symmetry

MDPI

Article

Socially Aware Robot Navigation with Probabilistic Long-Term Human Trajectory Estimation in Dynamic Environments

Seokjin Kang [1], Suhyeon Kang [2] and Heoncheol Lee [1,*]

[1] Department of IT Convergence Engineering, Kumoh National Institute of Technology, Gumi 39177, Republic of Korea; rkdtjrid@kumoh.ac.kr
[2] Advanced Institute of Convergence Technology, Suwon 16229, Republic of Korea; sh.kang@snu.ac.kr
* Correspondence: hclee@kumoh.ac.kr; Tel.: +82-54-478-7476

Abstract

This paper aims to improve the efficiency of 2D LiDAR-based robot navigation, which can be decreased in dynamic environments. In existing methods, dynamic objects are considered as LiDAR scan data itself, but human behavior patterns (trajectory, speed, etc.) are not considered, which may cause inconvenience to people while the robot is moving along the path. Therefore, in this paper, human behavior patterns (trajectory, speed) are added to a costmap to be considered as obstacles. This enables the robot to perform efficient socially aware robot navigation by considering the human information and avoiding humans in advance. The proposed method is compared with existing methods in a simulation environment by setting the human speed, human trajectory, and the robot's initial position and goal point under each different condition. The overall results show that the existing methods violated the social distance, while the proposed method does not disturb humans through efficient socially aware robot navigation that considers and avoids humans in advance.

Keywords: socially aware robot navigation; long-term probabilistic human trajectory estimation; dynamic environments

1. Introduction

Robot systems are currently being developed in various forms, including indoor service robots, cooperative robots, and autonomous robots [1,2]. Particularly, robots are being integrated with artificial intelligence, computer vision, and sensors to develop more advanced and complex system technologies beyond simple repetitive tasks. Robot navigation [3,4] and path planning are essential technologies for controlling autonomous robots. Path planning is the process of finding the best path to a goal point. Path planning is generally divided into global planning and local planning. The global planner is used to plan the overall path given the initial position of the robot and the goal point. Typical algorithms include A* [5], Dijkstra [6], and D* [7]. Global planners assume a static environment and set a path based on a specific cost function, so they typically prioritize fixed obstacles reflected in the map. The local planner then uses the global path and real-time sensor data to generate feasible motion commands. Based on the path planned by the global planner, the local planner reflects the robot's surroundings in real time through sensor data to guide the robot to the most efficient and optimal path while avoiding static obstacles or dynamic objects. Typical local planners include the Dynamic Window Approach (DWA) [8] and Model Predictive Control (MPC) [9]. Local planners avoid collisions by modifying the

Academic Editor: Chengxi Zhang

Received: 30 April 2026
Revised: 29 May 2026
Accepted: 1 June 2026
Published: 4 June 2026

route from a pre-created map in real time to adjust to the changing environment. Mapping, localization, and path planning are the key components of overall robot navigation.

However, most robot navigation algorithms are based on general static environments without dynamic objects. In dynamic environments, LiDAR sensor data can become distorted [10–12] due to the presence of dynamic objects that differ from the pre-created map data. Several studies have improved SLAM and localization robustness in challenging environments, including robust robot localization in dynamic corridor environments [13] and pitch variation filtering for LiDAR-only SLAM and localization in self-balancing mobile robots [14]. However, these studies mainly focus on localization robustness, whereas this paper reflects human behavior patterns such as speed and head pose in the navigation costmap. In addition, suddenly appearing dynamic objects can cause the path plan to be modified drastically, leading to inefficient path planning, which can increase the overall navigation time or even cause the robot to conflict with dynamic objects as it moves. Since dynamic objects are characterized by constantly changing positions, even if the robot avoids them by reflecting the dynamic object sensor data, if the direction the robot should go and the direction of the dynamic object's movement overlap, the robot's navigation time may increase rapidly or negatively affect the dynamic object. Therefore, dynamic factors caused by humans must be considered in terms of human safety and efficient navigation.

Recent studies have also investigated visual perception [15,16], target-driven visual navigation [17], dynamic obstacle avoidance [18], and smooth robot motion control [19] in robotic systems. These studies demonstrate that perception-aware planning and predictive control are important for reliable robot navigation in dynamic environments. However, the present study focuses on a rule-based costmap integration strategy that can be directly applied within the standard ROS navigation framework without replacing the existing global or local planners.

In Table 1, O indicates that the corresponding factor is explicitly considered, X indicates that it is not considered, and △ indicates that the factor is partially considered or indirectly addressed but not explicitly formulated. Human Trajectory Estimation refers to short-term human trajectory estimation, and Social Distance refers to the minimum distance that a robot should maintain from a person. Probability Approach refers to probabilistic methods such as Gaussian distribution, and Human Behavior Patterns refer to behavior-related cues such as human speed or head pose.

Table 1. Representative related works on robot navigation in dynamic and social environments.

Related Works	Human Trajectory Estimation	Social Distance	Probability Approach	Human Behavior Patterns
[20–24]	X	X	X	X
[25–27]	O	X	X	X
[28]	O	△	X	X
[29,30]	O	△	O	X
Proposed	O	O	O	O

Unlike end-to-end learning-based social navigation methods that replace the navigation policy, the proposed method adopts a rule-based costmap layering strategy that can be directly integrated into the standard ROS navigation stack. The main novelty of this work is the explicit incorporation of human behavior patterns, including walking speed and head pose direction, into a probabilistic trajectory costmap. Therefore, the proposed framework enables socially aware navigation without modifying the underlying global or local planners.

The existing methods [20–24] do not consider dynamic object behavior patterns, and the efficiency of path planning decreases rapidly when the robot's path and the human's trajectory overlap. Furthermore, social distance is not considered, which increases the risk of collision when the human is in motion. Other existing methods either consider trajectory but not social distance [25–27], or do not define or consider social distance clearly [28–30]. In general, the direction in which a person's head is facing is likely to be the direction in which the person is moving, but none of the existing methods consider these human behavior patterns.

To efficiently avoid and consider people as dynamic objects, estimating their trajectories would be helpful. Efficient socially aware robot navigation algorithms that estimate and consider a person's current trajectory in dynamic environments are needed. Although it is difficult to accurately estimate human trajectories, human behavior patterns that can be generally considered exist. A more accurate estimation of human trajectories can be achieved by considering human behavior patterns such as human speed and human head pose.

Therefore, an efficient socially aware robot navigation in dynamic environments that reflects human behavior patterns is proposed in this paper. The contributions of this paper are organized as follows.

(1) Socially aware robot navigation that considers social distance and human behavior patterns.
(2) Improved performance of the proposed method is demonstrated by comparing the results with existing methods.
(3) Validation of applicability to experimental environments through experimentation.

The remainder of this paper is organized as follows. Section 2 describes the problem description, such as system configuration, problems of robot navigation in dynamic environments, and human trajectory estimation. Section 3 describes the proposed method, efficient socially aware robot navigation in dynamic environments. Section 4 describes the results, including simulation results and experimental results. Finally, conclusions are discussed in Section 5.

2. Problem Description

2.1. System Configuration

The robot hardware configuration used in the paper is as follows. A 2D LiDAR is used for environment mapping, robot localization, path planning, and robot navigation. A depth camera is used for the object detection and head pose estimation algorithms to utilize human behavior patterns. The overall robot hardware configuration used is shown in Figure 1. A depth camera is attached to the front of the robot, and a 2D LiDAR is attached to the top for full data acquisition. ROS (Robot Operating System) is used for the overall algorithm development.

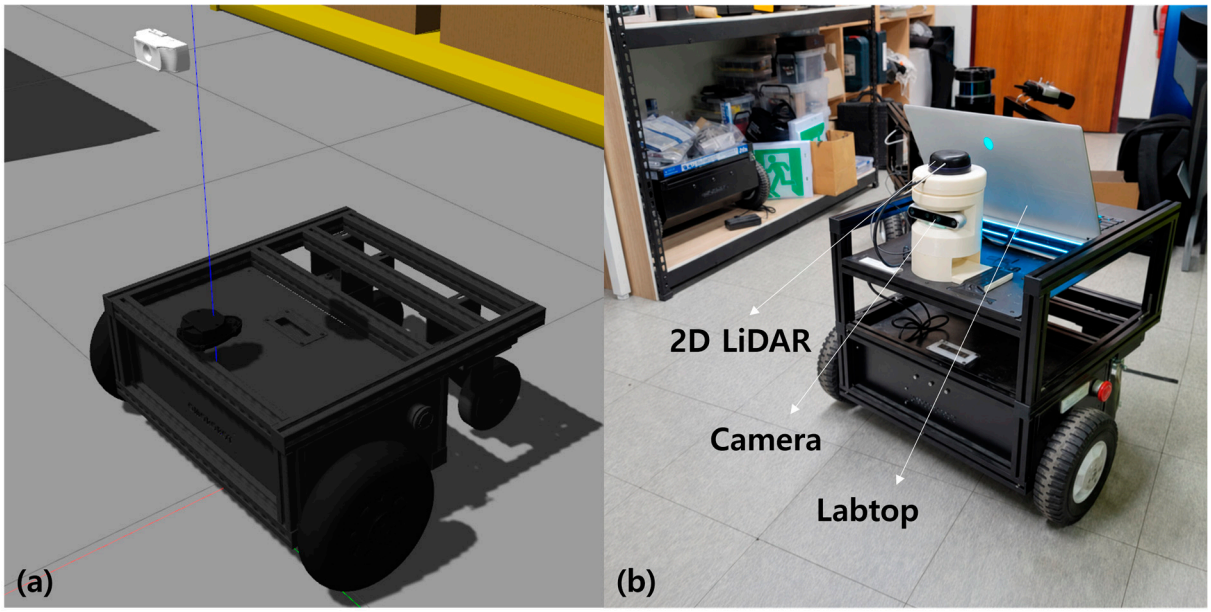

Figure 1. Robot system configuration: (**a**) configuration in simulations, (**b**) configuration in the experimental setups.

