ORCHARDBENCH: A Physically-Grounded, GPU-Parallel Apple-Orchard Simulation Benchmark for Agricultural Robotics

Humphrey Munn
School of EECS, The University of Queensland
h.munn@uq.net.au Project page & code: https://humphreymunn.github.io/orchardbench-page/

Fig. 1: **ORCHARDBENCH.** A physically-grounded, GPU-parallel apple orchard. *Left:* a mobile manipulator (Clearpath Ridgeback base and Franka arm) autonomously harvesting a compliant, fruit-bearing tree over bumpy terrain—here holding a picked apple beside the collection bucket, amid foliage that moves with the branches. *Right:* one hundred domain-randomized trees simulated in parallel, each a distinct plausible tree grown by a stochastic L-system, varying in growth habit, foliage, fruit and colour. Branches *snap* under load and apples *detach* at realistic pull forces (Section IV).

***Abstract*—Robotic tree-fruit harvesting is a flagship problem for agricultural automation, but progress is bottlenecked by the cost and irreproducibility of field experiments: an orchard is available only weeks a year, every tree is different, and a control error can permanently damage the crop or the plant. The tree models used in graphics and agronomy are geometrically detailed but physically inert, while the GPU-parallel simulators used in robot learning contain no plausible trees. We present ORCHARDBENCH, a physically-grounded, GPU-parallel simulation of apple-orchard trees on the NEWTON/MuJoCo-Warp engine. Each tree is grown by a stochastic L-system and instantiated as a fully articulated body: branches are compliant torsional spring-dampers whose stiffness follows Euler-Bernoulli beam theory, they break at a wood modulus of rupture and fall as free hinges, and apples are independent bodies on stem tethers that detach at literature-grounded pull forces and load the branch as they are pulled. A moving, density-controllable foliage layer occludes the canopy as real leaves do. Every physical parameter is tied to a published source. Per-environment domain randomization makes each batched world a distinct tree, and a mobile manipulator with a wrist depth camera closes the loop with geometric fruit perception and an autonomous harvesting baseline. Careful engineering of the solver and the model lets ORCHARDBENCH run many parallel environments at interactive rates on a single 8 GB laptop GPU. We define the tasks and a metric suite spanning harvest completeness, throughput, and plant damage (with a per-canopy-zone breakdown), and report baseline results across foliage, fruit load, terrain, canopy zone, and parallelism. The analytic baseline succeeds on about 40 % of the fruit it detects and harvests only about an eighth of the reachable fruit on a tree, leaving clear headroom for the learned and agentic methods the benchmark is built to measure. Code and videos are released at https://humphreymunn.github.io/orchardbench-page/.**

***Index Terms*—agricultural robotics, physics simulation, benchmark, fruit harvesting, domain randomization, manipulation, sim-to-real.**

I. INTRODUCTION

Tree-fruit harvesting is one of the most labour-intensive and least automated operations in agriculture. Apples are still picked overwhelmingly by hand, the seasonal workforce is shrinking and expensive, and the picking window for a given block is only a few weeks long. These pressures have driven two decades of work on robotic harvesters [1–3], but the problem remains

open: fruit is heavily occluded by foliage and by the plant's own structure, the manipulator must reach into a compliant canopy without snapping branches or bruising fruit, and every tree is geometrically unique. Crucially, the very thing that makes the task hard (physical contact with a living, breakable plant) also makes it expensive to study. A field trial requires a real orchard in season, is unrepeatable because no two trees or approaches are identical, and carries a real cost of failure: a mis-planned motion can tear a scaffold limb or strip fruit, damaging next year's crop.

Simulation is the standard escape from this bind in the rest of robotics, where GPU-parallel physics engines now train contact-rich manipulation policies at a scale field robots could never reach [4, 5]. Orchard robotics, however, sits in a gap between two mature but disjoint bodies of work. On one side, decades of research in computer graphics and functional–structural plant modelling produce exquisitely detailed trees from L-systems and related grammars [6–8], but these models are *procedural geometry*: they describe how a tree *looks*, not how it *moves*, bends, or breaks under a robot's touch. On the other side, physics-based robot learning benchmarks [9–11] offer rigorous contact dynamics but populate their worlds with rigid boxes, tools and articulated furniture, never a compliant, fruit-bearing tree. The effort that brings the two closest together, the contact-aware branch-manipulation work of Jacob et al. [12] (PCAP), simulates a procedural forest of compliant branches and learns to push them aside; but it uses a spring abstraction rather than beam-theoretic dynamics and models neither branch breakage, nor fruit, nor autonomous harvesting, and a benchmark with standardized tasks and metrics is absent (Table I).

We close this gap with ORCHARDBENCH (Figure 1), a simulation benchmark whose central commitment is *physical fidelity of the plant itself*. An apple tree in ORCHARDBENCH is not a decorative mesh but a fully articulated dynamical system: every internode is a rigid link, every branch junction a compliant torsional spring–damper whose stiffness follows Euler–Bernoulli beam theory, so that thin outer twigs are soft and the trunk is stiff exactly as in a real tree. Branches *rupture* when the transmitted bending moment exceeds the wood's modulus of rupture, and the freed sub-tree falls as a genuine hinge at gravity rate rather than vanishing or freezing. Apples are independent rigid bodies suspended from their spurs by spring–damper stem tethers; a firm, sustained pull, and only a firm, sustained pull, snaps the stem at a force drawn from the harvesting literature, while a light tug merely bends the branch. Every one of these numbers is grounded in a published source, from green apple wood's density, elastic modulus and rupture stress to the 14–23 N force needed to detach a ripe fruit (Section IV, Table III).

Around this plant model we build the rest of an orchard-robotics testbed. A Ridgeback–Franka mobile manipulator carries a wrist-mounted depth camera and executes a full autonomous harvest cycle (explore, approach, reach, grasp, detach, deposit), driven by a geometric, learning-free fruit detector chosen for its sim-to-real plausibility. A density-controllable *foliage* layer, rendered as leaves rigidly attached to the branch bodies so that it sways with the tree, occludes the fruit exactly as a real canopy does, and optional bumpy terrain adds outdoor variation. Per-environment domain randomization makes each of the N GPU-batched worlds a *different* plausible tree, varying geometry, growth habit, material, fruit load, foliage, and colour, so that a policy or detector is never trained or evaluated on a single instance. Finally, careful engineering of the solver and the model (a matrix-free constraint solve, in-place rupture with no recompile, reduced-DOF fruit, and instanced foliage) lets ORCHARDBENCH run many of these parallel worlds at interactive rates on a single 8 GB laptop GPU (Section VII).

Why one benchmark for four communities. The design is driven by the observation that a physically-grounded *and* diverse *and* fast orchard simulator unlocks several research programs that each existing tool serves only partially:

1) **Safe verification of analytic controllers.** A model-based or solver-based picking strategy can be stress-tested against thousands of randomized trees, measuring exactly how often it snaps a branch or drops fruit, *before* it is ever run on hardware, where such failures are costly and irreversible.
2) **A learning and sim-to-real substrate.** The environment exposes the standard reinforcement- and imitation-learning interface at GPU scale, with domain randomization built in for transfer, so policies can be pretrained in simulation before fine-tuning on a real robot.
3) **Controllable perception research.** Because foliage density, occlusion and lighting are dials with perfect ground truth, ORCHARDBENCH is a controlled testbed for fruit detection under variable foliage, a regime that is precisely what makes real orchard perception hard.
4) **A real-world-grounded optimization target.** The benchmark reports a vector of physically meaningful metrics, throughput, success by canopy zone, branches snapped, fruit dropped, that a learned or automated-research method can be tasked to improve, with damage and safety as first-class objectives rather than afterthoughts.

Contributions.

1) A GPU-parallel apple-orchard simulation in which the *plant* is physically grounded: compliant beam-theoretic branches, moment-of-rupture breaking with free-fall dynamics, and realistic detachable fruit that loads the branch, with every parameter tied to a published source.
2) A moving, density-controllable *foliage* layer that sways with the branches and occludes fruit as real leaves do, giving a controllable occlusion variable with perfect ground truth for perception research.
3) Per-environment domain randomization of tree geometry, growth habit, material, foliage and appearance that keeps parallel worlds structurally homogeneous (and therefore GPU-batched) while making each one a distinct tree.
4) Compute-efficiency techniques (a matrix-free constraint solve, in-place rupture with no model recompile, reduced-

DOF fruit, instanced foliage, and an on-device domain-randomization build) that let many parallel environments run at interactive rates on an 8 GB laptop GPU.

5) A complete closed-loop harvesting testbed and benchmark: mobile manipulator, wrist depth sensing, geometric fruit perception, an autonomous baseline controller, and a metric suite (harvest completeness, throughput, plant damage, per-canopy-zone success), with baseline results across foliage, fruit load, terrain, canopy zone, and parallelism.

