2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Manipulator Generative Design Optimization for Orchard Environments

Marcus Rosette[1], Tianhai Wang[2], James Burridge[2], Wei Guo[2], and Joseph R. Davidson[1]

***Abstract*— Manipulators are essential for orchard robotics tasks such as pruning and harvesting, which require precise, dexterous motion in cluttered and unstructured environments. Off-the-shelf industrial arms, while readily available, often lack the reach and dexterity required for these settings. In this paper, we present a simulation-driven, multi-objective optimization framework for task-specific manipulator kinematics, leveraging the NSGA-II evolutionary algorithm and physics-based evaluation. Candidate designs are encoded with high-level parameters – joint type, axis orientation, link length, and joint count – then automatically generated as URDF models and evaluated in simulation for reachability, manipulability, torque demand, and motion planning cost. Trade-offs are revealed on a Pareto front, enabling exploration across diverse designs. The framework is demonstrated on a real-world tree pruning task, using collected 3D scans of expert-pruned trees and an automated prune point identification pipeline to generate target points to guide the optimization. Results show that the proposed approach produces task-specific manipulator designs with improved workspaces and reduced operational constraints compared to a commercial industrial arm, offering a viable pathway toward deployable agricultural manipulation systems.**

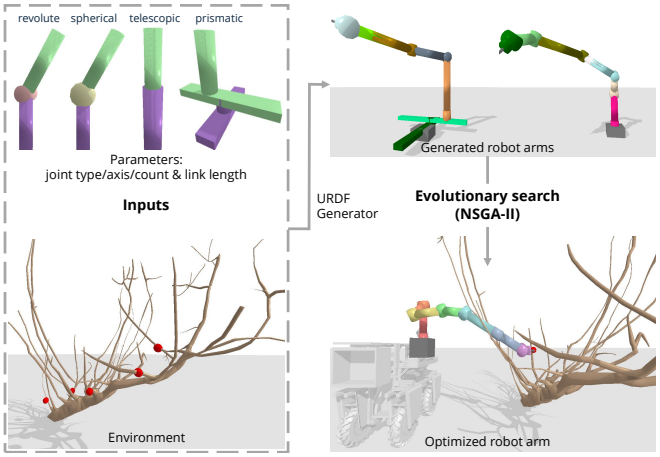

Fig. 1. The system takes as input the joint types, joint axes, link lengths, and number of joints/links, along with the environment model and target reachable points (e.g., a tree mesh with designated prune points). The NSGA-II–based generative design process iteratively searches for the optimal kinematic structure for operating within the target task space.

I. Introduction

There is a growing gap between labor demand and supply in specialty crop agriculture; labor shortages, rising wages, and dependency on seasonal labor threaten the economic viability of tasks that still require human dexterity and care [1]. While agricultural robots have been commercially adopted for some tasks such as weeding and spraying, dexterous manipulation in unstructured outdoor environments remains a major challenge. Tree fruit production, in particular, requires delicate and precise interaction with branches, stems, and fruit; tasks like pruning, thinning, and harvesting demand high dexterity, adaptability, and collision-free motion in cluttered and restricted spaces [2]. Robots designed for these types of labor-intensive tasks have not yet transitioned from the lab to widespread use on the farm.

Previous work in dexterous agricultural manipulation has often adopted general-purpose, off-the-shelf robotic arms, similar to the setup shown in Fig. 2. While utilizing such systems reduces development overhead, they are typically designed for indoor environments. In orchard environments, however, robots must contend with significant variability in orchard design (e.g., tree height, tree spacing, and row spacing), tree architecture, canopy density, etc. A mismatch between the robot's kinematic design and the orchard environment often leads to overly constrained workspaces, inefficient motion, higher energy consumption, and an increased risk of collisions. As a result, current designs struggle to meet the speed, reliability, and cost-effectiveness required for commercial adoption.

A promising alternative is to design the robot's kinematic structure specifically for the agricultural task and setting. This shifts the focus from adapting existing hardware to optimizing robot parameters (e.g., number of joints, joint types, link lengths, etc.) for the intended environment. However, determining an optimal design is challenging because tasks often involve conflicting objectives. For example, one may need to maximize the reachable workspace while minimizing joint torque, reduce joint count while maintaining dexterity, or improve path planning success while minimizing configuration changes. The search space is large, highly nonlinear, and strongly dependent on environmental constraints. This complexity makes multi-objective optimization methods particularly attractive.

This paper presents a multi-objective generative design optimization framework for robot manipulators, shown in Fig. 1, and applies it to a robotic pruning task in modern fruit tree orchards. Our method uses the NSGA-II evolutionary algorithm to explore a wide range of kinematic configurations in full 3D, with the option to seed the search with baseline

*This work was supported by the Washington Tree Fruit Research Commission and the AI Research Institutes program supported by NSF and USDA-NIFA under the AI Institute: Agricultural AI for Transforming Workforce and Decision Support (AgAID) Award No. 2021-67021-35344.

[1]Collaborative Robotics and Intelligent Systems (CoRIS) Institute, Oregon State University, Corvallis OR 97331, USA {rosettem, joseph.davidson}@oregonstate.edu

[2]Graduate School of Agricultural and Life Sciences, University of Tokyo, Tokyo, Japan {tianhai.wang@fieldphenomics.com; {burridge-j, guowei}@g.ecc.u-tokyo.ac.jp

Fig. 2. Real world dormant tree pruning (Prosser, WA, USA; January 2025) with an industrial UR5e robot arm mounted to a ground robot. Baseline example of an agricultural task to guide robot arm optimization.

designs. Candidate kinematic parameters are evaluated by dynamically generating URDF arm models, simulating their operation in a realistic orchard environment, and measuring performance across multiple objectives. The result is a Pareto front of candidate designs, enabling informed decisions about trade-offs between competing performance metrics. We evaluate our approach on a real-world pruning dataset derived from 3D scans of sweet cherry trees before and after pruning by a subject matter expert. Our contributions are: i) **a general-purpose evolutionary design approach** for optimizing manipulator kinematics and statics using NSGA-II in full 3D; and ii) **evaluation of the optimization framework on a real-world orchard pruning use case** that incorporates realistic 3D scans of fruit trees from a commercial orchard.

