Convex-Hull Based Dynamic Footprint of Mobile Manipulators for Collision-Safe Navigation in Nav2

Keerthi Sagar[1] Philip Long[2] and Carlos Garcia Santiago[1]

***Abstract*— Safe autonomous navigation of mobile manipulators demands that the robot's collision footprint accurately reflect the combined swept volume of the base and the arm, which is otherwise treated as static within the base footprint. Existing navigation frameworks treat the robot footprint as a static parameter, introducing unmodelled collision risk whenever the manipulator departs its transport configuration. We present a real-time, geometry-driven projection method that constructs a configuration-dependent robot footprint directly from the robot's kinematic chain. At each update cycle, the arm-link geometry is projected onto the ground plane, and the convex hull of the projected geometry is used as the updated mobile-manipulator footprint for navigation planning. The implementation supports both a lightweight disc-based approximation for real-time footprint updates and a mesh-based projection variant using URDF collision geometry for higher-fidelity footprint generation. We validate the proposed method on Nav2 by deploying on three mobile-manipulator platforms: (i) Neobotix MPO-700 rectangular mobile base with UR10 arm; (ii) TIAGo robot with circular base; (iii) KUKA KMR with 7-DOF iiwa manipulator arm. We demonstrate collision-safe navigation in scenarios where a static footprint fails. The framework provides a principled and computationally lightweight solution for collision-safe autonomous navigation in industrial mobile manipulation scenarios. Experiment videos and code repository are available here:** $arm\_dynamic\_footprint\_nav2$**.**

I. INTRODUCTION

Mobile manipulators are increasingly deployed in industrial environments for tasks such as machine tending, surface inspection, and logistics [1], where the robot must navigate autonomously through shared workspaces while executing manipulation tasks that may require the arm to extend beyond the base platform [2], [3]. Additionally, mobile manipulators [4] are also gaining more acceptance within indoor human-centered environments for disability assistance [5], transporting objects [6], and cleaning [7].

Safe autonomous navigation in such settings demands that the robot's collision boundary accurately reflect the full physical extent of the system at all times. For a bare mobile base this boundary is fixed and well-defined, but for a mobile manipulator it is configuration-dependent: as the arm moves through its workspace, its links project a swept volume onto the ground plane that changes continuously with the joint state.

This work has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement no. 101070254 for project "CORESENSE".

[1]Authors are with Robotics and Automation Group, Irish Manufacturing Research, Mullingar, Ireland firstname.lastname@imr.ie

[2]Author is with the Robotics Group, Atlantic Technological University, Galway, Ireland firstname.lastname@atu.ie

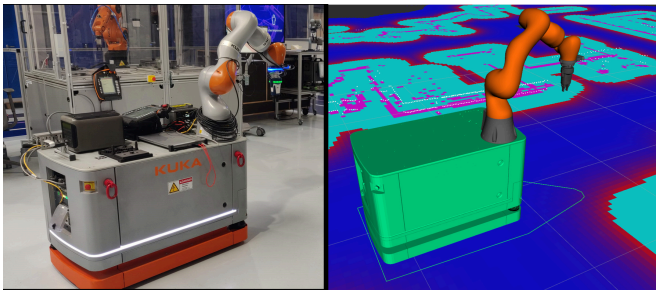

Fig. 1. Projection of the manipulator arm geometry onto the footprint of the mobile base in Nav2

The main limitation within existing navigation frameworks such as Nav2 is not the absence of footprint-based collision checking, but the assumption that the robot footprint is either fixed at launch or switched between a small number of predefined footprints during task phases such as loading or unloading. For mobile manipulators, this assumption is insufficient because the ground-plane projection of the robot changes with the arm configuration and any carried payload. A static base-only footprint therefore causes the planner and controller to collision-check only the mobile base, while the extended arm geometry remains absent from the navigation model, creating unmodelled collision risk near obstacles, workcells, or people. We present a real-time, geometry-driven footprint projection framework that computes an arm-aware collision polygon from the manipulator kinematic state and publishes it through the standard Nav2 footprint interface. The approach requires no navigation-stack modification or exteroceptive sensing, supports rectangular and circular bases, and is validated on three heterogeneous mobile-manipulator platforms. Fig. 1 illustrates the core idea of the proposed approach.

II. RELATED WORK

Configuration-dependent collision modelling for mobile manipulators has been addressed from several perspectives, including whole-body planning, controller-level safety, perception-based costmap updates, and dynamic footprint adaptation. During transport or waypoint navigation, the manipulator arm is commonly assumed to be stowed within the mobile base footprint. However, this assumption no longer holds when the arm is extended during navigation or when the robot carries external payloads. A similar issue arises in mobile-robot towing or shelf-carrying scenarios [8], where the effective footprint changes once the object is attached to the robot.

Whole-body manipulation frameworks address this problem by jointly planning arm and base motions in a shared

configuration space [9], [10], [2], [3]. These methods account for the full robot geometry during trajectory generation and are well suited to tightly coupled manipulation tasks. However, they typically require coupled arm–base planning and do not directly update the runtime collision footprint used by standard navigation planners during execution. Controller-level safety methods, including Control Barrier Function (CBF)-based whole-body controllers [11], encode collision constraints directly in the control law. While such methods can provide formal safety guarantees, they generally require access to the robot dynamics and are not directly exposed through standard navigation stack interfaces.

