Moving object detection for humanoid navigation in cluttered dynamic indoor environments using a confidence tracking approach

Prabin Kumar Rath[1], Alejandro Ramirez-Serrano[2] and Dilip Kumar Pratihar[3]

***Abstract*— Humanoid robot perception is challenging compared to perception in other robotic systems. The sensors in a humanoid are in constant state of motion and their pose estimation is affected by the constant motion of the tens of DOFs (Degrees of Freedom) which in turn affect the estimation of the sensed environmental objects. This is especially problematic in highly cluttered dynamic spaces such as indoor office environments. One of the challenges is identifying the presence of all independent moving/dynamic entities such as people walking around the robot. If available, such information would assist humanoids to build better maps and better plan their motions in unstructured confined dynamic environments. This paper presents a moving object detection pipeline based on relative motion and a novel confidence tracking approach that detects point clusters corresponding to independent moving entities around the robot. The detection does not depend on prior knowledge about the target entity. A ground plane removal tool based on voxel grid covariance is used for separating point clusters of objects within the environment. The proposed method was tested using a Velodyne VLP-16 LiDAR and an Intel-T265 IMU mounted on a gimbal-stabilized humanoid head. The experiments show promising results with a real-time computational time complexity.**

Index Terms– Relative motion, Pose transformation, Confidence tracking, Voxel grid covariance, SLAM

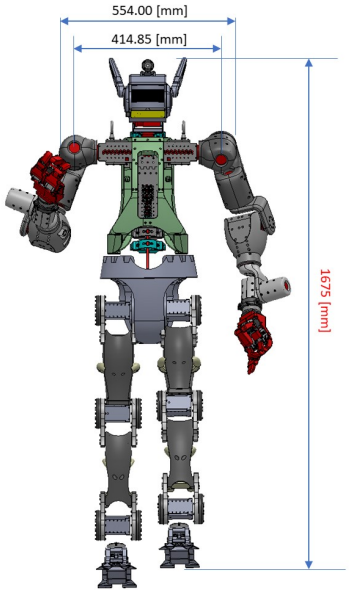


Fig. 1: Life-size Humanoid (CAD model) having 35 DOF developed in UVs Robotarium Lab, University of Calgary.

I. INTRODUCTION

Numerous research on humanoid robots have focused on developing them to be capable of assisting humans in different social environments [1]. Usually the workplace for such socially capable humanoids is dynamic and unstructured. Due to the environment's and the robot's motion the perception sensors of the robot undergo change in pose (position and orientation) during the mission execution process. Highly dynamic environments such as busy streets, shopping malls, and sporting events undergo frequent unanticipated changes due to the presence of numerous independent moving entities (e.g, people, vendors and vehicles) around the space. Humanoids (e.g, Fig. 1) navigating in such environments require frequent/fast knowledge about the spatial locations of the moving entities to enable them to be aware of the changes taking place in their surroundings. In the field of robot perception such a problem is commonly known as Detection And Tracking of Moving Objects (DATMO) [2]. One of the essential applications of DATMO is as a pre-processing step providing information for Simultaneous Localization and Mapping (SLAM) algorithms where there is a need to generate accurate maps in dynamic environments that are as good as the ones generated in a purely static environment. Numerous SLAM algorithms have been developed that are focused and bound to be used in static or quasi-static environments (e.g, [3]). In highly cluttered and dynamic spaces, however if slow or fast moving objects are not properly identified the generated maps will have a potentially large number of (undesirable) artifacts (considered static but actually being dynamic) (e.g, [4]).

In the last decade, 3D perception sensors have been successfully used for SLAM to a great extent. 3D Light Detection and Ranging (LiDAR) sensors having high data sampling frequencies in capturing the terrain topology with high precision have been used in somewhat effective DATMO algorithms. Wang et al. [5], and Vu et al. [6] among others have proposed the use of Bayesian formulations to integrate SLAM with DATMO. In contrast to an integrated model, Wolf et al. [7] used multi-occupancy grids while Mertz et al. [8] used a 2D scan matching method based on environmental features to detect moving objects. Asvadi et al. [9] used a 2.5D motion grid pointcloud representation to detect moving objects via historical sensor pose information. Dewan et al.

