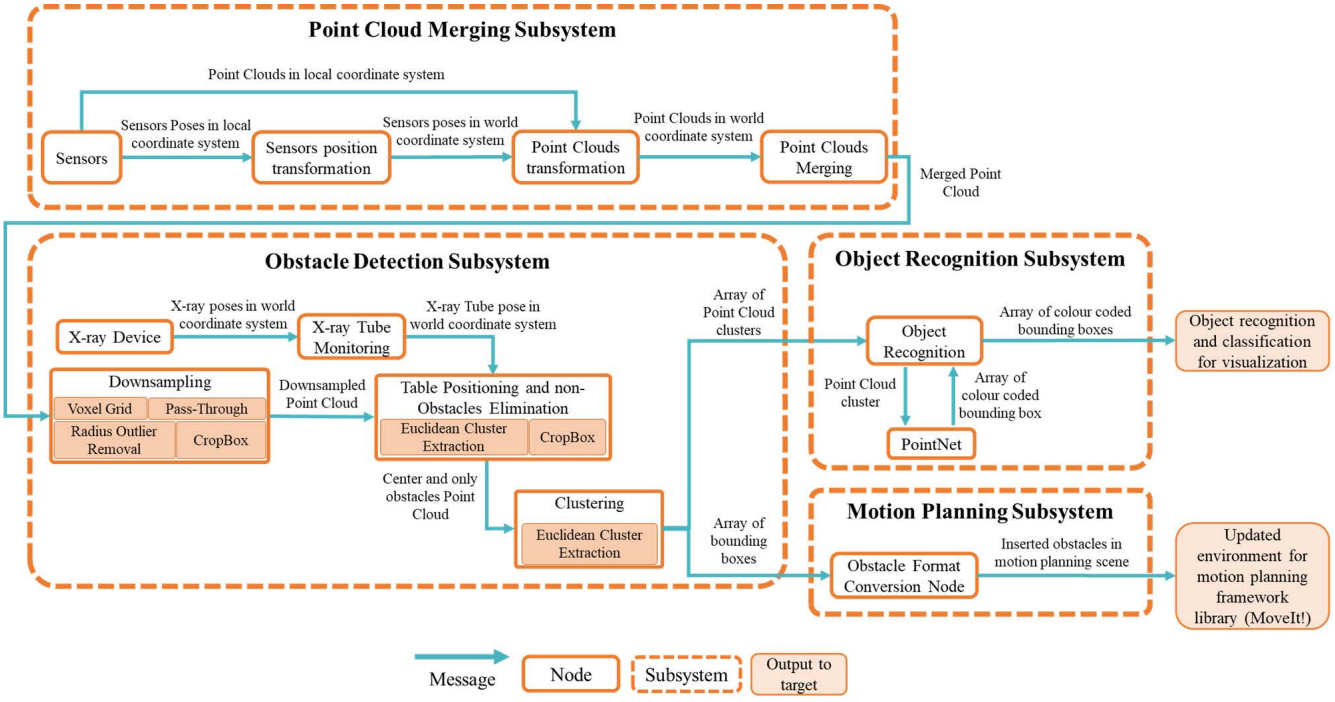

FIGURE 5. Architectural diagram of the integrated system architecture of the radiography equipment automation framework based on the previously described subsystems.