II. Prior Work

A. Manipulator Co-Design and Optimization

Manipulator co-design involves the simultaneous optimization of structural and control parameters to meet task-specific requirements. Prior analytical methods used brute-force enumeration of candidate kinematic configurations, evaluating workspace coverage and manipulability to select optimal designs [3], and Jacobian-based formulations to optimize link lengths, joint placement, and joint types under kinematic constraints [4]. With advances in computational power, simulation-aided co-design methods have emerged. These integrate physical simulation with optimization to jointly tune link lengths, actuator properties, and joint placement for improved efficiency and torque minimization [5], [6], [7]. Beyond conventional link geometries, unconstrained shape optimization has been explored, redistributing mass along the link [8] or parameterizing shapes using Bézier curves to reduce collisions in cluttered workspaces [9]. Frameworks such as GlobDesOpt [10] have made simulation-based methods more accessible, though they typically require a user-defined baseline configuration. More recent advances, such as MORPH [11] and RoboGrammar [12], use reinforcement learning or graph grammars to explore morphological design spaces without predefined topologies.

While these general approaches can produce optimized designs, they often assume simplified environments, planar workspaces, or incorporate partial parameter optimization. Fully integrated, 3D, task-specific kinematic structure generation with environmental collision constraints remains under-explored—particularly for irregular and cluttered agricultural settings.

B. Manipulator Design Optimization in Agriculture

Agricultural manipulation requires precise, collision-aware motion in complex, semi-structured environments. Prior work has addressed this challenge through task-specific kinematic optimization for greenhouse tomato pruning [13], vineyard pruning [14], and cucumber harvesting [15]. Most of these studies optimized only a limited set of parameters, such as link lengths for a fixed number of degrees of freedom, rather than exploring a broader design space of robot parameters. Multi-objective formulations balancing dexterity, reachability, and collision avoidance have also been explored [16], [17], but typically assumed a predefined manipulator configuration.

Beyond kinematic tuning, several works have considered system-level or environment-aware optimization. For example, Arikapudi and Vougioukas [18] optimized the placement of telescopic arms in multi-robot harvesting systems, while Levin and Degani [19] investigated modular link designs to expand workspace coverage. Broader environment-robot co-optimization has been studied, with evaluations of manipulator performance across multiple tree geometries and trellis configurations highlighting the value of joint robot-environment design [20].

Despite this progress, agricultural manipulator optimization remains constrained in three ways:

1) Most approaches assume a known baseline manipulator structure.
2) Environment modeling is often simplified to planar or partially planar conditions.
3) Optimization typically considers a narrow set of objectives, neglecting trade-offs between competing performance metrics.

As a result, there is still no agricultural-focused framework that can automatically generate novel kinematic structures in full 3D while explicitly accounting for crop geometry and task-specific targets such as pruning or harvesting points.

C. Evolutionary and Multi-Objective Optimization Methods

Evolutionary algorithms (EAs) have proven effective for manipulator design optimization due to their ability to search large, nonlinear, and discontinuous design spaces. Early implementations focused on planar manipulators, evolving

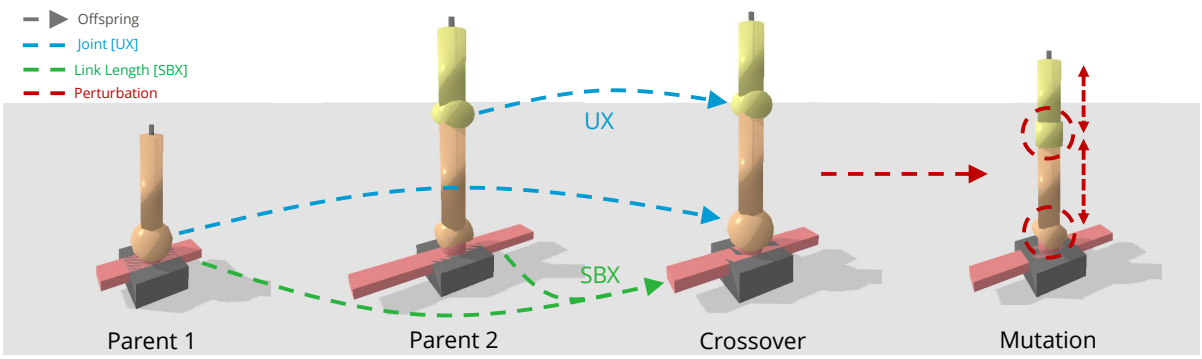

Fig. 3. Illustration of genetic operators in the evolutionary generative design of candidate manipulators. Two randomly sampled parent encodings (joint type, axis, and link length) undergo Uniform Crossover (UX), indicated by blue dashed arrows for integer variables, and Simulated Binary Crossover (SBX), indicated by green dashed arrows for floating-point variables, to produce an offspring inheriting traits from both. A subsequent mutation, highlighted in red, introduces a small change to the genome, slightly altering the arm's geometry.

link lengths, joint types, and DOF to maximize workspace or minimize torque, weight, and power consumption [21]. Others targeted constrained environments, minimizing tracking error, collisions, and arm length [22]. Multi-objective approaches are particularly well-suited for manipulation tasks with conflicting requirements, such as maximizing reachability while minimizing joint torque, or improving manipulability while reducing joint count. This methodology has been applied in both agricultural [14], [16] and industrial contexts [23].

