2024 22nd ACM-IEEE International Symposium on Formal Methods and Models for System Design (MEMOCODE) | 979-8-3503-7802-3/24/$31.00 ©2024 IEEE | DOI: 10.1109/MEMOCODE63347.2024.00021

Perception-based Runtime Monitoring and Verification for Human-Robot Construction Systems

Apala Pramanik
School of Computing
University of Nebraska
Lincoln, NE, United States
apramanik2@huskers.unl.edu

Sung Woo Choi
School of Computing
University of Nebraska
Lincoln, NE, United States
schoi9@huskers.unl.edu

Yuntao Li
School of Computing
University of Nebraska
Lincoln, NE, United States
yli2@huskers.unl.edu

Luan Viet Nguyen
Department of Computer Science
University of Dayton
Dayton, OH, United States
lnguyen1@udayton.edu

Kyungki Kim
Durham School of Architectural Engineering and Construction
University of Nebraska
Lincoln, NE, United States
kkim3@unl.edu

Hoang-Dung Tran
School of Computing
University of Nebraska
Lincoln, NE, United States
dtran30@unl.edu

***Abstract*—The rising use of robots in construction aims to ease labor-intensive and hazardous tasks. Ensuring safety in human-robot collaboration at construction sites is crucial, necessitating robust safety protocols and smooth interaction. This work aims to develop an open-source framework for monitoring and verifying safety in construction scenarios involving humans and robots. Our proposed framework includes the co-design of two modules: runtime monitoring against Signal Temporal Logic (STL) requirements and real-time reachability analysis using ProbStar. The runtime monitoring module effectively detects, localizes, and predicts human movements within the robot's operational field. By employing a Kalman filter, we accurately estimate the future paths of workers, which facilitates proactive monitoring of worker safety. This approach enables dynamic adjustments to the robot's trajectory, guided by quantitatively calculating robustness values of STL specifications in real-time. Our approach leverages real-time data from an RGB-D camera to promptly identify any deviations from expected behavior, further enhancing safety measures. To address uncertainties in localization that make the monitoring results inconclusive for safety judgments, the verification module employs real-time probabilistic reachability analysis to evaluate the likelihood of collisions between robots and obstacles within the robot's local view. We evaluate the proposed framework across various human-robot interaction scenarios at construction sites.**

***Index Terms*—Human-Robot collaboration, Construction sites, Signal Temporal Logic, Reachability Analysis, ProbStar**

I. Introduction

The construction industry faces persistent challenges related to productivity, worker safety, and health due to the various hazards and the ever-changing nature of its tasks. Robots, typically designed for structured environments, struggle to adapt to the complexities of unstructured construction sites. As a result, fully autonomous robots replacing humans may not be the most practical solution. Previous research [1]–[3] has explored human-robot collaboration for hazardous and repetitive tasks such as painting, wood framing, and welding. However, reports of worker discomfort when working alongside robots [4] have prompted further studies to better understand the interaction dynamics between workers and robotic systems in various environments [5]–[8]. This complexity underscores the critical need for robust monitoring and verification methods to ensure the safety and efficiency of autonomous robots in construction settings. There is a lack of perception-based, real-time monitoring and verification for robotics, which is essential for understanding how the presentation and training of a robot's internal state impact human attitudes towards robots in real-world scenarios.

This paper introduces perception-based runtime monitoring and collision verification algorithms for human-robot construction systems. Our algorithms address the challenges of deploying robots at a construction site through its three core components: *Localization*; *Predict and Monitor*; and *ProbStar Reachability*, as illustrated in Fig. 1.

Localization. Monitoring human-robot collaboration systems in real-time using Signal Temporal Logic (STL) specifications presents significant challenges. It requires the robot to avoid the human, demanding precise knowledge of both the robot's and the human's locations. Robots typically rely on camera and LiDAR sensors to detect obstacles within their local views. In the localization component, we estimate the coordinates of a human by localizing workers within the robot's local view and processing point cloud data obtained from the robot's RGB-D camera.

Predict and Monitor. After localization, we focus on prediction and monitoring. A major challenge in monitoring STL specifications that involve future temporal logic operators is that the calculation of robustness at time t depends on inputs at a future time $t' > t$, which are not yet available. To tackle this challenge, we utilize a Kalman filter to estimate the future trajectories of workers. This method allows us to monitor worker safety and dynamically adjust the robot's path according to the robustness values derived from future temporal logic-based STL specifications in real time. Importantly, in our case study involving multiple robot systems, while each robot

2832-6520/24/$31.00 ©2024 IEEE
DOI 10.1109/MEMOCODE63347.2024.00021

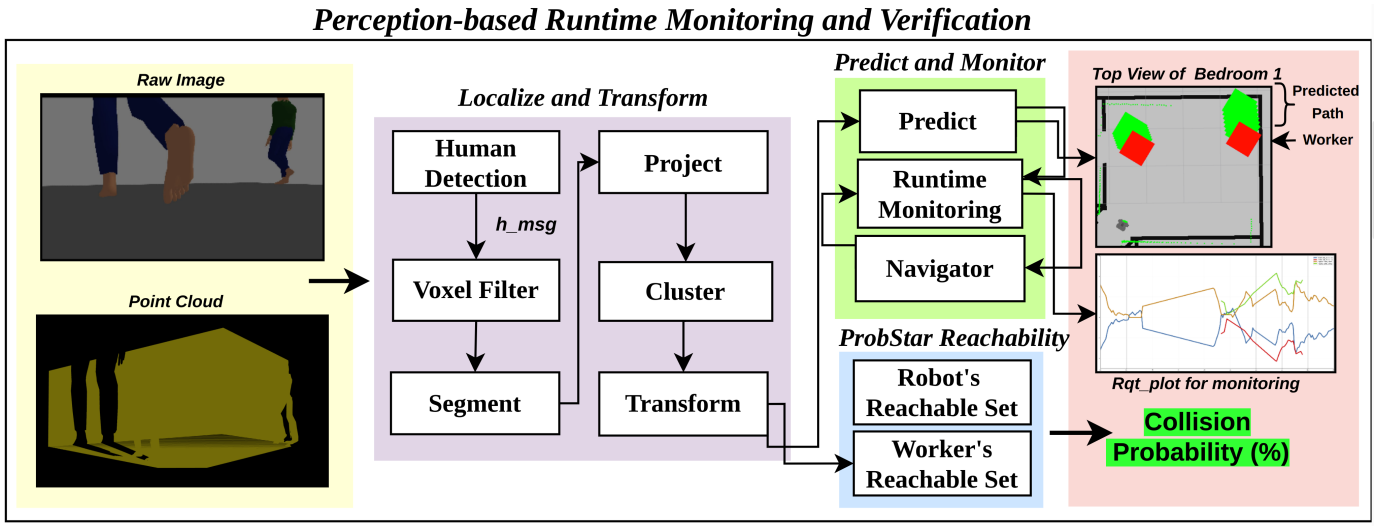


Fig. 1: Overview of the proposed vision-based runtime monitoring and verification algorithm.

localizes humans within its immediate vicinity, it does not localize other robots. Instead, their coordinates are obtained from a global perspective.

ProbStar Reachability. Despite employing a Kalman filter for estimation, uncertainties in the positions of humans and robots are inevitable in dynamic and unstructured environments due to occlusions, sensor noise, and human unpredictability. These uncertainties can result in incorrect monitoring outcomes, particularly when computed robustness values fluctuate within a narrow range of positive and negative values. To bolster safety justification under these conditions, we conduct probabilistic reachability and collision analysis for a human-robot construction system, modeled as a linear system with inputs.

Our approach quantitatively verifies the collision probability of the robot at runtime using the recent probabilistic star set [9] built on a set representation named probabilistic star (or ProbStar) which is a variant of the well-known star set used in DNNs [10]. Mathematically, a ProbStar [11] is an affine mapping of a truncated multivariate Gaussian distribution. We model the inputs as a ProbStar and propagate them through the control system to construct the reachable output set, which contains multiple ProbStars. Note that we perform real-time computation of reachable sets for the robot and workers to determine the collision probability. This analysis quantitatively verifies the risk and informs decision-making to avoid collisions.