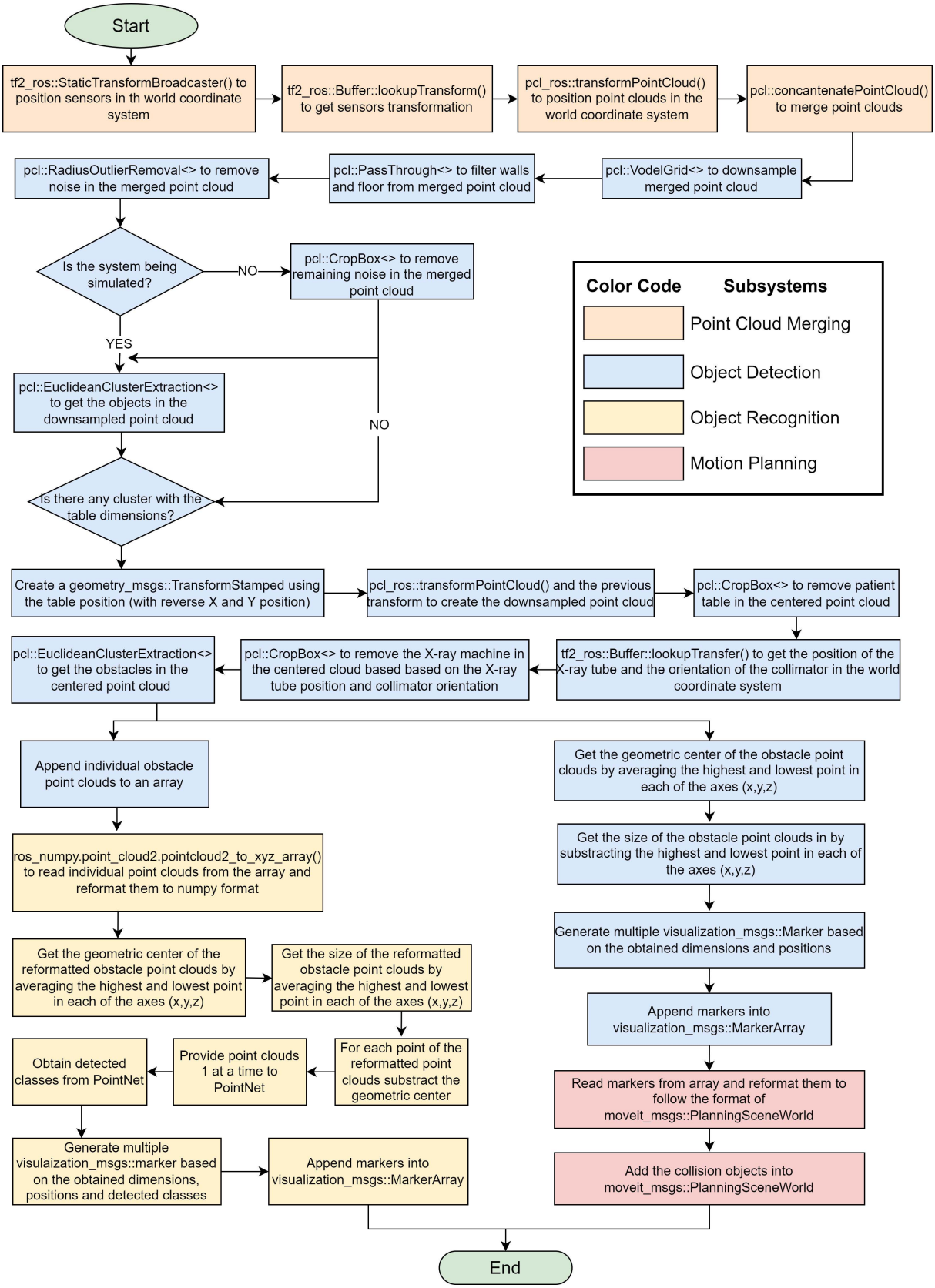

FIGURE 6. **Algorithmic flowchart of the integrated system architecture for collision avoidance automation framework in radiography equipment.**

The node created generated boxes only for obstacles with more than 1024 points, i.e., there was a possibility that an array of boxes did not contain all obstacles in the room. Therefore, we did not define the output of this subsystem as input to the motion planning subsystem.

D. MOTION PLANNING SUBSYSTEM

To implement motion planning in the architecture, we used MoveIt! because it is highly compatible with the framework used to develop the architecture (ROS) and integrates setup and visualisation tools that facilitated the implementation and subsequent testing of motion planning in ceiling-mounted X-ray machines Figure 4.

MoveIt! can plan motion from an initial position to a target position based on the information it has about the scene. Therefore, the scene needs to be constantly updated with obstacle information to avoid collisions during motion execution. It is important to mention that after initialising the motion planning framework, a topic is created to update the scene, which expects messages in a format specified by MoveIt!. Based on this information, we created a ROS node that subscribes to the array of boxes published by the obstacle detection subsystem and converts the boxes to the format specified by MoveIt! for defining obstacles. After converting the boxes to the correct message format, they were published to the previously mentioned topic to successfully update the scene based on the processed information from the sensors.

III. RESULTS

The creation of the various subsystems resulted in the system architecture shown in Figure 5.

After creating the architecture and with the goal of evaluating its functionality, we created two test platforms. The first was a simulation created with Gazebo. This simulation included the X-ray machine, the examination room, four simulated depth sensors (positioned in the upper corners of the room), and additional components found in such a room. The second test platform, on the other hand, was a scaled version of an examination room. The use case test platform consisted of an aluminium frame and acrylic panels to delineate the room, a gantry and telescope mechanism to represent the X-ray machine, four range sensors attached to the top corners of the model, and various 3D-printed components to characterise different components of an examination room.

In this section, we describe the results of implementing and testing the architecture on both test platforms. We also present the results of training the PointNet implementation.

A. ALGORITHM USED IN THE SYSTEM

Previously we have described the architecture in detail and as comprehensively as possible. By dividing the created architecture into sub-architectures and using diagrams to describe them, it was possible to explain in detail the workflows of the sub-architectures and illustrate how they are connected and what information they communicate with each other. However, in order to describe the architecture in more detail without splitting it into sub-architectures, and to show in more detail how we used specific packages and their respective functions, we created the flowchart (Figure 6), which describes the architecture in summary form while highlighting the main functions that make this architecture work.