[1]Prabin Kumar Rath is with Department of Computer Science and Engineering, National Institute of Technology, Rourkela, Odisha, 769008, India prabinrath123@gmail.com

[2]Alejandro Ramirez-Serrano is with Faculty of Mechanical Engineering, University of Calgary, Calgary, Alberta, T2N-1N4, Canada aramirez@ucalgary.ca

[2]Dilip Kumar Pratihar is with Faculty of Mechanical Engineering, Indian Institute of Technology, Kharagpur, West-Bengal, 721302, India dkpra@mech.iitkgp.ernet.in

[10] proposed a probabilistic approach to model moving objects using the Random SAmple Consensus (RANSAC) algorithm while Azim et al. [11] used octree data structures to detect inconsistencies between consecutive pointcloud frames to detect moving objects. Takabe et al. [12] proposed a detection method based on the minimization of an energy function from photometric and depth consistency data.

Most of the works on DATMO with 3D LiDARs have focused on outdoor environments where the spatial difference between consecutively captured pointclouds is large. Although such approaches work relatively well the developed solutions present problems when used in indoor environments where higher sensitivity and robustness of the detection algorithms is required. Previous works have mainly used Kalman filters for tracking the moving objects but they have proven somewhat ineffective in indoor environments due to the inconsistent motion cues of the moving entities.

Existing algorithms have not given full consideration to cluster based approaches that could provide effective in indoor applications where objects might posses both static and dynamic aspects simultaneously (e.g, a person waving his/her arms while standing at a fixed position) a problem that has not been fully considered in past work. To be effective in such cases the detection algorithms must not only detect the arms as moving but the entire point cluster belonging to the person must be identified as the moving entity.

The work presented in this paper (Fig 2) follows the goals of DATMO but incorporates cluster based approaches to identify all individual moving objects within the sensed space. The proposed algorithm is generic enough and can be used in various autonomous systems (e.g, rovers) and humanoids performing complex manoeuvres that typically compromise the robot's estimation process (e.g, transitioning from crawling to climbing, to jumping). The proposed algorithm has the aim to be used as an intermediary process in SLAM and in the navigation of humanoids in dynamic cluttered spaces where flexible objects of various geometries move in diverse (chaotic) ways, and at varying speeds.

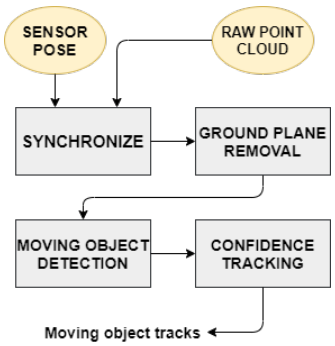


Fig. 2: Proposed moving object detection flowchart.

II. MOVING OBJECT DETECTION

The proposed work focuses on DATMO as a complementary functional block to SLAM and navigation in real-time. The real time output of the block is suitable to be used along other functions of interest such as SLAM, navigation, dynamic obstacle avoidance, etc. The proposed approach focuses to indoor environments where the collected environmental sensor data can be used as collected or cropped depending on the desired area where moving entities are to be identified around the sensor. In what follows it is assumed that the humanoid (autonomous robot) has the following basic sensors: LiDAR and IMU (inertia measuring unit) with suitable driver algorithms running on the robot. The detection of moving objects takes place in 7 steps described below.

A. STEP 1: Cloud Pose Synchronization

The pointcloud data from the LiDAR and the odometry data from the IMU are time synchronized to generate a set of pointcloud-pose pairs that is updated over time. Each pair includes a pointcloud (captured at time "t") and the corresponding sensor pose w.r.t the inertial frame at the time of capturing the pointcloud. The synchronization is achieved using the ROS (Robot Operating System) approximate time policy that ensures minimum data loss [13].