To evaluate our framework, we developed two case studies that simulate real-life construction site scenarios. In Case Study 1, we assess runtime monitoring as a robot navigates through five different locations within an apartment, encountering 1-3 human workers along the way. For each scenario, we tabulate the total navigation time, the minimum robustness value of STL specifications, and the Kalman filter prediction errors. In Case Study 2, we evaluate the collision verification module with three deployed robots. One robot navigates to two different locations, encountering both the other robots and a human worker. For each robot encounter, we report the collision probabilities under potential collision conditions. Finally, we evaluate the effectiveness of our co-design monitoring/verification strategy by analyzing the reduction in overall reachability analysis time and reporting the average processing times for each algorithm.

Our Contribution. In this paper, we introduce a framework designed to monitor and ensure safety requirements for autonomous human-robot collaboration systems. Utilizing real-time robot perception, this framework dynamically approximates human positions to prevent collisions proactively. Below we summarize the key contributions of the paper:

- We propose the first perception-based algorithm designed to localize and estimate human motion within a robot's local view, using these predictions to monitor human-robot systems' safety in real-time actively.
- We integrate monitoring with probabilistic reachability and collision analysis in real-time using ProbStar for multiple robots at runtime under different scenarios.
- We propose the implementation of Kalman filter prediction using ProbStar representation.
- We perform an extensive evaluation to validate the scalability and robustness of the proposed monitoring and reachability analysis algorithm.
- We implement the algorithm and publish it as an early prototype ROS-integrable package that can be used for other robotic applications. [1]

Paper Organization. The remainder of the paper is organized as follows: Section II introduces the preliminaries and outlines the problems addressed in this study. Sections III, IV, and V provide a comprehensive overview of our proposed algorithms to tackle these problems. Section VI describes the experimental setup and presents the results for each module of our algorithm. Section VII reviews related works in the field. Finally, we present our concluding remarks in Section VIII.

[1] https://github.com/apalapramanik/didactic-waffle

II. Preliminaries and Problem Formulation

Our algorithm involves the co-design of two modules: runtime monitoring against STL specifications and real-time reachability analysis using ProbStar. We start by defining the preliminary concepts for each module and outline the specific problems we aim to address for each within this paper.

A. STL Monitoring Preliminaries

*Definition 1 (**STL Syntax**):* Consider an STL requirement interpreted over discrete time using future logic operators. Let $X = \{x_1, \ldots, x_n\}$ be a set of real-valued variables. A valuation $v : X \to \mathbb{R}$ for $x \in X$ maps a variable x to a real value. A signal w defined over X is a function $T \to \mathbb{R}^X$ that gives the value $w(t)$ of the variables in X at time $t \in T$, where $T = \mathbb{R}^{\geq 0}$. The syntax of STL formula φ is defined as the following grammar:

$$\varphi := \top \mid f(Y) > c \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \mathcal{U}_I \varphi_2$$

where $Y \subseteq X$, $f : \mathbb{R}^Y \to \mathbb{R}$, $c \in \mathbb{R}$, and I is the time interval of the form [a, b] or [a, ∞) such that a, b $\in T$ and a $\leq$ b.

*Definition 2 (**STL Quantitative Semantics**):* Given a real valued function ρ of an STL formula φ, a signal w and a time t, the STL *quantitative (or robustness)* semantics is recursively defined as follows [12]:

$$\rho(\varphi, w, t) \geq 0 \iff (w, t) \vDash \varphi,$$

$$\rho(f(Y), w, t) = f(w_Y(t)) - c,$$

$$\rho(\neg\varphi, w, t) = \neg\rho(\varphi, w, t),$$

$$\rho(\varphi_1 \vee \varphi_2, w, t) = \max(\rho(\varphi_1, w, t), \rho(\varphi_2, w, t)),$$

$$\rho(\varphi_1 \wedge \varphi_2, w, t) = \min(\rho(\varphi_1, w, t), \rho(\varphi_2, w, t)),$$

$$\rho(\varphi_1 \mathcal{U} \varphi_2, w, t) = \sup_{t' \in [t \oplus I]} \min\left(\inf_{t'' \in [t, t']}^{\rho(\varphi_2, w, t')} \rho(\varphi_1, w, t'')\right),$$

where $\oplus$ denotes the Minkowski sum between t and I.

Temporal operators. The syntactic definition of STL is minimal and includes only basic operators. We can derive other standard temporal operators as follows:

eventually	$\Diamond_I \varphi$	$\equiv$	$\top\ \mathcal{U}_I\ \varphi$
always	$\Box_I \varphi$	$\equiv$	$\neg\Diamond_I \neg\varphi$
implies	$\varphi_1 \to \varphi_2$	$\equiv$	$\neg\varphi_1 \vee \varphi_2$

Intuitively, a signal w satisfies a formula $\varphi_1 \mathcal{U}_I \varphi_2$ at time t if there exists a time $t \in [t + a, t + b]$ such that w satisfies φ_2 and for all the times before then w satisfies φ_1.

Problem 1: (*Perception-based Runtime Monitoring*) Given an STL formula φ, a signal w over N observed time steps and t_0 is initial time, we require that $\rho(\varphi, w, t) \geq 0\ \forall\ t \in [t_0, t_0 + N]$. For a human-robot construction system, the objective is to detect and localize human workers using pointcloud data in the robot's local frame of reference and estimate their future trajectory $\hat{w}$ over a prediction of t_{pred} time step. We aim to ensure collision avoidance while monitoring the safety of the human worker by computing the quantitative robustness $\rho(\varphi, w, t)\ \forall\ t \in [t_0, t_0 + t_{pred}]$, for formula φ.

Uncertainties and noise can lead to localization and estimation errors, potentially causing incorrect monitoring outcomes. To mitigate these issues and enhance safety assessments, the next section introduces probabilistic reachability analysis along with the related problem statement.

B. Probstar Reachability Preliminaries

*Definition 3 (**Probabilistic Star Set (or ProbStar)**):* A probabilistic star (or simply ProbStar) Θ is a tuple $\langle c, V, \mathcal{N}, P, l, u\rangle$ where $c \in \mathbb{R}^n$ is the center, $V = \{v_1, v_2, \ldots, v_m\}$ is a set of m vectors in $\mathbb{R}^n$ called basis vectors, $P : \mathbb{R}^m \to \{\top, \bot\}$ is a predicate, l and u are the lower-bound and upper-bound vectors of the predicate variables, which are random variables of a Gaussian distribution $\mathcal{N}$. We restrict the predicates to be a conjunction of linear constraints, $P(\alpha) \triangleq C\alpha \leq d$ where, for p linear constraints, $C \in \mathbb{R}^{p \times m}$, α is the vector of m-variables, i.e. $\alpha = [\alpha_1, ..., \alpha_m]^T$, and $d \in \mathbb{R}^{p \times m}$.

The basis vectors are arranged to form the ProbStar's $n \times m$ basis matrix. The set of states represented by the ProbStar is given as:

$$[\![\Theta]\!] = \Big\{x \mid x = c + \sum_{i=1}^{m} (\alpha_i v_i), \alpha \sim \mathcal{N}, P(\alpha) \triangleq C\alpha \leq d, l_i \leq \alpha_i \leq u_i\Big\}$$

*Definition 4 (**Probability**):* Given a ProbStar Θ, the probability of the ProbStar is the probability of the predicate random variables $\alpha = [\alpha_1, \alpha_2, \ldots, \alpha_m]$ satisfying its constraints and bounds, i.e., $\mathcal{P}(\Theta) = \mathcal{P}(C\alpha \leq d \wedge l \leq \alpha \leq u, \alpha \sim \mathcal{N}(\mu, \Sigma))$, where $\mathcal{N}$ is a Gaussian distribution with mean μ and variance $\sum$. A ProbStar is an empty set if its probability is zero, i.e., $\mathcal{P}(\Theta) = 0$.