II. Related Work

OrchardBench sits at the intersection of four largely separate literatures: procedural plant modelling, physics simulation for robot learning, agricultural harvesting robotics, and orchard perception. Its contribution is best understood as joining the first two, bringing physically-grounded dynamics to procedurally generated, diverse trees, in service of the last two.

a) Procedural plant and tree modelling.: The generation of realistic plant geometry is a mature field founded on Lindenmayer systems [6], with the branching structure of trees captured by Honda's parametric models [7] and radii by the pipe model of Shinozaki [13, 14] and its hydraulic refinements [15]. For apple specifically, MAppleT [8] couples a stochastic L-system to a biomechanical model of gravimorphic bending to reproduce cultivar-specific architecture over years of growth. These functional–structural plant models are the state of the art in *describing* how a tree is shaped, and we adopt their machinery, a stochastic central-leader grammar with phyllotactic divergence [16] and pipe-model taper, to grow our canopies. The essential limitation for robotics is that they are models of *static geometry*: a MAppleT tree does not bend when pushed, transmit a moment, or break. OrchardBench takes their output as the rest configuration of a *dynamical* articulated body.

b) Physics simulation for robot learning.: GPU-parallel physics engines have transformed robot learning by simulating thousands of environments at once [4, 17], building on accurate contact solvers such as MuJoCo [5] and differentiable/data-parallel back-ends like NVIDIA Warp [18]. A now-crowded landscape, Brax [19], MuJoCo Playground [20], SAPIEN [21] and Genesis [22], delivers ever-faster batched simulation, yet none ships a compliant, breakable, fruit-bearing plant as an asset. We build on Newton [23], a Warp-based engine whose MuJoCo-Warp solver gives implicit, stable joint springs while running fully on the GPU. At the massive parallelism these engines are built for, however, what they are typically used to simulate are rigid robots and rigid or articulated *objects*; compliant, high-degree-of-freedom natural structures such as trees are rare, and a breakable, fruit-bearing one is absent. A recurring theme in our system design is making a large, stiff, sparsely coupled articulation (a tree is ∼hundreds of compliant joints plus dozens of free fruit bodies) behave stably and fast on hardware built for many small articulations (Section IV).

c) Robot-learning benchmarks.: Standardized benchmarks have been central to progress in manipulation, RLBench [9], Meta-World [10] and ManiSkill [11] among them, by fixing tasks, assets and metrics so methods can be compared. Their scenes are deliberately generic (blocks, tools, cabinets); none targets the specific and demanding physics of a compliant orchard canopy, nor reports agricultural metrics such as fruit throughput or plant damage. OrchardBench is, to our knowledge, the first benchmark whose object of manipulation is a physically-grounded breakable plant, with a metric suite to match.

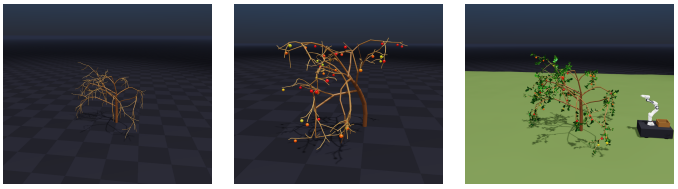
d) Agricultural harvesting robotics.: Robotic harvesting of tree fruit has been pursued for decades; reviews document the recurring obstacles of occlusion, variable illumination, delicate contact and the sheer geometric variability of plants [1, 3, 24]. Field systems such as the robotic apple harvester of Silwal et al. [2] demonstrate a full detect–reach–grasp–detach cycle but also expose the cost structure that motivates us: field integration and evaluation are slow, seasonal and unrepeatable, and controllers that damage the tree or fruit cannot be freely iterated on live plants. A simulator that faithfully reproduces the compliant, breakable canopy and the detachment mechanics of the fruit lets this iteration happen safely and at scale, which is precisely OrchardBench's purpose.

e) Fruit detection and perception.: Orchard perception is a field in itself, spanning sensor surveys for fruit detection and localization [25], the shift to deep learning for detection and yield estimation [26], and public datasets such as MinneApple [27]. The central difficulty is occlusion by foliage and structure, which no fixed dataset can vary in a controlled way. Because OrchardBench renders depth (and colour) with perfect ground-truth fruit poses while foliage density is a continuous dial, it is a controllable environment for exactly this problem, and our baseline detector is deliberately a classical, learning-free geometric method (Section V), so that perception performance reflects the scene rather than a pretrained network.

f) Closest work: physical tree/branch manipulation.: The nearest precedent is the contact-aware branch-manipulation work of Jacob et al. [12] (PCAP, "Proprioceptive Contact-Aware Policy"), which models tree branches as rigid cylinders joined by torsional spring–dampers with a beam-derived stiffness $K_{\mathrm{p}} = EI/\ell$ and per-level damping, and learns a proprioceptive policy that pushes branches aside with minimal contact force, transferring zero-shot to a real arm. We adopt their branch-compliance formulation as our starting point and extend it substantially. PCAP's goal is to *avoid* damaging branches while reaching through them; like us it trains on a domain-randomized procedural L-system forest, but its branch physics is a spring abstraction and it models neither branch *breakage*, nor *fruit* with detachment, nor a mobile base with fruit sensing, nor an autonomous harvesting task with metrics. OrchardBench adds exactly these capabilities, breaking, realistic detachable fruit that loads the whole-tree dynamics, a mobile manipulator with depth sensing, and a benchmark task suite, turning a branch-avoidance testbed into a harvesting benchmark. Table I summarizes the comparison.

TABLE I: **Feature comparison** of ORCHARDBENCH against a representative procedural plant model, the closest physical branch-manipulation system (PCAP / Jacob et al.), and general-purpose GPU physics/robot-learning simulators. ✓ = native/supported, ∼ = partial/limited, ✗ = absent.

Capability	**ORCHARDBENCH** (ours)	MAppleT [8]	PCAP / Jacob et al. [12]	Isaac Gym/ Lab [4]	MJX/Play-ground [20]
Procedural stochastic apple trees	✓	✓	∼	✗	✗
Pipe-model taper / botanical allometry	✓	✓	∼	✗	✗
Compliant articulated branch dynamics	✓	∼	✓	✗	✗
Branch **breaking** at rupture	✓	✗	✗	✗	✗
Realistic **fruit**: detach + loads tree	✓	✗	✗	✗	✗
Per-env domain-randomized *distinct* trees	✓	✗	∼	∼	∼
GPU-parallel batched envs	✓	✗	∼	✓	✓
Mobile manipulator + fruit sensing	✓	✗	∼	∼	∼
Autonomous harvesting task + metrics	✓	✗	✗	✗	✗
Lit.-grounded *mechanical* parameters	✓	∼	∼	N/A	N/A



(a) L-system skeleton (b) articulated + fruit (c) on terrain

Fig. 2: From grammar to dynamical body. A stochastic L-system produces a skeleton of internodes (a); each becomes a rigid capsule link joined to its parent by a compliant joint, with foliage and fruit on the outer spurs (b); the finished tree, here on a heightfield terrain with the manipulator, is placed in the scene (c).

III. TREE GENERATION

Each tree begins as a symbolic string produced by a parametric L-system and is interpreted by a 3-D turtle into a *skeleton*: an ordered list of internode segments, each with world-frame endpoints, a branch order, and an orientation quaternion whose local $+z$ axis is the branch heading. The skeleton is the rest configuration that the physics model (Section IV) is built to reproduce.

a) Grammar.: We support two generators. A ternary bracketed L-system [6] with Honda-style parametric branch angles and length ratios [7] (used for reproducible, canonical trees) applies, from an axiom trunk, the production

$$\begin{aligned} A \;\rightarrow\; & !(v_r)\,F(F_0)\,[\,\&(a)\,F(F_0)\,A\,]\,/(d_1) \\ & [\,\&(a)\,F(F_0)\,A\,]\,/(d_2)\,[\,\&(a)\,F(F_0)\,A\,], \end{aligned} \tag{1}$$

where F draws an internode, $\&(a)$ pitches the heading by branching angle a, $/(d)$ rolls by divergence d, $[\,]$ push/pop the turtle state, and $!(v_r)$ scales width. Internodes elongate ($F(\ell)\rightarrow F(\ell\, l_r)$) and thicken ($!(w)\rightarrow !(w\, v_r)$) each derivation. After n derivations the body count is $1 + 2(3^n - 1)$. The divergence angles are centred on the golden angle $d_1 = d_2 = 137.5°$, the phyllotactic divergence at the shoot apex [16].

The default generator, used throughout the benchmark, is a stochastic apple-tree grammar in the spirit of MAppleT [8]: a central leader emits a phyllotactic spiral of 3–6 scaffold limbs at wide crotch angles (42–65° from vertical, the range that maximizes limb strength and fruit-bud formation), and each limb recursively forks into a bounded number of laterals that arch downward under simulated gravimorphism. Every parameter, scaffold count, fork angles, per-internode droop, length decay, bud-abortion probability, and an overall "form" from upright to weeping, is drawn from a per-seed distribution, so each seed yields a distinct but botanically plausible tree (Figure 4).

b) Taper (pipe model).: Radii are assigned by the da Vinci/Murray pipe model [13, 15]: from a fixed tip radius r_{tip}, a parent radius satisfies

$$r_{\text{parent}}^{\beta} = \sum_{c \in \text{children}} r_c^{\beta}, \tag{2}$$

with exponent $\beta = 2.2$–2.3. Setting $\beta = 2$ is exactly area-preserving (da Vinci's rule); load-bearing tree wood stays near this value while hydraulic-optimal networks approach Murray's $\beta \approx 2.5$, so 2.0–2.5 is the defensible band [14]. The result is a monotone taper with the trunk thickest, which, together with the beam-theoretic stiffness of Section IV, produces the correct mechanical gradient from a stiff trunk to compliant twigs.

c) Grounding and placement.: The finished skeleton is anchored so the trunk base sits at the origin, and a forward-kinematic ground-clearance pass lifts any branch that would otherwise dip below ∼10 cm, mimicking the way real limbs grow away from the soil. Fruit and leaves are placed on the outer canopy (Section IV), and the whole tree is rescaled to a target height (2.4 m by default, a compact, modern trained tree so that most fruit falls within the fixed-base arm's reach envelope; Table III). Because all of these operations are purely geometric transforms of a fixed set of segments, they preserve the graph structure across environments, which is what makes the domain randomization of Section IV-D compatible with GPU batching.

