Chair of Computing in Civil and Building Engineering
TUM School of Engineering and Design
Technical University of Munich

Real-Time Motion Planning for Mobile Manipulation with VAMP and QuIK

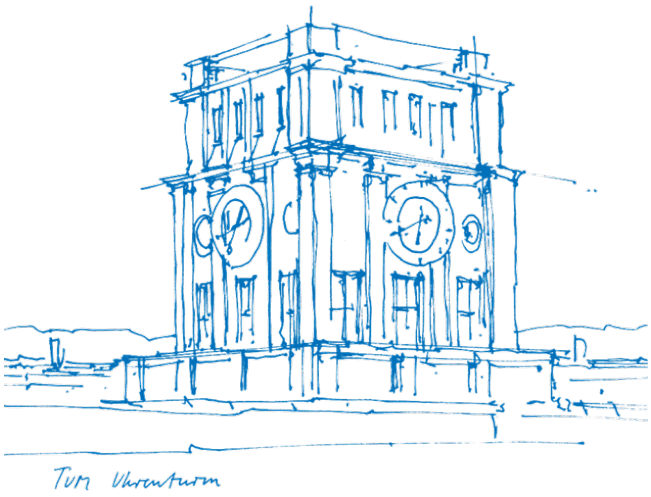
Interdisciplinary Project (IDP) - Final Report

Anil Zeybek

Examiner:
Prof. Dr.-Ing. André Borrmann

Supervisor:
Panagiotis Petropoulakis

Submitted:
Munich, September 15, 2026



Abstract

This study presents the architectural integration and performance evaluation of the VAMP (Vector-Accelerated Motion Planning) motion planner and the Quick Inverse Kinematics (QuIK) (Quick Inverse Kinematics) solver within a ROS2-native mobile manipulation framework. Focusing on a Clearpath Husky A200 platform equipped with a UFactory xArm6 manipulator, this work details the development of high-performance MoveIt2 plugins that leverage Single Instruction Multiple Data (SIMD)-accelerated collision checking and Halley's method-based inverse kinematics. A key contribution is the implementation of the Collision-Affording Point Tree (Collision-Affording Point Tree (CAPT)) data structure, enabling real-time, sensor-driven collision avoidance in dynamic environments. Experimental validation in high-fidelity Gazebo warehouse simulations and real-world experiments on the physical Husky A200 platform demonstrates that the integrated system achieves sub-3 millisecond planning latencies, a 19-fold improvement over standard Open Motion Planning Library (OMPL) with Kinematics and Dynamics Library (KDL) baselines, while ensuring robust collision avoidance in cluttered scenarios. These results confirm that the proposed framework enables the real-time planning and execution required for responsive manipulation tasks.

Contents

1 Introduction

Motion planning for robotic manipulators is a fundamental problem in robotics that involves computing collision-free trajectories from a start configuration to a goal configuration. Traditional sampling-based planners such as RRT (Rapidly-exploring Random Trees) [1] and PRM (Probabilistic Roadmaps) [2] have been widely adopted due to their probabilistic completeness and ability to handle high-dimensional configuration spaces. However, these planners can be computationally expensive, particularly when operating in cluttered environments with complex collision geometries.

Recent advances in parallel computing and vectorized algorithms have enabled significant speedups in motion planning. The VAMP (Vector-Accelerated Motion Planning) motion planner [3] is not a new planning strategy, but rather an accelerated implementation of standard sampling-based planners. VAMP supports various planning algorithms including probabilistic roadmaps (PRM) and others, but in this work we exclusively use RRT-Connect [4]. VAMP leverages SIMD instructions to perform collision checking and path planning operations in parallel, achieving planning times that are orders of magnitude faster than traditional implementations.

Similarly, inverse kinematics solvers play a critical role in robotic manipulation, converting desired end-effector poses into joint configurations. The QuIK (Quick Inverse Kinematics) [5] algorithm provides a fast and robust approach to solving Inverse Kinematics (IK) problems by applying Halley's method, a third-order root-finding technique that utilizes second-order derivative information.

In this project, we integrate both VAMP and QuIK into a ROS2-based mobile manipulation system. The target platform is a Clearpath Husky A200 mobile robot equipped with:

- UFactory xArm6 6-Degrees of Freedom (DoF) manipulator
- Robotiq 2F-85 parallel gripper
- Ouster Multi-Layer 3D-LiDAR OS0 REV 7 64 sensor
- Intel RealSense D455 depth camera

The main contributions of this work are:

1. Development of a MoveIt2 planner plugin for VAMP that supports the complete mobile manipulator kinematic chain
2. Development of a MoveIt2 kinematics plugin for QuIK that provides fast IK solutions
3. Integration of real-time point cloud processing for collision avoidance using VAMP's CAPT data structure
4. Performance evaluation comparing VAMP and QuIK against standard baselines
5. A demonstration application showcasing the integrated system in a simulated warehouse environment
6. Real-world experiments validating the system on the physical Husky A200 platform

2 Background and Related Work

2.1 Motion Planning

Motion planning algorithms can be broadly categorized into combinatorial and sampling-based approaches. Combinatorial methods compute exact solutions but are generally limited to low-dimensional spaces due to computational complexity. Examples include cell decomposition [6], visibility graphs [7], and roadmap methods [8]. Sampling-based methods were introduced by Kavraki et al. [2] and LaValle [1]. These methods trade completeness guarantees for computational efficiency by randomly sampling the configuration space.