*Proposition 1 (**Affine Mapping**):* Given a ProbStar set $\Theta = \langle c, V, \mathcal{N}, P, l, u\rangle$, an affine mapping of the ProbStar Θ with the mapping matrix A and offset vector b defined by $\bar{\Theta} = \{y | y = Ax + b, x \in \Theta\}$ is another ProbStar with the following characteristics:$\bar{\Theta} = \langle \bar{c}, \bar{V}, \bar{\mathcal{N}}, \bar{P}, \bar{l}, \bar{u}\rangle, \bar{c} = Ac + b, \bar{V} = \{Av_1, Av_2, \ldots, Av_m\}, \bar{\mathcal{N}} = \mathcal{N}, \quad \bar{P} \equiv P, \quad \bar{l} \equiv l, \quad \bar{u} \equiv u.$

*Proposition 2 (**Intersection**):* The intersection of a ProbStar $\Theta \triangleq \langle c, V, \mathcal{N}, P, l, u\rangle$ and a half-space $\mathcal{H} \triangleq \{x | Hx \leq g\}$ is another ProbStar with the following characteristics: $\bar{\Theta} = \Theta \cap \mathcal{H} = \langle \bar{c}, \bar{V}, \bar{\mathcal{N}}, \bar{P}, \bar{l}, \bar{u}\rangle$, where $\bar{c} = c$, $\bar{V} = V$, $\bar{\mathcal{N}} = \mathcal{N}$, $\bar{P} = P \wedge P', P'(\alpha) \triangleq (H \times V_m)\alpha \leq g - H \times c, V_m = [v_1 v_2 \ldots v_m]$, $\bar{l} = l$, $\bar{u} = u$

*Proposition 3 (**Kalman Filter using ProbStar**):* Given a ProbStar set, a Kalman filter [13] reachable set is the affine mapping of ProbStar Θ with the affine matrix $A = I_4 - KM$ and the offset vector $b = Kz$, where I_4 is a 4×4 identity matrix, K is the Kalman gain, M is the measurement matrix, and z is the measurement vector.

Problem 2: (*Perception-based Real-time Collision Verification*) Given the initial state of the robot from odometry and initial state of the worker in the robot's local view, the task is to generate an initial ProbStar set Θ_r for the robot and Θ_w for

the human worker. From these initial sets, compute k reachable ProbStar sets $[\![\Theta_r^k]\!]$ and $[\![\Theta_w^k]\!]$ for the robot and human worker, respectively. Verify the probability of collision based on robot's perception by checking the intersection (proposition 2) of these reachable sets, i.e., $\mathcal{P}_{\text{collision}}(i) = \mathcal{P}(\Theta_r^i \cap \Theta_w^i \neq \emptyset)$ from Definition 4.

III. Localization

This section explores the localization of humans in a robot's local view using point cloud data. Point clouds are a collection of data points in a 3D coordinate system that represent an object's surface, making them invaluable for tasks such as obstacle avoidance and human tracking. By processing point cloud data, robots can effectively perceive and localize humans, enabling safer and more efficient interactions.

In Algorithm 1, we use the YOLOv5 model [14] for human detection. When a human is detected, the flag h_msg triggers cloud processing steps, including point cloud filtering, segmentation, projection, clustering, and transformation. The depth camera captures point cloud data μ, which is downsampled using a 3D voxel grid filter (line 2) to create ds_pcl. The ground plane and background are removed using plane segmentation with the *SAC-MODEL PLANE* model and *SAC RANSAC* method [15], resulting in pl_pcl (line 3). The planeless cloud pl_pcl is projected onto the ground to generate $proj\_pcl$ (line 4). Using *DBSCAN* [16], the human's point cloud is clustered (line 5). The mean of the x and y coordinates of the cluster determines the worker's position $\mathcal{W}$ in the robot's local view.

Algorithm 1: Localization and Transformation

Input: Point Cloud from depth camera μ,
Yolov5 detection flag h_msg,
Robot's pose $\mathcal{R}$,
Robot's orientation quaternion q

Output: Worker's pose w.r.t robot $\mathcal{W}$

1 **if** h_msg **then**
/* Cloud Processing */
2 ds_pcl = VoxelFilter(μ)
3 pl_pcl = SACSegmentation(ds_pcl)
4 $proj\_pcl$ = Project(pl_pcl)
/* Clustering and tranformation */
5 $\mathcal{W}$ = mean(DBSCAN($proj\_pcl$))
6 $q\_rot = q_{current} \times conj(q_{prev})$
7 $\mathcal{W}'$ = Transform($q\_rot, \mathcal{W}_{current}, \mathcal{W}_{prev}$)
8 **return** $\mathcal{W}'$

Next, we transform the human's past coordinates to the robot's current frame of reference for accurate prediction while the robot is in motion (Algorithm 1, line 6-7). Consider the robot at point $O_1 = (o_{1_x}, o_{1_y}, o_{1_z})$ having quaternion $Q_1 = (x_1, y_1, z_1, w_1)$ at time t_1. Let $\mathcal{W}_1 = (\mathcal{W}_{1_x}, \mathcal{W}_{1_y}, \mathcal{W}_{1_z})$ be the position of the human with respect to O_1. Now, at time t_2 where $t_2 = t_1 + 1$, the new position of the robot is O_2 and the

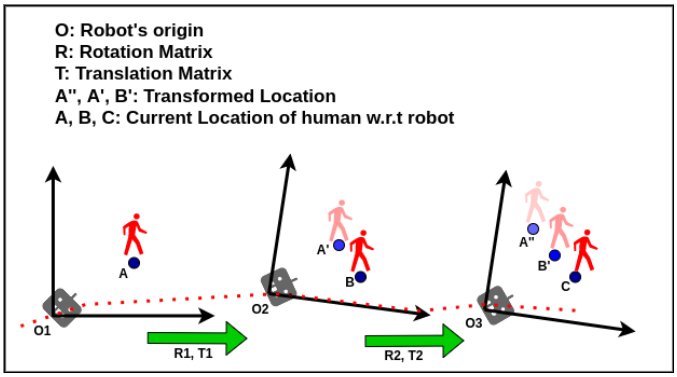


Fig. 2: Transformation of a human's location to the robot's coordinate system.

new position of the human is $\mathcal{W}_2$ w.r.t. O_2. Then, to perform prediction at time t_2, we require all the previous locations to be in the same frame of reference. To achieve that we transform the human's location $\mathcal{W}_1$ w.r.t. O_1 to $\mathcal{W}'_1$ in the new frame of reference O_2. For transforming any point $\mathcal{W}$ to $\mathcal{W}'$, the general equations are:

$$\mathcal{W}'_1 = \mathcal{W}_{rotation} + \mathcal{W}_{translation},$$

$$\mathcal{W}_{rotation} = Q_{rot} \times \mathcal{W}_1 \times conj(Q_{rot}),$$

$$\mathcal{W}_{translation} = \sqrt{(o_{2_x} - o_{1_x})^2 + (o_{2_y} - o_{1_y})^2 + (o_{2_z} - o_{1_z})^2}$$

, where $Q_{rot} = Q_2 \times conj(Q_1)$ and $conj(Q_1)$ is the conjugate of quaternion Q_1. In Fig 2 the robot traverses from O_1 to O_3 via O_2. At each location, the current position of the human in the robot's frame of reference is identified as A, B, or C, while the newly transformed locations from the previous frames are denoted as A', A'', and B'. The transformations are represented by R, which signifies the rotation quaternion, and T, which represents the translation. We use the transformed position for prediction as shown in Algorithm 2, line 2.

IV. Predict and Monitor

In this section, we discuss our algorithm for predicting human trajectory and applying runtime monitoring in detail.

A. Runtime STL Monitoring

Our approach utilizes a Kalman filter to predict human motion based on past and current positions. We have implemented these predictions into the STL-based runtime monitoring to continuously assess future states and dynamically adapt robot behavior, ensuring safe interactions. The approach is outlined in Algorithm 2. The Kalman filtering algorithm [13] leverages observed measurements to estimate unknown variables over time precisely. We apply it to predict human motion for the next t_{pred} steps (line 2). We select the number of time steps for predicting human motion, t_{pred}, to balance prediction accuracy and computational complexity for real-time performance. An alternative strategy involves dynamically adjusting t_{pred} according to the robustness of STL specifications.