2.2. Difficulty in Mobile Robot Navigation in Dynamic Environments

Unlike pre-drawn maps of static environments, which align with incoming sensor data, dynamic environments introduce dynamic objects with unknown movement directions, complicating efficient robot navigation. Existing robot navigation systems using 2D LiDAR consider scan data from dynamic objects but do not account for the direction, speed, or movement characteristics of humans. As a result, even if a robot–human collision is temporarily avoided using scan data, the risk of collision increases or the overall robot navigation time may rise, disturbing efficient navigation. Therefore, additional methods that account for human behavior patterns are necessary for socially aware robot navigation in dynamic environments to prevent collisions and ensure respect for people.

2.3. Need for Long-Term Human Trajectory Estimation

In this paper, human behavior patterns are defined as observable motion-related cues that can be used to infer a person's future movement direction, such as walking speed and head pose direction. Short-term human trajectory estimation refers to estimating a person's immediate movement direction based mainly on recent position changes. In contrast, long-term human trajectory estimation refers to estimating a probable future movement direction by additionally considering these behavior patterns. Humans generally move by looking in the direction they are going. Furthermore, they tend to look ahead to the next direction of travel and adjust their movement accordingly. Therefore, estimating human trajectory using only recent position changes may be insufficient, and long-term human trajectory estimation that reflects human behavior patterns is necessary for socially aware robot navigation. In this study, long-term trajectory estimation does not refer to predicting an exact future position at a fixed prediction time. Instead, it refers to generating a behavior-aware probabilistic trajectory costmap by considering walking speed and head pose direction as motion-related cues. This study does not assume that head pose perfectly determines the future walking direction. Instead, head pose is used as an auxiliary behavioral cue for estimating the likely direction of human motion. In general walking situations, pedestrians often look toward their intended direction of travel. However, this assumption may not hold when a person looks sideways while walking, turns the head during social interaction, or continuously looks away from the actual walking direction.

These unusual cases are outside the scope of the current framework and are discussed as limitations in Section 5.

3. Proposed Method

3.1. Overview of the Proposed Method

The overall system flowchart of the proposed method is shown in Figure 2. The raw scan data from a 2D LiDAR is used for robot localization along with the environment map. Object detection and head pose estimation algorithms are performed using RGB-D camera sensor data. Each estimated object, head pose, and human speed is used to define a new costmap for efficient robot navigation, considering human information.

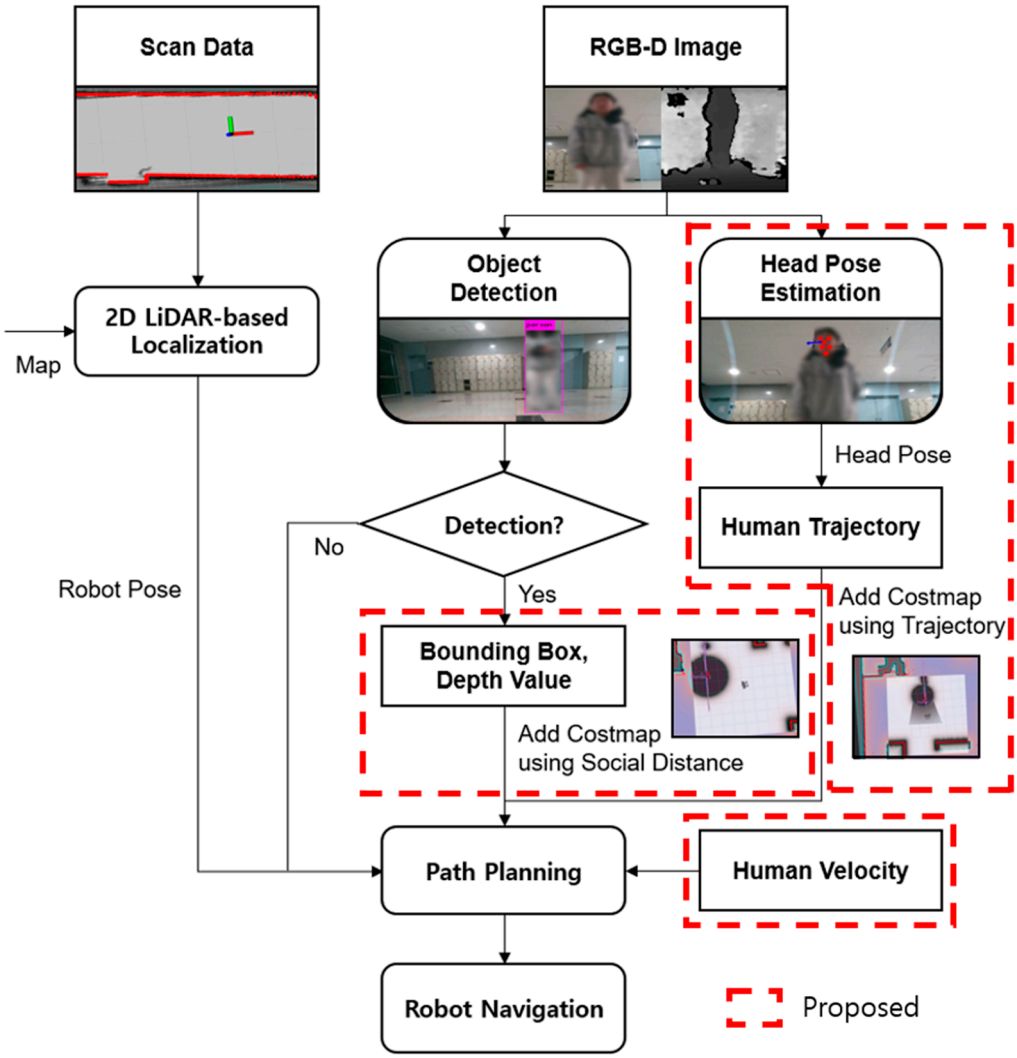

Figure 2. Flowchart of the proposed method.

3.2. Long-Term Human Trajectory Estimation by Head Pose Estimation

3.2.1. Social Distance Costmap

Social distance is the minimum distance a robot must maintain from a human in socially aware robot navigation. It is defined to prevent collisions and ensure human safety during path planning. In this paper, the social distance is applied to the human's central position for socially aware robot navigation and incorporated into the costmap. The default social distance value is set to 1 m and is chosen as a balanced value. A larger value could cause the robot to generate overly wide paths around a person, which reduces efficiency, while a smaller value might increase the risk of collisions.

In addition, the human speed value is added to the minimum social distance to create a circular social distance costmap. The faster the human moves, the larger the social distance. In the costmap, each cell can have a value between 0 and 255. However, the cells added to the social distance costmap are set to lethal obstacles. Since the social distance represents the minimum distance to respect people, the cells affected by this condition are not treated probabilistically. Instead, they are designated as areas where obstacles exist, potentially causing collisions. However, assigning Lethal_Obstacle to the social distance region may restrict the feasible path space in narrow corridors or highly constrained environments. In this study, this setting was used to prioritize human safety and to prevent the robot from entering the minimum personal space around a person. In more constrained environments, the social distance costmap can be extended to an adaptive high-cost region rather than a lethal obstacle region, allowing the robot to maintain social distance while reducing the risk of the local planner freezing. The pseudo code for adding the social distance costmap using the estimated human position and speed is shown in Algorithm 1.

Algorithm 1. Add Social Distance Costmap

function SD Costmap
    double *wx*, *wy*, *t*
    double *current_position*, *previous_position*, *human_speed* // Human
    double *dist*, *social_distance* // The distance between cells and human
    $dt \leftarrow (current\_time - previous\_time).toSec()$
    $dx \leftarrow current\_position.x - previous\_position.x$
    $dy \leftarrow current\_position.y - previous\_position.y$
    $moving\_distance \leftarrow \sqrt{dx^2 + dy^2}$
    $accumulated\_dt \leftarrow accumulated\_dt + dt$
    $accumulated\_distance \leftarrow accumulated\_distance + moving\_distance$
    if $accumulated\_dt \geq t$ **then**
        $human\_speed \leftarrow accumulated\_distance \,/\, accumulated\_dt$
        Initialize(*accumulated_dt*)
        Initialize(*accumulated_distance*)
    end if
    $social\_distance \leftarrow social\_distance + human\_speed$
    $(wx, wy) \leftarrow$ master_grid.mapToWorld(*cell.x*, *cell.y*)
    $\text{dist} \leftarrow \sqrt{(wx - person\_position.x)^2 + (wy - person\_position.y)}$
    if $dist \leq social\_distance$ **then**
        master_grid.setCost(*cell.x*, *cell.y*, *Lethal_Obstacle*)
    end if
end function

The results of adding human speeds of 0.2 m/s and 0.6 m/s to the basic social distance are shown in Figures 3a and 3b, respectively. By comparing the social distance regions in Figure 3a,b, the area of the social distance region changes depending on the human speed. This enables more efficient socially aware robot navigation by expanding the avoidance region according to human walking speed.