The Rapidly-exploring Random Tree Connect (RRT-Connect) algorithm [4] is particularly popular for manipulation planning as it grows two trees simultaneously from the start and goal configurations, often finding solutions faster than single-tree variants. The OMPL library [9] provides implementations of many sampling-based planners and is integrated with MoveIt Motion Planning Framework (MoveIt) [10] as the default planning backend.

2.2 VAMP Motion Planner

VAMP [3] is a recent motion planning framework that achieves significant speedups through three key innovations:

SIMD Vectorization. Modern CPUs support SIMD operations through instruction sets such as Advanced Vector Extensions (AVX). VAMP exploits these capabilities by batching multiple collision queries into single vector operations. Where traditional planners check one configuration at a time, VAMP evaluates 8 configurations simultaneously using 256-bit vector registers.

Sphere-based Collision Model. Rather than using complex mesh geometries for collision checking, VAMP approximates each robot link as a union of spheres. This representation enables highly efficient distance computations, checking whether a sphere collides with an environment primitive reduces to simple arithmetic operations that vectorize well.

Collision-Affording Point Tree (CAPT). VAMP accelerates collision checking against sensor data using the CAPT data structure [11]. Unlike standard k-d trees which require backtracking [12], CAPT uses a linear, array-backed Eytzinger layout for cache efficiency. The key innovation is storing "affordance sets" at leaf nodes, pre-computed sets of points relevant for collision checking within each spatial cell. This design enables branch-free, SIMD-parallel queries that can check for collisions in nanoseconds.

2.3 Inverse Kinematics

Inverse kinematics solvers compute joint configurations that achieve a desired end-effector pose. The problem is fundamental to robotic manipulation but presents several challenges:

Non-uniqueness. For robots with more than 6 DoF, infinitely many joint configurations may achieve the same end-effector pose. Even for 6-DoF robots, multiple discrete solutions typically exist.

Singularities. At certain configurations, the manipulator Jacobian becomes rank-deficient, causing numerical instabilities in iterative solvers.

Joint Limits. Physical constraints on joint angles must be respected, which can make otherwise valid IK solutions infeasible.

Common approaches include analytical solutions (limited to specific robot geometries), Jacobian-based iterative methods (Newton-Raphson, Levenberg-Marquardt), and optimization-based formulations.

The QuIK algorithm [5] applies Halley's method, a third-order root-finding technique that utilizes second-order derivative information. This approach provides faster convergence than traditional Jacobian-based methods while maintaining robustness near singularities through adaptive damping.

2.4 MoveIt Framework

MoveIt [10] is the most widely used motion planning framework for Robot Operating System (ROS)-based robots. Its plugin-based architecture allows different planning and kinematics algorithms to be swapped without modifying application code. Key components include:

- **Planning Pipeline**: Orchestrates motion plan requests through configurable sequences of planners and post-processors
- **Planning Scene**: Maintains a representation of the robot and its environment for collision checking
- **Kinematics Plugin Interface**: Standardized interface for IK solvers
- **Controller Interface**: Manages trajectory execution on real or simulated hardware

MoveIt2 is the ROS2 version, providing improved real-time capabilities and better integration with the ROS2 ecosystem.

3 Methodology

3.1 VAMP Robot Model Adaptation

VAMP's performance relies on compile-time code generation that produces robot-specific collision checking routines. Adapting VAMP to a new robot requires several steps:

3.1.1 Spherized Robot Representation

The first step is converting the robot's collision geometry from arbitrary meshes to unions of spheres. The initial spherized Unified Robot Description Format (URDF) is generated procedurally using FOAM [13] and subsequently refined manually to ensure tighter bounding volumes. Figure 3.1 illustrates the resulting spherized representation. This "spherization" process involves:

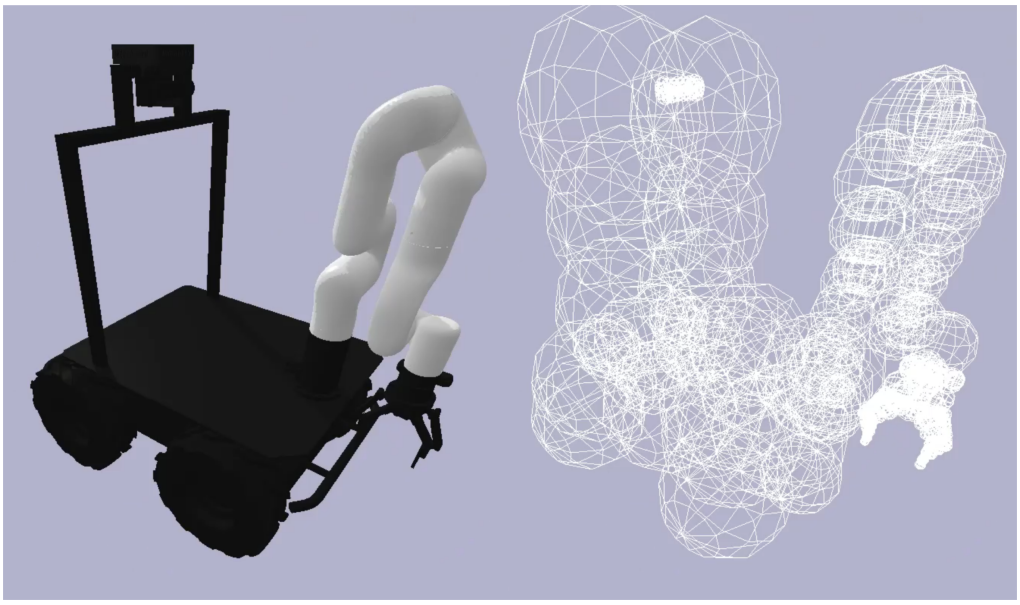


Figure 3.1 Spherized robot representation showing the union of spheres approximating the robot's collision geometry.

1. **Mesh Analysis**: Each link's collision mesh is analyzed to determine its bounding volume and geometric complexity.
2. **Sphere Fitting**: An optimization algorithm places spheres to approximate the mesh volume while minimizing both the number of spheres and the approximation error.
3. **Radius Selection**: Sphere radii are chosen to provide conservative (slightly larger) coverage of the original geometry, ensuring that any collision detected with the original mesh would also be detected with the sphere approximation.

For the Husky A200 with xArm6 system, the spherization process produced approximately 148 spheres distributed across the manipulator links, gripper, and robot base.

3.1.2 Forward Kinematics Generation

VAMP requires forward kinematics functions that compute sphere positions as functions of joint angles. These functions are generated automatically from the spherized URDF using Cricket [14], a library for tracing the forward kinematics of spherized robots. Cricket is built on Pinocchio [15] for forward kinematics. The generated code is encoded as compile-time constants, enabling:

- Loop unrolling by the compiler
- SIMD vectorization of position computations
- Elimination of runtime parsing overhead

The generated code for our robot model comprises approximately 100,000 lines, with the majority being unrolled collision checking logic.

3.1.3 Collision Pair Filtering

Not all pairs of robot links need to be checked for self-collision. The Semantic Robot Description Format (SRDF) (Semantic Robot Description Format) specifies which link pairs can be safely ignored because they:

- Are adjacent in the kinematic chain
- Never come close enough to collide due to joint limits
- Are always in collision (overlapping at rest)

For example, in the xArm6 robot, link1 and link2 cannot collide because they are adjacent links connected by a single joint, making collision geometrically impossible.

This filtering significantly reduces the number of collision checks required per configuration.

3.2 Point Cloud Integration

A key contribution of this work is the integration of real-time point cloud data for environment collision checking. The approach involves three stages:

3.2.1 Coordinate Frame Transformation

Point cloud data from the Ouster LiDAR arrives in the sensor frame. Before use in planning, points must be transformed to the robot's base frame. This transformation is computed using the TF2 library, which maintains the transformation tree between all robot frames.

3.2.2 Self-Filtering

A challenge with onboard LiDAR is that the sensor observes parts of the robot itself. These self-observation points would cause false collision detections if not removed. Our approach uses VAMP's sphere model to filter points:

1. Query the robot's current joint configuration
2. Compute sphere positions using forward kinematics
3. Remove any point that falls within any robot sphere (plus a safety margin)

This filtering runs each time a new point cloud is received, ensuring the collision environment remains consistent with the robot's current configuration.

3.2.3 CAPT Construction

The point cloud is first processed by a space-filling curve filter, which removes redundant data to reduce the effective point cloud size by approximately 80–90% while preserving structure. This filtered cloud is then organized into the Collision-Affording Point Tree (CAPT). Unlike standard k-d trees which require backtracking, CAPT uses a linear, array-backed Eytzinger layout [16] for cache efficiency. The key innovation is storing "affordance sets" at leaf nodes, pre-computed sets of points relevant for collision checking within each spatial cell.

The tree is constructed through three main steps:

1. **Partitioning**: Recursively organizing points into spatial cells.
2. **Affordance Computation**: Creating affordance sets for each cell by duplicating and storing local points relevant for collision checking.
3. **Layout**: Arranging the structure in an Eytzinger layout (linear, array-backed) for optimal cache performance and branch-free traversal.

During planning, the CAPT enables branch-free, SIMD-parallel queries that can check for collisions in nanoseconds.

3.2.4 Dynamic Environment Updates

A key advantage of the CAPT-based approach is its support for dynamic obstacles. VAMP continuously monitors point cloud updates from the sensor in a separate asynchronous thread. When new point cloud data arrives, the system automatically reconstructs the CAPT structure to reflect the current environment state. This asynchronous processing ensures that planning queries always operate on the most recent sensor data, enabling the system to handle moving obstacles and changing environments without requiring explicit replanning triggers. The thread-safe mechanism for sharing the CAPT between the point cloud processing thread and the planning thread ensures that collision checking remains accurate even as the environment evolves.