Recent trends integrate physics simulation within the optimization loop, enabling more realistic evaluation of collision risk, actuator effort, and workspace accessibility; however, few works combine evolutionary, multi-objective optimization with full 3D environmental modeling and automated kinematic structure generation from scratch. This integration is particularly critical for agricultural robotics, where environmental clutter and irregularity demand designs tuned to specific field conditions.

III. Manipulator Optimization Framework

This paper presents a simulation-driven optimization framework for automated manipulator design. The core component is an evolutionary algorithm–based search that can be applied to any task requiring a manipulator to reach a set of target positions or poses within a given environment. The framework supports multi-objective optimization, parallelized physics-based evaluation, and flexible kinematic parameterization. The algorithm setup is discussed in Section III-A, with its pipeline defined in Section III-B. To demonstrate its capabilities, the framework is later applied to the problem of orchard pruning (Section IV).

A. Algorithm User Definitions

The optimization requires two inputs:

1) **Target end-effector poses**: 3D positions or full 6-DOF poses.
2) **Environment model**: Workspace geometry with optional collision objects.

Optional settings:

- **Kinematic bounds:** Joint count, joint types, joint axes, and ranges for link lengths.
- **Optimization settings:** Number of generations, objectives, calibration samples, parallel workers, and population size.
- **Task region & base pose:** Manipulator base placement; window filtering of the target points to be within the desired task space operational volume.
- **Objective weights:** Relative importance of each objective (e.g., pose error vs. torque minimization).
- **Seeded initialization:** Baseline kinematic designs/URDFs to warm-start the search.

B. Evolutionary Algorithm Framework

Our framework includes the encoding of design parameters, Unified Robot Description Format (URDF) generation, an evolutionary search technique, and candidate scoring using evaluation metrics (related to kinematics, statics, and manipulability).

1) Manipulator Representation and URDF Generation: Candidate manipulators are represented compactly as decision vectors, which provide a bridge between the evolutionary algorithm and the URDF models used for simulation. This connection is enabled through a bidirectional pipeline that translates between these two representations.

Decision Vector Representation: Each candidate manipulator is encoded as a compact decision vector:

Decision Vector Representation: Each candidate manipulator is encoded as a compact decision vector:

$$\mathbf{x} = \left[n_j,\ (t_1, a_1, \ell_1),\ (t_2, a_2, \ell_2),\ \ldots,\ (t_{n_j}, a_{n_j}, \ell_{n_j})\right]$$

$$\begin{aligned} &n_j \in [\underline{n}, \overline{n}] \cap \mathbb{Z}, \\ &t_i \in \{0, 1, 2\} \quad \text{(0: prismatic, 1: revolute, 2: spherical)}, \\ &a_i \in \{0, 1, 2\} \quad \text{(0: x-axis, 1: y-axis, 2: z-axis)}, \\ &\ell_i \in [\underline{\ell}, \overline{\ell}] \subset \mathbb{R}. \end{aligned} \tag{1}$$

This encoding specifies the joint count n_j and, for each joint i, its type t_i, axis a_i, and link length ℓ_i, subject to the bounds defined in Eq. 1. Populations are initialized from a

mixture of expert-provided seed designs (optionally imported as URDFs) and Latin Hypercube Sampling (LHS) to balance exploitation and exploration.

Decision Vector to URDF: To evaluate candidates, this pipeline translates the decision vectors of high-level manipulator parameters into a complete URDF model. Kinematic chains are constructed iteratively from a fixed base link. Revolute and prismatic joints are mapped directly, while spherical joints are represented as logical joints internally expanded into three orthogonal revolutes for URDF compatibility. Joint mass is omitted due to the variability in physical joint realizations, allowing the current study to isolate structural and kinematic performance. Incorporating actuator-specific mass models is left to future work. Links are modeled as cylinders (radius 0.05 m, mass 1 kg) with geometry and inertial properties being automatically scaled according to these dimensions, and both collision and visual meshes are generated. Future work will introduce a density-based mass scaling to improve physical fidelity. A probe-shaped end-effector is appended to the final link to complete the serial chain.

URDF to Decision Vector: The reverse translation extracts structural parameters from any kinematic chain URDF, reformulating them into the compact vector format. This allows existing designs to be introduced into the optimization loop or used as initialization seeds.

This bidirectional pipeline closes the loop between optimization and simulation, enabling automated generation, evaluation, and refinement of manipulator structures at scale.

2) Evolutionary Optimization: The optimization problem is formulated as a multi-objective problem and solved using the NSGA-II [24] algorithm as implemented in `pymoo` [25]. Raw performance metrics are normalized to $[0, 1]$, and optionally re-weighted before being passed to the optimizer. NSGA-II evaluates candidates in the full multi-dimensional objective space, using non-dominated sorting and crowding-distance diversity to approximate the Pareto front rather than aggregating objectives into a single scalar.

Candidate solutions are evolved using elitist selection with tournament-based ranking. Variation operators include Simulated Binary Crossover (SBX) for continuous parameters, uniform crossover (UX) for integers, and polynomial mutation with random resampling for discrete variables (see Fig. 3). The framework supports both seeded and unseeded initialization modes, enabling either warm-starts from expert-provided designs or purely exploratory searches.

To manage computational cost while maintaining evaluation accuracy, the set of target points is introduced gradually through a fidelity schedule. At initialization, each candidate arm is evaluated on only a small subsample of the available target points. As generations progress, the number of targets grows linearly with the generation index until the full set is reached by the end of the optimization. This progressive schedule reduces evaluation time in early generations, when many poor designs are quickly discarded, while ensuring that later evaluations capture full task complexity.