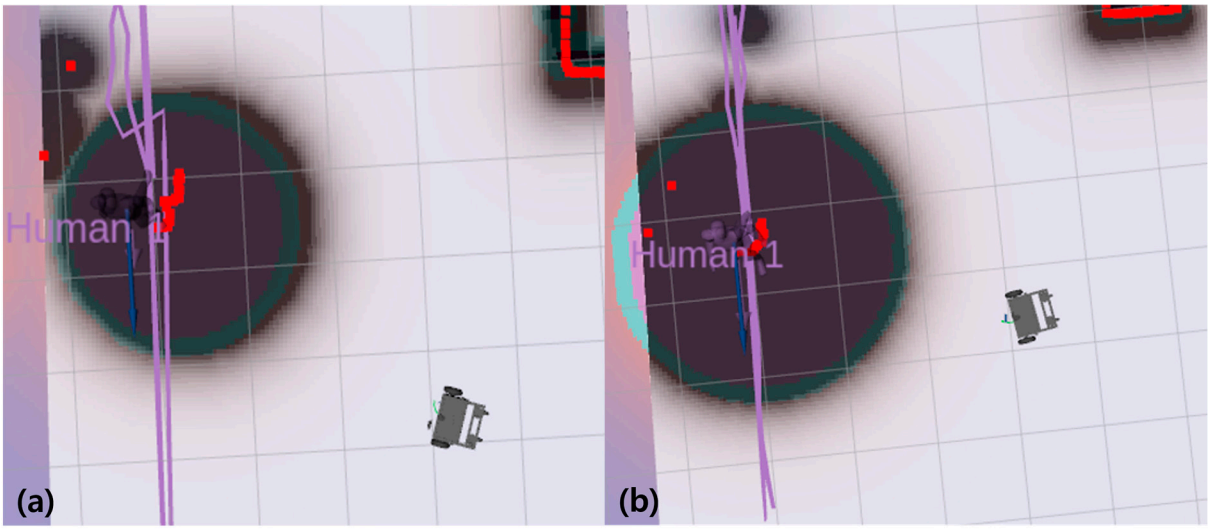

Figure 3. Social distance region according to human speed: (**a**) Social distance costmap with human speed of 0.2 m/s; (**b**) social distance costmap with human speed of 0.6 m/s. The purple line indicates the human trajectory, and the red points represent LiDAR scan data.

3.2.2. Long-Term Human Trajectory Costmap

This section describes the human head pose estimation algorithms needed to reflect human behavior patterns. Generally, people tend to look in the direction they need to go, so the head pose estimation algorithm is used to reflect this behavioral characteristic. The head pose estimation algorithm estimates the direction of a recognized human head from an input image or video. It mainly estimates the head pose in three angles: yaw, pitch, and roll. Head pose estimation is utilized in various fields such as driver status monitoring and interaction in autonomous vehicles. Typical approaches for head pose estimation algorithms include feature-based approaches, learning-based approaches, and 3D model-based approaches. Feature-based approaches detect features such as eyes, nose, and mouth on the face and use them to estimate the orientation of the head. Learning-based approaches train a model, such as a CNN, to estimate head orientation directly from an image. Finally, 3D model-based approaches use Perspective-n-Point (PnP) to extract feature points from 2D images and compare them with 3D models to estimate head pose. Recent vision-based recognition studies have shown that compact neural architectures can support real-time perception tasks in constrained environments. Although the proposed framework does not employ these learning-based recognition models, such studies indicate the importance of efficient visual perception modules for real-time robotic systems. In this paper, 3D model-based head pose estimation is used to estimate head pose using PnP, which has the advantage of relatively high accuracy and real-time processing. Pose estimation is performed using the PnP algorithm by using two types of information: the specified 3D model points and the 2D image points recognized in the current image. The overall head pose estimation process is as follows. First, the 3D model points of the face are defined. The values of the 3D model points defined in this paper are as shown in Equation (1). The points in Equation (1) are the coordinates of the 3D space defined in the model. The 3D facial model points are predefined reference coordinates used for 3D model-based head pose estimation. In this study, these values are used as generic facial landmark reference points rather than subject-specific parameters.

$$\mathbf{P}_3 = \begin{bmatrix} (0.0,\ 0.0,\ 0.0) & (Nose\ tip) \\ (0.0, -330.0, -65.0) & (Jaw) \\ (-255.0,\ 170.0, -135.0) & (Left\ corner\ of\ the\ left\ eye) \\ (255.0,\ 170.0, -135.0) & (Right\ corner\ of\ the\ right\ eye) \\ (-150.0, -150.0, -125.0) & (Left\ corner\ of\ the\ mouth) \\ (150.0, -150.0, -125.0) & (Right\ corner\ of\ the\ mouth) \end{bmatrix} \tag{1}$$

These 3D model points and the corresponding 2D image points are used to calculate rotations and transformations. The 2D image points used here are the feature points of the face currently recognized in the camera image. The 2D image points are calculated from the face landmarks after detecting the face first. The 2D image is defined as shown in Equation (2).

$$\mathbf{P}_2 = \begin{bmatrix} (u_1,\ v_1) & (Nose\ tip) \\ (u_2,\ v_2) & (Jaw) \\ (u_3,\ v_3) & (Left\ corner\ of\ the\ left\ eye) \\ (u_4,\ v_4) & (Right\ corner\ of\ the\ right\ eye) \\ (u_5,\ v_5) & (Left\ corner\ of\ the\ mouth) \\ (u_6,\ v_6) & (Right\ corner\ of\ the\ mouth) \end{bmatrix} \tag{2}$$

Using the 3D and 2D points defined in Equations (1) and (2), the rotation and translation vectors of the camera are calculated using the PnP algorithm. The PnP algorithm estimates the camera pose by minimizing the reprojection error between the observed 2D image points and the projected 3D model points. The processing of the PnP algorithm consists of the perspective projection, camera matrix, rotation vector, and translation vector. The perspective projection process, which projects 3D points onto a 2D image, is shown in Equation (3).

$$s \cdot \begin{bmatrix} u_i \\ v_i \end{bmatrix} = \boldsymbol{K} \cdot [\boldsymbol{R}|\boldsymbol{t}] \cdot \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix}, \text{for } i = 1, 2, \cdots, n \tag{3}$$

In Equation (3), $[u_i, v_i]^T$ is a 2D point in the current image, and the feature points used in this paper are the tip of the nose, jaw, left corner of the left eye, right corner of the right eye, left corner of the mouth, and right corner of the mouth, totaling n = 6. $[x_i,\ y_i,\ z_i]^T$ is the corresponding points of the predefined 3D model. $[R|t]$ is a transformation matrix consisting of a rotation matrix (**R**) and a transformation vector (**t**), K is the camera's intrinsic parameter matrix, and s is the depth scale factor, which is the value used when coordinates in 3D space are projected onto the 2D image plane. Used intrinsic parameter K values are shown in Equation (4), where $f_x,\ f_y$ are the focal lengths of the camera and $c_x,\ c_y$ are the center coordinates of the image.

$$\boldsymbol{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

$$\min_{R,t} \sum_i \| (u_i, v_i) - \pi(\boldsymbol{K}, \boldsymbol{R},\ \boldsymbol{t},\ (x_i,\ y_i,\ z_i)) \|^2 \tag{5}$$

Using the above variables and the perspective projection equation, the rotation vector r and translation vector t are found using the solvePnP function of the OpenCV library. The process of the solvePnP function is shown in Equation (5), with **r** and **t** as output values. With the rotation vector **r** as input to the Rodrigues function, the rotation matrix **R** is obtained through the process in Equation (6).

$$\boldsymbol{R} = Rodrigues(\boldsymbol{r}) \tag{6}$$

$$\theta = \|\boldsymbol{r}\| = \sqrt{r_x^2 + r_y^2 + r_z^2} \tag{7}$$

$$\hat{\boldsymbol{r}} = \frac{\boldsymbol{r}}{\|\boldsymbol{r}\|} = \left(\frac{r_x}{\theta}, \frac{r_y}{\theta}, \frac{r_z}{\theta}\right)^T \tag{8}$$

$$\boldsymbol{R} = \boldsymbol{I} + (\sin\theta)\boldsymbol{P} + (1 - \cos\theta)\boldsymbol{P}^2 \tag{9}$$

In Equations (7) and (8), the magnitude of the rotation vector r is the rotation angle θ, and the unit direction vector $\hat{r}$ refers to the rotation axis. The Rodrigues function can be expressed by solving the equation, as shown in Equation (9). In Equation (9), I is a 3×3 unit matrix, and P is an antisymmetric matrix based on the rotation axis $\hat{r}$. Expressing P as an equation is shown in Equation (10). The rotation matrix R obtained in Equation (9) and the transformation vector t obtained through the solvePnP function are applied to the perspective projection Equation (3) to find the location of the head pose. The rotation matrix R is represented by Equation (11), which is converted into the Euler angle of yaw, pitch, and roll as shown in Equations (12)–(14), respectively, to get the final output value of the head pose angle.

$$\boldsymbol{P} = \begin{bmatrix} 0 & -\hat{r}_z & \hat{r}_y \\ \hat{r}_z & 0 & -\hat{r}_x \\ -\hat{r}_y & \hat{r}_x & 0 \end{bmatrix} \tag{10}$$

$$\boldsymbol{R} = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \tag{11}$$

$$\boldsymbol{yaw} = \boldsymbol{atan}2(R_{21}, R_{11}) \tag{12}$$

$$\boldsymbol{pitch} = \boldsymbol{atan}2\left(-\boldsymbol{R}_{31}, \sqrt{\boldsymbol{R}_{32}{}^2 + \boldsymbol{R}_{33}{}^2}\right) \tag{13}$$

$$\boldsymbol{roll} = \boldsymbol{atan}2(\boldsymbol{R}_{32}, \boldsymbol{R}_{33}) \tag{14}$$

The Euler angle consists of three values: yaw, pitch, and roll, and since this paper performs navigation using 2D LiDAR, only the yaw value of rotation with respect to the ground is extracted and used. The extracted yaw value is published as a ROS topic, and the path-planning node receives this value and uses it to define a new costmap. The results of the head pose estimation algorithm in the simulation environment and the experimental environment are shown in Figure 4. In Figure 4, the red dots correspond to the left corner of the left eye, the right corner of the right eye, the tip of the nose, the left corner of the mouth, the right corner of the mouth, and the chin on the 2D image. The estimated yaw angle value is defined in the form of a new costmap, which is used for dynamic object avoidance.