A closely related line of work considers changing robot geometry during navigation. Hornung et al. [12] proposed a 3D navigation framework for mobile manipulation using layered environment representations and down-projected robot footprints. Outon et al. [13] presented a dynamic robot footprint module for an industrial dual-arm mobile manipulator, where arm joint positions are monitored and a bounding polygon is used to update the robot footprint. Dynamic footprint adaptation has also been studied in social navigation; Sun and Venture [14] adjust the footprint for human–robot side-by-side walking based on the relative human–robot configuration. These works demonstrate the importance of adapting the robot footprint online, but they either rely on layered 3D representations, modified navigation planners, or application-specific social-navigation models.

Perception-based methods project 3D sensor measurements into costmap representations to capture obstacles or swept volumes online [15], [16], [17]. Such approaches are useful in unstructured environments, but require continuous depth sensing and additional costmap-layer processing. Custom costmap layer architectures [16], [17] can extend the navigation stack to incorporate environmental changes, but do not directly address the robot's own configuration-dependent geometry as a runtime footprint consumed by the planner and controller.

Nav2 [18], the widely adopted open-source navigation stack for ROS 2, exposes a runtime footprint update interface through which the collision polygon can be updated without modifying the stack [19]. *SmacPlannerLattice* and *SmacPlannerHybrid* [20] perform full SE(2) footprint collision checking at each expanded state, while the MPPI controller [21] evaluates footprint-aware cost critics during trajectory optimisation. Therefore, the infrastructure to consume a dynamic footprint already exists within Nav2, but a lightweight mechanism is still required to compute this footprint from the configuration of a mounted manipulator arm. This gap motivates the present work.

The proposed method differs from existing approaches by addressing the problem at the navigation-footprint layer rather than through whole-body planning, controller redesign, 3D environment layering, or perception-based costmap augmentation. It computes a configuration-dependent footprint directly from TF link poses and calibrated link radii, requiring no additional depth sensing, custom costmap layer, or modification of Nav2. Existing Nav2 planners and controllers can therefore consume the updated footprint through the standard footprint interface.

The main contributions of this work are as follows:

- A lightweight convex-hull based arm-geometry projection algorithm for dynamic footprint generation, with toolbox support for both disc-based and mesh-based footprint variants.
- Seamless integration with Nav2 using costmap footprint update via ROS 2 lifecycle nodes.
- Open-source ROS2 repository implementation with platform examples and configuration utilities to support reproducible evaluation is provided [1].

The remainder of this paper is organised as follows. Section III introduces the notation and relevant Nav2 footprint preliminaries. Section IV describes the proposed dynamic footprint generation method. Section V presents experimental results across three mobile-manipulator platforms. Section VI concludes the paper and outlines future work.

III. PRELIMINARIES AND NOTATIONS

A. Notation

The notation below is defined for the lightweight disc-based footprint formulation, which is the main focus of this paper. The toolbox also supports a mesh-based convex-hull variant using the same Nav2 pipeline. Table I summarises the main notation used throughout this work. All footprint quantities are expressed in the mobile-base frame unless otherwise stated.

TABLE I

SUMMARY OF MATHEMATICAL NOTATION USED IN THIS WORK.

Symbol	Definition
$\mathbf{x} = (x, y, \theta)$	Mobile-base pose in $SE(2)$
$\mathbf{q} = [q_1, \ldots, q_n]^\top$	Manipulator joint configuration
$T_i(\mathbf{q})$	Pose of arm link l_i in the base frame
$\mathbf{p}_i = [x_i, y_i, z_i]^\top$	Translational component of $T_i(\mathbf{q})$
$\pi(\mathbf{p}_i)$	Ground-plane projection of link point $\mathbf{p}_i$
l_i	i-th selected manipulator link
L	Number of selected manipulator links
r_i	Calibrated radius enclosing link l_i in projection
$\mathcal{B}$	Base footprint point set
$\mathcal{B}_{rect}$	Rectangular base footprint point set
$\mathcal{B}_{circ}$	Circular base footprint point set
r_{base}	Radius of a circular mobile base
N_d	Number of samples used to discretise each disc/circle
N_s	Number of samples along each inter-link segment
$\mathcal{D}_i(\mathbf{q})$	Disc point set associated with link l_i
$\mathbf{s}_{i,j}$	j-th sample point between projected link origins
$\mathcal{D}^{seg}_{i,j}(\mathbf{q})$	Disc point set associated with segment sample $\mathbf{s}_{i,j}$
$\mathcal{P}(\mathbf{q})$	Complete point set used for convex-hull computation
$\mathcal{H}(\mathbf{q})$	Convex hull of $\mathcal{P}(\mathbf{q})$
δ	Radial safety padding margin
$\mathcal{F}(\mathbf{q})$	Final dynamic footprint published to Nav2

B. Mobile Manipulator Kinematics

A mobile manipulator is modelled as a kinematic chain comprising a mobile base with configuration $\mathbf{x} = (x, y, \theta) \in SE(2)$, where x and y denote the position of the base in the world frame and θ its heading angle, and a serial manipulator arm with joint configuration $\mathbf{q} = [q_1, \ldots, q_n]^\top \in R^n$. The

[1] github.com/KeerthiSagarSN/cs4mt_arm_dynamic_footprint_nav2

full system configuration is denoted $(\mathbf{x}, \mathbf{q})$, and the pose of each arm link i in the robot base frame is given by the forward kinematics map $T_i(\mathbf{q}) \in SE(3)$, evaluated at runtime by querying the TF tree in ROS 2, maintained by the robot state publisher from the `/joint_states` topic based on the URDF kinematic chain, without requiring explicit kinematic model inversion.