IV. PHYSICAL MODEL

The defining feature of ORCHARDBENCH is that the tree is a genuine dynamical system. Each internode segment becomes a

rigid link with a capsule collider along its local $+z$ axis; mass and inertia follow from a wood density ρ_w and the capsule geometry. The trunk base is welded to the world, and each branch connects to its parent by a joint whose rest transform reproduces the skeleton exactly,

$$T_{\text{joint}}^{\text{parent}} = (0, 0, \ell_{\text{parent}}) \cdot q_{\text{rel}}, \qquad q_{\text{rel}} = \bar{q}_{\text{parent}}\, q_{\text{child}}, \tag{3}$$

so that at rest the articulation is stress-free and matches the generated geometry. In rigid mode the joints are fixed; in the deformable mode used throughout the benchmark they are compliant.

A. Compliant branch dynamics

Each compliant joint is a torsional spring–damper acting on two bending degrees of freedom (a 6-DOF "D6" joint with the twist and translational axes locked). Following the branch model of Jacob et al. [12] and Euler–Bernoulli beam theory, the rotational stiffness of a branch of radius r and length ℓ is

$$K_{\text{p}} \;=\; \frac{\pi}{4}\,\frac{E\,r^4}{\ell} \;=\; \frac{EI}{\ell}, \qquad K_{\text{d}} \;=\; c_d\,K_{\text{p}}, \tag{4}$$

with E the wood's Young's modulus, $I = \pi r^4/4$ the second moment of area of the circular section, and c_d a stiffness-proportional damping coefficient (units of time). We stress that c_d is *not* a modal damping ratio: for a joint of rotational inertia J the effective ratio is $\zeta_{\text{eff}} = c_d\sqrt{K_{\text{p}}}/(2\sqrt{J})$, which varies per joint with stiffness and inertia; c_d is chosen for numerical stability (so the tree neither creeps nor rings) rather than tuned to a measured ζ. Because $K_{\text{p}} \propto r^4$ and r tapers by Equation (2), thin outer twigs are orders of magnitude more compliant than the trunk: this reproduces the mechanical behaviour of a real tree without any per-branch tuning. The restoring torque on a joint at bending angle $\boldsymbol{\theta}$ (angular velocity $\boldsymbol{\omega}$) is $\boldsymbol{\tau} = -K_{\text{p}}\,\boldsymbol{\theta} - K_{\text{d}}\,\boldsymbol{\omega}$. A per-level exponential model (the "rudimentary" abstraction of Jacob et al. [12]) is also provided. Under NEWTON's MuJoCo-Warp solver these are exact implicit position actuators, which keeps the articulation (roughly 150 internodes $\times\,2$ bending DOF plus $\sim$30 fruit $\times\,3$ DOF, i.e. ~ 400 DOF for a default apple tree) stable at the low substep counts we use.

Wood is not perfectly elastic: we add velocity-proportional damping $F = -c_v\,m\,v$, $\tau = -c_\omega\,I\,\omega$ (mass- and inertia-proportional so thin twigs are not over-damped). This is a phenomenological surrogate for aerodynamic and material losses, not literal v^2 air drag; together with a soft ground penalty it lets disturbed branches settle rather than ring indefinitely. The default $E \approx 7\,\text{GPa}$ of green apple wood is stiff enough that hand-scale forces barely deflect a scaffold limb (physically correct), so we report bending validation at a softer sapling modulus in Section VII.

B. Branch breaking

A branch ruptures when the transmitted bending moment exceeds the wood's modulus of rupture σ_r times the section modulus of a solid circular cross-section,

$$M_{\max} \;=\; \sigma_r\,\frac{\pi r^3}{4}, \qquad |M| = K_{\text{p}}\,\|\boldsymbol{\theta}\| \;>\; M_{\max} \;\Rightarrow\; \text{rupture}. \tag{5}$$

We test the quasi-static elastic moment $K_{\text{p}}\|\boldsymbol{\theta}\|$ (omitting the transient damper term $K_{\text{d}}\boldsymbol{\omega}$); an N-frame hysteresis then suppresses rate spikes so a whipping branch does not chain-snap the tree. Because living branches greenstick-fracture and buckle rather than snapping cleanly [28], we place σ_r near the low end of the green-wood range so the picker meets realistic resistance before a branch gives.

Making a constrained body *genuinely free at runtime* is the hard part on a MuJoCo-family solver: disabling the joint welds the child rather than freeing it, and editing the model mid-simulation forces a recompile that, under a hard pull, injects energy and cascades into a blow-up. We avoid this entirely. NEWTON's MuJoCo-Warp back-end stores each joint's position-actuator gains in device arrays of shape (worlds, n_u) that are read afresh every step; on rupture we simply zero the broken degrees of freedom's gain and bias rows *in place*, per environment, removing the spring *and its implicit damping* with no recompile and no CUDA-graph recapture. The branch becomes a true free hinge and swings down at gravity rate. A snapped sub-tree's linear drag is also reduced (so the surrogate damping does not fake a slow fall), and Coulomb friction is introduced at the break and ramped up $\sim 5\times$ over $\sim 1.5\,\text{s}$ as the torn fibres seize, so the limb falls, swings through once, and is then pinned; it does not pendulum back above its release height or sway forever. The entire operation is a per-joint flag and a small array write; there is no frame-rate cost relative to a non-breakable tree.

Applied and interactive forces are kept unconditionally stable by three rails on the body force (impulse cap, power fade, speed brake) that never affect static loading; we defer the details to Section B.

C. Fruit

Apples are placed on eligible outer spurs with a bias toward the well-lit outer canopy and instantiated as *independent* rigid bodies, not joints of the tree. A heavy fruit rigidly jointed to a thin compliant spur is numerically stiff and unstable; instead each apple is held at its hang point by a one-sided spring–damper *tether*

$$\boldsymbol{f}_{\text{tether}} = -k_s\,(\boldsymbol{x} - \boldsymbol{x}_{\text{hang}}) - c_s\,\boldsymbol{v}, \tag{6}$$

applied to the apple only. To keep the apple's dominant cost down (each free body is expensive in a batched solver), the default fruit body has three translational degrees of freedom (an apple never needs to spin), which halves its DOF count relative to a full free body.

a) Loading the branch.: A pull on the fruit must load its branch, or the canopy would feel rigid to the manipulator. We react the *elastic* part of the tether force, minus the apple's static weight (which the tree already carries at rest), onto the parent spur at the attach point,

$$\boldsymbol{f}_{\text{react}} = -\big(\boldsymbol{f}_s - mg\,\hat{\boldsymbol{z}}\big), \qquad \boldsymbol{f}_s = -k_s(\boldsymbol{x} - \boldsymbol{x}_{\text{hang}}), \tag{7}$$

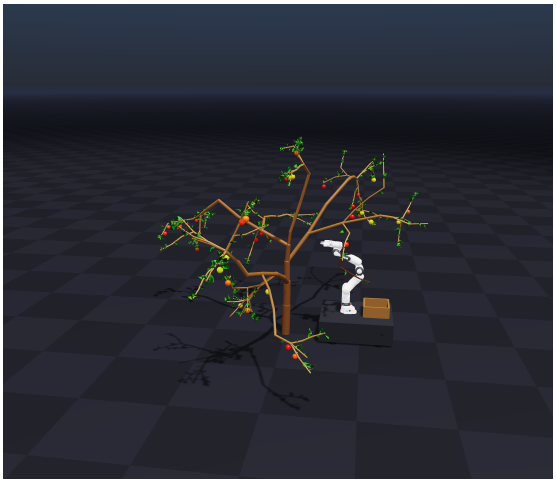


Fig. 3: Branch breaking under robot contact. As the manipulator pushes into the canopy, a limb loaded past its modulus of rupture snaps: it goes limp and droops (recoloured dead-brown), and drops out of the collision set so the debris cannot re-excite the solver. Damage of this kind is what the *branches-snapped* metric penalizes.

clamped to a maximum so a detaching yank can never chain-snap the tree. At rest the apple hangs a distance mg/k_s below $\boldsymbol{x}_{\text{hang}}$, so $\boldsymbol{f}_s = mg\hat{\boldsymbol{z}}$ and the reaction vanishes: only the elastic *perturbation* beyond rest is transmitted to the spur, and the rest pose and rupture margins are exactly those of a tree without fruit. Under a pull the spur visibly bends toward the hand, more on thin outer wood than on stiff scaffold.

b) Detachment.: The stem breaks along either of two paths: a direct pull force on the fruit exceeding a threshold f_{detach} sustained for a few frames, or the stem *tension* itself exceeding a slightly larger threshold for longer; the second is what lets a gripper holding the fruit purely by contact friction detach it, since contact forces never appear as a body force. Stems are strong: $f_{\text{detach}} \in [\,14, 23\,]$ N, grounded in measured apple detachment forces (Table III), so a light tug only bends the branch and a whipping branch never sheds fruit. A detached apple becomes a plain ballistic body that the drag and ground kernels settle; detachment itself is a single flag flip with no model edit.