3.3 QuIK Integration

The QuIK inverse kinematics solver was integrated through a service-based architecture, chosen for its flexibility and debugging capabilities.

3.3.1 Denavit-Hartenberg Parameterization

QuIK requires the robot kinematics to be specified using Denavit-Hartenberg (DH) parameters [17]. For the xArm6, these parameters were derived from the manufacturer's documentation and validated against the URDF model:

Table 3.1 xArm6 Denavit-Hartenberg Parameters [17]

Joint	a_i (m)	α_i (rad)	d_i (m)	θ_i offset (rad)
1	0	$-\pi/2$	0.267	0
2	0.2897	0	0	-0.186
3	0.0775	$-\pi/2$	0	0.186
4	0	$\pi/2$	0.3425	0
5	0.076	$-\pi/2$	0	0
6	0	0	0.097	0

Note that joint 2 has a non-standard offset angle due to the xArm6's mechanical design, where the upper arm is not aligned with the shoulder axis at zero position.

3.3.2 Tool Frame Configuration

The end-effector frame used for IK must account for the gripper and any additional tooling. A tool transformation matrix is configured to offset from the xArm6's flange frame to the gripper's fingertip center, a distance of approximately 0.248 m along the tool axis.

3.3.3 Algorithm Configuration

The QuIK solver was configured with parameters optimized for the xArm6's workspace:

- **Maximum iterations**: 200 (sufficient for convergence in typical cases)
- **Exit tolerance**: 10^{-12} (sub-millimeter position accuracy)
- **Damping factor**: 10^{-10} (light damping for singularity robustness)

3.3.4 MoveIt2 Plugin Architecture

The kinematics plugin wraps the QuIK service with MoveIt2's expected interface. Key design considerations include:

- **Fallback Mechanism**: If the QuIK service is unavailable, the plugin falls back to the KDL solver, ensuring system robustness.
- **Asynchronous Communication**: Service calls are made asynchronously with configurable timeouts, preventing planning from blocking indefinitely.
- **Seed Handling**: The initial joint configuration (seed) significantly affects IK convergence. The plugin passes the current robot state as the seed, promoting solutions near the current configuration.

3.4 Motion Planning Pipeline

The complete motion planning pipeline integrates all components:

1. **Goal Specification**: The application specifies a target pose for the end-effector.
2. **Inverse Kinematics**: The QuIK plugin converts the pose to a goal joint configuration.
3. **Environment Update**: The latest CAPT is loaded into the planning environment.
4. **Path Planning**: VAMP's accelerated implementation of RRT-Connect finds a collision-free path from current to goal configuration.
5. **Path Simplification**: Unnecessary waypoints are removed through shortcutting.
6. **Time Parameterization**: Velocities and accelerations are computed respecting joint limits.
7. **Execution**: The trajectory is sent to the robot controllers.

3.4.1 Error Handling and Recovery

Planning can fail due to infeasible goals, cluttered environments, or IK failures. The system implements several recovery strategies:

- **Goal Perturbation**: If planning fails, small random perturbations are applied to the goal pose and planning is reattempted.
- **Multiple IK Seeds**: Different seed configurations are tried to find alternative IK solutions.
- **Execution Monitoring**: During trajectory execution, the system continuously validates that the remaining path is still collision-free, triggering replanning if obstacles appear.

4 System Architecture

4.1 Hardware Configuration

The target platform consists of a Clearpath Husky A200 mobile base with manipulation and sensing capabilities. Table 4.1 summarizes the hardware components.

Table 4.1 System Hardware Components

Component	Model	Key Specifications
Mobile Base	Clearpath Husky A200	4-wheel differential drive, 75 kg payload
Manipulator	UFactory xArm6	6-DoF, 5 kg payload, 700 mm reach
Gripper	Robotiq 2F-85	85 mm stroke, parallel jaw
3D LiDAR	Ouster OS0 Rev 7 64	64 channels, 120 m range, 10-20 Hz
Depth Camera	Intel RealSense D455	RGB-D, 1280x720, 90 fps

The complete kinematic chain spans from the mobile base through the manipulator to the gripper end-effector. The URDF model defines 6 actuated joints for the xArm6 manipulator, with additional joints for the gripper mechanism.

4.2 Software Architecture

The software architecture follows a layered design built on ROS2 Humble. Figure 4.1 illustrates the main components and their interactions.