The robot's kinematic structure is described using the Unified Robot Description Format (URDF) [22], an XML-based specification that defines the robot as a tree of rigid links connected by joints, where each link carries a name, a collision geometry, and a visual geometry, and each joint defines the transformation between consecutive links as a function of the joint variable. The base platform geometry is described either as a convex polygon $\mathcal{B} = \{v_1, \ldots, v_m\} \subset R^2$ for rectangular or arbitrarily shaped bases, or as a circle of radius r_{base} centred at the base origin for circular platforms, the latter being sampled uniformly at N_d points to yield an equivalent polygon representation.

C. Navigation Stack and Footprint Representation

The robot footprint $\mathcal{F} \subset R^2$ is the 2D representation used by the navigation stack to evaluate obstacle clearance during path planning and trajectory execution, representing the ground-plane projection of the robot's physical extent. Nav2 [18], the most widely adopted open-source navigation stack for ROS 2, supports both polygonal and circular footprint representations natively through the `footprint` and `robot_radius` costmap parameters respectively, and exposes a runtime update interface via the costmap's `~/footprint` topic, through which the collision polygon can be updated without restarting the navigation stack [19]. Nav2 maintains two costmap instances: a *global costmap* over the full map extent used by the path planner, and a *local costmap* over a rolling window centred on the robot used by the local controller, subscribing to the `/global_costmap/footprint` and `/local_costmap/footprint` topics respectively.

The footprint polygon is actively consumed by both the global planner and the local controller during navigation execution. *SmacPlannerLattice* [20] performs an A^* search over the SE(2) state space (x, y, θ) using a precomputed motion primitive lattice that encodes omnidirectional kinematics, explicitly checking collision against the full polygon footprint at each expanded state without imposing a minimum turning radius. *SmacPlannerHybrid* [20] follows the same SE(2) search and footprint-checking procedure but constrains motion primitives to Reeds-Shepp curves, making it appropriate for non-holonomic platforms. By contrast, holonomic planners such as NavFn [23] and Theta* [24] operate on a circular footprint approximation and do not perform full SE(2) collision checking, making them unsuitable when the arm geometry extends asymmetrically beyond the base. At the control level, the Model Predictive Path Integral (MPPI) controller [21] optimises a batch of sampled trajectories against a set of cost critics, including explicit footprint-aware collision evaluation via `consider_footprint: true`, applicable to both polygonal and circular footprint representations.

In either case, the footprint is defined as a static parameter at launch time, which is sufficient for a bare mobile base but becomes inadequate when a manipulator arm is mounted, as the arm links extend beyond the base boundary in a configuration-dependent manner that a static parametrisation cannot capture. The infrastructure to consume a dynamically updated footprint polygon therefore already exists within Nav2, but no mechanism exists to compute this polygon from the kinematic state of a mounted manipulator arm. The proposed method, described in Section IV, closes this gap.

IV. PROPOSED METHOD

The proposed method consists of three main steps. First, a configuration-dependent point set is constructed from the mobile base geometry and the current manipulator configuration. For each selected arm link, the TF pose is evaluated in the mobile-base frame, its translation is projected onto the ground plane by discarding the vertical coordinate, and the link cross-section is conservatively approximated as a 2D disc of calibrated radius. Additional discs are sampled along consecutive link segments to capture the projected extent of elongated or irregularly shaped links. Second, the convex hull of the combined base, link-disc, and segment-disc point set is computed to obtain the smallest convex polygon enclosing the current base–arm geometry. Third, the hull is expanded using a radial safety-padding approximation before publication to Nav2, to account for localisation uncertainty, TF-frame/link-geometry offsets, and approximation error in the disc-based link model.

The framework is implemented as a ROS 2 lifecycle node using CoreSense cognitive architecture [25], composed of an *Afferent* layer subscribing to `/joint_states`, a *Core* layer performing forward kinematics evaluation and convex hull computation, and an *Efferent* layer publishing the resulting polygon to the Nav2 costmap footprint topics at 20 Hz.