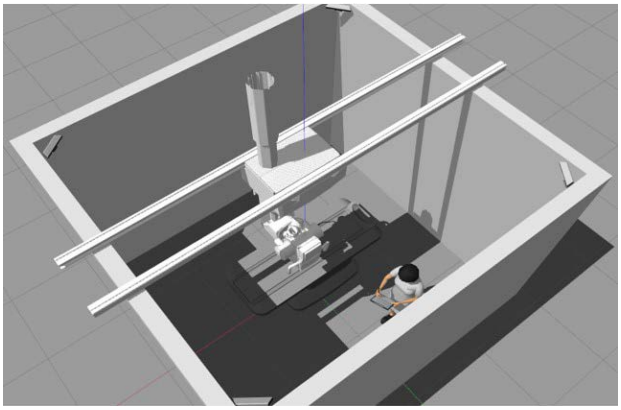

FIGURE 7. Gazebo simulation of radiography examination room.

It is important to mention that this architecture contains a large number of scripts written using Python and C++. The reason for using two programming languages to create the architecture is that some functions were faster and easier to generate and compute. This increases the complexity of the architecture. However, this is where ROS comes into play, by establishing common message types that allow communication between scripts. Also, it is important to note that tasks such as data type conversions have been omitted from the flowchart as they are implicit to working with certain functions. In summary, the flowchart is very similar to FIGURE 5. Nevertheless, it provides important insights that would not be as readily apparent in another type of diagram, but flowchart completes the concept view for reproducibility of the system. In flowchart we also have used different colour code to differentiate the algorithm used in different subsystems.

B. SYSTEM ARCHITECTURE IMPLEMENTATION IN GAZEBO SIMULATION

Gazebo Simulator allows the simulation of various sensors, such as depth sensors, and implements plugins to connect them to ROS.

Gazebo establishes this connection by creating ROS topics through which the simulated sensors publish their readings. Therefore, to connect the architecture to the sensors, we mapped their topics to the topics expected by the architecture. With the initialization of the simulation (Figure 7), the ROS architecture, and the MoveIt! framework, it was possible to visualise the different stages of the system architecture using RVIZ.

As can be seen in Figure 8a, the first subsystem correctly aligned the various point clouds and output the successfully

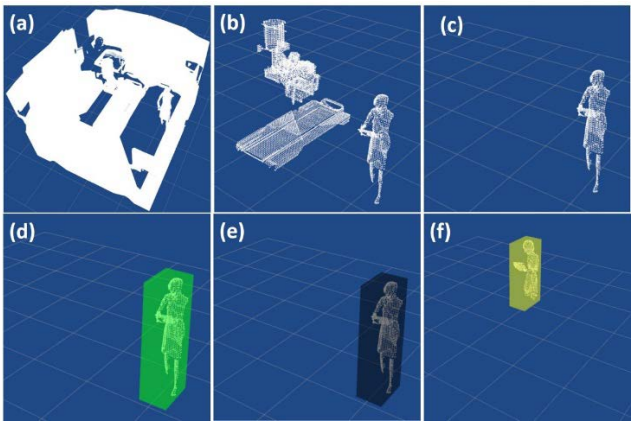

FIGURE 8. Architectural implementation results in Gazebo: (a) alignment and merging of point clouds; (b) downsampled point cloud; (c) elimination of patient table and X-ray device; (d) obstacle detection; (e) correct object detection of a person as black bounding box represents detection of a person; (f) incorrect object detection of a person as yellow bounding box represents detection of a TV Stand.

merged sensor point clouds, which simultaneously provided a nearly complete view of the room in a single point cloud.

After merging the point clouds, the obstacle detection subsystem was able to efficiently downsample the scene, leaving only the objects in the space (Figure 8b). In addition, this subsystem also eliminated the points representing the patient table and the X-ray machine based on their known positions (Figure 8c). Finally, the boxes output by the system architecture correctly enclosed the detected obstacle, as shown in Figure 8d. In addition, the subsystem published the array of the obstacle point clouds. However, since this was a custom message type, it was not possible to visualise this message type in RVIZ.

The object recognition subsystem received the array of point clouds, fed them with the PointNet implementation, and then published bounding boxes whose colour depended on the neural network's predictions. From the output boxes, it could be seen that the system could provide accurate predictions when the point cloud of an obstacle provided a nearly complete view of the real obstacle (Figure 8e). However, when the point cloud only partially described an object/person, the probability of an incorrect prediction increased significantly (Figure 8f).