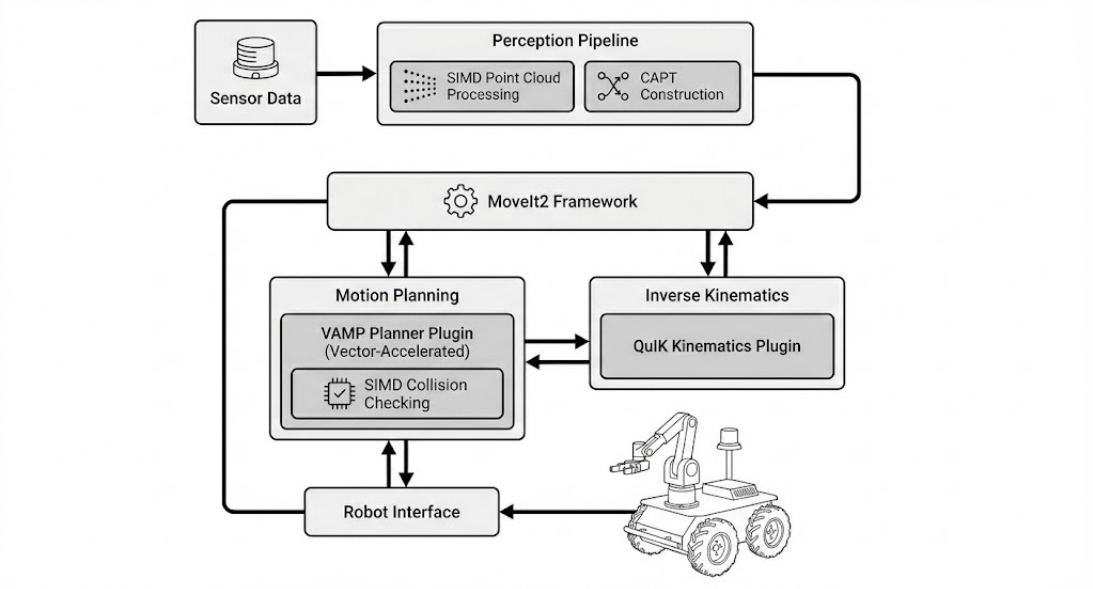


Figure 4.1 Layered software architecture showing the integration of VAMP and QuIK with MoveIt2.

The key design decisions in this architecture are:

1. **Plugin-based Integration**: Both VAMP and QuIK are integrated as MoveIt2 plugins, allowing them to be used with any MoveIt2-compatible application without code changes.

2. **Asynchronous Point Cloud Processing**: Point cloud data is processed in a separate callback thread, with the resulting CAPT structure shared with the planner through thread-safe mechanisms. This design enables dynamic obstacle handling, as VAMP continuously monitors sensor updates and reconstructs the CAPT to reflect changes in the environment, allowing the planner to operate on the most recent sensor data.

3. **Service-based IK**: The QuIK solver runs as a standalone ROS2 service, decoupling the IK computation from the planning process and enabling potential distribution across multiple machines.

5 Experiments and Results

5.1 Experimental Setup

The system was evaluated in both Gazebo simulation and real-world experiments on the physical Husky A200 platform.

5.1.1 Simulation Experiments

Both simulation and real-world experiments share the same experimental methodology. The experimental setup involved executing a set of 10 distinct movements, with each movement repeated 3 times, resulting in a total of 30 planning queries per configuration. All movements and starting points were identical across all experimental configurations to ensure fair comparison. For each planning query, the planning time was recorded. The results presented in the following sections represent the average planning times across these trials.

The experiments evaluated four different planning configurations:

- VAMP + QuIK
- VAMP + KDL
- OMPL with RRT-Connect + QuIK
- OMPL with RRT-Connect + KDL

The simulation runs were executed on a system equipped with an AMD Ryzen 7 PRO 7840U processor (8 cores, 16 threads, max frequency 5.13 GHz) with AVX-512 support, enabling efficient execution of the SIMD-accelerated VAMP planner.

5.1.2 Real-World Experiments

Real-world experiments were conducted on the physical Clearpath Husky A200 mobile manipulator platform, following the same experimental methodology described above.

Note on Point Cloud Collision Environment: Due to technical issues with the LiDAR sensor, none of the real-world experiments utilized point cloud collision checking. This limitation is important to note because collidable point cloud environments typically cause planning to be slower, and this is precisely where VAMP's CAPT-based collision checking demonstrates its greatest advantage. With point cloud collision checking enabled, the performance gap between VAMP and traditional planners would be expected to be even more pronounced, as VAMP's SIMD-accelerated collision queries against the CAPT data structure are specifically optimized for handling dense sensor data efficiently.

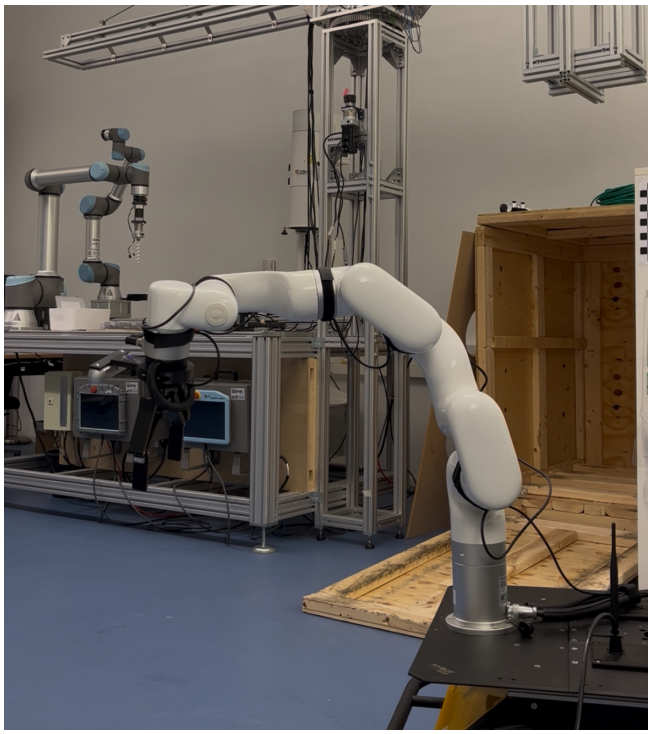


Figure 5.1 Real-world experimental setup showing the physical Clearpath Husky A200 mobile manipulator platform with the UFactory xArm6 manipulator during planning experiments.