The evolutionary loop begins by loading the environment, base platform, manipulator seeds, and target poses. A calibration batch is first run to normalize metrics. Each generation then proceeds as follows: (i) sampling new candidates by applying crossover and mutation, (ii) evaluating candidates in simulation, and (iii) selecting the top performing candidates via Pareto ranking and crowding distance, seeding the next generation. After G generations, the algorithm outputs a Pareto-optimal set (X, F) of candidate solutions, where X represents the set of decision vectors defined in Eq. 1 and F contains their evaluated objective values.

3) Evaluation Metrics: Each candidate manipulator is instantiated in PyBullet [26] and evaluated across a set of target end-effector poses $\mathcal{T} = \{T_1, \ldots, T_N\}$, where each $T_k \in SE(3)$ specifies a desired position and orientation. For each target, inverse kinematics (IK) is solved and collision-free joint configurations are identified. Targets with no feasible solution are considered unreachable and excluded from per-target averages.

Let $\mathcal{Q}(T_k, x)$ denote the set of collision-free IK solutions for design x, and let $\mathcal{T}_f(x) \subseteq \mathcal{T}$ denote the subset of targets for which $\mathcal{Q}(T_k, x) \neq \emptyset$. For each $T_k \in \mathcal{T}_f(x)$, a representative configuration $q_k \in \mathcal{Q}(T_k, x)$ is selected for evaluation. All per-target metrics are averaged over reachable targets via

$$\mathcal{A}_x[g] = \frac{1}{|\mathcal{T}_f(x)|} \sum_{T_k \in \mathcal{T}_f(x)} g(q_k, T_k; x). \tag{2}$$

The following metrics are evaluated:

Pose error:

$$m_{\text{pose}}(x) = \mathcal{A}_x \left[d(\text{FK}(q_k; x),\, T_k) \right], \tag{3}$$

where $d(\cdot, \cdot)$ measures position and orientation deviation.

RRT path cost:

$$m_{\text{rrt}}(x) = \mathcal{A}_x \left[\text{len}(\pi_{\text{RRT}}(q_{\text{home}}, q_k)) \right], \tag{4}$$

where π_{RRT} denotes the RRT-Connect plan in joint space [27] and $\text{len}(\cdot)$ its path length.

Gravity-compensation torque demand:

$$m_\tau(x) = \mathcal{A}_x \left[\|\tau_g(q_k; x)\|_2 \right], \tag{5}$$

where $\tau_g(q_k; x)$ denotes the static gravity torque vector.

Joint count penalty:

$$m_J(x) = n(x), \tag{6}$$

where $n(x)$ is the number of controllable joints in the decoded kinematic chain.

Global Conditioning Index (GCI):

$$\text{GCI}(x) = \frac{1}{K} \sum_{k=1}^{K} \kappa^{-1}\Big(J(q^{(k)}; x) \Big), \qquad \kappa(J) = \frac{\sigma_{\max}(J)}{\sigma_{\min}(J)}, \tag{7}$$

where $q^{(k)}$ are joint samples within limits and $J(\cdot; x)$ is the manipulator Jacobian.

Resolved-Rate Motion Control (RRMC):

$$m_\Delta(x) = \mathcal{A}_x \left[\Delta(q_k; x) \right], \tag{8}$$

$$m_p(x) = \mathcal{A}_x \left[e_p(q_k; x) \right], \tag{9}$$

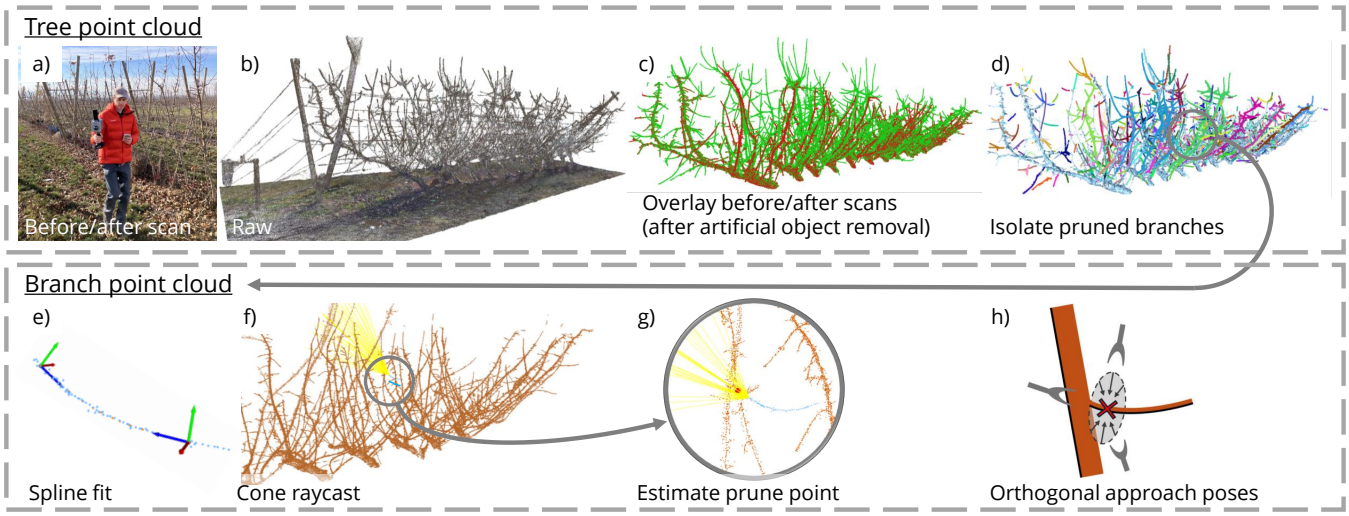

Fig. 4. Prune point identification pipeline. *Entire canopy:* (a-b) Collect before/after scans of pruned trees, (c) Clean point clouds by removing artificial objects and outlier points, followed by overlaying the before/after scans, (d) Cluster individual pruned branches. *Individual branches:* (e) Skeletonize, fit spline, and align end-point coordinate frames, (f) Cone-based raycast projecting from the base of the branch, (g) Estimate prune point by selecting the ray with shorted length, (h) Generate approach poses orthogonal to the pruned branch.

where $\Delta(\cdot)$ denotes accumulated joint deviation during resolved-rate servoing and $e_p(\cdot)$ the corresponding end-effector position error.