Finally, the motion planning subsystem continuously retrieved the boxes output by the obstacle detection subsystem and converted them to the format required by the motion planning framework. Using the Motion Planning Visualization GUI of MoveIt! which is an RVIZ plugin, it was possible to define the target position for the X-ray devices. If the target position represented a collision with an obstacle, the colour of the X-ray device turned red, which meant that the device could not move to that position and that the subsystem had correctly received the position of obstacles. On the other hand, the system was able to plan

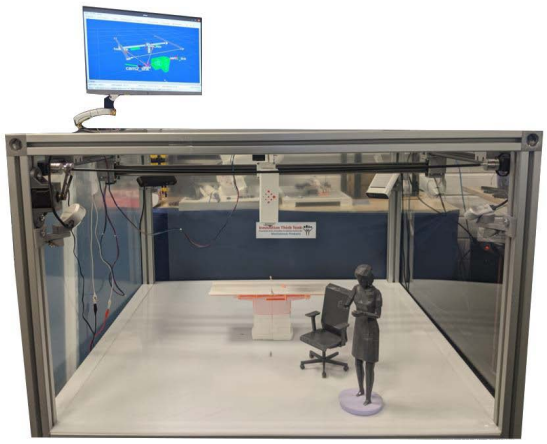

FIGURE 9. Radiography room Innovation Think Tank Use-Case Testing Platform (ITT-UCTP).

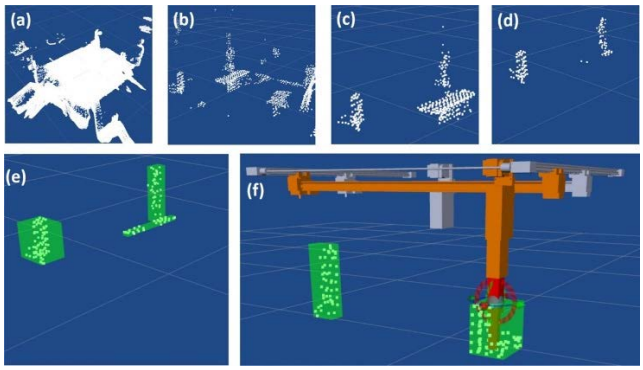

FIGURE 10. Architecture implementation results in the Use Case Testing Platform: (a) point cloud alignment and merging; (b) downsampled point cloud without crop box filter; (c) downsampled point cloud with crop box filter; (d) patient table and X-ray machine elimination; (e) obstacle detection; (f) motion planning.

movements to target positions located in an obstacle-free space without colliding with obstacles on the way to the target position.

When executing motions based on the motion plan created by MoveIt! it was found that the architecture could not run at high speed. The main reason for this was that the map update speed was slower than the movement speed, which meant that the device could not swerve when the obstacles near the system changed position or new obstacles appeared near the system. Since the elimination of the X-ray device depended on the joint states of this device, at high speed the filters of Crop Box would not correctly eliminate the points that made up the device. To solve these problems, we decided to limit the speed of the movement. This speed limit allowed the system architecture to be faster than the motion of the x-ray device.

Given the successful implementation of the architecture in Gazebo, the next step was to evaluate its functionality in the real world. Therefore, we tested the architecture in a use-case platform of the X-ray examination room.

C. SYSTEM ARCHITECTURE IMPLEMENTATION IN RADIOGRAPHY ROOM USE-CASE TESTING PLATFORM

Similar to the simulation, the first step was to execute the architecture and initialise the depth sensors, gantry, and telescope mechanism in the test platform for the use case (Figure 9). It is important to note that we had to rebuild the topics output by the depth sensors to the topics expected by the system architecture. Once this was done, we were able to monitor the different subsystems of the architecture.

In the case of the first subsystem, we could see that these clouds were more irregular than those from the simulation, but we were able to successfully align and merge them (Figure 10a).

The effectively merged multisensory data were then retrieved by the following subsystem (obstacle detection subsystem). As we described earlier, the first step performed by the obstacle detection subsystem was downsampling. However, the results obtained by downsampling were not as effective as those obtained in the simulation (Figure 10b). The irregularities of the point clouds retrieved from the real sensors were the main reason for the ineffectiveness. Therefore, we applied another filter (Crop Box filter) that resulted in a clear and correctly down-sampled output (Figure 10c).

Based on the known positions of the X-ray machine and the patient table, these objects were eliminated from the point clouds, resulting in a pure obstacle point cloud (Figure 10d). Furthermore, the obstacle detection subsystem was able to correctly generate and publish bounding boxes for each obstacle in the room (Figure 10e).

In addition to the bounding boxes, the obstacle detection subsystem continuously published arrays of obstacle point clouds. Then, the object recognition subsystem retrieved the obstacle point clouds. However, the object recognition subsystem was not effective because in the use-case platform, all detected obstacles contained less than 1024 points. Therefore, the neural network was not able to predict a class for these point clouds.