5.1.3 Reaction Time Experiments

To evaluate the system's ability to respond to dynamic changes in the environment, reaction time experiments were conducted in Gazebo simulation. The goal of these experiments was to compare VAMP with OMPL on their ability to detect and react to new obstacles that appear during the execution of a planned trajectory. This metric is critical for real-world applications where the environment may change unexpectedly, requiring rapid replanning to avoid collisions.

The experimental setup consisted of:

- A table surface at height 1.05 m
- A target object (brick) placed on the table
- The Husky A200 with xArm6 positioned facing the table
- Simulated Ouster LiDAR providing point cloud data

During execution of a planned trajectory, a new obstacle was introduced into the scene. The reaction time was measured as the duration from when the obstacle appeared in the LiDAR point cloud to when the system detected the collision and initiated replanning.

Results: The VAMP-based system with CAPT collision checking achieved a reaction time of approximately **70 ms**, while the standard OMPL with KDL configuration required approximately **1.4 seconds** to detect and react to the new obstacle. This represents a $\sim$20$\times$ improvement in reaction time, demonstrating VAMP's advantage for dynamic environments where rapid replanning is essential.

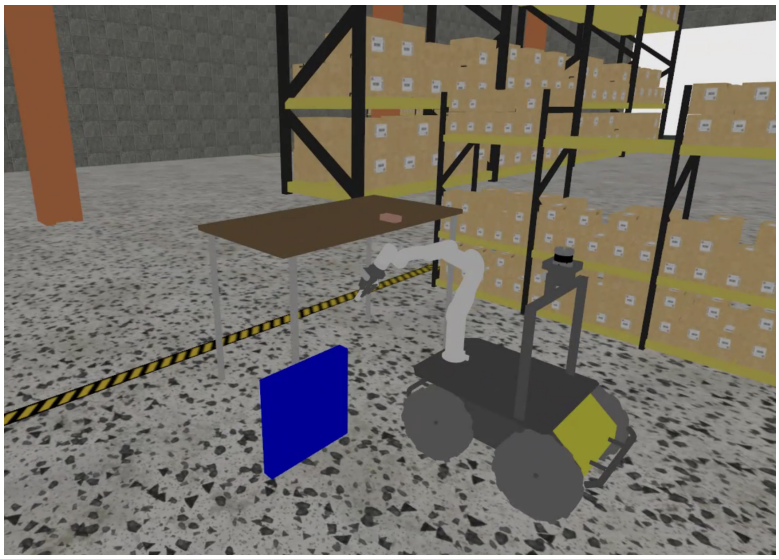


Figure 5.2 Simulation experimental setup showing the Gazebo warehouse environment with the Husky A200 mobile manipulator, table, and target object used for reaction time experiments.

5.2 Planning Performance

5.2.1 Simulation Planning Performance

Table 5.1 compares VAMP with QuIK against OMPL's RRT-Connect implementation with both KDL and QuIK IK solvers. All measurements were performed in the Gazebo simulation environment.

Table 5.1 Motion planning performance comparison in Gazebo simulation (30 trials)

Configuration	OMPL + KDL	OMPL + QuIK	VAMP + KDL	VAMP + QuIK
Average Planning Time	26.2 ms	24.1 ms	2.0 ms	1.3 ms

5.2.2 Real-World Planning Performance

Real-world planning performance was evaluated using the experimental setup described in Section 5.1.2. Planning times were averaged across the 30 trials for each of the four configurations. As noted in the experimental setup, these experiments were conducted without point cloud collision checking due to LiDAR sensor issues. VAMP's performance advantages are expected to be even more significant when point cloud collision environments are enabled, given its optimized CAPT-based collision checking.

Table 5.2 presents the average planning times for all four configurations tested on the physical Husky A200 platform.

Table 5.2 Motion planning performance comparison in real-world experiments (30 trials)

Configuration	OMPL + KDL	OMPL + QuIK	VAMP + KDL	VAMP + QuIK
Average Planning Time	30.1 ms	29.6 ms	2.4 ms	1.7 ms

The real-world results demonstrate similar performance characteristics to the simulation experiments, with VAMP maintaining its substantial speedup over traditional planners even on physical hardware. The performance improvements are consistent across both IK solver choices, confirming that VAMP's acceleration comes primarily from its vectorized collision checking rather than IK computation time.