Algorithm 2: Prediction and Runtime Monitoring

Input: Worker's pose w.r.t robot $\mathcal{W}'$,
Robot's pose $\mathcal{R}$,
Number of prediction time steps t_{pred},
STL formula φ,
Goal Location δ,
Yolov5 detection flag h_msg

Output: worker trajectory: $\hat{w} = \{(t_i, \hat{x}_i)\}_{i=0}^{t_{\text{pred}}}$,
Robustness value: $\rho(\varphi, \hat{w}, t)$

```
1  SetGoal(δ)
2  while goal not reached do
3      if h_msg then
4          ŵ ← KalmanPrediction(W', t_pred)
5          ρ ← (φ, ŵ, t_pred) /* section II-A */
6          if ρ < 0 then
7              Stop()
8              angle ← CalcTurnAngle(R, W')
9              if angle is not none then
10                 Turn(angle)
11             else
                   /* Goal cannot be accomplished */
12                 break
13         else
14             head_to (δ)
15     else
16         head_to (δ)
```

In Algorithm 2, we set a goal location for the robot (line 1) and start prediction and monitoring (lines 2-12) until the goal is reached. After predicting human motion using the Kalman filter (line 4), these predictions are used for robustness computation to assess potential collisions based on the STL specification φ (line 5). The robustness values for the distance between the robot and the human's future positions are computed for the next t_{pred} time steps. If robustness value (ρ) is less than zero (lines 6-12), indicating a potential safety violation, the robot stops and turns to avoid a collision. Otherwise, the robot continues moving towards the goal, continually checking robustness. When the robot stops, it adjusts its heading based on the angle calculated by $CalcTurnAngle$ (line 8), which computes the angle between the worker and the robot's current orientation. Under conditions of a head-on collision (angle of approach is 0°), the robot stops navigation completely (lines 11-12). The robot resumes navigation once robustness is positive, with a modified path due to the new heading angle.

V. ProbStar Reachability Analysis

In dynamic environments, uncertainties like unpredictable movements and localization errors pose significant safety challenges in monitoring. A probabilistic approach to calculating collision probability effectively addresses these uncertainties, providing a more robust and quantitative verification of worker safety. This section defines the fundamentals of probabilistic reachability analysis for real-time collision detection and outlines our approach in Algorithm 3.

A. Robot's System Model

First, we develop a system model for our robot, i.e., a two-wheeled differential drive robot. As shown in Fig 3, $\hat{x}_t$, $\hat{y}_t$, and $\hat{\gamma}_t$ represent the state vector x_t of the robot at time t, where $\hat{x}$ and $\hat{y}$ are the position coordinates and $\hat{\gamma}$ is the yaw angle. The system input u comprises v, i.e., the linear velocity of the robot, and ω, the angular velocity. We define the system dynamics in matrix form as:

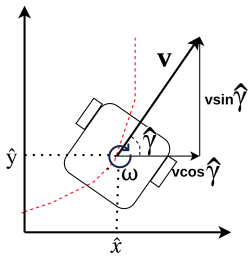


Fig. 3: Kinematic model for two-wheeled differential drive robot

$$\begin{bmatrix} \hat{x}_{t+1} \\ \hat{y}_{t+1} \\ \hat{\gamma}_{t+1} \end{bmatrix} = \begin{bmatrix} \hat{x}_t + v\cos\gamma_t \cdot dt \\ \hat{y}_t + v\sin\gamma_t \cdot dt \\ \hat{\gamma}_t + \omega \cdot dt \end{bmatrix},$$

$$\implies \begin{bmatrix} \hat{x}_{t+1} \\ \hat{y}_{t+1} \\ \hat{\gamma}_{t+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{x}_t \\ \hat{y}_t \\ \hat{\gamma}_t \end{bmatrix} + \begin{bmatrix} \cos\hat{\gamma}_t \cdot dt & 0 \\ \sin\hat{\gamma}_t \cdot dt & 0 \\ 0 & dt \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}$$

$$x_{t+1} = A_r x_t + B_r u, \tag{1}$$

where A_r is the state transition matrix and B_r is the control input matrix for the robot.

B. Probabilistic Reachable Set and Collision Probability

Algorithm 3 highlights the probabilistic reachability and collision analysis for both the robot and the human. First, we define the respective initial states for computing the reachable sets of both the robot and the human. As shown in the Algorithm lines 2-3, we then define the probabilistic initial set.

Probabilistic Initial Set. We get the initial state vector of the robot from the odometry sensor and generate the initial ProbStar with mean, variance, upper bound, and lower bound as the input; the basis vector is based on the space occupied by the robot (length, width) and the center is the initial state vector. For the initial configuration, the μ is set to 0 and the σ is set to 1. Thus, the lower and upper bounds are chosen based on the equation: $\mu_{x[0]} - (a\sigma_{x[0]}) \leq x[0] \leq \mu_{x[0]} + (a\sigma_{x[0]})$, where a is a positive coefficient. When a increases, the probability of the inputs lying between their lower and upper bounds of interest increases. In this case study, we choose $a = 4.5$. The variance of the distribution is $\sum = diag(\sigma_1^2, \sigma_2^2, \sigma_3^2)$, where $diag$ stands for a diagonal matrix. Similarly, for the human, we consider a normal distribution for the initial set with pose coordinates of the human as the center and an average width and breadth of 1.79 meters for the basis vector.

Further, to compute the reachable sets of the robot we iteratively affine map (Proposition 1) the initial ProbStar to get the set $[\![\Theta_r^k]\!]$ of the next k reachable sets (line 8) using the system matrices A_r and B_r (line 5) defined in section V-A(eq

Algorithm 3: Probabilistic Reachability Analysis

Input: Worker's pose w.r.t robot $\mathcal{W}'$,
Robot's state $\mathcal{R}$,
Reachable set time steps k
Output: List of robot's k reachable sets $[\![\Theta_r^k]\!]$,
List of worker's k reachable sets $[\![\Theta_w^k]\!]$,
List of k collision probabilities $[\![\mathcal{P}_{collision}^k]\!]$

1 **Function** `REACH`$(\mathcal{R},\ \mathcal{W}',\ k)$**:**
/* ProStar initial set: section V-B */
2 $\Theta_r = \texttt{InitialSet}(\mathcal{R}.pose)$
3 $\Theta_w = \texttt{InitialSet}(\mathcal{W}'.x, \mathcal{W}'.y)$
4 $[\![\Theta_r^k]\!].append(\Theta_r)$, $[\![\Theta_w^k]\!].append(\Theta_w)$
/* define system matrices from V-A */
5 $A_r \leftarrow I_3$, $B_r \leftarrow$ from equation 1
/* define matrices from Proposition 3 */
6 $A_w \leftarrow I - KM$, $B_w \leftarrow Kz$
7 **for** *i in range(k)* **do**
/* from Proposition 1 */
8 $\Theta_r \leftarrow \Theta_r.\texttt{AffineMap}(A_r, B_r u_r)$
9 $\Theta_w \leftarrow \Theta_w.\texttt{AffineMap}(A_w, B_w)$
10 $[\![\Theta_r^k]\!].append(\Theta_r)$, $[\![\Theta_w^k]\!].append(\Theta_w)$
/* from Proposition 2 */
11 $\Theta_{overlap} \leftarrow \texttt{Intersection}(\Theta_r, \Theta_w)$
/* from definition 4 */
12 $\mathcal{P}_{rw} \leftarrow \texttt{estimateProbability}(\Theta_{overlap})$
13 $[\![\mathcal{P}_{collision}^k]\!].append(\mathcal{P}_{rw})$
14 **return** $[\![\Theta_r^k]\!], [\![\Theta_w^k]\!], [\![\mathcal{P}_{collision}^k]\!]$

1) and control input as velocity v and heading angle ω from the odometer.