On the other hand, the motion planning subsystem effectively converted the format of the bounding boxes to the format expected by ''MoveIt!''. We verified this successful conversion by using the Motion Planning Visualization GUI and setting the target position to be a position where an object was located, which gave the expected results (Figure 10f). If the target position was an obstacle-free position, MoveIt! was able to plan a movement to that position.

Finally, in order to execute a movement, we needed to establish a link between the X-ray device representation (gantry and telescope mechanism) and ROS. Therefore, we created scripts for the movement of the mechanism based on the joint states given by MoveIt! during the movement execution. We defined this motion based on the joint states to maintain the same position as the virtual X-ray mechanism represented in RVIZ. The result was successful collision avoidance during motion execution. In addition, since the architecture was implemented in a scaled representation of the X-ray space, no velocity issues were encountered as in the simulation.

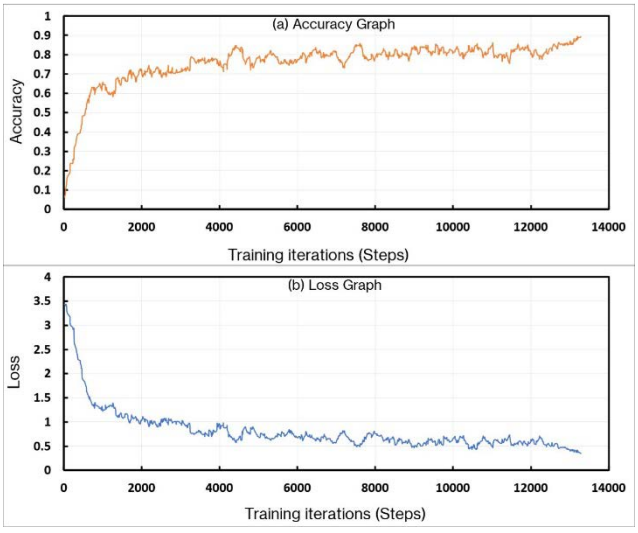

FIGURE 11. Training plot of the PointNet implementation smoothed with an exponential moving average with a smoothing factor of 0.9. It includes the (a) accuracy (orange) and (b) loss (blue) achieved by the model.

D. POINTNET IMPLEMENTATION TRAINING

As mentioned in the previous sections, PointNet was defined as a model for classifying the objects in the examination room. More specifically, we used the implementation presented in [34] as it provides a straightforward alternative to integrate PointNet into TensorFlow 2, the version of TensorFlow used in this work. In addition, as described earlier, we adapted ModelNet40, originally comprising 12,311 CAD models from the 40 most common object categories in the world and reduced its content to 15 classes representing objects found in an X-ray examination room. These classes were bed, bottle, chair, curtain, desk, door, lamp, laptop, monitor, nightstand, person, plant, stool, table, and television stand.

Once the model and dataset were ready, we trained the model as shown in Figure 11(a) & (b). We also used the Adam optimizer for training with an initial learning rate of 0.001, batch size of 16, a decay step of 200,000, and a decay rate of 0.7.

Training on the custom ModelNet40 with a GeForce GTX 1080 Ti GPU ran until accuracy was above 90%, as the main goal of this work was to prove the functionality, which did not require an extremely accurate model. Furthermore, as mentioned in the second section, the evaluation of the model resulted in an accuracy of 92.02% and an average class accuracy of 92.94%.

IV. DISCUSSION AND CONCLUSION

The aim of the present work was to create a system architecture that would enable the development of intelligent and autonomous X-ray devices. The developed system had

multiple subsystems that integrated multisensory data fusion, obstacle detection, object classification and motion planning.

To build the above system architecture, we used the ROS framework to support the processes that needed to be executed by the different subsystems. Other supporting frameworks and libraries were also required, with MoveIt! the Point Cloud Library, and TensorFlow being the most important.

The system architecture included four subsystems. The first subsystem was tasked with aligning and merging data from multiple sensors. The second subsystem processed the point cloud output from the previous subsystem to produce a point cloud that contained only the relevant information (obstacles) required for the subsequent extraction of point cloud clusters. Based on the clusters, arrays of simplified representations of the obstacles (green bounding boxes) and point clouds of the obstacles were output. The third subsystem retrieved this final array and fed these point clouds into a PointNet implementation. The network implementation later generated and published bounding boxes in specific colours (the colour depended on the recognised class) for each obstacle cloud that consisted of more than 1024 points. On the other hand, the fourth subsystem subscribed to the bounding boxes published by the second subsystem and converted these obstacles into the format required by the motion planning framework to update the map of the scene and correctly avoid obstacles during motion planning and execution.

With the integration of the system architecture, we implemented the architecture in two test platforms to verify its functionality. The first testing approach was to use a simulation created with the Gazebo Simulator.