B. STEP 2: Ground Plane Removal

Ground plane points represent the data points that connect different point clusters together within the pointcloud. With the pairs from Step 1 a ground plane removal is done to isolate distinct clusters present in the pointcloud. Traditional ground plane removal methods (e.g, RANSAC [14]) do not perform well with sparse LiDAR pointclouds in indoor environments due to unavailability of enough ground plane points for plane fitting. Pointcloud density variation w.r.t distance and ring-shaped ground plane features [15] make the process even more difficult for LiDAR data. As the concerned environment of operation is indoor, it has been assumed that the ground surfaces do not have much inclinations or slopes. The proposed approach used in this paper is a modification of the method proposed by Douillard et al. [16] where the following 3-step process is used:

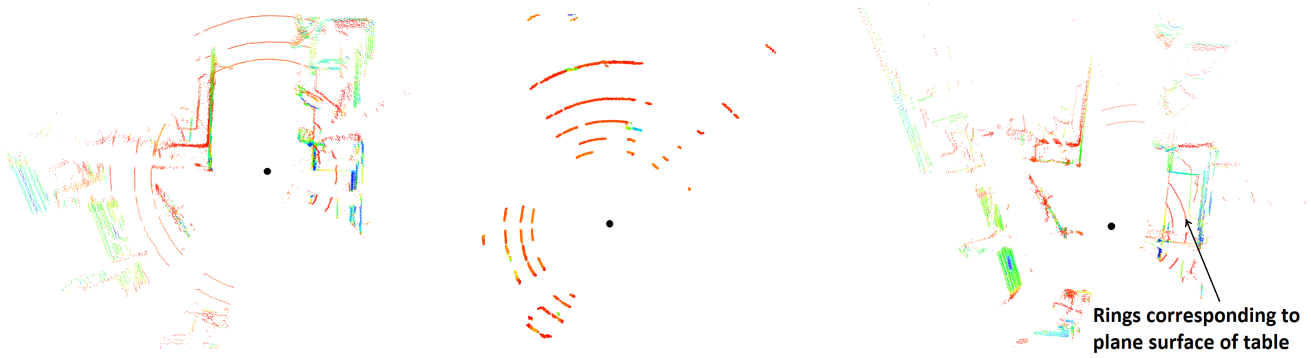
(a) The pointcloud is partitioned using a voxel grid [17] approach where the size of each voxel holding a small subset of the pointcloud depends on the density of the captured pointcloud.

(b) Assuming the ground plane's normal axis is aligned with z-axis (gimbal stabilized humanoid head) of the sensor and ground plane points must have very little variance along the z-axis of the sensor. Under such conditions the xz and yz covariance for each voxel is determined using the Eqn. 1. The voxels having a covariance less than a threshold are clustered into a group of voxels called Ground Voxel Plane (GVP).

$$cov(a;z) = \frac{\sum_{i=1}^{n} (a_i - \bar{a})(z_i - \bar{z})}{n} \qquad (1)$$

$cov(a;z) <$ where a can be either x or y

(c) Depending on the environment, the GVP obtained in Step (b) may include voxels corresponding to surfaces (e.g, table surfaces, stair landings, etc.) other than the



(a) Raw pointcloud. (b) Extracted ground plane. (c) Pointcloud after ground plane removal.

Fig. 3: Ground plane removal based on voxel grid covariance. LiDAR location shown as a black dot.

desired ground plane. To identify the ground plane, the true ground plane is assumed to be the plane with the maximum number of points. To determine the dominant plane w.r.t point count, all the points within the voxels present in the GVP are binned w.r.t their z-coordinate value. Such process divides the z-axis domain (z_{min} to z_{max}) of GVP into equal subdivisions (bins, each having size $\mathbf{G}$) and distributes the points among the bins according to their z-coordinate values (Eqns. 2 and 3). The bin having the highest number of points is considered to be the ground plane and all such points are removed from the raw pointcloud.

$$\text{No of bins } N = ceil(\frac{z_{max} \quad z_{min}}{\mathbf{G}}) \tag{2}$$

$$\text{Range of } i^{th} \text{ bin} = [z_{min} + (i \quad 1) \quad \mathbf{G}; \quad z_{min} + i \quad \mathbf{G}] \tag{3}$$
$$i = 1, 2, 3 \ldots N$$

The results of the proposed ground plane removal applied to a Velodyne VLP-16 LiDAR point cloud is shown in Fig 3 where the plane surface corresponding to a table is preserved in the pointcloud.