D. Per-environment domain randomization

Domain randomization is a standard route to robust policies and sim-to-real transfer [29, 30]; we apply it to the whole-tree structure. To evaluate on a *population* of trees rather than one instance, every GPU-batched environment is a different tree. The constraint is that a batched solver requires the parallel worlds to be structurally homogeneous (identical body, joint and shape counts and types, position by position), while their *continuous values* may differ freely. We therefore generate one base skeleton and perturb only continuous quantities per world (Figure 4), holding the discrete topology fixed. One axis is treated specially. The per-branch *dimensions* (global scale and stockiness, and per-segment length and thickness) are the only perturbation whose *per-world* variation makes the batched solve expensive: distinct dimensions across the worlds of a step re-dimension every link and gate the parallel solve (Section VII). By default we therefore *share* one dimension draw across the whole batch, so the worlds stay dimensionally identical and step at the homogeneous rate, and resample it across resets so tree shape still varies over training. Every other axis varies per world for free, since it perturbs frames or values rather than link dimensions: per-segment bend and whole-tree growth *habit* (a gravimorphic droop multiplier applied inside a forward-kinematic re-walk so droop compounds per internode exactly as real arching does, plus whole-tree lean, fork spread and an env-wide phyllotactic twist); wood density, Young's modulus and rupture stress; fruit size, mass and visible count; leaf size and density; and coloration (wood tint, foliage colour, and an apple-palette hue shift). Per-world branch dimensions remain available for maximal visual variety, at a lower batched step rate. Fruit and leaf *counts* are held structurally fixed: per-tree visible fruit count is varied by shrinking a random subset to sub-pixel size rather than by adding or removing bodies, so the topology, and hence the batch, is preserved. The build replicates the base environment and patches the per-world values directly into the finalized device arrays, so startup stays fast and cross-environment render instancing survives (in full when dimensions are shared).

We stress that fixing the topology is a *within-batch* requirement for GPU batching, not a limit on the randomization. The topology itself is a stochastic draw of the generator: a fresh seed re-samples the scaffold count, the branching pattern, and the whole architecture, so different runs (or re-generations) produce genuinely different *structures*, not just different continuous values of one structure. Structural variation is therefore covered by running several batches, each a distinct topology randomized continuously within it; a single batch trades structural variety for the homogeneity that makes thousands of worlds step in parallel.

E. Ground, foliage, and solver

a) Terrain.: An optional heightfield provides a bumpy outdoor ground: a few octaves of value noise on a random lattice, bilinearly upsampled to a fine grid and flattened under each trunk with a smooth disc so the tree stays planted. For multi-env batches the noise is generated *periodic* at the display-grid pitch and tiled across the grid, and shared as a single static shape across all worlds (batching is unaffected). Its amplitude is grounded in agricultural soil random-roughness data (Table III); the manipulator's base rides the surface through a terrain-following vertical servo.

b) Foliage.: Leaves are folded, curled elliptical blade meshes placed with phyllotactic spacing on the outer twigs. Each leaf is rigidly attached to its parent branch body, so the whole canopy *moves with the tree*, swaying as branches bend and occluding the fruit the way a real canopy does, rather than being a static backdrop. Efficiency comes from instancing: all leaves of one of three discrete size classes share a single mesh, so the whole canopy renders in three draw batches regardless

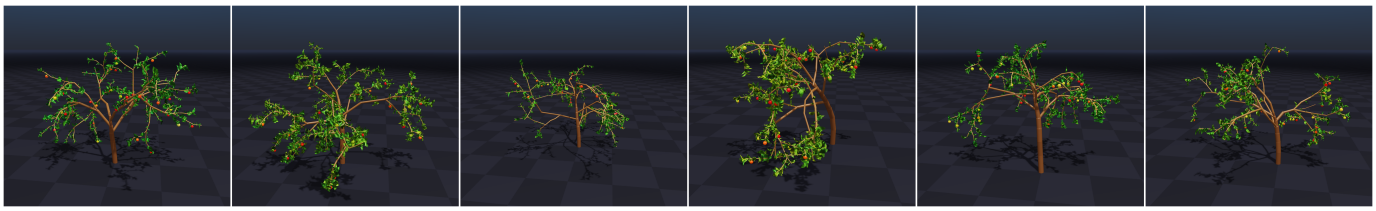


Fig. 4: **Domain randomization.** A single grammar and a fixed topology, randomized continuously, span a diverse population of distinct but plausible trees varying in scale, growth habit (upright to weeping), fork spread, taper, foliage, fruit and coloration. Within one GPU batch the branch *dimensions* (scale and taper) are shared across worlds so the parallel solve stays homogeneous and full-speed, and are resampled across batches; growth habit, material, fruit, foliage and coloration vary per world within the batch, and per-world dimensions are available as a slower option. Every world stays structurally identical and therefore GPU-batchable.

of leaf count, and the blades are massless and non-colliding so their physics cost is exactly zero. Foliage density is a continuous dial that controls leaf count and, above a threshold, populates inner branches to occlude the tree interior: this is the controlled occlusion variable for our perception experiments (Section VII).

c) Solver and contact.: The MuJoCo-Warp constraint solver offers two algorithms; we run its conjugate-gradient algorithm by default, which on a single large articulation is $\sim 7\times$ faster than the blocked-Cholesky algorithm that is NEWTON's default. On our regression suite (rest drift, static bends, detachment, break-fall timing, and full-speed rams) the two algorithms agree to within solver tolerance, so the speed-up does not change the physics; Section VII reports both step-rates. When the manipulator is present, contact uses NEWTON's own collision pipeline with group filtering (tree and fruit collide with the robot but never with each other, leaves never collide), so only a few hundred shapes reach the broad phase and robot–tree contact is nearly free. The delicate case is the thinnest twigs: a gram-scale branch against a 130 kg robot is a $\sim 10^4$:1 mass ratio that an unconditioned rigid contact cannot resolve—the usual reason such contacts are simply dropped, which lets the manipulator pass straight through a small branch. Instead of dropping them we *condition* the ratio: each thin-twig joint carries extra *armature*—added inertia in DOF space that regularises the contact impulse without changing the branch's static bend—and a small contact *margin* catches the contact before the fast arm has stepped through the thin capsule. The manipulator then genuinely cannot penetrate a branch: one it cannot push aside stalls the base, or, pushed hard enough, *ruptures*, the breaking model doubling as the physical release valve. This holds a twig centreline out of the chassis to within measurement noise while a full autonomous pick—arm deep in the canopy—stays stable, at a handful of extra contacts only when the arm is actually among the branches.

V. MANIPULATOR, PERCEPTION AND AUTONOMY

A. Mobile manipulator

The benchmark's embodied agent is IsaacLab's [17] RidgebackFranka: a Clearpath Ridgeback omnidirectional base carrying a Franka arm (instantiated from the real FR3 URDF).

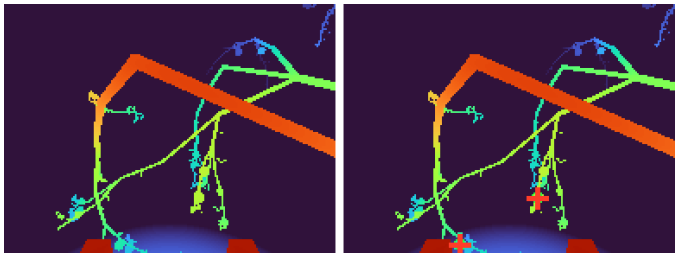


Fig. 5: Wrist depth camera (left) and geometric fruit detection (right): depth-continuous patches are fitted with spheres and gated; accepted apples are back-projected to world markers, self-detections of the robot's own links are masked from kinematics.

Following common practice, the base is not simulated wheel-by-wheel but driven through three planar degrees of freedom (world x, y, yaw) with velocity actuators whose effort is limited to a wheel-traction-scale bound, so the base stalls against obstacles like a real platform rather than acting as an unbounded crusher; a fourth vertical servo tracks the terrain height. One robot is added identically to every environment, preserving batch homogeneity. A wrist-mounted depth camera (NEWTON's GPU-raycast tiled-camera sensor) renders all worlds in a single call at a realistic 5 Hz, with a field of view and range matched to a consumer RGB-D sensor (Table III).