Metric values are normalized using calibration bounds $(m_i^{\min}, m_i^{\max})$:

$$\text{lin}(m_i(x)) = \text{clip}\left(\frac{m_i(x) - m_i^{\min}}{\max(\varepsilon,\, m_i^{\max} - m_i^{\min})}, 0, 1\right). \quad (10)$$

Each objective may optionally be reweighted by a scalar coefficient $w_i \in \mathbb{R}^+$. In this study, $w_i = 1.0$ for all objectives.

IV. Tailored Arm Optimality: Dormant Tree Pruning

The generic optimization framework can be specialized for specific domains by defining task-relevant target poses and environment models. This work demonstrates its application to dormant fruit tree pruning (i.e. pruning bare trees in the winter) by generating an orchard environment model from real world LiDAR data. The data was collected in January 2025 from a sweet cherry orchard (location: Prosser, WA, USA). A row of trees was scanned before and after pruning by professional orchard workers using a handheld LiDAR scanner (FJDynamics Trion, Singapore).

A. Prune Point Identification

This section describes how raw 3D scans are transformed into a set of 6-DOF pruning poses suitable for manipulator optimization. There are five main stages (Fig. 4):

1) Scan Alignment and Mesh Construction: The first step is to load dense 'before-' and sparse 'after-' pruning point clouds in Open3D [28]. The outliers are then removed through noise filtering, artificial objects (e.g. wooden posts, trellis wires, and irrigation lines) are manually removed, downsample, and center both point clouds at the origin. Next, the 'after' point cloud is aligned with the 'before' point cloud using coarse-to-fine Iterative Closest Point (ICP) within a cropped region of interest, in this case the points encapsulated between the ground plane and the vertical location on the thick trunk where smaller limbs become visible. This region was chosen so as to find common points between the 'before' and 'after' scans such that ICP has a better chance of converging. From the two cleaned point clouds, the pipeline upsamples points, generates an alpha-shape mesh, fills holes, removes noise, and optionally applies smoothing. The 'before' mesh serves as the orchard environment's ground-truth geometry.

2) Branch Clustering and Pruned Region Detection: Step 2 isolates individual branch point clouds from the two original tree point clouds. The unsigned distance from each point in the 'before' point cloud to the 'after' mesh is first computed using raycasting. Points beyond a distance threshold (e.g., 3–5 cm) are flagged as candidate pruned material. These candidates are then denoised using statistical outlier removal to filter isolated points. Finally, Open3D's implementation of DBSCAN [29] is used to isolate clusters and group the remaining points into discrete branch segments. Each cluster's centroid is checked against the reference mesh, and only those within a specified distance threshold are retained as valid pruned branches. The result is a set of branch-level point clouds that represent individual cut regions for downstream processing.

3) Branch Skeletonization and Frame Placement: During Step 3, for each isolated branch point cloud, a skeletal curve that represents its central axis is extracted. The default method uses the Laplacian-based contraction (LBC) method from PC Skeletor [30] to generate a skeleton and corresponding topology graph. In cases where LBC fails due to sparsity or insufficient point support, the pipeline falls back to a voxel-based approach: the point cloud is voxelized

at an adaptive resolution, skeletonized via 3D thinning, and converted into a graph. In either case, the longest path through the skeleton is identified and a cubic B-spline is fitted to it, ensuring smooth interpolation along the branch. Tangent vectors are computed at the spline endpoints, and local Cartesian frames are placed with their z-axes aligned to these tangents. To resolve ambiguity, the endpoint frame with the higher local point density in the original cloud is designated as the branch base.

4) Prune-Point Localization via Ray Casting: From the selected base frame, rays are cast both along the principal branch axis and within a surrounding cone of 30°. The cone sweep accounts for uncertainty in spline fitting and frame alignment caused by sparse or noisy branch point clouds, and by sampling multiple nearby directions it ensures at least one valid intersection with the 'after' mesh. The prune point is defined as the intersection with the shortest travel distance among all rays, representing the nearest feasible cut location along the branch direction.

5) Pose Generation: After identifying each prune point and its local branch frame, full 6-DOF end-effector poses for cutting are computed. Prune points are filtered based on their Euclidean distance to the robot base. For each point, an approach plane is generated orthogonal to the branch axis, and a 0.1 m radius ring of candidate approach poses is sampled. Candidate approach poses are evaluated by solving inverse kinematics from the manipulator's starting configuration. Among the resulting solutions, the first collision-free configuration is selected and parameterized as a 3D position with quaternion orientation. Prune points with no feasible, collision-free IK solutions are classified as unreachable. The table of all approach poses for a tree is passed to the multi-objective optimizer for manipulator evaluation.

B. Optimization Task Definition

The optimization is conducted within a constrained operating region defined relative to the Farm-ng Amiga platform seen in Fig. 2. The manipulator base is fixed to a designated mounting location on the top side of the Amiga, ensuring consistent initialization for all candidate designs. The combined system (Amiga plus manipulator) is then positioned in front of a single reconstructed tree mesh out of the whole orchard dataset, establishing the environment for evaluation.

Target prune points are filtered to those lying within a user-specified operational volume, which bounds the reachable task space of the arm. This constraint ensures that optimization focuses only on branches that are both task-relevant and physically accessible from the Amiga's mounting location. New manipulator designs generated during the evolutionary loop are instantiated at this same base frame, and their ability to reach the target points within the operating region forms the foundation of the optimization process.