C. STEP 3: Euclidean Clustering

The pointcloud without the ground plane obtained in Step 2 is clustered based on the euclidean distance between individual points as described in [18]. This step generates distinct point clusters corresponding to different objects around the sensor. In what follows, a point cluster captured at time "t" is referred as PC_t and the number of points within such a cluster is denoted as N_{PC_t}.

D. STEP 4: Cluster Correspondence

A Global Nearest Neighbour (GNN) approach using a Kd-tree data structure [19] is used to match the centroids of the point clusters identified in Step 3 between consecutive frames. The correspondence assumes the following constraints.

The sensor has a high sampling rate (e.g, 10Hz).

The sensor's relative velocity w.r.t the ground is within a pre-selected range (i.e, 0m/s and 6m/s).

The associations between point clusters representing the same object captured in two consecutive pointcloud frames are stored as a correspondence map $\mathbf{M}$. Such a collection of maps is used for tracking the clusters through multiple consecutive frames. In order to prevent false positive correspondences, the matching operation is reciprocal in nature. Additionally, a volume constraint (Eqn. 4) on the bounding box (BB) of the clusters is used to ensure correct correspondence. In Eqn. 4 the terms $PC_{(t-1)}$ and $PC_{(t)}$ represent the point clusters of the same object in two consecutively captured pointcloud frames at times t-1 and t, respectively.

$$\begin{aligned} v_1 &= Volume(BB(PC_{(t \quad 1)})) \\ v_2 &= Volume(BB(PC_{(t)})) \\ \frac{jv_1 \quad v_2 j}{jv_1 + v_2 j} &< \\ &= \text{Volume difference threshold} \end{aligned} \tag{4}$$

E. STEP 5: Cloud pose transformation

For a point cluster PC_t the associated sensor pose (Step 1) is referred as P_t. The local pose of the point cluster observed from the sensor's frame is referred as P_{Lt}. The change in global pose of the cluster P_{Gt} as observed from the inertial frame is used to indicate whether the cluster is moving or static. The global pose of a cluster is thus represented in terms of the local pose and the sensor pose as per Eqn. 5.

$$P_{Gt} = P_t \quad P_{Lt} \tag{5}$$

For the corresponding point clusters $PC_{(t-1)}$ and $PC_{(t)}$ (Step 4) the local pose are denoted as $P_{L(t-1)}$ and $P_{L(t)}$ respectively. Herein, a 3D 4x4 transformation matrix T is defined to transform the local pose of the point cluster $PC_{(t-1)}$ from $P_{L(t-1)}$ to $P_{L(t)}$ (Eqn. 6). Thus, the transformed point cluster $PC_{(t \quad 1)}$ is an approximation of $PC_{(t)}$ in the sensor's frame.

$$\begin{aligned} P_{L(t)} &= T \quad P_{L(t \quad 1)} \\ T &= P_{L(t)} \quad P^{1}_{L(t \quad 1)} \\ PC_{(t \quad 1)} &= T \quad PC_{(t \quad 1)} \end{aligned} \tag{6}$$

Assuming the concerned object is static/non-moving, the global pose of the given clusters $P_{G(t-1)}$ and $P_{G(t)}$ must be

equal. Under such an assumption the transformation matrix T can be represented solely in terms of the sensor's pose matrices $P_{(t-1)}$ and $P_{(t)}$ (Eqn. 7).