B. Geometric fruit perception

We deliberately use a classical, learning-free detector so that perception performance reflects the *scene*, occlusion, foliage, geometry, rather than a pretrained network, and so that the same pipeline could run unchanged on a real depth camera (the geometry-first localization philosophy of field harvesters). Apples are the only sphere-like surfaces in an orchard, and a sphere is identifiable from depth alone. The pipeline back-projects the depth image to a point cloud, segments it into depth-continuous patches, and for each patch fits a sphere by linear least squares: writing each point as $\boldsymbol{p}_i$, the identity $\|\boldsymbol{p}_i - \boldsymbol{c}\|^2 = R^2$ linearizes to

$$2\,\boldsymbol{p}_i^{\top}\boldsymbol{c} + b = \|\boldsymbol{p}_i\|^2, \qquad b = R^2 - \|\boldsymbol{c}\|^2, \tag{8}$$

Algorithm 1: Autonomous harvesting baseline (per environment).

Input: depth stream, robot state; **Output:** fruit deposited in bucket

loop *every control step*

- **Scan**: mask self/ground pixels; if canopy unseen, orient to belief; else survey-orbit for ripe fruit
- **Align**: drive base to a reach-aware stand-off facing the target
- **Reach**: DLS-IK to a pre-grasp pose behind the fruit
- **Grasp**: advance, close fingers, engage contact grip
- **Pull**: withdraw along approach axis until stem tension $>$ threshold
- **Deposit**: carry over bucket, release when centred and slow
- **if** *unreachable / timeout / stem too strong* **then** blacklist, **Scan**

solved for $(\boldsymbol{c}, b)$ and hence centre $\boldsymbol{c}$ and radius $R = \sqrt{b + \|\boldsymbol{c}\|^2}$. Candidates are then gated on radius (2–5.8 cm), millimetric fit residual, convexity toward the camera, silhouette isolation, and a *pixel-count consistency* check, a sphere of radius R at range z can subtend only $\approx \pi(Rf/z)^2$ pixels, which rejects leaf clusters that happen to fit a small sphere. Two *anti-branch* gates reject foreshortened limb sections that fit a small sphere alarmingly well: a footprint-elongation cap (a sphere cap is isotropic; a branch section stretches along its axis) and an explicit sphere-versus-cylinder residual comparison. Self-detections of the robot's own rounded links are masked from known kinematics, and the picker adds temporal persistence before committing. In a design-time ablation on 48 fixed canopy vantages these gates cut wood false positives from 14 to 3 (precision $0.78 \rightarrow 0.95$) at a -13% single-view recall cost that the survey orbit and temporal persistence recover; Section VII reports precision and recall under varying foliage.

C. Autonomous harvesting

The baseline controller is an analytic state machine mirroring the pick cycle of field systems (Algorithm 1). Exploration is driven by a per-pixel *scene mask*: pixels belonging to the robot's own body or to the ground/terrain are excluded before any "do I see the tree?" decision, and the picker maintains a persistent tree-centre belief so that when the canopy leaves view it reorients toward the remembered tree rather than rotating blind. Arm motion is damped-least-squares inverse kinematics solved on a separate arm-only model (a full-model solve would perturb the tree's joints), slewed into the arm's position servos. Grasping uses a real contact grasp centred between the fingers, whose closing force feeds the same stem-tension detachment detector as a manual pull; the fruit is then withdrawn gently along the approach axis until the stem gives, and deposited in a bucket on the robot's back. Failures (unreachable target, timeout, stem too strong) blacklist the target and return to exploration. Under multi-environment operation, an independent picker with its own perception and metrics runs on every world, with viewer overlays on the first; physics stays batched and per-robot host work is kept to small array uploads.

VI. Benchmark Tasks and Metrics

OrchardBench is a benchmark, not only a simulator: it fixes a set of tasks, a metric vocabulary, and a randomized evaluation protocol so that controllers, policies and detectors can be compared. All quantities are logged per attempt and per environment and saved as JSON for offline analysis.

a) Tasks.: The primary task is *autonomous harvesting*: starting from a stand-off, detect, approach, grasp, detach and deposit as much fruit as possible within a fixed episode, on a randomized tree, without breaking branches or dropping fruit. The environment also supports component tasks that isolate parts of the pipeline, *fruit detection* (report fruit poses from the depth stream against ground truth), *reaching/grasping* a specified fruit, and *gentle interaction* (push through the canopy while minimizing transmitted moment), and it exposes the underlying physics for open-loop verification of analytic controllers.

b) Metrics.: An episode reports:

- **Harvest completeness (primary):** fruit deposited divided by the *reachable* fruit present (fruit whose centre lies within the arm's reach sphere from an admissible stand-off). Unlike a per-attempt rate, this cannot be gamed by a controller that only attempts easy fruit.
- **Efficiency (secondary):** per-attempt place-success rate, grasp and detach rates, throughput (fruit deposited per simulated minute), and mean pick-cycle time.
- **Damage / loss:** *branches snapped* and *fruit dropped* (detached but not deposited), emitted unconditionally (default 0). These are first-class safety costs, since a harvester that maximizes throughput by damaging the tree is a failure.
- **Effort:** maximum stem pull force applied. As a peak transient it can exceed the quasi-static detachment band (the grip clamp permits a brief overshoot); we report it as a safe-force diagnostic for a real end-effector.
- **Perception:** detection precision, the fraction of reported detections that match a real fruit (a true positive is an estimated centre within one apple radius of an unmatched ground-truth fruit, one-to-one assignment). Per-frame recall against visible ground-truth fruit is supported by the environment and left to future evaluation.
- **Compute:** simulation step rate, environment-steps per second available to a policy, and peak GPU memory, so results carry their cost.

c) Canopy-zone breakdown.: Because a harvester's difficulty varies systematically over the canopy, success is also reported by *zone*. Each environment self-calibrates a zone frame from its own reachable fruit: the trunk axis is the robust horizontal median of fruit positions, the vertical extent is split into lower/middle/upper thirds between the 5th and 95th height percentiles, and the radial extent is split at the median distance from the trunk axis into inner/outer. Every pick is tagged with its (vertical $\times$ radial) zone and the reachable-fruit census of each zone is recorded, so that per-zone success can be normalized by the fruit actually present there (Figure 10a).

TABLE II: Baseline autonomous-harvesting results (nominal condition: 40 apples, foliage density 0.6, no terrain; pooled over 18 domain-randomized trees, 3 seeds). The analytic baseline leaves substantial headroom, especially in fruit dropped, which is the point of a benchmark.

Metric	Value
Success per detected fruit (per attempt)	0.41
Harvest completeness (picked / reachable)	0.12
Throughput per robot (fruit/min)	1.9
Mean pick-cycle time (s)	7.3
Detection precision	0.90
Branches snapped per tree	$\sim$0.2
Fruit dropped per tree	$\sim$6
Max stem pull force (N, peak)	29

This directly quantifies the intuition that deep-inner and high fruit are hardest, which a single aggregate success rate hides.

d) Evaluation protocol.: Because the closed-loop task is chaotic (small perturbations in contact ordering change which branch is nudged and whether a grasp slips), single episodes are not meaningful. We therefore fix an official, versioned evaluation seed set (ORCHARDBENCH-V1: K held-out seeds, each a domain-randomized population, disjoint from any tuning seeds) and a fixed episode horizon (a sim-time budget; an episode ends on that budget, on all-reachable-fruit picked, or on an unrecoverable stall). A method reports each metric as a mean with a 95% bootstrap confidence interval over trees $\times$ seeds; for the shared-tree ablation sweeps we use paired tests and report effect sizes, and we separate within-tree (run-to-run) from between-tree variance. This ensemble discipline is part of the benchmark, not an afterthought. For the agentic setting, a submission optimizes harvest completeness subject to damage constraints (branches snapped $\leq b$, fruit dropped $\leq d$) and reports the achieved Pareto point; the leaderboard ranks on constrained completeness. A one-line mapping of these metrics onto the four use cases of Section I is deferred to that section.

VII. EXPERIMENTS

a) Setup.: Unless noted, experiments use the stochastic apple generator with breaking enabled, the autonomous baseline of Section V, and N domain-randomized environments run in parallel on a single NVIDIA RTX 2000 Ada laptop GPU (8 GB). Each sweep fixes the random seed so the *same* population of trees is reused across conditions, making every sweep a paired comparison in which only the swept variable changes. Metrics are pooled across the parallel trees and reported with 95% bootstrap confidence intervals; the canopy-zone table pools across all twelve autonomous runs (80 trees). Table II gives the nominal-condition baseline. Each robot deposits about 1.9 fruit per minute, below the $60/7.3 \approx 8$ per minute that a continuous 7.3 s pick cycle would allow, because much of the episode is spent searching, on failed attempts, and in recovery; throughput is reported per robot (the parallel environments are independent single-robot trials, not one multi-robot stand).