A. Robot Model and Configuration

Let the robot be described by a kinematic chain of L arm links $\{l_1, l_2, \ldots, l_L\}$, selected from the full link set of the robot model as those whose ground-plane projection contributes to the configuration-dependent collision boundary. Each link l_i is characterized by its pose $T_i(\mathbf{q}) \in SE(3)$ in the robot base frame, obtained by evaluating the forward kinematics map at the current joint configuration $\mathbf{q}$, and by a calibrated radius $r_i > 0$ that conservatively encloses the physical cross-section of the link including any attached components. The ground-plane projection of the link origin is denoted:

$$\pi(\mathbf{p}_i) = (x_i, y_i) \tag{1}$$

where $\mathbf{p}_i = [x_i, y_i, z_i]^\top$ is the translational component of $T_i(\mathbf{q})$ and the z coordinate is discarded to obtain the 2D ground-plane position. The base platform geometry $\mathcal{B} \subset R^2$ defines the minimum footprint of the robot in the absence of

any arm contribution. For a *rectangular* base parameterised by half-extents (h_x, h_y):

$$\mathcal{B}_{rect} = \{(\pm h_x,\ \pm h_y)\} \subset R^2 \tag{2}$$

For a *circular* base of radius r_{base}, the geometry is sampled at N_d uniformly distributed points:

$$\mathcal{B}_{circ} = \{r_{base}(\cos\theta_k,\ \sin\theta_k)\}_{k=0}^{N_d-1} \tag{3}$$

In both cases, $\mathcal{B} \in \{\mathcal{B}_{rect},\ \mathcal{B}_{circ}\}$ is expressed as a finite point set in R^2, entering the point set construction step without any structural distinction between base types.

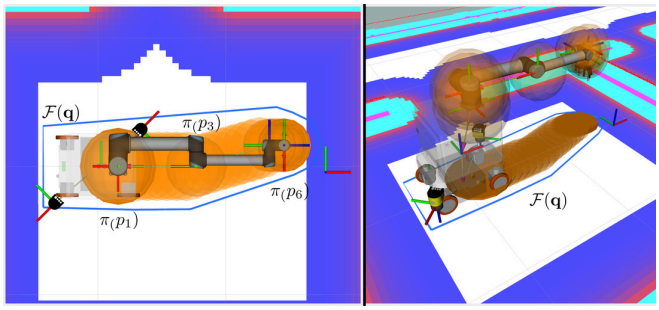

Fig. 2. RViz visualisation of the dynamic footprint framework on the MPO-700 + UR10 platform. Transparent orange spheres of calibrated radius r_i are centred at each TF link origin $\pi(\mathbf{p}_i)$ along the kinematic chain from l_1 (shoulder) to l_L (tool). Their ground-plane projections, combined with segment sample discs interpolated along each link axis, form the point set $\mathcal{P}(\mathbf{q})$ whose convex hull yields the published footprint $\mathcal{F}(\mathbf{q})$, shown as the orange filled region bounded by the blue polygon.

B. Configuration-Dependent Point Set

Each arm link l_i is approximated in the ground plane as a circle/disc of calibrated radius $r_i > 0$ centred at its projected origin $\pi(\mathbf{p}_i)$, sampled at N_d uniformly distributed points around its circumference. The centre point is also explicitly included to ensure the link origin itself is always enclosed in the final hull, yielding the per-link point set:

$$\mathcal{D}_i(\mathbf{q}) = \{\pi(\mathbf{p}_i)\} \cup \{\pi(\mathbf{p}_i) + r_i(\cos\theta_k,\ \sin\theta_k)\}_{k=0}^{N_d-1} \tag{4}$$

where $\theta_k = \frac{2\pi k}{N_d}$ for $k = 1, \ldots, N_d - 1$.

To capture the swept volume of elongated links whose physical geometry extends significantly between consecutive joint origins, the segment connecting $\pi(\mathbf{p}_i)$ and $\pi(\mathbf{p}_{i+1})$ is additionally sampled at N_s uniformly spaced intermediate points, with a disc of radius r_i placed at each sample:

$$\mathbf{s}_{i,j} = \pi(\mathbf{p}_i) + \frac{j}{N_s}\big(\pi(\mathbf{p}_{i+1}) - \pi(\mathbf{p}_i)\big), \quad j = 0, \ldots, N_s \tag{5}$$

If $T_i(\mathbf{q})$ is unavailable for a given link, that link is skipped and the footprint is computed from the remaining links, ensuring graceful degradation under partial kinematic information. The complete point set is assembled as:

$$\mathcal{P}(\mathbf{q}) = \mathcal{B} \cup \bigcup_{i=1}^{L} \mathcal{D}_i(\mathbf{q}) \cup \bigcup_{i=1}^{L-1} \bigcup_{j=0}^{N_s} \mathcal{D}_{i,j}^{seg}(\mathbf{q}) \tag{6}$$

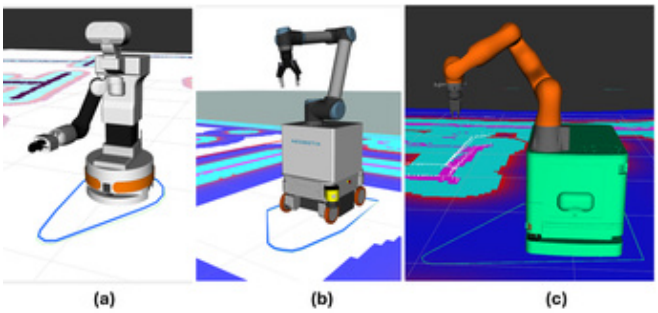

Fig. 3. Dynamic footprint $\mathcal{F}(\mathbf{q})$ visualised for (a) TIAGo with circular base ($r_{base} = 0.30$ m) and a single manipulator arm; (b) Neobotix MPO-700 with rectangular base and UR10 arm in an extended configuration; (c) KUKA KMR iiwa with rectangular base and 7-DOF LBR iiwa arm operating in a cluttered environment. In each case the blue polygon denotes the published footprint $\mathcal{F}(\mathbf{q})$ consumed by the Nav2 global planner and MPPI local controller, adapting continuously to the current arm configuration.

where $\mathcal{D}_{i,j}^{seg}(\mathbf{q})$ denotes the disc point set centred at $\mathbf{s}_{i,j}$ with radius r_i, defined analogously to (4). If $|\mathcal{P}(\mathbf{q})| < 3$, the method falls back to $\mathcal{B}$ to ensure a valid footprint is always available.