$$
\begin{aligned}
&P_{G(t\ \ 1)} = P_{(t\ \ 1)}\ \ P_{L(t\ \ 1)} \quad \text{(using Eqn. 5)}\\
&P_{G(t)} = P_{(t)}\ \ P_{L(t)} \quad \text{(using Eqn. 5)}\\
&P_{(t\ \ 1)}\ \ P_{L(t\ \ 1)} = P_{(t)}\ \ P_{L(t)} \quad \text{(as } P_{G(t-1)} \text{ equals } P_{G(t)})\\
&P_{(t)}^{\ 1}\ \ P_{(t\ \ 1)} = P_{L(t)}\ \ P_{L(t\ \ 1)}^{\ 1}\\
&T = P_{(t)}^{\ 1}\ \ P_{(t\ \ 1)} \quad \text{(substituting in Eqn. 6)}
\end{aligned}
\tag{7}
$$

The local pose transformation matrix T for a static object is derived using the change in inertial pose of the sensor. Hence, the approximated point cluster $PC_{(t\ \ 1)}$ and $PC_{(t)}$ have the same local pose $P_{L(t)}$, only if the concerned object is non-moving. In such a case the point clusters completely overlap in the sensor's frame. However, if the object is moving, the two clusters will not overlap as T does not account for the object's motion. Hence, the extent of overlap between the approximated point cluster $PC_{(t\ \ 1)}$ and the measured point cluster $PC_{(t)}$ is an indication of the object's sate of motion w.r.t the inertial frame.

F. STEP 6: Cluster overlap detection

Every pair of corresponding point clusters in two consecutive pointcloud go through the pose transformation process. The extent of overlap between the approximated point cluster $PC_{(t\ \ 1)}$ and the measured point cluster $PC_{(t)}$ is determined using the octree pointcloud change detection algorithm proposed in [20]. The process determines the amount of spatial change detected between the corresponding clusters while maintaining a structural consistency using double buffered octrees. If the number of points that spatially differ between the two clusters, herein termed as $'$, satisfies a normalized threshold constraint (Eqn. 8), the corresponding object is identified as moving.

$$
' > \frac{N_{PC^{*}_{(t-1)}} + N_{PC_{(t)}}}{} \tag{8}
$$

The normalization parameter in Eqn. 8 represents the sensitivity of the detection process. With a high value of , the constraint gets satisfied for small object movements like within cluster changes (e.g, a person waving his/her hand while standing at a fixed place). However, with high sensitivity values, the number of false-positive detections increases. The results of the detection for the clusters of a pointcloud frame are stored in a boolean result vector **R**. A collection of such result vector stores the detection results of the clusters through time.

G. STEP 7: Confidence based cluster tracking

Using the approach detailed in Steps 1 to 6 the following two observations were made:

Moving objects continuously get detected as moving on multiple consecutive frames. However, occasionally they get erroneously identified as static due to inconsistent motion cues.

Due to the variation in the sensor's motion, static objects get occasionally detected as moving (i.e, false positive detections).

In order to reduce the number of false positive detections a confidence tracking approach was developed that maintains the belief on the objects' motion state(s). A point cluster is classified as moving, only if it has continuously been detected as moving in the last (moving confidence) point cloud frames. The detection results of last frames are stored in result vectors and cluster correspondences between the frames are stored in the last 1 correspondence maps. Eqn. 9 represent the queues that are updated with every incoming pointcloud frame.

$$
\begin{aligned}
\mathbf{M_b} &= \mathbf{M_{t-(\ -1)}}\ :\ :\ :\ \mathbf{M_{t-2}}\ \ \mathbf{M_{t-1}}\\
\mathbf{R_b} &= \mathbf{R_{t-}}\ \ :\ :\ :\ \mathbf{R_{t-2}}\ \ \mathbf{R_{t-1}}
\end{aligned}
\tag{9}
$$