The main results of the simulation tests were successful collision avoidance during motion planning and execution, and correct recognition of obstacles, whose point clouds provided a nearly complete view of the objects. However, when the point clouds did not provide a complete view of the objects, they were usually not classified correctly. In addition, the system architecture exhibited significant latency for most processes due to its computational complexity and linearity. Consequently, the motion could not be executed at a very high speed because the update rate of the scene map was slower than necessary. This meant that during high-speed motion execution, the system was not able to detect in time when a new obstacle had appeared or when an object had changed its position. To address these issues in the future, we plan to implement multithreading as this would be an alternative to speed up the processing of the point cloud and reduce latency. We also plan to train the neural network with a richer and more diverse dataset to classify point clouds that partially describe an object.

After successfully implementing the system architecture in a simulation, we tested the architecture in a real environment using a use case testing platform. The testing platform was a scaled representation of an X-ray examination room. Similar to the simulation, it contained the elements commonly found in this room. The most promising result of the test on the use case test platform was the correct motion planning and execution, successfully avoiding the obstacles in the scene. On the other hand, object classification was inconclusive, as each of the obstacles in the scaled test space ranged from 70 to 150 points, which meant that the PointNet implementation could not handle them (the number of points required for classification is 1024). To address the inability to classify objects with fewer than 1024 points, some future work will focus on implementing upsampling methods to increase the number of points in these point clouds.

Moreover, in the current system architecture, the point clouds from the different sensors are first manually aligned and later the position transformation is stored to avoid the need to manually align the point cloud in the future. However, accidentally moving the depth sensors would affect the alignment. Therefore, it is also important to implement a method to automatically align the point cloud. This work lies in the scope of future work which will lead the concept to the next technology readiness level (TRL) also translating the concept into mathematical model as well.

In summary, the present work has shown that the implementation of the defined system architecture in ceiling mounted radiography systems is a feasible solution. In other words, the architecture would contribute to the automation of the device through the analysis and processing of multisensory data, allow the simplification of an overly complex workflow in the radiography department, reduce the workload of MTRAs and speed up scanning operations. On the other hand, in addition to the challenges mentioned above (such as reducing latency, implementing upsampling methods, and automatic camera alignment), there are several real-world challenges in successfully implementing this architecture in the real world. First, our architecture is trained with only a certain number of classes and a limited amount of data. In this regard, the classification accuracy of the neural network will not increase with time. Therefore, the real-world implementation should be able to use all the data it can continuously collect to continuously train the network with new information. This would increase the robustness of the architecture in classification and reduce misclassification. Another challenge is to numerically demonstrate in future studies that the savings and efficiencies gained by implementing our system architecture outweigh the cost of sensors and adapting the examination room to integrate the system components. Finally, patient and physician acceptance play a key role in the real-world implementation of the architecture. If either of these groups does not consider a fully autonomous X-ray machine to be safe or useful, then the implementation of this work in practise will be highly unlikely.

REFERENCES

[1] (2016). SCoR. *The Radiography Workforce Current Challenges and Changing Needs*. The College Radiographers. [Online]. Available: https://www.sor.org/sites/default/files/document-versions/appg_a4.pdf.

[2] V. Tacher and T. de Baere, ''Robotic assistance in interventional radiology: Dream or reality?'' *Eur. Radiol.*, vol. 30, no. 2, pp. 925–926, Feb. 2020, doi: 10.1007/s00330-019-06541-w.