For computing the worker's reachable set, we generate the initial ProbStar with the initial state vector of x, y, $v_x$and v_y (line 5) which is computed from Algorithm 1 and then compute the Kalman filter reachable set $[\![\Theta_w^k]\!]$ of next k reachable sets by affine mapping (line 9) the initial ProbStar Θ_w using matrices A_w and B_w (line 6) as defined in Proposition 3 and affine map to get the Kalman filter reachable set. Lastly, the robot's collision probability with a worker is calculated. We find the overlapping region of the robot's reachable set with a half-space occupied by the worker (line 11) which gives us an overlapping ProbStar (Proposition 2). Thus, we calculate the probability of the overlapping ProbStar $\Theta_{overlap}$ (definition 4) which gives us the collision probability $\mathcal{P}_{rw}$ (line 12).

Remark: STL-based monitoring with ProbStar Reachability Integration. In our approach, we integrate STL-based monitoring with probabilistic reachability analysis to optimize computational efficiency. We verify collision probabilities only when the robot is close to an obstacle, reducing unnecessary calculations when the robot is distant from hazards. When monitoring results are inconclusive due to localization uncertainties, the verification module employs real-time probabilistic reachability analysis to evaluate collision likelihood. These results are prioritized to prevent collisions. This strategy reduces computational load and maximizes resource utilization.

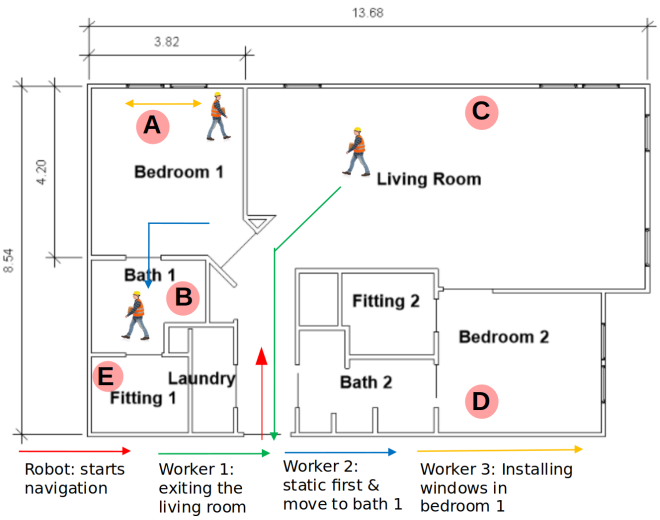


Fig. 4: Case Study 1: robot navigating from apartment entrance (red arrow) to various locations (red dots) detecting and maintaining a safe distance from workers during navigation.

Proximity between robots and static obstacles is determined using odometer readings. For proximity determination between robots and static obstacles, we rely on odometer readings.

VI. Experimental Evaluation

The current research employed the Turtlebot3 waffle model which was fitted with a 360-degree LiDAR and an RGBD camera for sensing. Modified animated models, mimicking real construction workers' movements, were employed in simulations to match the case study's emphasis. The experiments were executed on an Intel(R) Core(TM) i7-10700 CPU @ 2.90GHz x 8 cores, 64-bit Ubuntu 20.04.4 LTS system. The package was created and tested on ROS Noetic and coded in Python 3.8.10 and C++14.

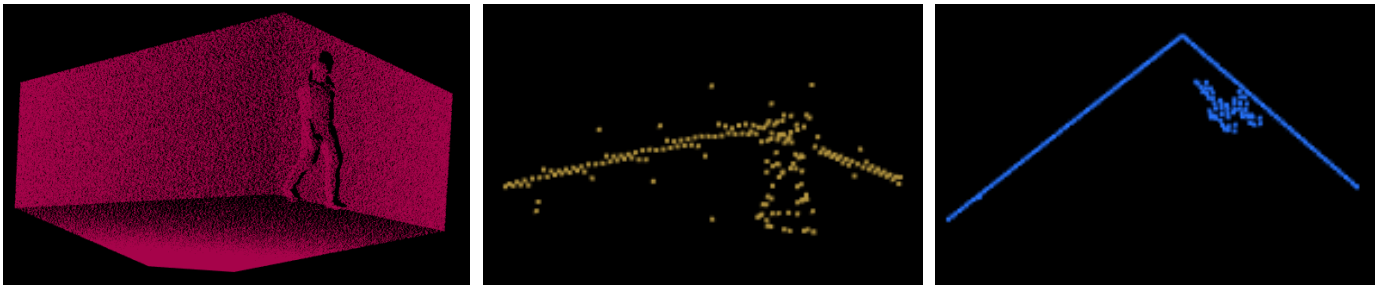
A. Signal Temporal Logic Specification

In this section, we carefully evaluate the algorithm at each stage, looking closely at the trends in the outcomes for a definite case study depicted in Fig 4. In a residential apartment construction site created on Gazebo, we focus on the interior finishing stage with a completed building structure and four workers. Fig 4 illustrates our robot's goal: navigating from the apartment entrance to a user-defined delivery location (δ) to deliver window materials to a construction worker. During navigation, the robot detects humans, calculates safety distance, and monitors time-to-collision. If safety distance or time-to-collision (TTC) falls below the set threshold, the robot pauses, executes a turn away from the approaching human, and then renavigates towards the goal. In this research, we monitor the system behavior in the case study scenario with respect to the following STL requirements.

$$\varphi_{safety} = \square_{[t,t+t_{pred}]}(d \geq d_{safe}) \tag{2}$$

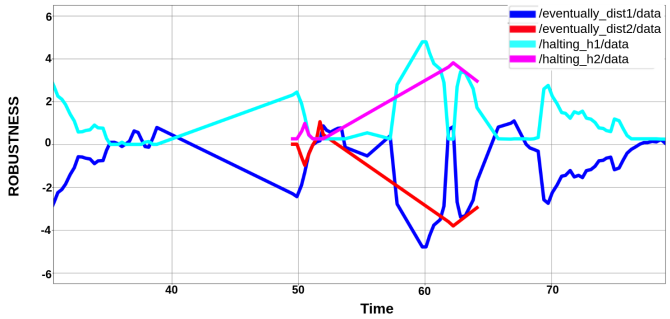
$$\varphi_{halting} = \Diamond_{[t,t+t_{pred}]}(d \leq d_{safe}) \rightarrow (v = 0) \tag{3}$$

$$\varphi_{distance} = (\square_{[t,t+t_{pred}]} d \geq d_{safe}) \mathcal{U} (\varphi_{halting}), \tag{4}$$

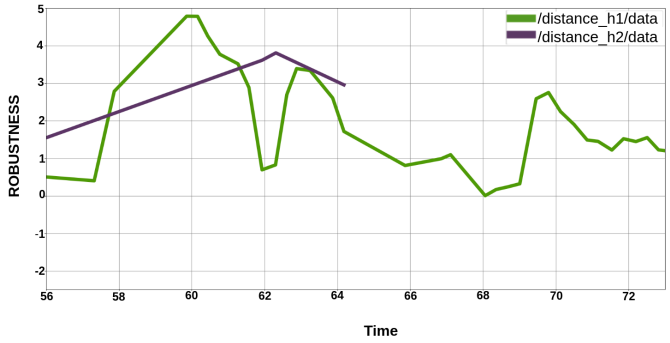


(a) Input Point Cloud, μ (b) Plane extracted point cloud: pl_pcl (c) Projected Point Cloud : $proj\_pcl$

Fig. 5: The Fig shows the input (red point cloud) and output after extracting features from the point cloud (yellow point cloud) and projecting the cloud on the ground plane (blue point cloud)



(a) Robustness plot for $\varphi_{halting}$



(b) Robustness plot for $\varphi_{distance}$

Fig. 6: Quantitative Robustness plots for STL Specifications

where $d = \sqrt{(x^2 + y^2 + z^2)}$, d_{safe} is the minimum safety distance required, v is the constant velocity of robot and t is the current time, t_{pred} is the number of prediction time steps and $t' \in [t, t + t_{pred}]$. Colloquially, specification φ_{safety} in equation 2 requires that "the distance of the human should be greater than d_{safe} meters from time t to $t + t_{pred}$" and specification $\varphi_{halting}$ (equation 3) additionally requires that "when eventually the human comes closer than d_{safe} meters from time t to $t + t_{pred}$ implies that the speed of the robot is zero". Thus, the robot should stop navigating when the $d \leq d_{safe}$. This ensures that the robot stops to avoid any collision with the human. Further, in equation 4 we expect to monitor the motion and the distance of the human from the robot.