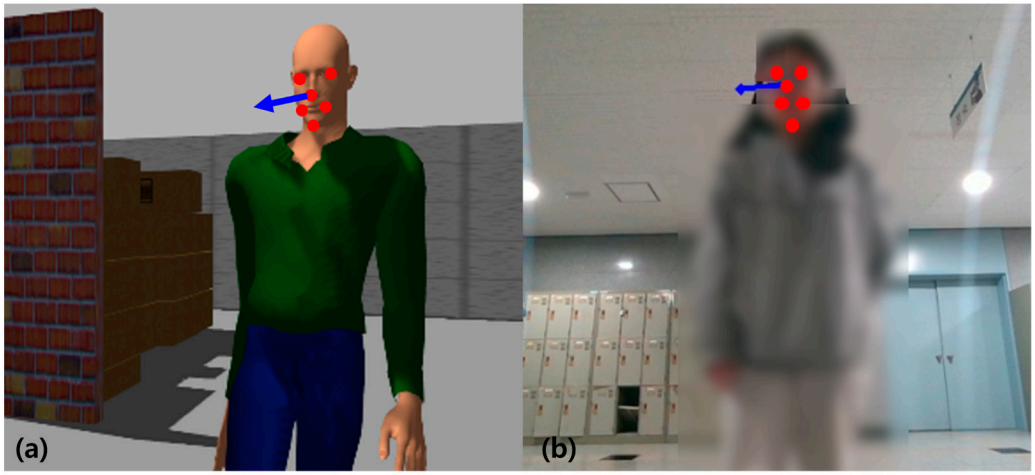

Figure 4. Results of the head pose estimation algorithm. The red dots indicate the facial landmarks used for head pose estimation, and the arrow indicates the estimated head pose direction: (**a**) results in simulation; (**b**) results in the experimental environment.

3.3. Probabilistic Human Trajectory Estimation by Costmap

The head pose recognized by the head pose estimation algorithm is added as a trajectory costmap. In addition to the social distance costmap added after object recognition, the estimated head pose is added as a trajectory costmap when the head pose is recognized. Before adding the head pose output, moving horizon estimation (MHE) is applied. The MHE method groups data into clusters, discarding old data as newer data comes in and constantly updating the clusters with new data to keep the data up-to-date. This makes the head pose estimates more robust, even when the head pose values contain outliers, because such outliers are mitigated by MHE. MHE is used to reduce the influence of temporary outliers in the estimated head pose sequence by considering a finite moving window of recent estimates. This also helps reduce abrupt changes in the trajectory costmap caused by temporary fluctuations in head pose estimation. However, since the local planner may still be affected by rapidly changing costmap layers in dynamic environments, quantitative evaluation of control smoothness, such as angular acceleration and trajectory jitter, will be considered in future work. Adding a trajectory costmap forms an inverted triangle around the person, which weights the cost values of each cell in the inverted triangle by applying a Gaussian distribution function based on the distance of the cell from the person. The Gaussian distribution function used is shown in Equation (15).

$$Gaussian\_cost = A \cdot e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (15)$$

In Equation (15), A is the maximum output value of the Gaussian function and is set to Lethal_Obstacle, which is the maximum cost value used in path planning. The variable x is the dist value defined in Algorithm 1, representing the distance from the person to each costmap cell. Therefore, the Gaussian mean μ was set to 0 so that the maximum cost is assigned to cells closest to the person, and the cost gradually decreases as the distance from the person increases. The standard deviation σ controls the spread of the trajectory cost region and was set to 2 in this study. The main costmap parameters were fixed for all simulation and experimental validation cases. The default social distance was set to 1 m, and the estimated human walking speed was added to this value to expand the social distance costmap. The ROS costmap cell value ranges from 0 to 255, and cells inside the social distance costmap were assigned Lethal_Obstacle. The pseudo code for the process of adding a trajectory costmap is shown in Algorithm 2.

Algorithm 2. Add Long-Term Probabilistic Trajectory Costmap

function Tcostmap(velocity_obstacles_list, master_grid)
  for all *obs* in velocity obstacles list **do**
    $(mx, my) \leftarrow$ master_grid.worldToMap($obs.x$, $obs.y$)
    if Conversion succeeds **then**
      $sector\ length \leftarrow obs.radius$
      $sector\ cells \leftarrow \lfloor sector\ length /$ master grid.getResolution$() \rfloor$
      for all $dx \in [-sector\_cells, sector\_cells]$ **do**
        for all $dy \in [-sector\_cells, sector\_cells]$ **do**
          $cell\_x \leftarrow mx + dx$, $cell\_y \leftarrow my + dy$
          $(wx, wy) \leftarrow$ master grid.mapToWorld$(cell_x, cell_y)$
          $dist \leftarrow \sqrt{(wx - obs.x)^2 + (wy - obs.y)^2}$
          $angle_p \leftarrow atan2(wy - obs.y,\ wx - (obs.x + social\ distance))$
          $angle_m \leftarrow atan2(wy - obs.y,\ wx - (obs.x - social\ distance))$
          if Cell(angle) falls within angular constraints **then**
            A $\leftarrow$ Lethal_Obstacle
            $cost \leftarrow A \cdot e^{-\frac{dist^2}{2 \cdot sigma^2}}$
            $final\ cost \leftarrow$ Normalize(cost, 0, Lethal Obstacle)
            master grid.setCost($cell\_x$, $cell\_y$, $final\_cost$)
          end if
        end for
      end for
    end if
  end for
end function

First, the person's data, including position and trajectory calculated by other nodes, are received, and all cells in the sector are traversed. The Euclidean distance between the current cell and the person is stored in the dist variable. The variables angle_p and angle_m define the angular boundaries of the trajectory cost region around the person's estimated moving direction. If the angle of a cell falls within these angular boundaries, the cell is included in the long-term probabilistic trajectory costmap. The trajectory cost region is determined based on the person's direction of movement, and the costmap is updated dynamically as the direction changes. A Gaussian distribution function is applied to each cell, assigning a higher final_cost to cells closer to the person. The result of applying the trajectory costmap using head pose information is shown in Figure 5. In Figure 5, the circle-shaped costmap is the social distance costmap added by the object detection algorithm. In addition, the inverted triangle-shaped costmap is the trajectory costmap added based on the head pose value. By applying a Gaussian distribution function to this trajectory costmap, the cost of each cell varies depending on the distance. Cells closer to the person have a higher cost and are darker in color, while cells farther away have a lower cost and are lighter in color.

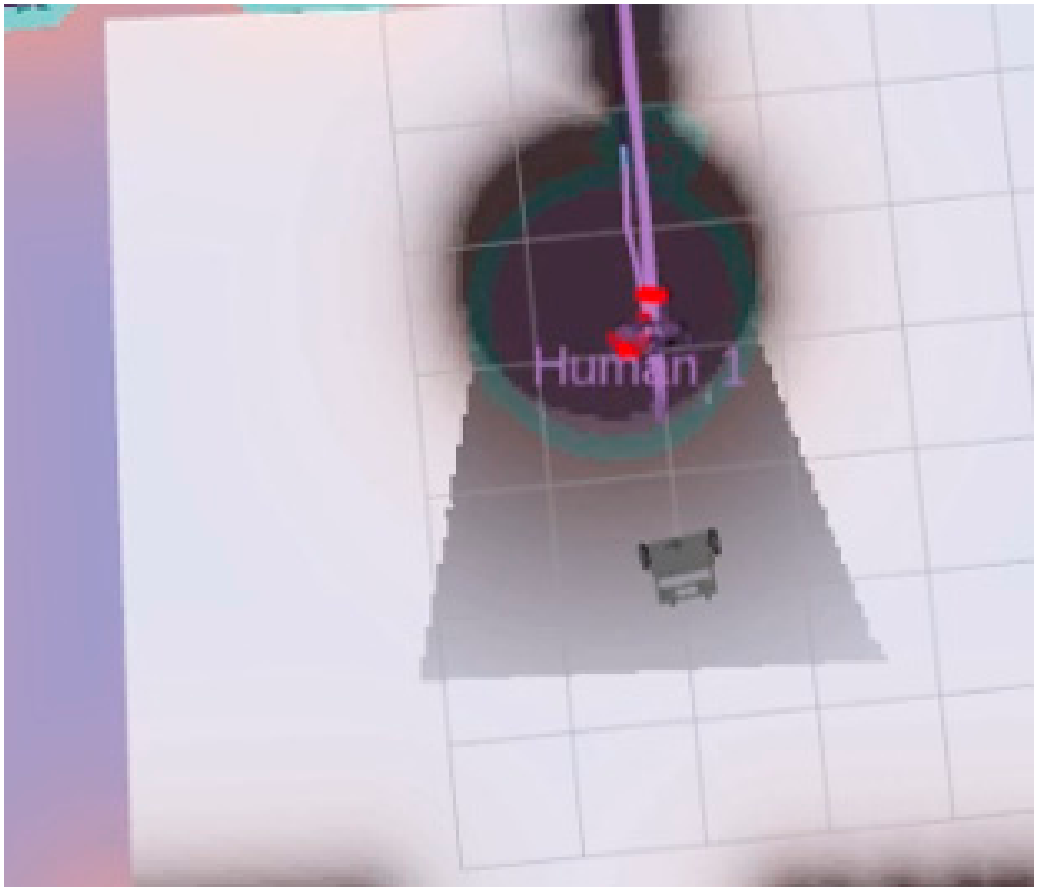

Figure 5. Result of adding a long-term probabilistic trajectory costmap in simulation. The circular region represents the social distance costmap around the detected human, and the inverted triangular region represents the long-term probabilistic trajectory costmap generated based on the estimated head pose direction. Darker colors indicate higher cost values.