C. Convex Hull, Padding, and Publication

Given $\mathcal{P}(\mathbf{q})$, the configuration-dependent footprint is obtained by computing the convex hull $\mathcal{H}(\mathbf{q}) = \text{conv}(\mathcal{P}(\mathbf{q}))$, defined as the smallest convex polygon enclosing all points in $\mathcal{P}(\mathbf{q})$:

$$\text{conv}(\mathcal{P}) = \left\{ \sum_i \lambda_i \mathbf{p}_i \;\middle|\; \mathbf{p}_i \in \mathcal{P},\ \lambda_i \geq 0,\ \sum_i \lambda_i = 1 \right\} \tag{7}$$

The convex hull naturally subsumes both base geometry cases, for a rectangular base the corner points anchor the hull to the base platform extent, while for a circular base the uniformly sampled disc points ensure the hull encloses the full base radius. The hull is computed using the Jarvis march algorithm [26], with time complexity $O(nh)$ where $n = |\mathcal{P}(\mathbf{q})|$ and h is the number of hull vertices, which remains small in practice due to the structured geometry of the arm link point set. The resulting hull is padded using a radial centroid-based approximation with safety margin $\delta > 0$ to account for localisation uncertainty and model approximation error. Each hull vertex v_k is displaced away from the polygon centroid $\mathbf{c}$:

$$v_k^{\delta} = v_k + \delta \cdot \frac{v_k - \mathbf{c}}{\|v_k - \mathbf{c}\|} \tag{8}$$

where $\mathbf{c} = \frac{1}{|\mathcal{H}|}\sum_k v_k$. The radially padded hull $\mathcal{H}^{\delta}$ constitutes the final dynamic footprint $\mathcal{F}(\mathbf{q})$, which is published as an ordered polygon to the `/local_costmap/footprint` and `/global_costmap/footprint` topics at 20 Hz. Since Nav2 represents both polygonal and circular footprints uniformly as an ordered vector of 2D points [19], the published polygon is consumed identically by the planner and controller regardless of the underlying base geometry type, ensuring both planning levels operate with a consistent

conservative approximation of the robot's configuration-dependent physical extent. The notations are represented as shown in Fig. 2 and the dynamic footprint for both circular and rectangular mobile robot bases are shown in Fig. 3 respectively.

V. EXPERIMENTAL RESULTS

The proposed method was evaluated across three mobile manipulator platforms - (i) Neobotix MPO-700 rectangular mobile base with UR10 arm; (ii) TIAGo robot with circular base; (iii) KUKA KMR with 7-DOF iiwa manipulator arm, to validate both the generality of the approach across heterogeneous base geometries and its effectiveness in preventing collision events that a static footprint parametrisation would fail to avoid. All experiments used Nav2 with *SmacPlanner-Lattice* as the global planner in ROS 2 Humble, selected for its omni-directional motion primitive support appropriate to all three platforms evaluated. *SmacPlannerHybrid* was also tested and produced smoother trajectories in unconstrained environments, though at the cost of reduced flexibility in tight spaces. A C++ implementation is publicly available as an open-source ROS 2 Humble package to support reproducibility.

A. Experimental Setup

Simulation experiments were conducted in Gazebo using an indoor industrial world with corridors, walls, and static obstacles representative of a manufacturing shopfloor. Two arm configurations were evaluated for each platform: a *tucked* configuration in which the arm is retracted close to the base, and an *extended* configuration in which the arm protrudes significantly beyond the base boundary. Navigation goals were set such that the planned path required traversal through constrained spaces where the extended arm footprint intersects static obstacles that the base-only footprint would either not detect or would not find a feasible trajectory. Each scenario was executed ten times for each footprint strategy: static base-only, static enlarged, and the proposed dynamic footprint. To further reduce the deployment barrier across platforms, a helper utility `fetch_urdf_links.py` is provided as part of the open-source release. The utility subscribes to the `/robot_description` topic, parses the URDF kinematic tree at runtime, and automatically filters out non-arm components including base, wheel, laser, and sensor links yielding a candidate list of arm and end-effector links printed directly in YAML format, ready to be pasted into the parameter file. In practice, deploying the proposed framework on a new platform requires mainly running the utility, calibrating the per-link radii to match the physical link cross-sections, and launching the lifecycle node, a setup process that took under ten minutes for each of the three platforms evaluated in this work. This parameter-driven architecture, makes the framework readily accessible to practitioners working with heterogeneous mobile manipulator platforms.