A. Parallelism and cost

For learning research the key question is how many environments the benchmark can run at once. Figure 6 sweeps parallel trees physics-only (no rendering) across three randomization modes and both constraint solvers. A single tree steps at 60–100 fps (CG, depending on the drawn shape); as environments are added the per-step rate falls but aggregate throughput keeps climbing. Homogeneous (non-randomized) batches and *shared-dimension* domain-randomized batches (our default, Section IV-D) track each other closely: both reach 512 trees and plateau near 3500–4400 environment-steps/s, confirming that shared-dimension DR steps at essentially the homogeneous rate. *Distinct-geometry* DR (per-world branch dimensions) does not scale—its throughput flatlines near 140 environment-steps/s and its step rate collapses (66 versus 4.4 fps at 32 trees, a $\sim 15\times$ gap)—because heterogeneous kinematic geometry across worlds breaks the batched solve. The conjugate-gradient solver reaches 512 trees ($\sim$170 000 bodies) using only $\sim$1.5 GB of the 8 GB laptop GPU, so the throughput plateau is compute-bound with ample memory headroom; the default blocked-Cholesky ("Newton") algorithm caps at 256 trees, runs 2–3$\times$ slower and uses roughly twice the memory, so the matrix-free CG solve is what makes laptop-scale batches practical. In-place rupture and reduced-DOF fruit keep breaking and fruit essentially free; the bottleneck at scale is the constraint solver, not asset storage.

Two mechanisms underlie this, one at build time and one per step. The build: computing each world's distinct geometry with a host-side forward-kinematic re-walk is $O(N)$, capping distinct-geometry builds at a few tens of trees. We remove this ceiling with an *on-device* build that perturbs every world's branch geometry in a single Warp kernel (one thread per world; fruit and foliage re-homing remain on the host path); it reproduces the replicated base model exactly at zero strength and produces stable, diverse trees at a build cost comparable to a homogeneous batch (e.g. 6.5 s versus 29 s for the host path at 64 trees, and 22 s at 512 where the host path takes minutes). Second, and independent of the build path, is a *per-step* cost, but we find it is confined to a single randomization axis. Decomposing the per-world perturbation and measuring each in isolation (8 settled worlds), randomizing per-world *mass*, joint *stiffness* and *damping* runs at the homogeneous step rate over wide ranges, individually and combined (100–133 fps against $\sim$110 fps for identical worlds), as does randomizing the growth-habit *angles* (droop, lean, spread, twist; $\sim$83 fps) and all *fruit and foliage* attributes (apple size, mass, count and colour; leaf size and colour), since fruit are independent free bodies and leaves are massless render-only cards. The only expensive axis is per-branch *size*, segment length, thickness, and global scale, which re-dimensions every link and drops the batch to 8–13 fps; the slowdown persists after the trees settle to rest, so it is a solver cost of the heterogeneous *kinematic geometry*, not a transient. The practical consequence is favourable: the standard reinforcement-learning randomizations (dynamics and gains) and the visually dominant shape variation (growth habit)

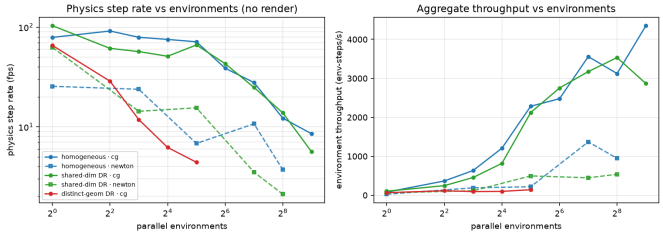


Fig. 6: Physics step rate (left) and aggregate environment throughput (right) versus the number of parallel trees (no rendering), for three randomization modes and both constraint solvers. *Shared-dimension* DR (green) tracks the *homogeneous* batch (blue) and both scale to 512 trees, whereas *distinct-geometry* DR (red) flatlines as its heterogeneous per-world geometry breaks the batched solve; CG (solid) far outscales the Newton solver (dashed). Single laptop GPU; each step-rate cell is a median over repeats to average run-to-run clock variation.

are essentially free at scale, and only re-dimensioning each branch gates the batched solve. Crucially, this cost is incurred only when the per-branch size differs *simultaneously* across the worlds of a single batched step; it is not a cost of varying size at all. Size variation is therefore recovered at scale by holding one size draw *shared* across the batch, so every step stays dimensionally homogeneous and runs at the full rate, and resampling that draw *across resets* so the training distribution still spans diverse shapes over time. A large diverse batch can thus randomize dynamics, materials, habit and posture per world at full speed while sharing one branch-size realization across the batch, trading within-batch geometric decorrelation, at the cost of a per-reset rebuild, for a homogeneous, full-speed step. The rates above are physics-only: unlike the solve, rendering is not batched across worlds, so visualising a large batch on-screen is far slower (a windowed view of 100 worlds drops to single-digit fps). Large-batch data collection therefore runs headless, with rendering reserved for inspecting a handful of worlds at a time.

B. Effect of foliage (perception under occlusion)

Foliage density is swept with all else fixed, so the same trees are seen through increasing occlusion. As density rises from 0 to 1.5, detection precision falls from 0.98 to 0.81 (more leaf clutter produces more false positives that the geometric gates must reject) and the per-attempt place rate falls from 0.47 to 0.34 (Figure 7). This is the variable-foliage perception and manipulation difficulty the moving foliage layer is designed to expose, here with perfect ground truth.

C. Effect of fruit load and terrain

Figures 8a and 8b sweep the number of apples and the terrain amplitude. More fruit raises per-robot throughput (2.1–4.0 fruit/min from 20 to 60 apples) but lowers completeness (0.26 to 0.13), since a denser tree cannot be cleared within the episode horizon. Terrain up to the 20 cm orchard-alley ceiling

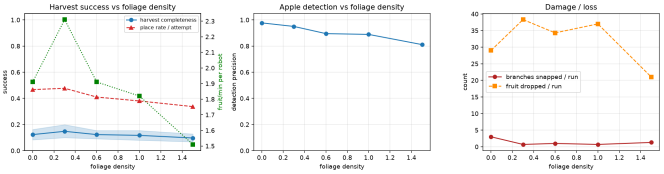
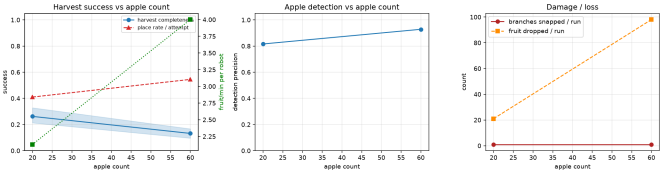
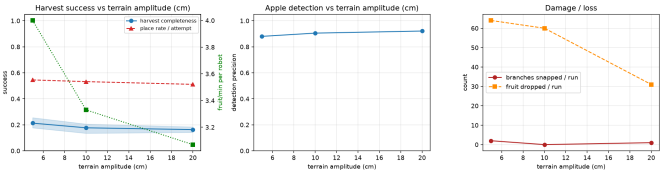


Fig. 7: Foliage sweep: harvest success and per-robot throughput (left), detection precision (centre), and damage/loss (right) versus foliage density. Both detection precision and picking success degrade as foliage occludes the fruit. Shaded band: 95% bootstrap confidence interval over trees (3 seeds × 6 trees per point).



(a) Fruit-load sweep (20–60 apples).



(b) Terrain-amplitude sweep (5–20 cm).

Fig. 8: Sweeps over fruit load and terrain roughness.

remains drivable: the per-attempt place rate falls only mildly, from 0.55 at 5 cm to 0.51 at 20 cm, so navigation on bumpy ground is not the dominant failure mode. The failure-reason breakdown (Figure 9) attributes most misses to false-positive detections (*no fruit at detection*) and grasp stalls rather than navigation.

D. Success by canopy zone

Figure 10a reports place-success over the 3×2 canopy grid, pooled across all runs (80–191 attempts per zone). Success is highest in the middle canopy (~ 0.55) and lowest in the lower-outer and upper zones (0.37–0.42), matching the intuition that fruit hanging below the outer canopy and high on the tree are hardest to reach and see.

E. Tree-to-tree variation and repeatability

Across the 18 trees of the baseline and extra-seed runs, the number of apples deposited per tree is 2.4 ± 1.5 (mean $\pm$ s.d., range 0–5; Figure 10b). This spread, together with the chaotic per-episode dynamics, is why we evaluate on a population and report confidence intervals: a single episode on a single tree would not be representative.

VIII. Discussion

The baseline results sketch the shape of the problem OrchardBench poses. Perception is the dominant bottleneck:

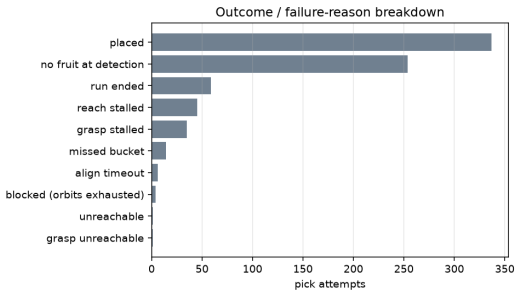
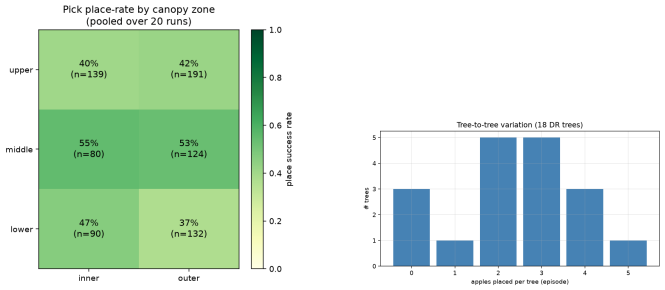


Fig. 9: Pooled outcome and failure-reason breakdown. Most misses are false-positive detections and grasp stalls rather than navigation, which makes each sweep interpretable.