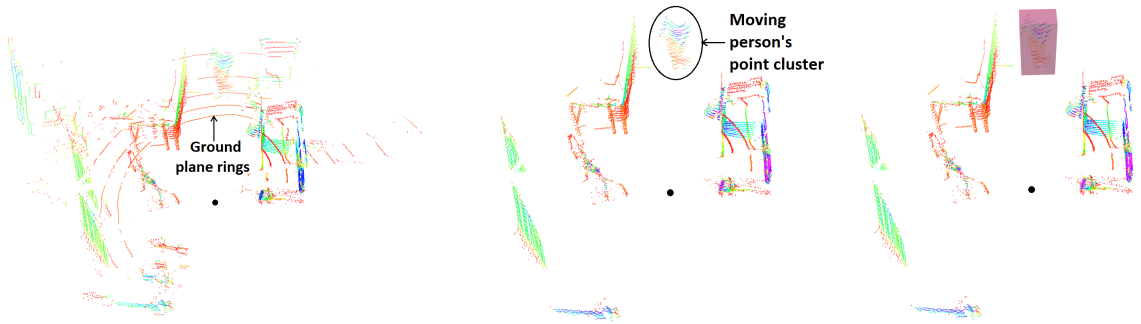
The buffer $\mathbf{M_b}$ contains the correspondence relations while buffer $\mathbf{R_b}$ holds the obtained results (moving clusters/objects). The result buffer $\mathbf{R_b}$ is analyzed using the correspondence relations from $\mathbf{M_b}$ to determine clusters with a continuous detection chain in the last frames. If such a chain is found then the centroid of the cluster is added to a tracking vector $\mathbf{T_m}$ with a non-moving confidence value = 0. The non-moving confidence parameter holds a value that identifies the number of consecutive pointcloud frames in which a given tracked cluster has been detected as non-moving. In order to prevent entry of a redundant cluster centroid which has already been tracked, a new centroid is added to $\mathbf{T_m}$ only if the euclidean distance of the centroid to all the existing centroids in $\mathbf{T_m}$ is greater than a threshold distance $_{catch\ up}$. However, if a centroid match is found (i.e., distance to one of centroids in $\mathbf{T_m}$ is less than $_{catch\ up}$) then the coordinate of the existing centroid in $\mathbf{T_m}$ is updated with the coordinate of the new centroid for further tracking. The parameter for the corresponding centroid (cluster) is updated according to the following three policies:

(1) If the matched cluster is detected as non-moving then is incremented by one (= +1). Such an increment is observed when a moving cluster momentarily stops moving or goes out of the sensor's range or a detection miss occurs due to partial or complete occlusion of the cluster.

(2) If the euclidean distance to the matched cluster's centroid is greater than a threshold $_{leave\ off}$ then is incremented by one unit (= +1). If the distance to the matched cluster centroid is large, it is assumed that the matched centroid is a wrong correspondence for the cluster.

(3) If the matched cluster is detected as moving, then is decremented by one unit (= -1). Such a decrement ensures that moving clusters which occasionally get detected as non-moving due to motion inconsistencies, continue to be classified as moving.

A centroid retains its existence in $\mathbf{T_m}$ until its maximum value condition $_{max}$ is satisfied (i.e., if a centroid's



(a) Raw pointcloud from Velodyne VLP-16.

(b) Pointcloud obtained after euclidean clustering step.

(c) Moving object detected after cluster tracking.

Fig. 4: Results of the proposed DATMO model. LiDAR location shown as a black dot within the scans.

exceeds the threshold then it is removed from $\mathbf{T_m}$). max is thus defined as the maximum number of consecutive frames in which a centroid in $\mathbf{T_m}$ can be detected as non-moving. also holds a minimum value condition of 0 (i.e., if is equal to 0 then its value is no longer decreased). is a variable used to ensure that all the cluster centroids within $\mathbf{T_m}$ are continuously detected as moving by the detection algorithm (Step 6). The parameter ensures detection accuracy while the parameter max ensures detection consistency and hence both parameters are considered critical for the functioning of the proposed DATMO process.

III. RESULTS

The proposed approach was implemented using a gimbal stabilized Humanoid head (Fig 5) with a Velodyne VLP-16 LiDAR and an Intel RealSense tracking camera sensor. The RealSense T265 tracking camera performs an internal V-SLAM algorithm from which precise odometry (position and orientation) data of the VLP-16 sensor was obtained w.r.t the inertial frame.