B. Platform A: Neobotix MPO-700 + UR10 (Simulation)

The MPO-700 + UR10 platform with a Robotiq 2F-140 gripper was evaluated in the `neo_workshop` Gazebo simulation environment with a rectangular base footprint parameterised by half-extents $h_x = 0.35$ m and $h_y = 0.30$ m. In the extended arm configuration, the dynamic footprint expanded to enclose the full arm extent, with the footprint area ($0.8315\ \mathrm{m}^2$) increasing by approximately 97.976% relative to the static base-only footprint ($0.42\ \mathrm{m}^2$). With the static footprint, the planner consistently generated paths that brought the extended arm into contact with surrounding obstacles, resulting in a navigation success rate of 20% across ten trials, as shown in Table III.The static enlarged footprint improved safety but remained over-conservative, achieving only 40% success because feasible paths were often blocked by the fixed enlarged collision boundary. With the dynamic footprint module activated, the planner correctly accounted for the arm geometry, replanning around constrained regions and achieving a success rate of 90% with zero collision events. The failures in the dynamic footprint were either due to goal-failure as the mobile robot entered into a region violating cost constraints, or the generated trajectory is infeasible. In the tucked configuration, the dynamic footprint converged to the base rectangle and navigation behaviour was identical to the static case, confirming that the module introduces no unnecessary conservatism when the arm is retracted. Fig. 4 shows the comparison and effectiveness of the proposed methodology, where a collision-free trajectory is generated using the dynamic projected footprint.

TABLE II

FOOTPRINT COMPUTATION TIME (50 MS UPDATE, 5000 TICKS).

Platform	Config	Mean (μs)	Min (μs)	Max (μs)	Vertices (pts)
NBX	Tucked	689.0	269.7	1565.8	5 (1003)
NBX	Extended	713.3	269.1	1618.8	8 (1003)
TIAGo	Tucked	856.5	406.6	2675.4	19 (1195)
TIAGo	Extended	856.9	382.9	2675.4	20 (1195)

TABLE III

NAVIGATION OUTCOME COMPARISON ACROSS FOOTPRINT STRATEGIES IN GAZEBO SIMULATION OVER 10 TRIALS.

Platform	Footprint Strategy	Success Rate	Dominant Failure Mode
MPO-700 + UR10	Static base-only	20%	Arm geometry ignored leading to collision
MPO-700 + UR10	Static enlarged	40%	Over-conservative / blocked paths
MPO-700 + UR10	Dynamic proposed	90%	Goal/controller infeasibility
TIAGo	Static base-only	30%	Arm geometry ignored collision in passageway entry
TIAGo	Static enlarged	50%	Over-conservative / blocked paths
TIAGo	Dynamic proposed	90%	Goal/controller infeasibility in certain poses

C. Platform B: TIAGo (Simulation, Circular Base)

To validate the method's applicability to circular base platforms, the framework was deployed on a simulated TIAGo robot, whose circular base is parameterised by radius $r_{base} = 0.30$ m and sampled at $N_d = 16$ uniformly distributed points to yield the base polygon $\mathcal{B}_{circ}$. The dynamic footprint correctly expanded beyond the circular base boundary when the TIAGo arm was extended, and the Nav2 planner successfully accounted for obstacles that the static `robot_radius` parametrisation would have ignored. The results are shown in Table .III. This result confirms that the unified point set representation described in Section IV handles circular base platforms without any structural changes to the pipeline.

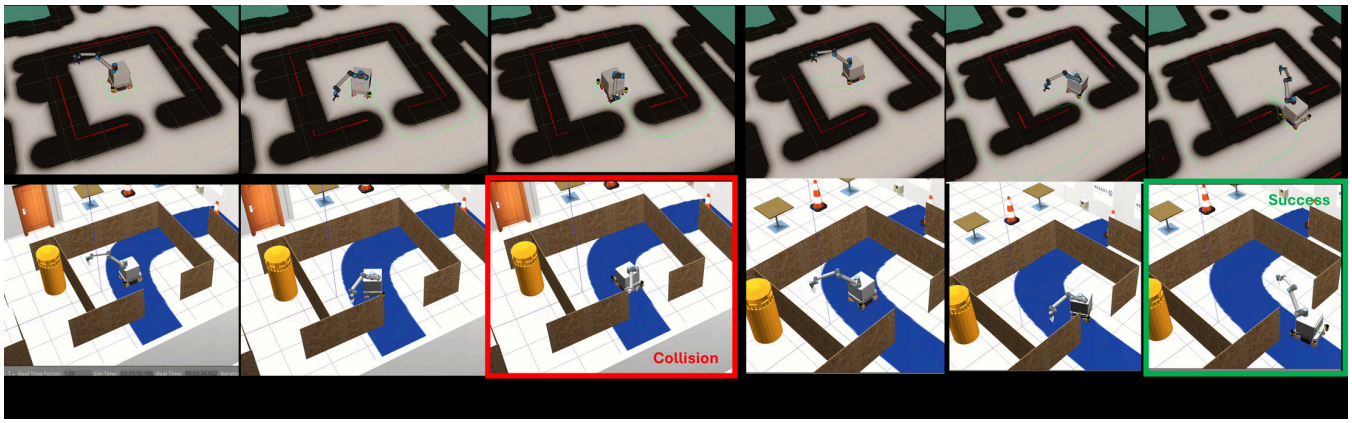

Fig. 4. Snapshot of a trajectory. (a) Neobotix robot navigating using static footprint without accounting for arm configuration, leading to collision (red outline), (b) Arm configuration aware footprint using proposed method achieving collision-free navigation.