Figure 5 illustrates the experimental setup, showing the Amiga platform, tree mesh, and constrained workspace (2.5 m × 2.25 m × 3.0 m) along with the optimization objectives: pose reachability, path planning, manipulability, resolved-rate motion control (RRMC), joint count penalty, and torque compensation. The robot base was positioned 2.25 m in front of the tree to reflect the natural growth pattern of branches, which extend outward toward the platform. This placement ensures the arm begins within the canopy to maximize reachability, while avoiding initial collisions when candidate designs are loaded. The figure highlights how manipulator mounting, workspace definition, and multi-objective criteria jointly shape the optimization problem.

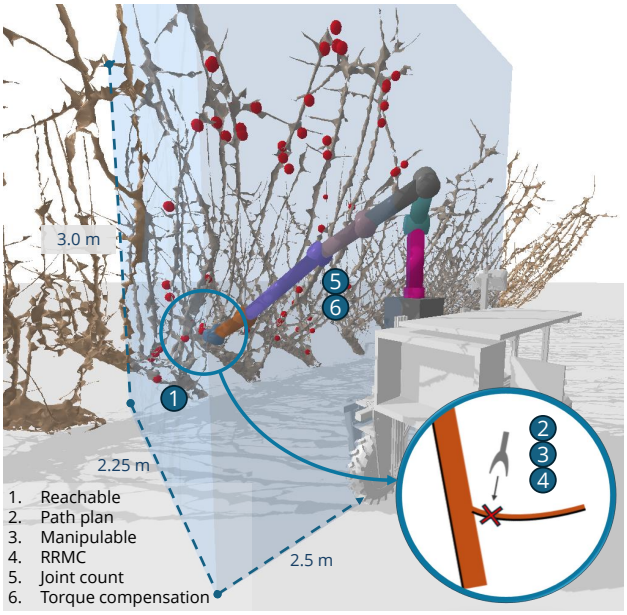

Fig. 5. Constrained workspace evaluation for pruning: (1) Reach target prune point, (2) Path plan to offset approach point, (3) Assess manipulability, (4) Perform resolved-rate motion control for final approach, (5) Minimize joint torques for gravity compensation.

Each generation used a population size of 44, matching the number of CPUs available on the HPC partition. Twenty calibration samples were included, ten of which were expert-seeded designs, and the optimization proceeded for 175 generations. The kinematic constraints were defined as follows: joint types $0, 1, 2$, permitting prismatic, revolute, and spherical joints; joint axes $0, 1, 2$, corresponding to the x-, y-, and z-axes; and link lengths bounded within $[0.05, 0.75]$ m. The evolutionary operators were configured with a seeded mixed sampling strategy, mixed-variable crossover, and mixed-variable mutation, with real-valued mutation probability $1/(1+3\cdot\overline{n})$ and integer mutation probability 0.1. Duplicate elimination was enabled throughout. All experiments were executed on a Dell PowerEdge R740 HPC node with two 24-core Intel Xeon Gold 6254R processors (3.0 GHz, 48 cores total) and 192 GB of RAM, with parallelization across 44 allocated cores matching the population size.

V. Results and Discussion

To assess the performance of the optimized manipulators, we benchmarked them against the UR5e (Universal Robots, Odense, Denmark) industrial arm, a standard platform in robotic pruning research [31], [32]. For each candidate design, a URDF was generated from its decision vector,

TABLE I

OPTIMIZED KINEMATIC DESIGNS FOR VARYING JOINT COUNTS, COMPARED TO THE UR5E BASELINE. 'M' DENOTES A MOCK SPHERICAL JOINT, REPRESENTED AS THREE ORTHOGONAL REVOLUTE JOINTS (3-DOF PER 'M').

Design	Number of Joints	DOF	Joint types	Joint axes	Link Length
UR5e [baseline]	6	6	[1, 1, 1, 1, 1, 1]	[2, 1, 1, 1, 2, 0]	[0.163, 0.138, 0.425, 0.392, 0.127, 0.100]
A	5	7	[1, 1, 1, 1, 2]	[0, 0, 1, 2, M]	[0.72, 0.74, 0.75, 0.35, 0.12]
B	5	9	[0, 2, 0, 1, 2]	[0, M, 2, 2, M]	[0.74, 0.66, 0.61, 0.39, 0.05]
C	6	14	[2, 0, 0, 2, 2, 2]	[M, 2, 2, M, M, M]	[0.75, 0.7, 0.66, 0.39, 0.05, 0.09]
D	6	12	[0, 2, 0, 1, 2, 2]	[0, M, 2, 2, M, M]	[0.68, 0.66, 0.61, 0.47, 0.12, 0.11]
E	7	15	[1, 0, 2, 2, 2, 1, 2]	[2, 0, M, M, M, 0, M]	[0.71, 0.75, 0.72, 0.42, 0.12, 0.65, 0.45]
F	7	9	[0, 0, 1, 1, 1, 0, 2]	[0, 1, 2, 0, 1, 2, M]	[0.73, 0.75, 0.74, 0.47, 0.51, 0.45, 0.10]

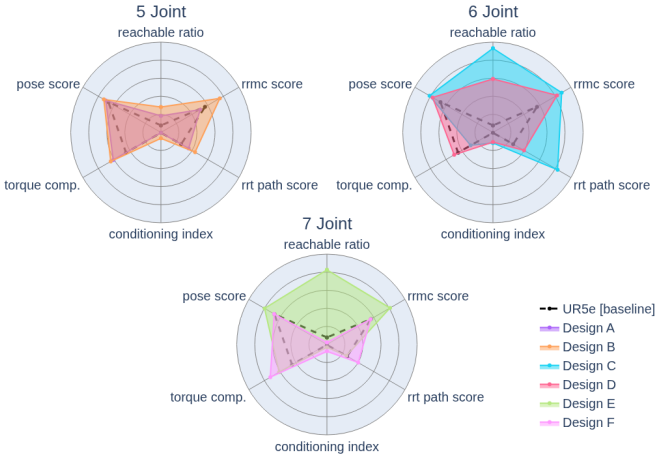

Fig. 6. Radar plots of manipulator performance across the optimization objectives. Each subplot compares two optimized designs of a given joint count category with the baseline UR5e, highlighting trade-offs in reachability, torque, manipulability, and path planning scores. A larger area occupied in the plot indicates better performance across objectives.

mounted on the Amiga in the same location relative to the tree mesh, and evaluated under identical conditions. The UR5e was staged and tested in the same manner. Each evaluation followed the fixed pipeline shown in Fig. 5, and performance was quantified by the number of reachable prune points within the operating window and by the same normalized metrics used during optimization. In total, the arms were tasked with reaching 64 identified prune points distributed within the defined operational volume, allowing direct comparison across designs.