(a) place-rate by zone (b) per-tree spread

Fig. 10: (a) Place-success rate over the 3×2 canopy grid, colour-coded, with per-zone attempt counts. (b) Histogram of apples deposited per tree across the 18-tree population: most trees yield one to four, illustrating the tree-to-tree and run-to-run spread that motivates population-level evaluation with confidence intervals.

the most common failure is a committed detection with no fruit actually there, and detection precision falls steadily as foliage occludes the canopy, dragging picking success down with it. The analytic baseline also exposes a sharp damage-versus-throughput tension: it succeeds on about two in five of its attempts but knocks roughly six fruit per tree to the ground in the process, so a method that merely maximized throughput would score poorly on the damage metrics that a real grower cares about. The canopy-zone profile localizes the difficulty to the deep-inner and upper canopy, suggesting that better viewpoint planning and reach strategies, not faster motion, are where the gains are. These are exactly the levers a learned or agentic method would tune, and the metric suite is designed to reward doing so without trading away plant safety.

A. Limitations

We are deliberately conservative about what ORCHARDBENCH does and does not establish. First, the physics is *grounded* in the literature, not *validated* against physical branch-break or grasp experiments: the Euler–Bernoulli torsional-spring lumping is an idealisation, and the green-wood modulus and rupture stress are reduced estimates from dried-wood references rather than direct measurements on live apple branches (Table III). The detachment force is set within, not calibrated to, a broad (9–40 N) reported spread. Establishing quantitative agreement with real measurements is important future work, and until then results should be read as *relative* comparisons within the simulator rather than absolute predictions of field performance. Second, the fruit model omits pedicel *twisting*: detachment is triggered by pull and stem tension but not by the twist-and-pull motion many pickers use. Third, sensing is depth-only and the renderer is not a validated photometric model, so RGB-based perception transfer is out of scope here. Fourth, and most importantly, *sim-to-real transfer is a design goal we support, not a result we demonstrate*: no real-robot experiment is reported, and the domain randomization is motivated by, not shown to achieve, transfer. Finally, we position ORCHARDBENCH as a substrate for learned policies and agentic optimization and report the environment throughput available to a policy, but we evaluate only the analytic baseline here; a learned or agentic baseline is future work, and the four use cases should be read as the interface we provide rather than as validated results.

B. Future work

ORCHARDBENCH is designed as a substrate that these extensions slot into without re-architecting:

- **Learned policies and perception.** The GPU-batched, randomized environment is ready for reinforcement- and imitation-learning of harvesting policies and for learned (RGB-D) detectors, with our analytic controller and geometric detector as baselines to beat.
- **Sim-to-real validation.** The most valuable next step is closing the loop with a real Ridgeback–Franka in an orchard or on cut branches, to calibrate the wood and detachment parameters and measure the transfer gap: this is the experiment that would upgrade our claims from grounded to validated.
- **Fruit twisting.** Adding a twist degree of freedom and a torsion-dependent detachment criterion to the stem model is a small extension that would capture the twist-and-pull strategy of human and robotic pickers.
- **Photorealistic rendering.** A validated photometric renderer (realistic foliage and lighting) would extend the benchmark to RGB perception and appearance-based sim-to-real, if the target application requires it.
- **Other crops and structures.** The generation and physics pipeline is not apple-specific; trellised vines, stone fruit, and other trained systems are natural targets, making ORCHARDBENCH a template for a family of physically-grounded agricultural benchmarks.
- **A benchmark for automated research.** Because the task is drawn from a real agricultural problem and the metric vector explicitly includes damage and safety, ORCHARDBENCH is a real-world-grounded target for automated research systems that co-optimize perception and control to push throughput up while holding branch breakage and fruit drop down.

C. Conclusion

ORCHARDBENCH brings the physical realism of a compliant, breakable, fruit-bearing tree, long the missing ingredient, into a GPU-parallel, domain-randomized simulation that runs on a laptop, and packages it as a benchmark with a metric suite matched to what agricultural harvesting actually requires: not just success, but throughput at bounded damage. Our analytic baseline clears only a fraction of the reachable fruit and drops more than it keeps, leaving ample room for the learning, perception, and agentic methods the benchmark is built to measure. By grounding every physical parameter in the literature and releasing the environment openly, we aim to let these communities iterate on orchard robotics safely and at a scale the field itself can never offer.

REFERENCES

[1] C. Wouter Bac, Eldert J. van Henten, Jochen Hemming, and Yael Edan. Harvesting robots for high-value crops: State-of-the-art review and challenges ahead. *Journal of Field Robotics*, 31(6):888–911, 2014.

[2] Abhisesh Silwal, Joseph R. Davidson, Manoj Karkee, Changki Mo, Qin Zhang, and Karen Lewis. Design, integration, and field evaluation of a robotic apple harvester. *Journal of Field Robotics*, 34(6):1140–1159, 2017.

[3] Gert Kootstra, Xin Wang, Pieter M. Blok, Jochen Hemming, and Eldert van Henten. Selective harvesting robotics: Current research, trends, and future directions. *Current Robotics Reports*, 2:95–104, 2021.

[4] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, et al. Isaac gym: High performance GPU-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.

[5] Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5026–5033, 2012.

[6] Przemyslaw Prusinkiewicz and Aristid Lindenmayer. *The Algorithmic Beauty of Plants*. Springer-Verlag, New York, 1990.

[7] Hisao Honda. Description of the form of trees by the parameters of the tree-like body. *Journal of Theoretical Biology*, 31(2):331–338, 1971.

[8] Evelyne Costes, Colin Smith, Michael Renton, Yann Guédon, Przemyslaw Prusinkiewicz, and Christophe Godın. MAppleT: simulation of apple tree development using mixed stochastic and biomechanical models. *Functional Plant Biology*, 35(10):936–950, 2008.

[9] Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J. Davison. RLBench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters (RA-L)*, 5(2):3019–3026, 2020.

[10] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Proceedings of the 3rd Conference on Robot Learning (CoRL)*, volume 100 of *Proceedings of Machine Learning Research*. PMLR, 2019.

[11] Tongzhou Mu, Zhan Ling, Fanbo Xiang, et al. ManiSkill: Generalizable manipulation skill benchmark with large-scale demonstrations. In *NeurIPS Datasets and Benchmarks Track*, 2021.

[12] Jayadeep Jacob, Shizhe Cai, Paulo Vinicius Koerich Borges, Tirthankar Bandyopadhyay, and Fabio Ramos. Gentle manipulation of tree branches: A contact-aware policy learning approach. In *Proceedings of the 8th Conference on Robot Learning (CoRL)*, volume 270 of *Proceedings of Machine Learning Research*. PMLR, 2024. URL https://proceedings.mlr.press/v270/jacob25a.html. PCAP: Proprioceptive Contact-Aware Policy.

[13] Kichiro Shinozaki, Kyoji Yoda, Kazuo Hozumi, and Tatuo Kira. A quantitative analysis of plant form – the pipe model theory: I. basic analyses. *Japanese Journal of Ecology*, 14(3):97–105, 1964.

[14] Romain Lehnebach, Robert Beyer, Véronique Letort, and Patrick Heuret. The pipe model theory half a century on: a review. *Annals of Botany*, 121(5):773–795, 2018.

[15] Katherine A. McCulloh, John S. Sperry, and Frederick R. Adler. Water transport in plants obeys murray's law. *Nature*, 421(6926):939–942, 2003.

[16] Takuya Okabe. Biophysical optimality of the golden angle in phyllotaxis. *Scientific Reports*, 5:15358, 2015.

[17] Mayank Mittal, Calvin Yu, Qinxi Yu, Jingzhou Liu, Nikita Rudin, David Hoeller, Jia Lin Yuan, Ritvik Singh, Yunrong Guo, Hammad Mazhar, Ajay Mandlekar, Buck Babich, Gavriel State, Marco Hutter, and Animesh Garg. Orbit: A unified simulation framework for interactive robot learning environments. *IEEE Robotics and Automation Letters*, 8(6):3740–3747, 2023. doi: 10.1109/LRA.2023.3270034. Framework now maintained as Isaac Lab, https://isaac-sim.github.io/IsaacLab/ (accessed 2026).

[18] Miles Macklin. Warp: A high-performance python framework for GPU simulation and graphics. In *NVIDIA GTC / GitHub (NVIDIA/warp)*, 2022.

[19] C. Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax – a differentiable physics engine for large scale rigid body simulation. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2021.

[20] Kevin Zakka, Baruch Tabanpour, Qiayuan Liao, Mustafa Haiderbhai, Samuel Holt, Jing Yuan Luo, Arthur Allshire, Erik Frey, Koushil Sreenath, Lueder A. Kahrs, Carmelo Sferrazza, Yuval Tassa, and Pieter Abbeel. MuJoCo playground: An open-source framework for GPU-accelerated robot learning and sim-to-real transfer. *arXiv preprint arXiv:2502.08844*, 2025.