5.2.3 Key Observations

Comparing the integrated VAMP + QuIK system against the OMPL + KDL baseline (the starting configuration for this project), the performance improvements are substantial when averaged across both simulation and real-world experiments:

- **Baseline Performance (OMPL + KDL)**: 28.15 ms average planning time (26.2 ms in simulation, 30.1 ms in real-world)
- **Optimized System (VAMP + QuIK)**: 1.5 ms average planning time (1.3 ms in simulation, 1.7 ms in real-world)
- **Overall Speedup**: $\sim$18.8$\times$ faster planning compared to the original baseline

This nearly 19-fold improvement in planning performance demonstrates the combined benefits of VAMP's SIMD-accelerated collision checking and QuIK's fast inverse kinematics solver, enabling real-time motion planning suitable for reactive manipulation tasks.

5.3 Inverse Kinematics Performance

Table 5.3 compares QuIK against the default KDL solver.

Table 5.3 Inverse kinematics solver comparison (100 random poses)

Metric	QuIK	KDL
Mean Solution Time	0.8 ms	2.1 ms
Success Rate	98%	95%
Mean Iterations	12	45
Position Error	$< 10^{-6}$ m	$< 10^{-4}$ m

QuIK's third-order convergence (via Halley's method) results in fewer iterations and faster solutions while achieving higher accuracy.

Real-world IK performance was validated through the planning experiments described in Section 5.1.2, which evaluated both QuIK and KDL solvers. The consistent performance improvements observed in the real-world planning results (Table 5.2) confirm that QuIK's faster convergence translates to practical benefits in physical robot deployments, with both VAMP + QuIK and OMPL + QuIK configurations showing improved planning times compared to their KDL counterparts.

5.4 Point Cloud Processing

The CAPT construction and self-filtering performance was measured:

- **Input point cloud size**: 65,536 points (typical Ouster OS0 Rev 7 scan)
- **Points after self-filtering**: 58,000–62,000 points
- **CAPT construction time**: 8–12 ms
- **Effective points during planning**: 5,000–10,000 (after CAPT pruning)

The CAPT structure reduces the effective collision checking burden by approximately 85%.

6 Discussion

6.1 Advantages of the Integrated System

The combination of VAMP and QuIK provides several advantages for mobile manipulation:

Reactive Planning. With planning times under 100 ms, the system can respond quickly to changing goals or environments. This enables applications such as visual servoing where targets may move during manipulation.

Sensor-Based Collision Avoidance. The integration of point cloud data allows the robot to avoid obstacles not present in the static world model. This is essential for operation in unstructured environments.

Predictable Performance. VAMP's low variance in planning time makes it suitable for real-time applications with timing constraints.

6.2 Limitations and Challenges

Robot Model Complexity. Generating the VAMP robot model requires significant effort, including mesh spherization and code generation. Changes to the robot configuration require regenerating this model.

Approximation Conservatism. The sphere-based collision model is inherently conservative, potentially rejecting valid paths that would be accepted with exact geometry checking.

Service Latency. Utilizing VAMP and QuIK via MoveIt introduces MoveIt-related communication overheads, which can sometimes exceed the duration of the planning queries themselves. For applications requiring maximum performance, a direct integration bypassing MoveIt would be preferable. However, this project prioritizes integration with MoveIt as it provides rich visualizations in RViz and represents the standard approach for motion planning in the ROS ecosystem.

6.3 Future Work

The natural next step for this project is **grasp planning integration**. While the current system provides fast and reactive motion planning, completing a full pick-and-place pipeline requires the ability to compute stable grasps for target objects. This would involve integrating a grasp planning module that analyzes object geometry from sensor data, generates candidate grasp poses, and selects optimal grasps based on task requirements and kinematic feasibility. Combined with the sub-3 ms motion planning capabilities demonstrated in this work, grasp planning integration would enable the Husky A200 platform to perform complete autonomous manipulation tasks in dynamic environments.

7 Conclusion

This project successfully integrated the VAMP motion planner and QuIK inverse kinematics solver into a ROS2-based mobile manipulation system for the Clearpath Husky A200 with UFactory xArm6 manipulator.

The key achievements are:

1. **VAMP Integration**: A complete MoveIt2 planner plugin was developed, including robot model generation, point cloud processing, and CAPT-based collision checking. Averaged across simulation and real-world experiments, planning times were reduced from 28.15 ms (OMPL + KDL) to 1.5 ms (VAMP + QuIK), achieving a $\sim$19$\times$ speedup. In simulation alone, VAMP + QuIK achieved 1.3 ms planning times compared to 26.2 ms for the baseline.

2. **QuIK Integration**: A MoveIt2 kinematics plugin was developed using a service-based architecture. The QuIK solver provides 2.6$\times$ faster IK solutions with improved accuracy.

3. **Real-Time Point Cloud Processing**: The system processes LiDAR data in real-time, filtering self-observations and constructing efficient collision checking structures.

4. **Dynamic Async Planning**: The system supports dynamic obstacles through asynchronous point cloud processing, where VAMP continuously monitors sensor updates in a separate thread and reconstructs the CAPT structure to reflect environment changes, enabling real-time collision avoidance in dynamic environments.