3.4. Path Replanning and Control Based on the Long-Term Probabilistic Human Trajectory Estimation

The proposed method does not replace the global or local planner. Instead, it modifies the navigation costmap by adding the social distance costmap and the long-term probabilistic trajectory costmap as additional cost layers. The resulting costmap is then used by the existing ROS navigation framework for path planning and control.

The comparison results of the proposed method with the added social distance costmap and long-term probabilistic human trajectory costmap as obstacles for path-planning, and other existing methods, are shown in Figure 6. In Figure 6a–c, the path planning results of the existing methods do not consider humans in advance and calculate possible collision paths when planning the overall path through the global planner. In contrast, Figure 6d shows that the proposed method generates an efficient path using the added costmaps.

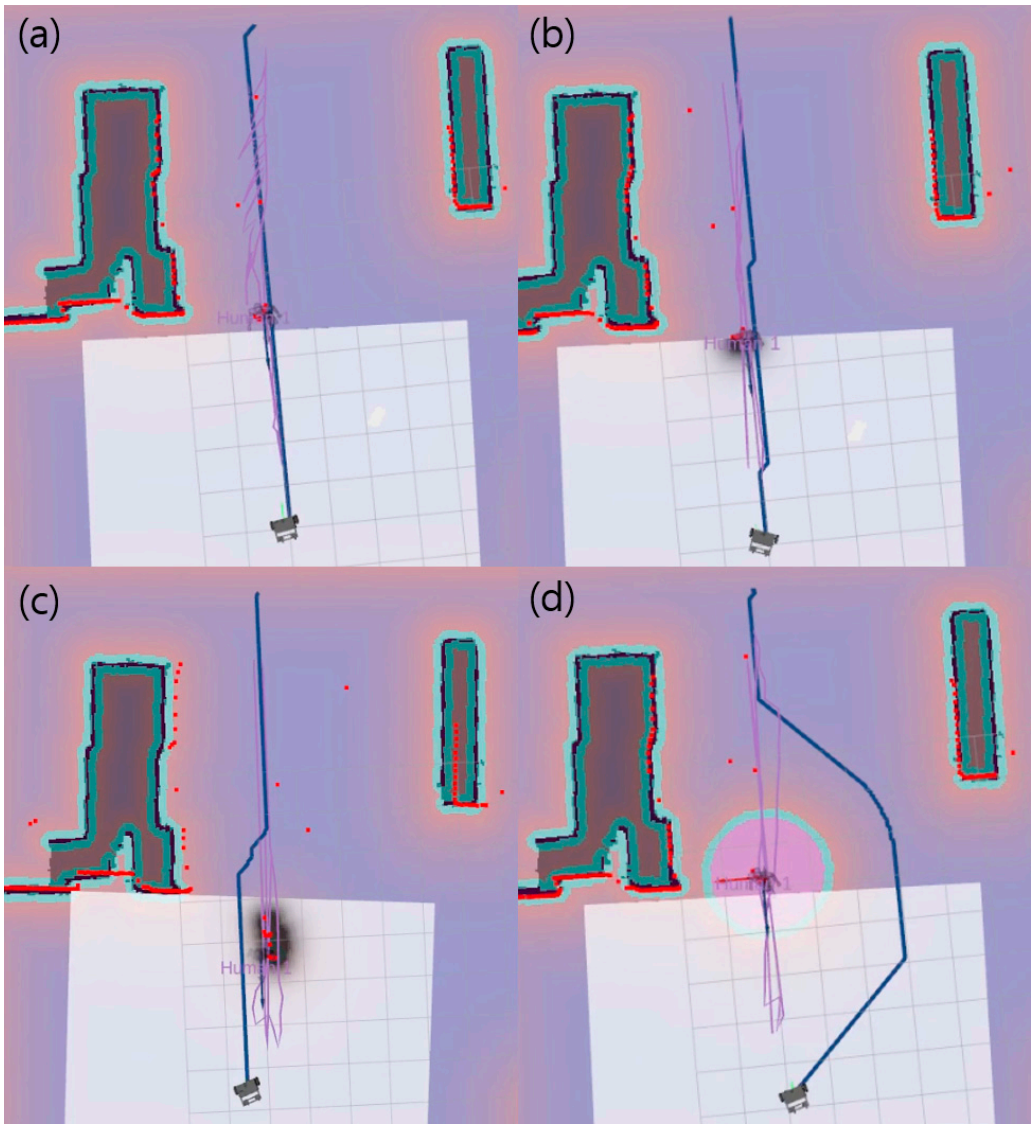

Figure 6. Results of the overall global path planning: (**a**) D* method, (**b**) A* method, (**c**) Dijkstra method, (**d**) proposed method with social distance costmap.

4. Results

4.1. Simulation Results

4.1.1. Simulation Setup

The proposed method is tested in a simulated environment on a desktop. The software setup is based on Ubuntu 20.04 Noetic and consists of a 12 Core i7 and 16 GB RAM. The overall modules of the proposed method consist of a mapping module through SLAM, a robot localization module, an object detection module, a head pose estimation module, and a path planning and navigation module. The entire algorithm is implemented based on ROS. Although the MHE-based filtering used in this study is implemented as a finite-window filtering process, a detailed module-wise computational cost analysis was not conducted in this study. The overall framework includes object detection, head pose estimation, MHE-based filtering, costmap updates, and path planning, and these modules may affect the real-time navigation loop depending on the hardware configuration and environmental complexity. Therefore, module-wise execution time, latency, and real-time update frequency will be quantitatively evaluated in future work.

4.1.2. Environment Configurations

The simulation environment is set up as shown in Figure 7. The number of dynamic objects used in the simulation is 1–2. The simulation environment consists of dynamic objects and static obstacles. The experiment is conducted by changing the position and trajectory of the dynamic object without changing the static obstacles.

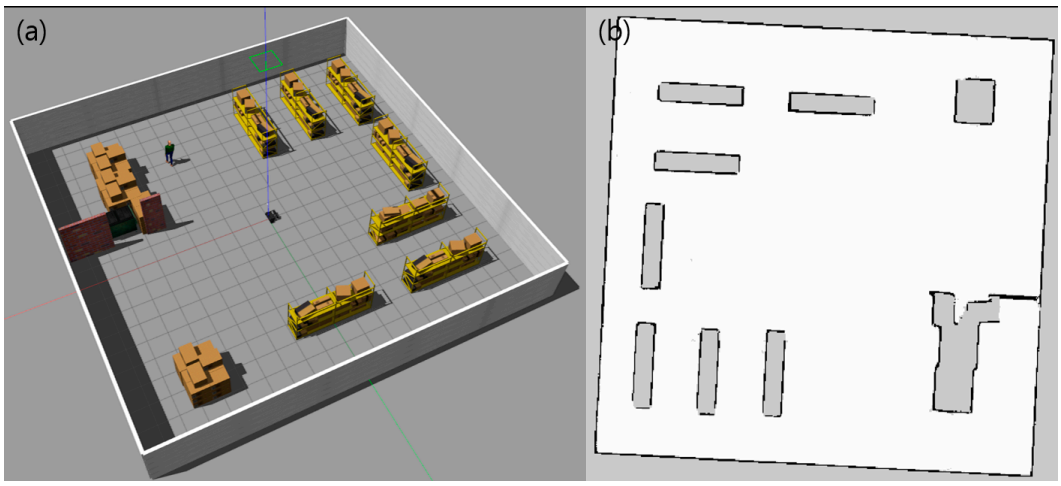

Figure 7. Simulation environment: (**a**) Environment configuration with dynamic objects and static obstacles, (**b**) map of the simulation environment.

Before proceeding with the experiments and analysis in the simulation environment, an environment map was created, which is essential for robot navigation and localization algorithms. Since the mapping should consist of only static obstacles, the dynamic objects shown in Figure 7a are excluded. The Cartographer algorithm based on 2D LiDAR was used for the mapping. The created map result is shown in Figure 7b. The origin of the map is the middle point in Figure 7b.

4.1.3. Trajectory Results

To validate the performance of the proposed method, a comparison and analysis with existing methods were conducted. The trajectory of the human, the robot, and the initial position and goal point of the dynamic object are plotted to show the overall path. Quantitative results included total navigation time, social distance violation time, and average human–robot distance.

First, the results for Case 1 are described. The human's trajectory is plotted on the map with a starting point of (3, 2) and a goal point of (3, −8). The human repeatedly moves between these points in the simulation environment. The human with this trajectory is referred to as Human1. In all experiments, DWA was used as the local planner. For the baseline methods, the global planner was changed among D*, A*, and Dijkstra. For the proposed method, the same ROS navigation framework was used, but the proposed social distance and long-term probabilistic trajectory cost layers were added to the navigation costmap. It should be noted that D*, A*, and Dijkstra are used as representative conventional path-planning baselines rather than modern socially aware navigation baselines. The purpose of this comparison is to evaluate the effect of adding the proposed human-aware costmap layers to the conventional ROS navigation framework under the same local planner setting. Since the proposed method is designed as an additional costmap layer rather than a replacement for the global or local planner, direct comparison with Social Force Model-based navigation, time-dependent networks, and learning-based social navigation

methods will be considered in future work. In Case 1, the human speed was set to two values: 0.3 m/s and 0.6 m/s.