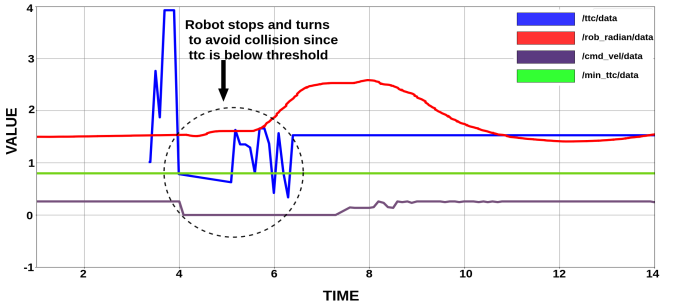


Fig. 7: Plot depicting collision avoidance over time by turning when TTC is below threshold.

In our quest to optimize performance and ensure collision prevention, we employ the time-to-collision (TTC) metric. TTC is calculated using the relative velocity and distance between the human and the robot. Thus we assess the estimated time until a potential collision occurs. A proactive safety measure is triggered when the calculated TTC dips below a predetermined threshold through iterative testing. Specifically, the robot initiates a turn in the direction opposite to the approaching human, strategically mitigating the risk of collision. This approach has prompted the establishment of additional safety specifications, designed to continually monitor and uphold the integrity of the system. The ensuing specifications further contribute to a robust framework for maintaining a secure and efficient robotic operation in dynamic environments.

$$\varphi_{stop} = (v = 0) \wedge (rad = 1.5) \quad (5)$$

$$\varphi_{ttc} = \square_{[t,t+t_{pred}]}([ttc < ttc_{min}] \rightarrow \varphi_{stop}) \quad (6)$$

$$\varphi_{turn} = (v > 0) \wedge (rad > 1.5) \quad (7)$$

$$\varphi_{avoid} = \varphi_{stop} \rightarrow (\Diamond_{[t',t+t_{pred}]}(ttc > ttc_{min}) \wedge \varphi_{turn})) \quad (8)$$

These specifications primarily emphasize the assessment of time-to-collision (TTC) to prevent collisions. This involves implementing measures such as stopping, turning, and adjusting motion to evade potential collisions effectively.

B. Localization and Estimation

In this research, the point cloud dataset input (Fig 5a) was subjected to voxel grid filtering. The filtering output was

achieved by setting the voxel grid's leaf size to Δ where $\Delta = (0.07, 0.0, 0.07)$. Fig 5 illustrates the progressive output acquired after feature extraction (Fig 5b) from the point cloud and subsequent projection onto the ground plane (Fig 5c). Further, we used the Kalman filter to predict the next 6 steps ($t_{pred} = 6$) of a human. The Root Mean Squared Error (RMSE) of prediction was calculated to evaluate the accuracy of the predictions and was found to be 0.08 meters.

C. Monitoring

In light of the Kalman filter's predicted positions of the human for the upcoming t_{pred} time steps, the values for $\varphi_{halting}$ were computed. Fig 6a quantifies STL specification robustness. The dark blue line corresponds to the "eventually" operator (in Eq.3). Robustness stays negative when the human is distant, turning positive within 1.25 meters. The light blue line (representing $\varphi_{halting}$) approaches zero when $d \leq 1.25$ meters , signaling the robot to stop. Pink and red plots in Fig 6b depict system robustness for the second worker on camera. The system adheres to $\varphi_{distance}$ (Eq. 4) for both workers, with consistently positive values. The second human exits the camera view at $t = 64$. Similarly, safety can be gauged using Eq. 6 and Eq. 8, visually confirmed in Fig 7. Here, the plot shows the system status when the time-to-collision (TTC) (blue line) drops below the threshold (green line). The robot's speed hits zero (purple line), and its orientation changes (red line) to avoid a collision.

To comprehensively assess the performance of the monitoring module, we conducted a series of tests across various scenarios and organized the results in tables. In table I, our experiments show that the proposed approach efficiently ensures safety in complex scenarios, even with multiple human encounters in construction environments. The RMSE for Kalman filter prediction varies in the range of $0.20 - 0.24m$. The system maintains a collision-free environment despite increasing human workers, since the minimum $\varphi_{distance}$ is always positive. However, task completion time rises with more workers, with substantial increases like 164.46% in Bedroom 2. In some cases, such as (B)Bath 1 and (E)Fitting 1, task completion failed due to frequent specification violations, highlighting the need for advanced safety control strategies beyond basic collision avoidance. This calls for further research into using runtime monitoring results to improve safe planning, control, and learning processes.

D. Reachability Analysis and Collision Probability

For this evaluation, we conducted a detailed case study using three Turtlebot robots, a static obstacle, and a human worker within the same apartment (Fig. 8). Our goal was to determine the collision probability of a robot with a human, other robots, and obstacles. For convenience, we represent the three turtlebot robots as $tb0$, $tb1$, and $tb2$, the human worker as h, and a static obstacle as ob. Each robot was positioned strategically, and $tb0$ navigated various areas while safety monitoring and reachability algorithms verified potential collisions.

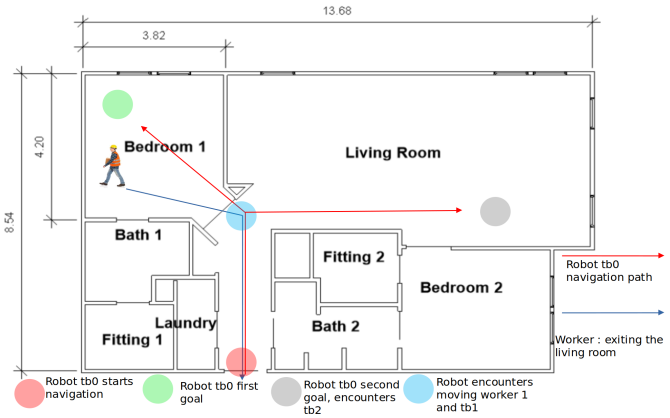


Fig. 8: Case Study 2: Robot $tb0$ begins its navigation from the apartment's entrance, marked by a red dot to the first goal location in bedroom 1 marked by green dot. $tb0$ encounters a human worker h and robot $tb1$ along its path, indicated by a blue dot. After this interaction, $tb0$ navigates to the living room i.e. the second goal location marked by grey dot, where it encounters $tb2$.

In our experiment, we thoroughly analyze collision probabilities between the Turtlebot robot and various entities when they are in proximity by computing k reachable sets for each entity, where $k = t_{pred}$). We consider four different cases and, for each scenario, determine the proximity of $tb0$ to the respective entity. Reachability analysis is initiated when the distance falls below the safe threshold d_{safe}, which is set to 1.25 meters in our study. For a human worker, we monitor the safety conditions from the defined STL specifications. When a safety violation occurs or monitoring results are inconclusive due to localization uncertainties, we trigger the reachability analysis and verify collision probability. For other robots, the proximity is determined from the odometry values, while a static obstacle is simply modeled as a half-space. By leveraging the probabilistic reachable sets, we compute reachable sets for $tb0$ for the next six time stamps (t_1 to t_6). The collision probability is evaluated based on the intersection of $tb0$'s reachable set with the ProbStar representation of other entities. In table II, we report the collision probabilities for each of those cases. Results show significant insights into collision probabilities. For example, $tb0$'s collision probability with $tb1$ starts at 0% at t_1 and rises to 82.9% by t_6. Similar trends are seen with $tb2$, a human (h), and a static obstacle (ob). These findings highlight the need for dynamic risk assessment in robotic navigation, requiring continuous monitoring and adaptation of advanced collision avoidance strategies.