5. **System Demonstration**: The integrated system was validated in both simulation and real-world experiments, successfully completing manipulation tasks with sub-100ms planning times on the physical Husky A200 platform.

The results demonstrate that modern vectorized algorithms can provide substantial performance improvements for mobile manipulation while maintaining compatibility with standard robotics frameworks. The plugin-based architecture ensures that these improvements are accessible to any application built on MoveIt2. The asynchronous dynamic planning capability enables the system to operate effectively in changing environments, making it suitable for real-world applications where obstacles may move or appear during task execution. Real-world experiments on the physical Husky A200 platform confirmed the system's performance and robustness in actual deployment scenarios.

Acronyms

AVX Advanced Vector Extensions. 2

CAPT Collision-Affording Point Tree. ii, 1, 2, 6, 7, 10–14, 16

DH Denavit-Hartenberg. 6

DoF Degrees of Freedom. 1, 2, 9

IK Inverse Kinematics. 1–3, 7, 8, 10, 13, 14, 16

KDL Kinematics and Dynamics Library. ii, 7, 11–14, 16

MoveIt MoveIt Motion Planning Framework. 2, 3, 15

OMPL Open Motion Planning Library. ii, 2, 11–14, 16

QuIK Quick Inverse Kinematics. ii, 1, 3, 6, 7, 9–11, 13–16

ROS Robot Operating System. 3, 15

RRT-Connect Rapidly-exploring Random Tree Connect. 2

SIMD Single Instruction Multiple Data. ii, 1, 2, 5, 6, 11, 14

SRDF Semantic Robot Description Format. 5

URDF Unified Robot Description Format. 4–6, 9

Bibliography

[1] Steven LaValle. “Rapidly-exploring random trees: A new tool for path planning”. In: *Research Report 9811* (1998).

[2] Lydia E Kavraki et al. “Probabilistic roadmaps for path planning in high-dimensional configuration spaces”. In: *IEEE transactions on Robotics and Automation* 12.4 (2002), pp. 566–580.

[3] Wil Thomason, Zachary Kingston, and Lydia E. Kavraki. *Motions in Microseconds via Vectorized Sampling-Based Planning*. 2023. arXiv: 2309.14545 [cs.RO]. URL: https://arxiv.org/abs/2309.14545.

[4] James J Kuffner and Steven M LaValle. “RRT-connect: An efficient approach to single-query path planning”. In: *Proceedings 2000 ICRA. Millennium conference. IEEE international conference on robotics and automation. Symposia proceedings (Cat. No. 00CH37065)*. Vol. 2. IEEE. 2000, pp. 995–1001.

[5] Steffan Lloyd, Rishad A Irani, and Mojtaba Ahmadi. “Fast and robust inverse kinematics of serial robots using Halley’s method”. In: *IEEE Transactions on Robotics* 38.5 (2022), pp. 2768–2780.

[6] Jacob T Schwartz and Micha Sharir. “On the “piano movers” problem. II. General techniques for computing topological properties of real algebraic manifolds”. In: *Advances in applied Mathematics* 4.3 (1983), pp. 298–351.

[7] Tomás Lozano-Pérez and Michael A Wesley. “An algorithm for planning collision-free paths among polyhedral obstacles”. In: *Communications of the ACM* 22.10 (1979), pp. 560–570.

[8] John Canny. *The complexity of robot motion planning*. MIT press, 1988.

[9] Ioan A Sucan, Mark Moll, and Lydia E Kavraki. “The open motion planning library”. In: *IEEE Robotics & Automation Magazine* 19.4 (2012), pp. 72–82.

[10] Sachin Chitta, Ioan Sucan, and Steve Cousins. “Moveit![ros topics]”. In: *IEEE robotics & automation magazine* 19.1 (2012), pp. 18–19.

[11] Clayton W. Ramsey et al. *Collision-Affording Point Trees: SIMD-Amenable Nearest Neighbors for Fast Collision Checking*. 2024. arXiv: 2406.02807 [cs.RO]. URL: https://arxiv.org/abs/2406.02807.

[12] Jon Louis Bentley. “Multidimensional binary search trees used for associative searching”. In: *Communications of the ACM* 18.9 (1975), pp. 509–517.

[13] Sai Coumar et al. *Foam: A Tool for Spherical Approximation of Robot Geometry*. 2025. arXiv: 2503.13704 [cs.RO]. URL: https://arxiv.org/abs/2503.13704.

[14] CoMMALab. *Cricket: Tracing Compilation for Spherized Robots*. https://github.com/CoMMALab/cricket. Accessed: 2024. 2024.

[15] Justin Carpentier et al. “The Pinocchio C++ library – A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives”. In: *IEEE International Symposium on System Integrations (SII)*. 2019.

[16] Paul-Virak Khuong and Pat Morin. “Array layouts for comparison-based searching”. In: *Journal of Experimental Algorithmics (JEA)* 22 (2017), pp. 1–39.

[17] Peter I Corke. “A simple and systematic approach to assigning Denavit–Hartenberg parameters”. In: *IEEE transactions on robotics* 23.3 (2007), pp. 590–594.