The optimization produced a Pareto front of 44 (equivalent to the population size) candidate arms spanning different kinematic structures. Because performance can vary by the number of joints in each design, we selected the two top performing designs from each joint count category (5, 6, and 7) based on the fraction of reachable prune points. Their relative performance against the UR5e is summarized in Fig. 6. In these radar plots, higher scores indicate better performance across objectives. The norther axis represents the reachable ratio (fraction of successfully reached targets), while remaining axes show normalized pose, torque, path, and RRMC metrics. All six objectives are normalized to the range [0, 1], with 0 at the center and 1 at the outermost ring.

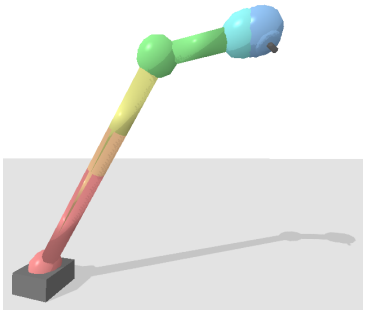

Fig. 7. Top performing optimized 6 joint robot arm (Design C - see Tab. I)

The results show that our optimized arms consistently outperformed the UR5e on the pruning task. Even with fewer joints, Design A (5 joints) achieved higher overall scores. Despite Design C having the same number of joints as the UR5e, it exceeded in reachability and all other metrics. Among the 7 joint designs, Design E improved pose accuracy, torque efficiency, and RRMC, but its added redundancy increased motion planning costs, leading to lower RRT performance as compared with the 6 jointed Design C. All optimized designs in Tab. I included at least one spherical joint, suggesting that higher dexterity has a large impact when operating in a cluttered and unstructured environment. The best-performing design overall was a 6 jointed arm with four spherical joints and two overlapping telescopic joints (Fig 7). For this pruning task, the combination of long reach from telescopic joints and dexterity from spherical joints offered clear advantages in kinematic feasibility, though realizing and controlling such a design in physical hardware would present significant complexity.

VI. CONCLUSION AND FUTURE WORK

This work presented a 3D simulation-driven framework for generative manipulator design, where candidate arms defined by high-level parameters are automatically converted into URDF models and evaluated in a large-scale evolutionary search. Using a multi-objective NSGA-II algorithm, designs were optimized across objectives including pose accuracy, torque demand, path planning cost, manipulability, and joint count. For evaluation, we considered a tree pruning task, where the simulation environment was reconstructed from 3D scans of orchard trees, with target prune points automat-

ically localized by comparing before- and after-pruning data. The resulting Pareto-optimal designs across 5–7 joint configurations outperformed the UR5e baseline in reachability and kinematic performance, with competitive or superior results even for fewer jointed arm designs. These results highlight both the feasibility and the benefits of tailoring manipulator designs directly to agricultural tasks and environments, rather than adapting general-purpose arms. While demonstrated here for pruning, the framework generalizes to other dexterous orchard operations such as thinning and harvesting, as well as non-agriculture tasks where cluttered geometries and task-specific reachability constrain manipulator effectiveness.

Future work will extend the optimization framework to leverage the complete orchard dataset rather than a single-tree instance. This expansion is currently constrained by computational cost, as full-dataset optimization is significantly more resource-intensive. Additional efforts will focus on systematic hyperparameter tuning and normalization methods for per-generation population calibration. Additionally, spherical joint selection will be reweighted in the optimization process to better capture hardware 'realism' and control complexity. This work will also incorporate higher-fidelity physical models of joints and links, including actuator-specific joint mass and density-based link modeling, to improve dynamic evaluation and hardware realism. URDF generation will be extended to allow joint offsets along link lengths, and an optimized design will ultimately be constructed and deployed in field trials. By combining generative kinematic design with task-driven evaluation, this framework moves toward practical agricultural manipulators that can bridge the gap between laboratory prototypes and reliable on-farm deployment.

REFERENCES

[1] K. F. Cooper, "Navigating labor challenges and finding solutions," Jul 2025.

[2] A. Navone, M. Martini, and M. Chiaberge, "Autonomous robotic pruning in orchards and vineyards: a review," 2025.

[3] S. V. Brecht, J. S. A. Voegerl, and T. C. Lueth, "Automated design and construction of a single incision laparoscopic system adapted to the required workspace," in *Intelligent Robots and Systems*, Oct. 2020, p. 3112–3118.

[4] A. Muñoyerro, A. Hernández, M. Urizar, and O. Altuzarra, "A general automatic method for mechanism optimization based on kinematic constraints and analytical jacobian matrix," *Journal of Mechanical Engineering Science*, vol. 237, no. 14, p. 3181–3197, Jul. 2023.

[5] S. Ha, S. Coros, A. Alspach, J. Kim, and K. Yamane, "Computational co-optimization of design parameters and motion trajectories for robotic systems," *Journal of Robotics Research*, vol. 37, no. 13–14, p. 1521–1536, Dec. 2018.