The results of each experiment for Human1 are shown in Figure 8. In Figure 8, (a), (b), and (c) are the results of the D*, A*, and Dijkstra methods, respectively. Part (d) is the trajectory result after applying the proposed method. The labels (1), (2), (3), and (4) are the results of changing the human's velocity, the robot's initial position, and the goal point, respectively. In all cases, the human position and trajectory are the same. In Figure 8, social distance violation means that the robot has violated the human's personal space. No social distance violation means that the trajectories of the human and robot overlapped at different timestamps. The trajectory results of the existing methods do not consider human behavior patterns and may generate paths that potentially conflict with human trajectories when planning the overall path. This caused the robot to violate the human's personal space while traveling and forced the local planner to modify the path suddenly. On the other hand, the trajectory result of the proposed method is different from the trajectory of the existing methods, and the added long-term probabilistic human trajectory costmap generates a path that avoids humans in advance.

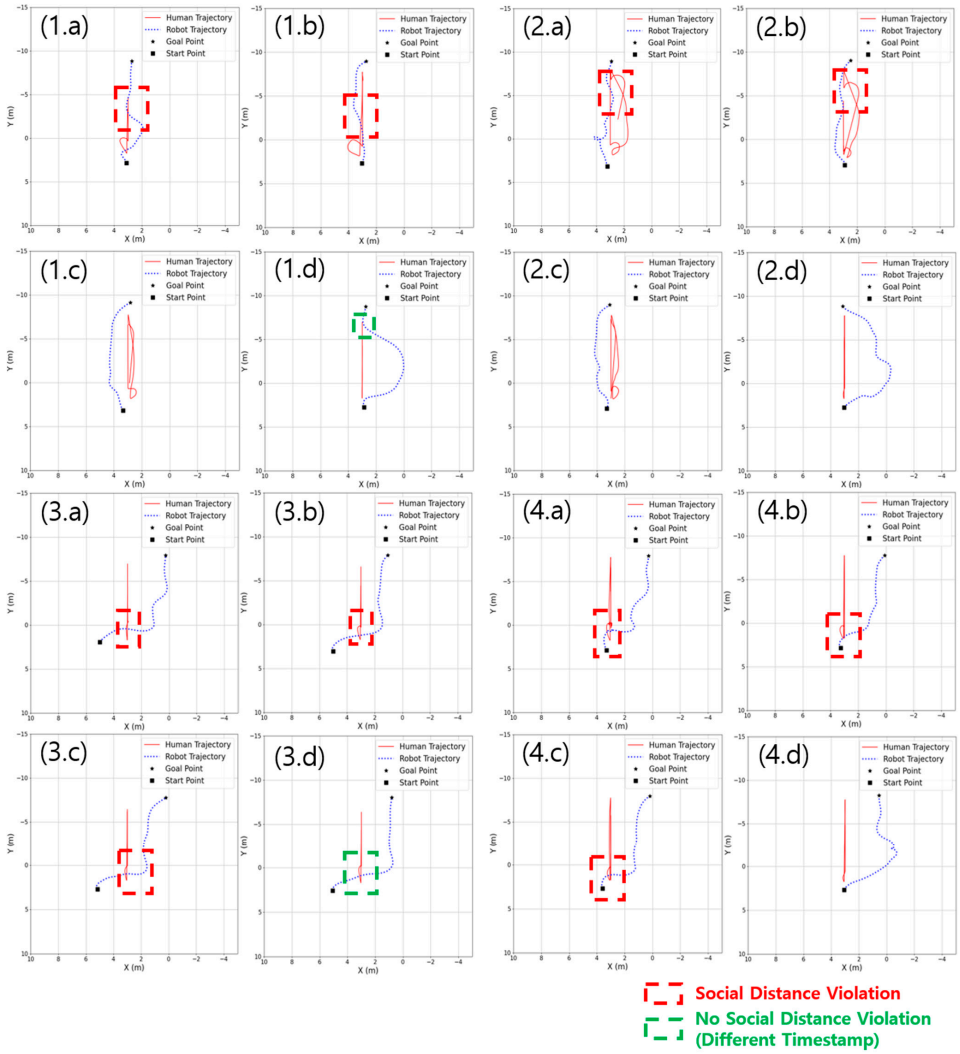

Figure 8. Trajectory results for Human1. The labels (**1**–**4**) indicate different experimental conditions with different human velocity, robot initial position, and goal point. In each condition, (**a**–**c**) represent the results of D*, A*, and Dijkstra, respectively, and (**d**) represents the result of the proposed method. Red dashed boxes indicate social distance violation, while green dashed boxes indicate no social distance violation at the same timestamp.

Figure 9 shows the results for Human2 with a different trajectory from Human1. Human2 has an initial position (9, 6) and a goal point (4, −2). The human repeatedly moves between these two points. Similarly, the trajectory results of the existing methods do not consider human behavior patterns and may generate potentially colliding paths when planning the overall path. By contrast, the trajectory results of the proposed method generate efficient socially aware paths that avoid people in advance by the added long-term probabilistic human trajectory costmap, unlike the trajectories of existing methods.

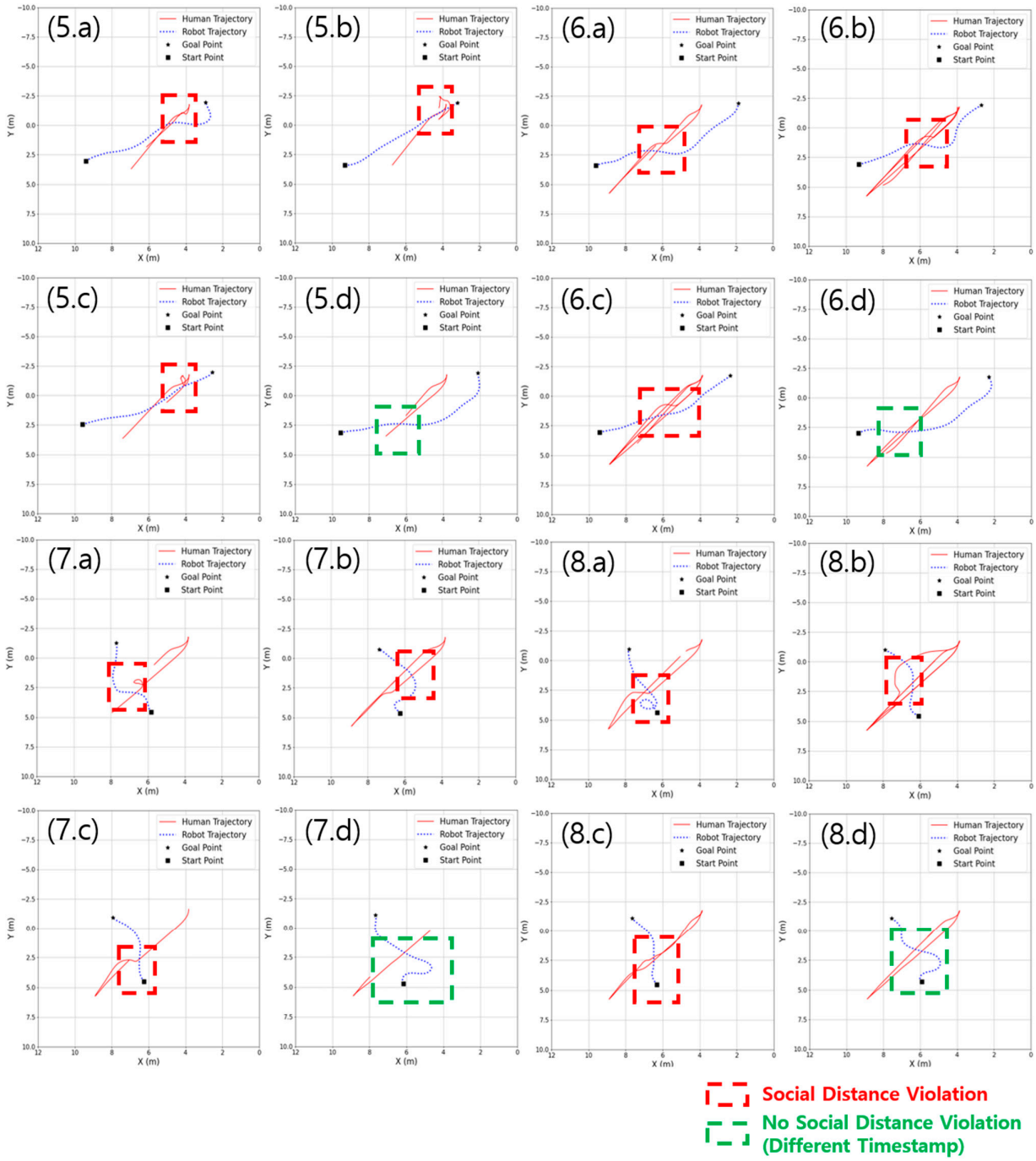

Figure 9. Trajectory results for Human2. The labels (**5**–**8**) indicate different experimental conditions with different human velocity, robot initial position, and goal point. In each condition, (**a**–**c**) represent the results of D*, A*, and Dijkstra, respectively, and (**d**) represents the result of the proposed method. Red dashed boxes indicate social distance violation, while green dashed boxes indicate no social distance violation at the same timestamp.

4.1.4. Comparison Results

The quantitative comparison results for Figures 8 and 9 are shown in Tables 2 and 3, respectively. In Tables 2 and 3, an asterisk (*) indicates the best result for each criterion. The comparison is based on three criteria: social distance violation time, navigation time, and average human–robot distance. Table 2 shows the quantitative comparison results for Human1, and Table 3 shows the quantitative comparison results for Human2. In Table 2, the proposed method shows a small social distance violation in only one of the four situations, while the existing methods all show high social distance violation times. In Table 3, the proposed method showed no social distance violation in all four situations. To summarize, the overall results show that the proposed method enables efficient socially aware robot navigation by reducing social distance violations and maintaining a larger average human–robot distance.