[3] O. Bwanga, "Barriers to continuing professional development (CPD) in radiography: A review of literature from Africa," *Health Professions Educ.*, vol. 6, no. 4, pp. 472–480, Dec. 2020, doi: 10.1016/j.hpe.2020.09.002.
[4] B. T. Andersson, S. M. Lundgren, and M. Lundén, "Trends that have influenced the Swedish radiography profession over the last four decades," *Radiography*, vol. 23, no. 4, pp. 292–297, Nov. 2017, doi: 10.1016/j.radi.2017.07.012.
[5] M. Mahmeen, M. R. Melconian, S. Haider, M. Friebe, and M. Pech, "Next generation 5G mobile health network for user interfacing in radiology workflows," *IEEE Access*, vol. 9, pp. 102899–102907, 2021, doi: 10.1109/ACCESS.2021.3097303.
[6] N. V. K. Philips, "Radiology staff in focus," Tech. Rep. 4522 991 53691, 2019.
[7] R. Siegwart and I. R. Nourbakhsh, "Introduction to autonomous mobile robots," *Choice Rev. Online*, vol. 49, no. 3, p. 49, 2011, doi: 10.5860/choice.49-1492.
[8] M. Zollhöfer, "Commodity RGB-D sensors: Data acquisition," in *Advances in Computer Vision and Pattern Recognition*. Cham, Switzerland: Springer, 2019, pp. 3–13, doi: 10.1007/978-3-030-28603-3_1.
[9] A. Bicchi, M. A. Peshkin, and J. E. Colgate, "Safety for physical human-robot interaction," in *Springer Handbook of Robotics*. Springer, 2008.
[10] R. Mohan, *Tip of the Iceberg: Exploring Robot Autonomy for Interventional Healthcare*. The Netherlands: Philips Healthcare, 2020.
[11] Samsung. *Digital Radiography GU60A | Samsung Healthcare Global*. Accessed: Aug. 21, 2021. [Online]. Available: https://samsunghealthcare.com/en/products/DigitalRadiography/GU60A/Radiology/benefit
[12] K. Incetan, R. Mohan, H. Stoutjesdijk, N. Fernandes, and B. de Jager, "RGB-D camera based collision prediction and avoidance for X-ray rotational angiography," in *Proc. IEEE SENSORS*, Oct. 2019, pp. 1–4.
[13] M. Azizian, J. Sorger, L. Blohm, C. Niebler, and H. Kunze, "Collision avoidance during controlled movement of image capturing device and manipulatable device movable arms," U.S Patent 9 259 282, Feb. 16, 2016.
[14] C. A. Hinton and G. Szejna, "Radiographic gantry with software collision avoidance," U.S. Patent 5485 502, Jan. 16, 1996.
[15] G. Shanmugavel and H. Krishnaswami, "Proximity detector and radiography system," U.S. Patent 6 985 556 B2, Jan. 10, 2006.
[16] H. Kikuchi and H. Otani, "Collision preventive device for medical equipment," U.S. Patent 4 969 170, Nov. 6, 1990.
[17] R. C. Luo, C.-W. Kuo, and Y.-T. Chung, "Model-based 3D object recognition and fetching by a 7-DoF robot with online obstacle avoidance for factory automation," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2015, pp. 2647–2652, doi: 10.1109/ICRA.2015.7139556.
[18] S. A. Bello, S. Yu, C. Wang, J. M. Adam, and J. Li, "Review: Deep learning on 3D point clouds," *Remote Sens.*, vol. 12, no. 11, p. 1729, May 2020, doi: 10.3390/rs12111729.
[19] L. Jiao, F. Zhang, F. Liu, S. Yang, L. Li, Z. Feng, and R. Qu, "A survey of deep learning-based object detection," *IEEE Access*, vol. 7, pp. 128837–128868, 2019, doi: 10.1109/ACCESS.2019.2939201.
[20] Y. Xu, T. Fan, M. Xu, L. Zeng, and Y. Qiao, "SpiderCNN: Deep learning on point sets with parameterized convolutional filters," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 87–102, doi: 10.1007/978-3-030-01237-3_6.
[21] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 77–85, doi: 10.1109/CVPR.2017.16.
[22] D. Maturana and S. Scherer, "VoxNet: A 3D convolutional neural network for real-time object recognition," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2015, pp. 922–928, doi: 10.1109/IROS.2015.7353481.
[23] M. Quigley. *ROS: An open-source Robot Operating System*. Accessed: Nov. 12, 2009. [Online]. Available: http://stair.stanford.edu
[24] L. Vinet and A. Zhedanov, "A 'missing' family of classical orthogonal polynomials," *J. Phys. A, Math. Theor.*, vol. 44, no. 8, p. 144, 2011, doi: 10.1088/1751-8113/44/8/085201.
[25] R. B. Rusu and S. Cousins, "3D is here: Point cloud library (PCL)," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 1–4, doi: 10.1109/ICRA.2011.5980567.
[26] I. A. Şucan, M. Moll, and L. E. Kavraki, "The open motion planning library," *IEEE Robot. Autom. Mag.*, vol. 19, no. 4, pp. 72–82, Dec. 2012, doi: 10.1109/MRA.2012.2205651.
[27] M. Abadi, "BigArray?: A system for large-scale machine learning," in *Proc. 12th USENIX Symp. Oper. Syst. Design Implement. (OSDI)*, vol. 16, 2016, pp. 265–283. [Online]. Available: https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi
[28] D. Coleman, I. Sucan, S. Chitta, and N. Correll, "Reducing the barrier to entry of complex robotic software: A MoveIt! Case study," 2014, *arXiv:1404.3785*.
[29] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, vol. 3, Oct. 2004, pp. 2149–2154, doi: 10.1109/iros.2004.1389727.
[30] S. Haider. (2005). *Innovation Think Tank: Proactively Drive Innovation to Improve Human Life*. Accessed: Sep. 21, 2021. [Online]. Available: https://www.siemens-healthineers.com/vn/careers/innovation-think-tank
[31] C. Moreno and M. Li, "A comparative study of ?ltering methods for point clouds in real-time video streaming," in *Proc. World Congr. Eng. Comput. Sci.*, vol. 1, 2016, pp. 389–393.
[32] L. E. R. De Carvalho, "Literature review for 3D object classifcaton / recogniton," *Pattern Anal. Appl.*, vol. 22, pp. 1243–1292, Sep. 2017, doi: 10.13140/RG.2.2.12899.25127.
[33] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3D ShapeNets: A deep representation for volumetric shapes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1912–1920, doi: 10.1109/CVPR.2015.7298801.
[34] H. Zeng. *PointNet2-Keras: PointNet2 Implemented Using Keras and TensorFlow*. Accessed: Aug. 21, 2021. [Online]. Available: https://github.com/HarborZeng/pointnet2-keras