In Figure 9, we present the visual representation of an instance of a potential collision with a robot (Fig 9a), human (Fig 9b), and static obstacle (Fig 9c). It shows the estimated reachable position of the robot and illustrates the overlap of its future position with the obstacle. These figures serve to visually demonstrate the probabilistic collision analysis, highlighting the regions where the robot's reachable set may intersect with

Goal	(A) Bedroom 1			(B) Bath 1			(C) Living Room			(D) Bedroom 2			(E) Fitting 1		
	$n=1$	$n=2$	$n=3$	$n=1$	$n=2$	$n=3$	$n=1$	$n=2$	$n=3$	$n=1$	$n=2$	$n=3$	$n=1$	$n=2$	$n=3$
Time (sec)	27.65	60.90	107.79	36.84	74.14	—	69.14	80.33	115.87	45.87	90.62	121.31	33.82	94.27	—
$RMSE_{KF}$ (m)	0.2281	0.2367	0.2113	0.2137	0.2737	0.2009	0.2107	0.2187	0.2087	0.2392	0.2338	0.2167	0.2398	0.2071	0.2373
Min $\varphi_{distance}$	0.0033	0.0030	0.01490	0.0005	0.0003	0.00147	0.0011	0.0396	0.01254	0.0057	0.0060	0.00213	0.0045	0.0011	0.01377

TABLE I: Performance evaluation based on the total time taken to reach a goal location, prediction RMSE for Kalman filter (KF), and minimum STL specification robustness during navigation to various locations (see Fig 4) where n represents the number of humans encountered on the way.

(a) Case 1: $tb0$ is close to $tb1$ (b) Case 2: $tb0$ is close to h (c) Case 3: $tb0$ is close to ob

Fig. 9: *Instance of collision verification*: The figure shows the overlap of $tb0's$ reachable set(green boxes) in the next 6 time stamps with (9a) $tb1$'s reachable set (blue boxes) , (9b) human worker's reachable set h (red boxes) and (9c) static obstacle ob (red box)

Reachable Set Time Stamp	Collision Probability (%)			
	$tb0 \rightarrow tb1$	$tb0 \rightarrow tb2$	$tb0 \rightarrow h$	$tb0 \rightarrow ob$
t_1	0.00	0.3	0.9	33.8
t_2	10.8	5.7	2.8	45.3
t_3	50.5	12.4	15.2	62.2
t_4	62.9	34.7	36.5	79.9
t_5	75.2	53.5	76.3	82.4
t_6	82.9	89.3	96.8	99.9

TABLE II: Collision Probability for four different cases with six reachable sets at time t_1, t_2, t_3, t_4, t_5, t_6

the space occupied by other entities.

E. Timing Analysis

Table III shows the total reachability analysis time with and without a co-design strategy, for the robot while it encounters multiple obstacles and stops to alter its path. Our co-design involves verifying collision probability only when an obstacle is near the robot. By implementing this selective reachability analysis, we substantially reduce the computational time required for real-time navigation. Specifically, this optimization results in a significant reduction in the total reachability analysis time, which decreases by approximately $57\% - 68\%$. This efficiency gain not only enhances the robot's overall performance but also ensures timely and accurate path adjustments in dynamic environments.

No of Obstacle Encounters	Time without co-design (s)	Time with co-design (s)	Reduction (%)
1	28	12	57.14
2	60	20	66.66
3	110	35	68.18

TABLE III: Percentage reduction in total reachability analysis time with co-design for multiple obstacle encounters

Node	Yolo Object Detection	Localization and Clustering	Monitoring	Reachability and Collision
Processing Time (s)	0.22	0.30	0.002	0.03

TABLE IV: Average processing times for different nodes

Lastly, Table IV highlights the processing time for each node in our algorithm. The results indicate that both the monitoring and reachability analysis components indicate low processing times. This efficiency is critical for real-time applications,

where prompt and reliable performance is crucial for safety. The low processing times not only demonstrate the algorithm's capability to handle real-time data streams effectively but also ensure that the system can respond swiftly to dynamic changes in the environment.

VII. Related Works

Temporal logic-based monitoring. E. Bartocci et al. [17] proposed monitoring for cyber-physical systems' temporal behaviors in real-time or simulation. They use temporal logic to create monitors that observe the system and check property satisfaction or violation, focusing on real-time properties using Signal Temporal Logic (STL). The AMT tool [18] provides a graphical interface to plot boolean satisfaction of user-provided STL properties. In [19], authors introduced a dynamic programming algorithm to compute the robustness of temporal logic specifications for automotive system models using the S-Taliro toolbox [20]. Breach [21] enables compositional verification of complex automotive systems with multiple sub-modules.

For semi-formal verification of STL requirements over real-valued and Boolean signals, various methods include testing-based verification [22], falsification [23], synthesis [24], [25], and parameter mining. RTAMT [26] parses textual specifications into abstract parse trees (APT) using ANTLR4 parser generator, then generates online monitors to evaluate signal robustness to the specification using algorithms [27]. Real-time monitoring of STL specifications requires knowledge of future events. Our proposed approach relies solely on the robot's vision to predict human motion in advance and effectively monitor human-robot interactions on the fly.

Probabilistic Real-Time Reachability Analysis. Satisfiability Modulo Convex (SMC) [28] and VeriSig [29] validate LiDAR-based control systems [30], using abstract observation models for mapping robot positions to LiDAR images. These are known to be challenging and time-consuming. In another method, the verification of a camera-based control system in [31] employed programming analysis tools such as CBMC [32]. The authors utilized piece-wise set value functions to approximate the perception module for verification, also assuming a structured environment to derive an approximate perception model. In the context of camera-based autonomous landing systems, [33] describes a verification process where a mathematical model of the relationship between aircraft states and inputs to the neural network controller is developed. This model is then encoded as another neural network and combined with the controller for verification. VerifAI [34] is a notable testing framework that uses simulation traces to refute system-level specifications for perception-based control systems.

The verification methods discussed for perception-based control systems operate during the design phase and rely on a significant assumption about the environment to create an observation model for closing the loop. Furthermore, these methods offer qualitative outcomes, such as determining whether a system is safe, unsafe, or unknown. In contrast, our proposed approach addresses the issue of unrealistic environmental assumptions by functioning effectively in dynamically changing, unstructured environments as often seen at construction sites. Furthermore, our verification process relies entirely on the robot's local perspective. This leads us to believe that our approach is practical and suitable for real-time autonomous robot systems.

VIII. Conclusions and Future Work

In summary, our research introduces a novel method to ensure safe autonomy in cyber-physical systems, particularly at construction sites where robots collaborate with human workers. By localizing humans and predicting their trajectories, we achieve precise distance measurements between robots and humans, ensuring a minimum safety distance is maintained during navigation. We perform real-time monitoring of STL specifications and generate robustness plots, enhancing safety in various industrial applications through real-time autonomous control. To address uncertainties in unstructured environments, we conduct real-time probabilistic reachability analysis to compute collision probabilities under proximity conditions. We also evaluate our algorithm's timing and performance. This work lays the foundation for deploying autonomous robots in safety-critical environments, with potential applications across industries to improve safety and efficiency for workers and society.

Looking ahead, monitoring and reachability analysis can enhance path planning strategies by providing real-time feedback and predicting potential obstacles, ensuring safer and more efficient navigation. Moreover, practical implementation on construction sites takes into account the algorithm's accuracy and scalability, addressing challenges posed by varying human speeds through robustness. Future research will focus on scalability with fast-moving humans, integrating parallelizable adaptive filtering techniques, and expanding hazard considerations beyond human motion. Additionally, we plan to dynamically choose the number of prediction timestamps for Kalman filtering based on feedback from STL specifications, enhancing adaptability and performance in real-world scenarios. Real-life testing will employ the Husky UGV for construction site research.