Table 2. Simulation comparison results for Human1.

	SD Violation Time (s)	Navigation Time (s)	Average Human–Robot Distance
D*, DWA	6.633	35.443	2.723
A*, DWA	3.755	40.510	2.578
Dijkstra, DWA	1.450	42.793	2.503
Proposed	0.098 *	31.995 *	3.608 *

SD (social distance) violation time—Total time a robot violated a person's private space that it should have kept away from to respect the person. Navigation time—Total navigation time. Average human–robot distance—The overall average distance between the human and robot, showing how far the robot moved away from the human. * Best result.

Table 3. Simulation comparison results for Human2.

	SD Violation Time (s)	Navigation Time (s)	Average Human–Robot Distance
D*, DWA	2.220	19.905 *	3.123
A*, DWA	2.110	25.058	3.258
Dijkstra, DWA	2.960	21.133	3.125
Proposed	0.0 *	20.543	3.873 *

SD (social distance) violation time—Total time a robot violated a person's private space that it should have kept away from to respect the person. Navigation time—Total navigation time. Average human–robot distance—The overall average distance between the human and robot, showing how far the robot moved away from the human. * Best result.

4.2. Experimental Implementation Validation

This experiment verifies whether the proposed costmaps can be generated and integrated into the ROS navigation framework in a real-world environment. The map used in the experimental environment is shown in Figure 10. The experiments were conducted using a robot with sensors attached, as shown in Figure 1b.

The result in RVIZ (ROS Visualization Tool), which uses object detection and depth values to estimate the position of a person and adds a social distance costmap, is shown in Figure 11a. In Figure 11a, the red arrow is the goal point, and the blue line is the planned path. The green lines in the added circular social distance costmap are the dynamic object data detected by LiDAR. This shows that the position of the person has been accurately estimated and a costmap has been added based on its center. The result of applying the estimated human position information and the head pose information estimated by the head pose estimation algorithm to the long-term probabilistic trajectory costmap in RVIZ is shown in Figure 11b. Also, the green line in the added long-term probabilistic trajectory costmap is the dynamic object data detected by LiDAR. From this, a social distance costmap and a long-term probabilistic trajectory costmap with a Gaussian distribution function are successfully added.

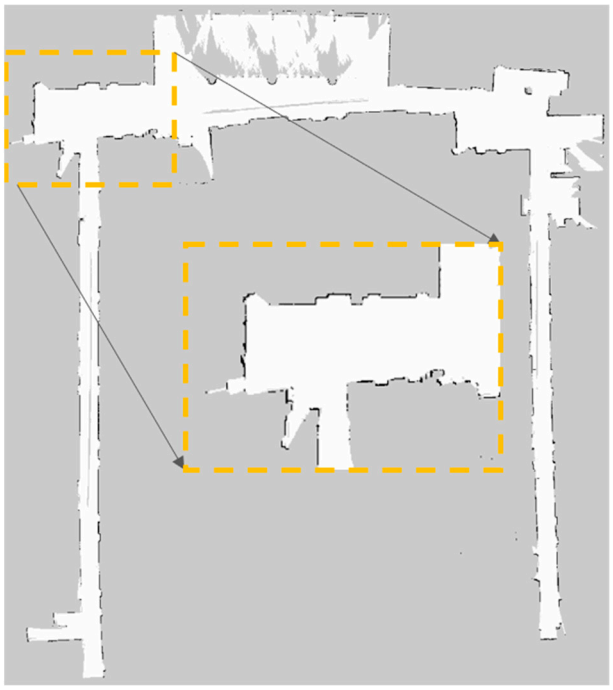

Figure 10. The map used for experimental validation. The box indicates the area where the experimental validation was conducted.

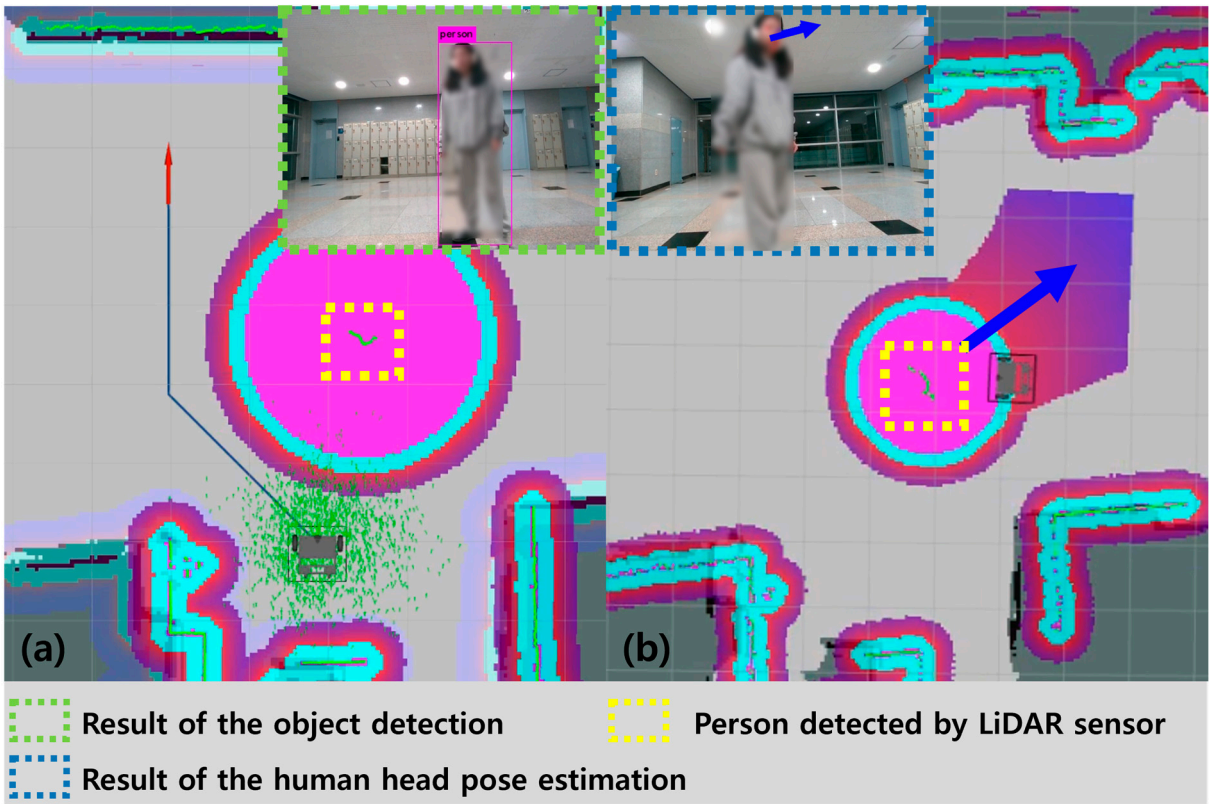

Figure 11. Experimental validation results of the proposed method: (**a**) Social distance costmap added by object detection and human velocity, (**b**) social distance costmap and long-term probabilistic trajectory costmap added by human behavior patterns. The blue arrow indicates the estimated head pose direction.

5. Conclusions

In this paper, human information as a dynamic object, including speed and long-term probabilistic human trajectory estimation, was estimated and added in the form of costmaps to enable efficient socially aware robot navigation. First, the social distance costmap is added to respect human personal space. This allows for efficient navigation even if the human trajectory is not estimated. In addition to the basic social distance value, the speed of the person is estimated so that the faster the person approaches the robot, the more efficient the path can be planned to avoid the person. To reflect human behavior patterns, the head pose angle obtained using the head pose estimation algorithm is applied in the form of a long-term probabilistic human trajectory costmap. Through long-term probabilistic human trajectory estimation, the cost values are weighted according to the distance from the human. The comparison of the proposed method and the existing methods in the simulation environment under each condition, including human trajectory, robot initial position, goal point, and human velocity, shows that the proposed method performs efficient socially aware robot navigation by considering humans in advance and reducing social distance violations compared with existing methods. The proposed costmaps were also implemented and validated in an experimental environment. However, the proposed method has several limitations. First, the current validation was conducted in environments with one or two dynamic objects, and the performance in highly crowded scenes with dense human interactions has not yet been fully evaluated. Second, the head pose cue may become unreliable when a person looks away from the actual walking direction, turns their head during social interaction, or is observed from the back or side. Third, the proposed framework depends on 2D LiDAR and RGB-D camera sensing; therefore, sensor occlusion, poor facial landmark detection, or depth measurement failure may reduce the reliability of the generated costmaps. Future work will address these limitations by extending the proposed method to crowded environments, improving robustness to unusual human behavior, evaluating the system under sensor uncertainty and failure cases, conducting component-level ablation studies on the social distance costmap, human-speed-based expansion, and head-pose-based trajectory costmap, and performing quantitative analyses of control smoothness, module-wise execution time, latency, and real-time update frequency.

Author Contributions: Conceptualization, S.K. (Seokjin Kang), S.K. (Suhyeon Kang) and H.L.; methodology, S.K. (Seokjin Kang) and S.K. (Suhyeon Kang); software, S.K. (Seokjin Kang) and S.K. (Suhyeon Kang); validation, S.K. (Seokjin Kang) and S.K. (Suhyeon Kang); formal analysis, S.K. (Seokjin Kang) and S.K. (Suhyeon Kang); investigation, S.K. (Seokjin Kang) and S.K. (Suhyeon Kang); resources, H.L.; data curation, S.K. (Seokjin Kang) and S.K. (Suhyeon Kang); writing—original draft preparation, S.K. (Seokjin Kang) and S.K. (Suhyeon Kang); writing—review and editing, S.K. (Seokjin Kang), S.K. (Suhyeon Kang) and H.L.; visualization, S.K. (Seokjin Kang) and S.K. (Suhyeon Kang); supervision, H.L.; project administration, H.L.; funding acquisition, H.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the Institute of Information Communications Technology Planning Evaluation (IITP)-Innovative Human Resource Development for Local Intellectualization program grant funded by the Korea government (MSIT) (IITP-2026-RS-2020-II201612, 50%), and in part by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2026-RS-2024-00438430, 50%) supervised by the IITP (Institute for Information Communications Technology Planning Evaluation).