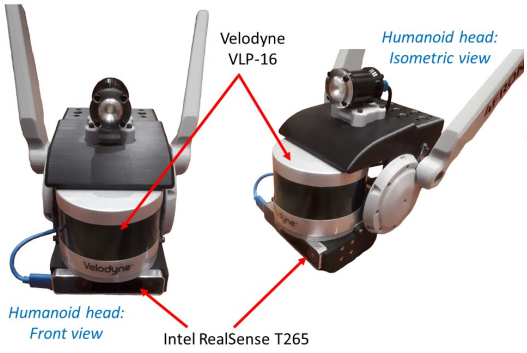


Fig. 5: Humanoid's experimental head with a Velodyne VLP-16 and an Intel RealSense T265 sensors.

The sensor setup mounted on a humanoid platform (Fig. 1) was moved through an office corridor (simulating what was thought would be a representative motion of the mobile robot during a typical office task such as distributing the mail). The environment had one moving object (i.e., a person walking towards and away from the sensor's field of view) and had multiple static objects (e.g, tables, chairs, cubicles, a bicycle, a white board, etc.). A sample result of the proposed moving object detection is shown in Fig 4 which show the generated pointclouds at different steps of the proposed approach, up to the point where the cluster of the walking person was found (i.e red bounding box) and tracked.

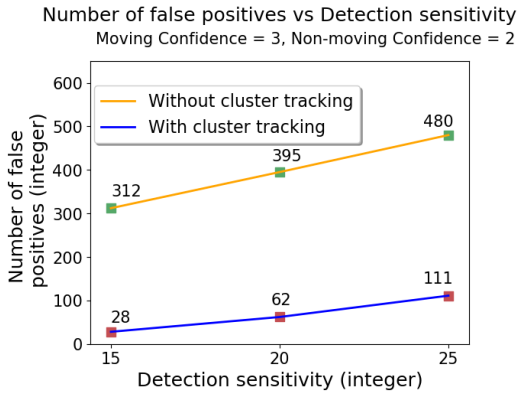


Fig. 6: Number of false positive detections w.r.t .

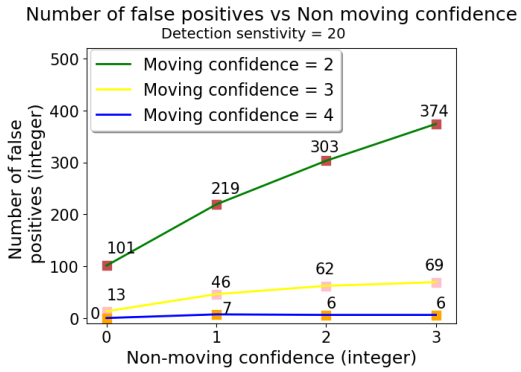


Fig. 7: Number of false positive detections w.r.t and max.

During the experimental tests performed with cluster tracking the parameters $catch\ up$ and $leave\ off$ were fixed to 0.2 and 0.5 meters respectively. The obtained variation in the number of false positive detections w.r.t the

variation in the detection sensitivity parameter is shown in Fig 6. The number of false positives was found to be reduced drastically with the use of the proposed confidence based cluster tracking when compared to the results obtained without using cluster tracking. Fig. 6 also shows that the number of false positive detections increases with increase in the detection sensitivity.

The obtained variation in the number of false positives w.r.t the variation in the tracking parameters and $_{max}$ is shown in Fig 7. It shows the increasing trend of the number of false positives with increase in the non-moving confidence parameter. The graph also shows that the number of false positive detections reduces significantly when the moving confidence value is increased.

For each of the numerous experiments conducted, the ground truth about the captured pointcloud data was obtained and used to estimate the accuracy of the detection as defined in Eqn. 10:

$$\text{Accuracy } (\%) = \frac{\text{True positive detections}}{\text{Ground truth detections}} \quad 100 \qquad (10)$$

The variation in the detection accuracy with respect to the variation in the tracking parameters is shown as a heat map in Fig 8. An analysis of the experimental results can be used