Acknowledgments

This work is supported by the National Science Foundation (NSF) under grants NSF-SLES-2331937, NSF-FMitF-2422189, and NSF-CRII-2245853. Any opinions, findings, conclusions, or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of NSF.

References

[1] S. Kim, M. Peavy, P.-C. Huang, and K. Kim, "Development of bim-integrated construction robot task planning and simulation system," *Automation in Construction*, vol. 127, p. 103720, 2021.

[2] O. W. Chong, J. Zhang, R. M. Voyles, and B.-C. Min, "Bim-based simulation of construction robotics in the assembly process of wood frames," *Automation in Construction*, vol. 137, p. 104194, 2022.

[3] C. Brosque, E. G. Herrero, Y. Chen, R. Joshi, O. Khatib, and M. Fischer, "Collaborativewelding and joint sealing robots with haptic feedback," in *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction*, vol. 38, pp. 1–8, IAARC Publications, 2021.

[4] C. Bartneck, D. Kulić, E. Croft, and S. Zoghbi, "Measurement instruments for the anthropomorphism, animacy, likeability, perceived intelligence, and perceived safety of robots," *International journal of social robotics*, vol. 1, pp. 71–81, 2009.

[5] G. Albeaino, M. Gheisari, and R. Issa, *Human-Drone Interaction (HDI): Opportunities and Considerations in Construction*, pp. 111–142. 01 2022.

[6] I. Jeelani and M. Gheisari, "Safety challenges of uav integration in construction: Conceptual analysis and future research roadmap," *Safety science*, vol. 144, p. 105473, 2021.

[7] Z. Zhu, A. Dutta, and F. Dai, "Exoskeletons for manual material handling – a review and implication for construction applications," *Automation in Construction*, vol. 122, p. 103493, 2021.

[8] M. Kurien, M.-K. Kim, M. Kopsida, and I. Brilakis, "Real-time simulation of construction workers using combined human body and hand tracking for robotic construction worker system," *Automation in Construction*, vol. 86, pp. 125–137, 2018.

[9] H.-D. Tran, S. Choi, H. Okamoto, B. Hoxha, G. Fainekos, and D. Prokhorov, "Quantitative verification for neural networks using probstars," in *Proceedings of the 26th ACM International Conference on Hybrid Systems: Computation and Control*, pp. 1–12, 2023.

[10] H.-D. Tran, D. Manzanas Lopez, P. Musau, X. Yang, L. V. Nguyen, W. Xiang, and T. T. Johnson, "Star-based reachability analysis of deep neural networks," in *Formal Methods–The Next 30 Years: Third World Congress, FM 2019, Porto, Portugal, October 7–11, 2019, Proceedings 3*, pp. 670–686, Springer, 2019.

[11] H. D. Tran, S. Choi, H. Okamoto, B. Hoxha, G. Fainekos, and D. Prokhorov, "Quantitative verification for neural networks using probstars," in *The 26th ACM International Conference on Hybrid Systems: Computation and Control (HSCC)*, May 2023.

[12] T. Yamaguchi, B. Hoxha, and D. Ničković, "Rtamt – runtime robustness monitors with application to cps and robotics," *International journal on software tools for technology transfer : STTT.*, vol. 26, no. 1, 2023-10-05.

[13] S. Y. Chen, "Kalman filter for robot vision: A survey," *IEEE Transactions on Industrial Electronics*, vol. 59, no. 11, pp. 4409–4420, 2012.

[14] G. Jocher, "Yolov5 by ultralytics," 2020.

[15] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*, (Shanghai, China), IEEE, May 9-13 2011.

[16] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[17] E. Bartocci, J. Deshmukh, A. Donzé, G. Fainekos, O. Maler, D. Ničković, and S. Sankaranarayanan, "Specification-based monitoring of cyber-physical systems: a survey on theory, tools and applications," in *Lectures on Runtime Verification*, pp. 135–175, Springer, 2018.

[18] D. Nickovic and O. Maler, "Amt: A property-based monitoring tool for analog systems," in *Formal Modeling and Analysis of Timed Systems* (J.-F. Raskin and P. S. Thiagarajan, eds.), (Berlin, Heidelberg), pp. 304–319, Springer Berlin Heidelberg, 2007.

[19] G. E. Fainekos, S. Sankaranarayanan, K. Ueda, and H. Yazarel, "Verification of automotive control applications using s-taliro," in *2012 American Control Conference (ACC)*, pp. 3567–3572, 2012.

[20] Y. Annpureddy, C. Liu, G. Fainekos, and S. Sankaranarayanan, "S-taliro: A tool for temporal logic falsification for hybrid systems," in *Tools and Algorithms for the Construction and Analysis of Systems* (P. A. Abdulla and K. R. M. Leino, eds.), (Berlin, Heidelberg), pp. 254–257, Springer Berlin Heidelberg, 2011.

[21] A. Donzé, "Breach, a toolbox for verification and parameter synthesis of hybrid systems," in *Computer Aided Verification* (T. Touili, B. Cook, and P. Jackson, eds.), (Berlin, Heidelberg), pp. 167–170, Springer Berlin Heidelberg, 2010.

[22] G. E. Fainekos, A. Girard, and G. J. Pappas, "Temporal logic verification using simulation," in *International Conference on Formal Modeling and Analysis of Timed Systems*, pp. 171–186, Springer, 2006.

[23] H. Abbas, G. Fainekos, S. Sankaranarayanan, F. Ivančić, and A. Gupta, "Probabilistic temporal logic falsification of cyber-physical systems," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 12, no. 2s, pp. 1–30, 2013.

[24] X. Jin, A. Donzé, J. V. Deshmukh, and S. A. Seshia, "Mining requirements from closed-loop control models," in *Proceedings of the 16th international conference on Hybrid systems: computation and control*, pp. 43–52, 2013.

[25] H. Yang, B. Hoxha, and G. Fainekos, "Querying parametric temporal logic properties on embedded systems," in *IFIP International Conference on Testing Software and Systems*, pp. 136–151, Springer, 2012.

[26] D. Nickovic and T. Yamaguchi, "Rtamt: Online robustness monitors from stl," 2020.

[27] A. Donzé and O. Maler, "Robust satisfaction of temporal logic over real-valued signals," in *International Conference on Formal Modeling and Analysis of Timed Systems*, pp. 92–106, Springer, 2010.

[28] X. Sun, H. Khedr, and Y. Shoukry, "Formal verification of neural network controlled autonomous systems," in *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, pp. 147–156, 2019.

[29] R. Ivanov, J. Weimer, R. Alur, G. J. Pappas, and I. Lee, "Verisig: verifying safety properties of hybrid systems with neural network controllers," in *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, pp. 169–178, 2019.

[30] R. Ivanov, T. J. Carpenter, J. Weimer, R. Alur, G. J. Pappas, and I. Lee, "Case study: verifying the safety of an autonomous racing car with a neural network controller," in *Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control*, pp. 1–7, 2020.

[31] C. Hsieh, Y. Li, D. Sun, K. Joshi, S. Misailovic, and S. Mitra, "Verifying controllers with vision-based perception using safe approximate abstractions," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 11, pp. 4205–4216, 2022.

[32] E. Clarke, D. Kroening, and F. Lerda, "A tool for checking ansi-c programs," in *Tools and Algorithms for the Construction and Analysis of Systems: 10th International Conference, TACAS 2004, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2004, Barcelona, Spain, March 29-April 2, 2004. Proceedings 10*, pp. 168–176, Springer, 2004.

[33] U. Santa Cruz and Y. Shoukry, "Nnlander-verif: A neural network formal verification framework for vision-based autonomous aircraft landing," in *NASA Formal Methods Symposium*, pp. 213–230, Springer, 2022.

[34] T. Dreossi, D. J. Fremont, S. Ghosh, E. Kim, H. Ravanbakhsh, M. Vazquez-Chanlatte, and S. A. Seshia, "Verifai: A toolkit for the formal design and analysis of artificial intelligence-based systems," in *International Conference on Computer Aided Verification*, pp. 432–442, Springer, 2019.