Data Availability Statement: The data presented in this study are available from the corresponding author upon reasonable request.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ROS	Robot Operating System
RGB-D	Red, Green, Blue-Depth
LiDAR	Light Detection and Ranging
DWA	Dynamic Window Approach
MPC	Model Predictive Control
SLAM	Simultaneous Localization and Mapping
PnP	Perspective-n-Point
MHE	Moving Horizon Estimation
RVIZ	ROS Visualization Tool

References

1. Liu, Y.; Wang, S.; Xie, Y.; Xiong, T.; Wu, M. A Review of Sensing Technologies for Indoor Autonomous Mobile Robots. *Sensors* **2024**, *24*, 1222. [CrossRef] [PubMed]
2. Ullah, I.; Adhikari, D.; Khan, H.; Anwar, M.S.; Ahmad, S.; Bai, X. Mobile Robot Localization: Current Challenges and Future Prospective. *Comput. Sci. Rev.* **2024**, *53*, 100651. [CrossRef]
3. Arpitha Shankar, S.I.; Shivakumar, M. Sensor Fusion Based Multiple Robot Navigation in an Indoor Environment. *Int. J. Interact. Des. Manuf.* **2024**, *18*, 4841–4852. [CrossRef]
4. Semborski, J.; Idzkowski, A. A Review on Positioning Techniques of Mobile Robots. *Robot. Syst. Appl.* **2024**, *4*, 30–43. [CrossRef]
5. Hart, P.E.; Nilsson, N.J.; Raphael, B. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Trans. Syst. Sci. Cybern.* **1968**, *4*, 100–107. [CrossRef]
6. Dijkstra, E.W. A Note on Two Problems in Connexion with Graphs. In *Edsger Wybe Dijkstra: His Life, Work, and Legacy*; Association for Computing Machinery: New York, NY, USA, 2022; pp. 287–290. [CrossRef]
7. Stentz, A. Optimal and Efficient Path Planning for Partially-Known Environments. In Proceedings of the 1994 IEEE International Conference on Robotics and Automation, San Diego, CA, USA, 8–13 May 1994; pp. 3310–3317. [CrossRef]
8. Fox, D.; Burgard, W.; Thrun, S. The Dynamic Window Approach to Collision Avoidance. *IEEE Robot. Autom. Mag.* **1997**, *4*, 23–33. [CrossRef]
9. Mayne, D.Q.; Rawlings, J.B.; Rao, C.V.; Scokaert, P.O.M. Constrained Model Predictive Control: Stability and Optimality. *Automatica* **2000**, *36*, 789–814. [CrossRef]
10. Jeong, H.; Lee, H. CNN-Based Fault Detection of Scan Matching for Accurate SLAM in Dynamic Environments. *Sensors* **2023**, *23*, 2940. [CrossRef] [PubMed]
11. Kang, S.; Kim, D.; Lee, H. A Combinational Approach to Robust Robot Localization in Dynamic Environments. In Proceedings of the 2nd International Conference on Maritime IT Convergence (ICMIC 2023), Jeju Island, Korea, 23–25 August 2023; pp. 181–182.
12. Woo, J.; Jeong, H.; Lee, H. Comparison and Analysis of LIDAR-Based SLAM Frameworks in Dynamic Environments with Moving Objects. In Proceedings of the 2021 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia), Gangwon, Korea, 1–3 November 2021; pp. 1–4. [CrossRef]
13. Kang, S.; Lee, H. Robust Robot Localization with Visually Adaptive Consensus Filters in Dynamic Corridor Environments. *Eng. Sci. Technol. Int. J.* **2025**, *64*, 101998. [CrossRef]
14. Kim, D.; Lee, H.; Choi, K.H. Pitch Variation Filter for LiDAR-Only SLAM and Localization in Self-Balancing Mobile Robot. *IEEE Robot. Autom. Lett.* **2025**, *10*, 12836–12843. [CrossRef]
15. Riaz, W.; Ji, J.; Ullah, A. TriViT-Lite: A Compact Vision Transformer–MobileNet Model with Texture-Aware Attention for Real-Time Facial Emotion Recognition in Healthcare. *Electronics* **2025**, *14*, 3256. [CrossRef]
16. Ullah, A.; Song, Z.; Riaz, W.; Qi, X.; Hossain, M.M. Hand-Gesture-Based Biometric Verification and Identification Using Embedded-STQNet Deep Neural Architecture in Security-Oriented Systems. *IEEE Internet Things J.* **2026**, *13*, 24561–24574. [CrossRef]
17. Zhao, X.; Wang, T.; Li, Y.; Zhang, B.; Liu, K.; Liu, D.; Wang, C.; Snoussi, H. Target-Driven Visual Navigation by Using Causal Intervention. *IEEE Trans. Intell. Veh.* **2024**, *9*, 1294–1304. [CrossRef]
18. Zhou, L.; Li, Z.; Li, Y.; Bai, S. Parallel MPPI with Gradient-Velocity Modulated SDF Cost for High-Performance Real-Time Dynamic Obstacle Avoidance by Robot Manipulators. *IEEE Trans. Robot.* **2025**, *41*, 5149–5168. [CrossRef]
19. He, D.; Lin, Y.; Dai, Z.; Yang, S.X. Neurodynamics-Based Visual Servo Predictive Control for Improving Smooth Movement of Logistics Omnidirectional Robots. *IEEE Trans. Ind. Electron.* **2025**, *72*, 14646–14655. [CrossRef]

20. Wahab, N.A.; Mahmud, M.S.A.; Alqaraghuli, H.; Samsuria, E.; Romdlony, M.Z. Efficient Autonomous Navigation in Dynamic Environments: Algorithm Evaluation and Multi-Robot Coordination. In Proceedings of the 2024 IEEE International Conference on Automatic Control and Intelligent Systems (I2CACIS), Shah Alam, Malaysia, 29 June 2024; pp. 433–438. [CrossRef]
21. Huang, Y.; Ding, H.; Zhang, Y.; Wang, H.; Cao, D.; Xu, N.; Hu, C. A Motion Planning and Tracking Framework for Autonomous Vehicles Based on Artificial Potential Field Elaborated Resistance Network Approach. *IEEE Trans. Ind. Electron.* **2020**, *67*, 1376–1386. [CrossRef]
22. He, N.; Su, Y.; Guo, J.; Fan, X.; Liu, Z.; Wang, B. Dynamic Path Planning of Mobile Robot Based on Artificial Potential Field. In Proceedings of the 2020 International Conference on Intelligent Computing and Human-Computer Interaction (ICHCI), Sanya, China, 4–6 December 2020; pp. 259–264. [CrossRef]
23. Chiang, C.-H.; Ding, C. Robot Navigation in Dynamic Environments Using Fuzzy Logic and Trajectory Prediction Table. In Proceedings of the 2014 International Conference on Fuzzy Theory and Its Applications (iFUZZY2014), Kaohsiung, Taiwan, 26–28 November 2014; pp. 99–104. [CrossRef]
24. Faisal, M.; Hedjar, R.; Al Sulaiman, M.; Al-Mutib, K. Fuzzy Logic Navigation and Obstacle Avoidance by a Mobile Robot in an Unknown Dynamic Environment. *Int. J. Adv. Robot. Syst.* **2013**, *10*, 37. [CrossRef] [PubMed]
25. Chen, Y.F.; Everett, M.; Liu, M.; How, J.P. Socially Aware Motion Planning with Deep Reinforcement Learning. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 1343–1350. [CrossRef]
26. Do, N.T.; Pham, T.D.; Son, N.H.; Ngo, T.D.; Truong, X.T. Deep Reinforcement Learning Based Socially Aware Mobile Robot Navigation Framework. In Proceedings of the 2020 7th NAFOSTED Conference on Information and Computer Science (NICS), Ho Chi Minh City, Vietnam, 26–27 November 2020; pp. 226–231. [CrossRef]
27. Yao, S.; Chen, G.; Qiu, Q.; Ma, J.; Chen, X.; Ji, J. Crowd-Aware Robot Navigation for Pedestrians with Multiple Collision Avoidance Strategies via Map-Based Deep Reinforcement Learning. In Proceedings of the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Prague, Czech Republic, 27 September–1 October 2021; pp. 8144–8150. [CrossRef]
28. Samsani, S.S.; Muhammad, M.S. Socially Compliant Robot Navigation in Crowded Environment by Human Behavior Resemblance Using Deep Reinforcement Learning. *IEEE Robot. Autom. Lett.* **2021**, *6*, 5223–5230. [CrossRef]
29. Kim, M.; Kwon, Y.; Yoon, S.-E. Group Estimation for Social Robot Navigation in Crowded Environments. In Proceedings of the 2022 22nd International Conference on Control, Automation and Systems (ICCAS), Busan, Korea, 27 November–1 December 2022; pp. 1421–1425. [CrossRef]
30. Hoang, V.B.; Nguyen, V.H.; Ngo, T.D.; Truong, X.-T. Socially Aware Robot Navigation Framework: Where and How to Approach People in Dynamic Social Environments. *IEEE Trans. Autom. Sci. Eng.* **2023**, *20*, 1322–1336. [CrossRef]