Fig. 5. KUKA KMR hardware experiment. (a) Static-footprint path planned close to the table obstacle. (b) Proposed dynamic footprint accounting for the extended arm during obstacle avoidance. (c) RViz visualisation of the costmap and planned trajectories.

D. Platform C: KUKA KMR iiwa (Hardware)

The proposed method was additionally validated on physical hardware as shown in Fig. 5 using the KUKA KMR mobile manipulator. A point-to-point navigation task was tested with the arm held in an extended configuration of $\mathbf{q} = [170°, -31.22°, 0.0°, +83.92°, 0.0°, -64.88°, 60.0°]^\top$. A table obstacle was placed to create a constrained passage where collision with the extended arm was clearly possible. The dynamic footprint module was deployed on the onboard computation platform and activated via the ROS 2 lifecycle interface, with the resulting footprint visualised in real time in RViz. The arm was tested in both stationary and dynamic poses commanded through a sequence of configurations ranging from fully tucked to laterally extended, and the Nav2 costmap footprint was updated continuously at 20 Hz throughout, tracking the arm geometry without interruption to the navigation stack. Navigation goals were issued prior to arm motion, and the planner replanned in response to footprint boundary changes driven by the manipulator's joint configuration. Across 10 trial runs from the same initial pose, navigation with the proposed module achieved 100% collision-free execution, whereas the static footprint resulted in collision or near-miss events in 8 of 10 trials. These results indicate that the proposed method significantly improves collision safety on physical hardware by allowing Nav2 to account for the configuration-dependent footprint of the manipulator during execution.

E. Discussion

Across the evaluated simulation and hardware scenarios, the dynamic footprint module improved navigation safety compared with static footprint baselines. The footprint update rate of 20 Hz was sustained on all platforms both in simulation and in hardware without measurable impact on navigation stack performance, confirming the computational lightweight nature of the approach. The computational overhead of the proposed footprint update pipeline remained low across all evaluated platforms. As summarised in Table II, the mean footprint computation time remained well below the 50 ms update period required for 10-20 Hz operation. The ROS 2 lifecycle node interface allowed the module to be activated and deactivated at runtime without restarting the navigation stack, which was particularly demonstrated on the KMR iiwa hardware where the module was toggled during a live navigation session. Although configuring the dynamic footprint was straightforward with the proposed approach, achieving reliable operation with the Nav2 controller required careful tuning of the associated Nav2 parameters.

The absolute percentages in Table III are specific to the evaluated Gazebo environments, start–goal pairs, obstacle layouts, and Nav2 parameter settings. The key result is the consistent trend across both simulated platforms: the base-only footprint is unsafe when the arm extends beyond the base, the enlarged static footprint is safer but over-conservative, and the proposed dynamic footprint provides the best success rate by adapting the collision boundary to the current arm configuration.

The open-source release of the framework is intended to lower the barrier to adoption and support reproducible evaluation within the mobile-manipulation community.

Practical Implications and Applications: The method improves collision awareness for mobile manipulators without requiring additional depth sensing, custom costmap layers, or modifications to Nav2. It is particularly relevant for applications where the effective footprint changes with arm posture or payload, such as machine tending, inspection, logistics, and service robotics.

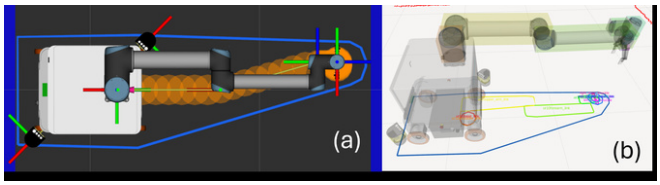

Fig. 6. Disc- and mesh-based footprint generation for the same joint configuration. (a) Disc approximation using calibrated radii at TF link origins. (b) Mesh-based convex hull from projected URDF collision geometry.

F. Limitations

The main focus of this paper is the lightweight disc-based dynamic footprint method. As shown in Fig. 6(a), it provides a simple real-time approximation, but can be conservative when TF frame origins are offset from the physical link geometry or when links are elongated. The toolbox also supports the mesh-based convex-hull variant in Fig. 6(b), improving geometric fidelity when suitable URDF collision meshes are available, but with higher processing cost. A practical limitation is that reliable Nav2 navigation with a dynamically changing footprint required careful selection and tuning of the planner, controller, costmaps, and recovery behaviours. Future work will focus on speed optimisation, planner-aware footprint prediction using future arm poses, and tighter footprint representations for constrained navigation.

VI. CONCLUSION

The results presented in this paper demonstrate that configuration-dependent footprint projection is a practical and effective approach to collision-safe navigation for mobile manipulators with both circular and polygonal base geometry. Validation of the proposed method on both circular base - TIAGo robots and polygon base - Neobotix MPO-700 with a UR10 arm, KUKA KMR iiwa confirmed that the framework operates reliably in both simulation and on physical hardware, correctly reflecting the arm's ground-plane extent across configurations without imposing a prohibitive computational burden. Also, the results suggest that the gap between static footprint parametrisation and the true collision geometry of a mobile manipulator is worth closing at the navigation layer, and would not require modifications to the mature underlying Nav2 framework by utilizing ROS2 lifecycle nodes. Future work will focus on planner-aware footprint prediction, where future arm poses are used to estimate the evolving footprint over a short horizon, enabling Nav2 planners and controllers to reason about upcoming robot geometry during navigation, while also identifying minimal parameter sets for reliable collision avoidance.