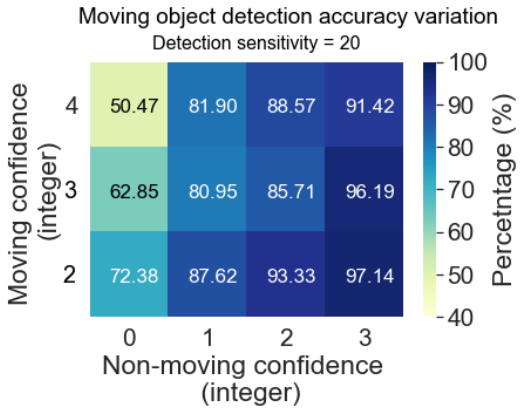


Fig. 8: Accuracy variation w.r.t and $_{max}$.

to select the optimal set of parameters for a given application environment. For the indoor office environment discussed in this paper, the optimal set of parameters is =4, $_{max}$=3, and =20 which provide a 91.42% accuracy with 6 false positive detections. Choice of optimal parameter set is subject to the existing trade-off between accuracy and false positive detections.

For a more extended validation of the research work, experiments were also conducted with the publicly available KITTI Campus dataset [21]. The proposed work has been specifically curated for cluttered indoor environments. However, the campus dataset from KITTI can be considered as a similar dynamic environment that a humanoid may come across while navigating in indoor environments. The dataset is collected using a Velodyne HDL-64E lidar attached to the roof of a moving car. Odometry data is obtained from an OXTS RT 3003 inertial and GPS navigation system. Fig 9a shows an RGB image of the scene, obtained from the front camera of the car. Fig 9b shows 3 persons (2-pedestrians and 1-cyclist) being detected and tracked by the proposed approach. The work only considers an area close to the sensor (indoor range), hence, the presented pointclouds have been zoomed in for providing a clear and enlarged view of the concerned region.

Computation time is a major constraint while processing 3D pointcloud data. The proposed approach for moving object detection has 7 steps which are executed serially. On an average, total computation time per pointcloud frame is about 60-70ms while running on a standard Intel-i9 processor with 8GB of RAM. Major computationally intensive tasks are ground plane removal (32ms) and euclidean clustering (10ms) while the other remaining steps collectively take about 20-25ms. Many methods discussed in Section-I use bayesian formulations for tracking of moving objects which is computationally expensive. The advantage in computation time when compared to these methods is achieved using GNN based approaches and confidence based cluster tracking algorithm.

IV. CONCLUSIONS

A new DATMO model for humanoid navigation in indoor cluttered environments using clustering approaches was proposed. The voxel grid covariance based ground plane removal was found to be effective in handling the translation variations in the sensors' motion. The confidence tracking approach was able to handle motion inconsistencies of indoor moving objects and kept the number of false-positive detections under an acceptable limit.

The algorithm is suitable when the moving objects are either well separated from each other or moving close to each other. However, in highly crowded environments, the cluster tracker faces difficulties in distinguishing one moving object from another due to the use of nearest neighbour approaches. The approach uses a set of parameters that need to be tuned according to diverse application environments. Formal techniques need to be developed for tuning such parameters automatically according to the spatial properties of the pointcloud.

REFERENCES

[1] F. Erich, M. Hirokawa, and K. Suzuki, "A systematic literature review of experiments in socially assistive robotics using humanoid robots," *arXiv preprint arXiv:1711.05379*, 2017.

[2] Á. Llamazares, E. J. Molinos, and M. Ocaña, "Detection and tracking of moving obstacles (datmo): A review," *Robotica*, pp. 1–14.

[3] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. Leonard, "Past, present, and future of simultaneous localization and mapping: Towards the robust-perception age," *IEEE Transactions on Robotics*, vol. 32, no. 6, p. 1309–1332, 2016.

[4] P. Siritanawan, M. D. Prasanjith, and D. Wang, "3d feature points detection on sparse and non-uniform pointcloud for slam," in *2017 18th International Conference on Advanced Robotics (ICAR)*. IEEE, 2017, pp. 112–117.

[5] C.-C. Wang, C. Thorpe, S. Thrun, M. Hebert, and H. Durrant-Whyte, "Simultaneous localization, mapping and moving object tracking," *The International Journal of Robotics Research*, vol. 26, no. 9, pp. 889–916, 2007.