Verification and Validation Report: 2D-RAPP

Ziyang Fang

April 18, 2025

1 Revision History

Date	Version	Notes
April 16, 2025	1.0	Initial version of the V&V Report

2 Symbols, Abbreviations and Acronyms

symbol	description
A	Assumption
DD	Data Definition
GD	General Definition
GS	Goal Statement
IM	Instance Model
LC	Likely Change
PS	Physical System Description
R	Requirement
SRS	Software Requirements Specification
TM	Theoretical Model
IK	Inverse Kinematics
FK	Forward Kinematics
A*	A-star Pathfinding Algorithm
DOF	Degrees of Freedom
EE	End-Effector
2D-RAPP	2D Robot Arm Path Planning

Contents

3 Functional Requirements Evaluation

Collision-Free Path Generation

Test ID	Description	Result
T1	Basic collision-free path planning	Pass
T2	Path planning with multiple obstacles	Pass

Comments: The path planner module reliably generates valid, collision-free trajectories. The A* algorithm performs well on a toroidal joint-space grid.

Inverse Kinematics Solver Validation

Test ID	Description	Result
T3	Feasibility of IK solution	Pass
T4	IK for complex configurations	Pass

Comments: The IK solver computes valid and optimal solutions, even in redundant configurations.

4 Nonfunctional Requirements Evaluation

4.1 Performance

Test ID	Description	Result
N1	Planning under high obstacle density	Pass
N2	Scalability with increased DOF	Pass

Comments: The system maintains real-time performance and memory usage within acceptable limits.

5 Comparison to Existing Implementation

Not applicable.

6 Unit Testing

Test ID	Module	Coverage	Result
U1	Collision Detection	100%	Pass
U2	IK Solver	100%	Pass
U3	Path Planner	100%	Pass

Comments: Unit tests cover all edge cases, and functional outputs match expectations under various scenarios.

7 Changes Due to Testing

Based on using experience and personal feedback:

- Refined obstacle representation for better precision.
- Clarified collision detection: defined tangent cases as non-colliding.
- Improved GUI visualization of configurations and trajectories.

8 Automated Testing

The following tools were employed for continuous integration and quality assurance:

- **pytest** for automated unit testing.
- **coverage.py** to ensure high code coverage.
- **flake8** for code style and static analysis.
- **GitHub Actions** for CI on every commit.

9 Trace to Requirements

Requirement	Test Case(s)	Status
FR1: Obstacle avoidance	T1	Pass
FR2: Multiple obstacles	T2	Pass
FR3: IK feasibility	T3, T4	Pass
NFR1: Performance	N1, N2	Pass

10 Trace to Modules

Module	Test IDs	Status
Collision Detection	U1	Pass
IK Solver	U2	Pass
Path Planner	U3	Pass

11 Code Coverage Metrics

Module	Statements	Missed	Coverage
astar_planner.py	63	0	100%
collision.py	40	0	100%
joint_limits.py	25	0	100%
nlink_arm.py	75	0	100%

Total Coverage: 100%
Comments: All core modules are fully tested and verified.

Appendix — Reflection

The system design and testing activities were guided by the requirements defined in the SRS (**?**), while the modular architecture was described in the MG (**?**) and detailed in the MIS (**?**). The current report follows the methodology outlined in the V&V plan (**?**).

1. The testing framework and modular design made this deliverable smooth. The tests matched well with the planned architecture.

2. Some edge cases in collision detection were challenging (e.g., tangent contacts). We resolved this by refining geometric definitions and test logic.

3. Peer and supervisor feedback helped shape key sections like test case design and UI presentation; the rest followed internal planning.

4. The actual V&V activities closely followed the plan. Minor modifications were introduced during execution (e.g., visual tweaks, detection precision), which emerged from real testing scenarios. In future projects, allocating time for such edge refinement would be beneficial.