MOHD MAHMEEN was born in 1989. He received the B.S. degree in mechanical engineering from Uttar Pradesh Technical University, Greater Noida, India, in 2011, and the M.S. degree in industrial engineering and management from Gautam Buddha University, Greater Noida, in 2015. He is currently pursuing the Ph.D. degree with the Radiology and Nuclear Medicine Department, Otto von Guericke University, Magdeburg, in collaboration with Siemens Healthcare GmbH, Germany.

From 2011 to 2013, he worked as a Provisional Term Lecturer at the Greater Noida Institute of Technology, Greater Noida, where he was supporting academic activities and mentoring start-up ideas for technological feasibility. From 2015 to 2018, he was an Assistant Professor and a Research Assistant along with as a Mentor for start-up ideas for the Incubation Center, IIMT Group of Colleges. In 2018, he started working as a doctoral employee at Innovation Think Tank Mechatronic Products, Siemens Healthineers, Kemnath, Germany. His research interests include healthcare automation frameworks, healthcare of the future, healthcare mechatronics, robotics, and sensor ecosystems.

RAUL DAVID DOMINGUEZ SANCHEZ was born in Valencia, Spain, in 1998. He received the B.S. degree in mechatronics engineering from the Instituto Tecnológico y de Estudios Superiores de Monterrey and the B.S. degree in automation and mechatronics from the Hochschule Zittau/Görlitz.

From 2020 to 2021, he was a Young Innovator Fellow and a bachelor's thesis student at the Innovation Think Tank Division, Siemens Healthineers, Kemnath, Germany. His research interests include artificial intelligence for robotics and autonomous systems, collision avoidance, and computer vision.

MICHAEL FRIEBE (Senior Member, IEEE) received the B.Sc. degree in electrical engineering, the M.Sc. degree in technology management from Golden Gate University, San Francisco, and the Ph.D. degree in medical physics in Germany.

After his B.Sc. degree, he spent five years in San Francisco, as a Research and Design Engineer with an MRI and ultrasound device manufacturer. He is a German citizen with expertise in diagnostic imaging and image guided therapies, as a/an Founder/Innovator/CEO/Investor and a Scientist. He is also a Research Fellow with the Technical University of Munich, Munich; an Adjunct Professor with the Queensland University of Technology, Brisbane; and a Professor of image guided therapies with Otto von Guericke University, Magdeburg, Germany. Since 2022, he has been a Professor of biomedical engineering innovation with the AGH University of Science and Technology, Kraków, Poland. He is a listed inventor of more than 80 patents and has authored over 200 papers. He is a Board Member of four medical technology start-up companies and an investment partner of a MedTec-fund. From 2016 to 2018, he was a Distinguished Lecturer of the IEEE EMBC teaching innovation generation and MedTec entrepreneurship.

MACIEJ PECH was born in 1968. He studied medicine at the TU Berlin and received the doctorate degree from Charité—Universitätsmedizin Berlin.

The specialist in diagnostic radiology took over the management of the University Clinic for Radiology and Nuclear Medicine, Magdeburg, in 2018. The personal research focus and strategic direction of the Department of Radiology and Nuclear Medicine is focused on the development and validation of minimally invasive image-guided interventions; the conduct of clinical trials for positioning minimally invasive oncology, including the development of appropriate biomarkers; and the symbiotic development of MR diagnostics and MR therapy. He is, among other things, a Founding Member of the German Interdisciplinary Society for Vascular Anomalies e. V. (DIGGefA).

SULTAN HAIDER is the Global Head of Innovation Think Tank (ITT), Siemens Healthineers (SHS), which was established in 2005. His inspiring vision of innovation culture formed Innovation Think Tank to become a global infrastructure of 72 activity locations (Innovation Labs and Innovation Think Tank certification programs) in Germany, China, U.K., India, the USA, United Arab Emirates, Turkey, Canada, Australia, Egypt, Saudi Arabia, Portugal, Switzerland, Brazil, and South Africa. He has filed over 500 inventions and patents; and under his leadership, ITT teams have worked on over 2500 technology, strategy, and product definition projects worldwide. He is a Principal Key Expert at (SHS), a title awarded to him by the SHS Managing Board, in 2008, for his outstanding innovation track record. He has also been awarded honorary directorships and professorships, and has developed innovation infrastructures and implemented innovation management certification programs for top institutions.