[6] G. Fadini, T. Flayols, A. D. Prete, and P. Soueres, "Simulation aided co-design for robust robot optimization," *Robotics and Automation Letters*, vol. 7, no. 4, p. 11306–11313, Oct. 2022.

[7] G. Fadini, T. Flayols, A. Del Prete, N. Mansard, and P. Souères, "Computational design of energy-efficient legged robots: Optimizing for size and actuators," in *Int'l Conf. on Robotics and Automation*, May 2021, p. 9898–9904.

[8] A. Dogra, S. Sekhar Padhee, and E. Singla, "Optimal synthesis of unconventional links for modular reconfigurable manipulators," *Journal of Mechanical Design*, vol. 144, no. 083304, Apr. 2022.

[9] T. Kot, Z. Bobovský, M. Brandstötter, V. Krys, I. Virgala, and P. Novák, "Finding optimal manipulator arm shapes to avoid collisions in a static environment," *Applied Sciences*, vol. 11, no. 1, p. 64, Jan. 2021.

[10] F. Cursi, W. Bai, E. M. Yeatman, and P. Kormushev, "Globdesopt: A global optimization framework for optimal robot manipulator design," *IEEE Access*, vol. 10, p. 5012–5023, 2022.

[11] Z. He and M. Ciocarlie, "Morph: Design co-optimization with reinforcement learning via a differentiable hardware model proxy," no. arXiv:2309.17227, Sep. 2023.

[12] A. Zhao, J. Xu, M. Konaković-Luković, J. Hughes, A. Spielberg, D. Rus, and W. Matusik, "Robogrammar: graph grammar for terrain-optimized robot design," *ACM Trans. Graph.*, vol. 39, no. 6, pp. 188:1–188:16, Nov. 2020.

[13] Y. Ma, Q. Feng, Y. Sun, X. Guo, W. Zhang, B. Wang, and L. Chen, "Optimized design of robotic arm for tomato branch pruning in greenhouses," *Agriculture*, vol. 14, no. 33, p. 359, Mar. 2024.

[14] F. Molaei and S. Ghatrehsamani, "Kinematic-based multi-objective design optimization of a grapevine pruning robotic manipulator." *AgriEngineering*, vol. 4, no. 3, pp. 606–625, Jul. 2022.

[15] E. Van Henten, D. Slot, C. Hol, and G. van Willigenburg, "Optimal manipulator design for a cucumber harvesting robot," *Computers and Electronics in Agriculture*, vol. 65, p. 247–257, Mar. 2009.

[16] C. Lehnert, T. Perez, and C. McCool, "Optimisation-based design of a manipulator for harvesting capsicum," 2015.

[17] N. Singh, V. K. Tewari, P. K. Biswas, L. K. Dhruw, R. Ranjan, and A. Ranjan, "Optimizing cotton-picking robotic manipulator and inverse kinematics modeling using evolutionary algorithm-assisted artificial neural network," *Journal of Field Robotics*, vol. 41, p. 2322–2342, Sep. 2023.

[18] R. Arikapudi and S. G. Vougioukas, "Robotic tree-fruit harvesting with telescoping arms: A study of linear fruit reachability under geometric constraints," *IEEE Access*, vol. 9, p. 17114–17126, 2021.

[19] M. Levin and A. Degani, "A conceptual framework and optimization for a task-based modular harvesting manipulator," 2019.

[20] V. Bloch, A. Degani, and A. Bechar, "A methodology of orchard architecture design for an optimal harvesting robot," *Biosystems Engineering*, vol. 166, p. 126–137, Feb. 2018.

[21] R. Saravanan, S. Ramabalan, N. G. R. Ebenezer, and R. Natarajan, "Evolutionary bi-criteria optimum design of robots based on task specifications," *Journal of Advanced Manufacturing Technology*, vol. 41, no. 3–4, p. 386–406, Mar. 2009.

[22] S. Rubrecht, E. Singla, V. Padois, P. Bidaud, and M. de Broissia, "Evolutionary design of a robotic manipulator for a highly constrained environment," in *New Horizons in Evolutionary Robotics*, ser. Studies in Computational Intelligence, S. Doncieux, N. Bredèche, and J.-B. Mouret, Eds. Berlin, Heidelberg: Springer, 2011, p. 109–121.

[23] R. Al-Khulaidi, R. Akmeliawati, S. Grainger, and T.-F. Lu, "Structural optimisation and design of a cable-driven hyper-redundant manipulator for confined semi-structured environments." *Sensors*, vol. 22, no. 22, Nov. 2022.

[24] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, p. 182–197, Apr. 2002.

[25] J. Blank and K. Deb, "pymoo: Multi-objective optimization in python," *IEEE Access*, vol. 8, pp. 89 497–89 509, 2020.

[26] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," 2016–2021.

[27] J. Kuffner and S. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Int'l Conf. on Robotics and Automation*, vol. 2, 2000, pp. 995–1001.

[28] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," 2018.

[29] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Knowledge Discovery and Data Mining*, ser. KDD'96. AAAI Press, 1996, p. 226–231.

[30] L. Meyer, A. Gilson, O. Scholz, and M. Stamminger, "Cherrypicker: Semantic skeletonization and topological reconstruction of cherry trees," 2023.

[31] A. You, N. Parayil, J. G. Krishna, U. Bhattarai, R. Sapkota, D. Ahmed, M. Whiting, M. Karkee, C. M. Grimm, and J. R. Davidson, "Semiautonomous precision pruning of upright fruiting offshoot orchard systems: An integrated approach," *IEEE Robotics Automation Magazine*, vol. 30, no. 4, pp. 10–19, 2023.

[32] A. Silwal, F. Yandun, A. Nellithimaru, T. Bates, and G. Kantor, "Bumblebee: A path towards fully autonomous robotic vine pruning," *Field Robotics*, vol. 2, pp. 1661–1696, 2022.