REFERENCES

[1] N. Ghodsian, K. Benfriha, A. Olabi, V. Gopinath, A. Arnou, Q. Charrier *et al.*, "Toward designing an integration architecture for a mobile manipulator in production systems: Industry 4.0," *Procedia CIRP*, vol. 109, pp. 443–448, 2022.

[2] M. Spahn, B. Brito, and J. Alonso-Mora, "Coupled mobile manipulation via trajectory optimization with free space decomposition," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 12 759–12 765.

[3] G. Rizzi *et al.*, "Robust sampling-based control of mobile manipulators for interaction with articulated objects," *IEEE Transactions on Robotics*, vol. 39, no. 3, 2023.

[4] C. C. Kemp, A. Edsinger, H. M. Clever, and B. Matulevich, "The design of stretch: A compact, lightweight mobile manipulator for indoor human environments," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 3150–3157.

[5] L. D. Riek, "Healthcare robotics," *Communications of the ACM*, vol. 60, no. 11, pp. 68–78, 2017.

[6] K. He, P. Simini, W. P. Chan, D. Kulić, E. Croft, and A. Cosgun, "On-the-go robot-to-human handovers with a mobile manipulator," in *2022 31st IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2022, pp. 729–734.

[7] R. Yang, Y. Kim, R. Hendrix, A. Kembhavi, X. Wang, and K. Ehsani, "Harmonic mobile manipulation," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 3658–3665.

[8] P. Y. Leong and N. S. Ahmad, "Exploring autonomous load-carrying mobile robots in indoor settings: A comprehensive review," *IEEE Access*, vol. 12, pp. 131 395–131 417, 2024.

[9] J.-R. Chiu, J.-P. Sleiman, M. Mittal, F. Farshidian, and M. Hutter, "A collision-free mpc for whole-body dynamic locomotion and manipulation," in *2022 international conference on robotics and automation (ICRA)*. IEEE, 2022, pp. 4686–4693.

[10] S. Chitta, I. Sucan, and S. Cousins, "MoveIt! [ROS topics]," *IEEE Robotics & Automation Magazine*, vol. 19, no. 1, pp. 18–19, 2012.

[11] B. Chen, Y. Wei, R. Liu, C. Han, H. Liu, C. Xia, L. Han, and B. Liang, "Safe expeditious whole-body control of mobile manipulators for collision avoidance," *Biomimetic Intelligence and Robotics*, p. 100301, 2026.

[12] A. Hornung, M. Phillips, E. G. Jones, M. Bennewitz, M. Likhachev, and S. Chitta, "Navigation in three-dimensional cluttered environments for mobile manipulation," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 423–429.

[13] J. L. Outón, I. Villaverde, H. Herrero, U. Esnaola, and B. Sierra, "Innovative mobile manipulator solution for modern flexible manufacturing processes," *Sensors*, vol. 19, no. 24, p. 5414, 2019.

[14] Z. Sun and G. Venture, "Dynamic footprint adjustment for navigation in human-robot side-by-side walking," in *ICRA Workshop on Advances in Social Robot Navigation: Planning, HRI, and Beyond*, 2025.

[15] E. Marder-Eppstein *et al.*, "The office marathon: Robust navigation in an indoor office environment," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, 2010.

[16] D. V. Lu, D. Hershberger, and W. D. Smart, "Layered costmaps for context-sensitive navigation," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 709–715.

[17] S. Macenski, T. Moore, D. Lu, A. Merzlyakov, and M. Ferguson, "From the desks of ROS maintainers: A survey of modern and capable mobile robotics algorithms in ROS 2," *Robotics and Autonomous Systems*, vol. 168, 2023.

[18] S. Macenski, F. Martín, R. White, and J. Clavero, "The marathon 2: A navigation system," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[19] Open Navigation LLC, "Setting up the robot's footprint," Nav2 Documentation, https://docs.nav2.org/setup_guides/footprint/setup_footprint.html, 2023.

[20] S. Macenski, M. Booker, and J. Wallace, "Open-source, cost-aware kinematically feasible planning for mobile and surface robotics," arXiv:2401.13510, 2024.

[21] G. Williams, P. Drews, B. Goldfain, J. Rehg, and E. Theodorou, "Aggressive driving with model predictive path integral control," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[22] Open Robotics, "Unified robot description format (URDF)," http://wiki.ros.org/urdf, 2023.

[23] E. Marder-Eppstein *et al.*, "NavFn: Navigation function planner," http://wiki.ros.org/navfn, 2010.

[24] K. Daniel *et al.*, "Theta*: Any-angle path planning on grids," *Journal of Artificial Intelligence Research*, vol. 39, pp. 533–579, 2010.

[25] Intelligent Robot Lab, "cs4home_core: Cognitive architecture for ROS 2," GitHub, https://github.com/CoreSenseEU/cs4home, 2023.

[26] R. Jarvis, "On the identification of the convex hull of a finite set of points in the plane," *Information Processing Letters*, vol. 2, no. 1, pp. 18–21, 1973.