[21] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, Li Yi, Angel X. Chang, Leonidas J. Guibas, and Hao Su. SAPIEN: A simulated part-based interactive environment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*

(CVPR), pages 11097–11107, 2020.

[22] Genesis Authors. Genesis: A universal and generative physics engine for robotics and beyond. https://github.com/Genesis-Embodied-AI/Genesis, 2024. Open-source project (accessed 2026).

[23] NVIDIA, Google DeepMind, and Disney Research. Newton: An open-source GPU-accelerated physics engine for robotics simulation. https://github.com/newton-physics/newton, 2025. Built on NVIDIA Warp and OpenUSD; contributed to the Linux Foundation. Newton 1.3, docs https://newton-physics.github.io/newton/ (accessed 2026).

[24] Zhao Zhang, C. Igathinathane, Jieqing Li, Haiyan Cen, Yu Lu, and Paulo Flores. Technology progress in mechanical harvest of fresh market apples. *Computers and Electronics in Agriculture*, 175:105606, 2020.

[25] Amol Gongal, Suraj Amatya, Manoj Karkee, Qin Zhang, and Karen Lewis. Sensors and systems for fruit detection and localization: A review. *Computers and Electronics in Agriculture*, 116:8–19, 2015.

[26] Anand Koirala, Kerry B. Walsh, Zhenglin Wang, and Cheryl McCarthy. Deep learning – method overview and review of use for fruit detection and yield estimation. *Computers and Electronics in Agriculture*, 162:219–234, 2019.

[27] Nicolai Häni, Pravakar Roy, and Volkan Isler. MinneApple: A benchmark dataset for apple detection and segmentation. In *IEEE Robotics and Automation Letters (RA-L)*, 2020.

[28] Sercan Ozden and A. Roland Ennos. Why don't branches snap? the mechanics of bending failure in three temperate tree species. *Trees*, 28:1657–1667, 2014.

[29] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[30] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[31] Forest Products Laboratory. Wood handbook—wood as an engineering material. Technical Report FPL-GTR-190, U.S. Department of Agriculture, Forest Service, 2010.

[32] Eric Meier. Apple (Malus spp.) — the wood database. https://www.wood-database.com/apple/, 2020. Reports: specific gravity 0.61 (basic)/0.83 (12% MC), MOR 88.3 MPa, elastic modulus 8.76 GPa, crushing strength 41.6 MPa, Janka hardness 7700 N. Accessed 2026.

[33] Karl J. Niklas and Hanns-Christof Spatz. Worldwide correlations of mechanical properties and green wood density. *American Journal of Botany*, 97(10):1587–1594, 2010.

[34] David E. Kretschmann. Mechanical properties of wood. In *Wood Handbook—Wood as an Engineering Material*, General Technical Report FPL-GTR-190, chapter 5, pages 5–1–5–46. U.S. Forest Products Laboratory, Madison, WI, 2010.

[35] Kenneth R. James. A study of branch dynamics on an open-grown tree. *Arboriculture & Urban Forestry*, 40(3): 125–134, 2014.

[36] Richard S. Smith, Soazig Guyomarc'h, Therese Mandel, Didier Reinhardt, Cris Kuhlemeier, and Przemyslaw Prusinkiewicz. A plausible model of phyllotaxis. *Proceedings of the National Academy of Sciences*, 103(5): 1301–1306, 2006. doi: 10.1073/pnas.0510457103.

[37] Terence L. Robinson. The evolution towards more competitive apple orchard systems in the USA. *Acta Horticulturae*, 772:491–500, 2008.

[38] Lingxin Bu, Chengkun Chen, Guangrui Hu, Jianguang Zhou, Adilet Sugirbay, and Jun Chen. Experimental and simulation analysis of the detachment force of apple fruit. *Scientia Horticulturae*, 261:108937, 2020.

[39] Jianfeng Li, Manoj Karkee, Qin Zhang, Kanghong Xiao, and Tao Feng. Characterizing apple picking patterns for robotic harvesting. *Computers and Electronics in Agriculture*, 127:633–640, 2016. doi: 10.1016/j.compag.2016.07.024.

[40] M. Parameswarakumar and C. P. Gupta. Design parameters for vibratory mango harvesting system. *Transactions of the ASAE*, 34(1):14–20, 1991. doi: 10.13031/2013.31615.

[41] Clearpath Robotics. Ridgeback omnidirectional indoor robot platform (datasheet). https://clearpathrobotics.com/ridgeback-indoor-robot-platform/, 2024. Nominal max speed 1.1 $\mathrm{m\,s^{-1}}$, $\sim$135 kg payload class; accessed 2026.

[42] Intel Corporation. Intel realsense depth camera D435i (product datasheet). https://www.intelrealsense.com/depth-camera-d435i/, 2024. Depth FOV $\sim$87°$\times$58°, range $\sim$0.3–3 m; accessed 2026.

[43] Niko E. C. Verhoest, Hans Lievens, Wolfgang Wagner, Jesús Álvarez Mozos, M. Susan Moran, and Francesco Mattia. On the soil roughness parameterization problem in soil moisture retrieval of bare surfaces from synthetic aperture radar. *Sensors*, 8(7):4213–4248, 2008. doi: 10.3390/s8074213.

Appendix A
Parameter Provenance

A central design principle of OrchardBench is that physical parameters are not free knobs but are tied to published measurements, so that the simulator's behaviour is defensible and its numbers are meaningful across the four target use cases. Table III lists the principal parameters, their default values, and their source. Where a value is a deliberate simulation compromise (e.g. tree height or apple size, chosen for the arm's reach envelope or the gripper's opening), the note states so and gives the real figure; where a value is a numerical/solver setting rather than a physical one, it is marked as such. Green-wood elastic modulus and rupture stress are reduced estimates from dried-wood references (green wood runs $\sim$10–40 % below the 12%-moisture values) and should be treated

as literature-grounded ranges rather than validated constants (Section VIII-A).

APPENDIX B
NUMERICAL STABILITY

Interactive and applied forces (the mouse pick, the autonomous grip spring, the twig brush) are made unconditionally stable by three rails on each body's accumulated force, none of which affects static loading, bending a branch until it snaps still sees the full force. (1) An *impulse cap*: an applied force may not accelerate a body past a speed limit in one substep. (2) A *power fade*: the component of an applied force along a body's own velocity fades out above a threshold speed, "a hand cannot keep pushing something flying away." (3) A soft *speed brake* above the limit. These matter because the pick clamp scales with the *whole articulation's* mass, so the picker may legally apply a large force to a gram-scale twig; without the rails, the instant a rupture frees such a twig it would take an explosive velocity in one substep and NaN the articulation. Gram-scale twigs also take no rigid contacts (their extreme mass ratio is unresolvable); the robot pushes them aside with a bounded penalty "brush" force routed through the same rails. Snapped sub-trees additionally drop out of collision and have their applied forces disabled, so debris cannot be re-excited.

APPENDIX C
REPRODUCTION

All experiments run on a single NVIDIA RTX 2000 Ada laptop GPU (8 GB) with NEWTON 1.3 / Warp 1.14. Each condition fixes the random seed so the same domain-randomized population is reused, and metrics are pooled across parallel environments. The experiment runner executes the matrix one job at a time, writing per-condition JSON and a combined results table, and an analysis script regenerates every figure and table from those files. The picking sweeps share a single base seed so the same domain-randomized population recurs across conditions (a clean paired design); the full command matrix, exact seeds, and the official evaluation seed set accompany the code release at https://humphreymunn.github.io/orchardbench-page/.

TABLE III: Principal physical parameters, defaults, and their provenance. Ranges in brackets are the domain-randomization or literature spread. "sim. choice" marks deliberate compromises; "solver" marks numerical-stability settings. Entries with a citation are grounded in that source; parenthetical notes marked "nominal" are representative values from standard horticultural practice or manufacturer datasheets, chosen as reasonable defaults rather than measured constants.

Parameter	Default (range)	Source / justification
Wood material (green apple, Malus domestica*)*		
Density ρ_w	850 kg m^{-3} ([780,1000])	Basic SG 0.61; [31, 32]
Young's modulus E	7 GPa ([6,8])	Dried ≈8.8 GPa; green ∼10–25% lower [31, 33, 34]
Modulus of rupture σ_r	50 MPa ([45,60])	Dried ≈88 MPa; green ∼35–40% lower; low end for greenstick tolerance [28, 31]
Stiffness-prop. damping coeff. c_d	0.1 s	*Solver* value ($K_\mathrm{d} = c_d K_\mathrm{p}$; *not* a modal ratio, see Equation (4)); physical branch $\zeta \approx 0.01$–0.075 [35]
Morphology		
Beam stiffness law	$K_\mathrm{p} = \frac{\pi}{4} E r^4 / \ell$	Euler–Bernoulli; branch model of [12]
Pipe-model exponent β	2.2–2.3	Area-preserving (2) to Murray (2.5) [13–15]
Phyllotactic divergence	137.5°	Golden angle [16, 36]
Scaffold crotch angle	42–65°	Strong-crotch / fruit-bud range (nominal, horticultural practice)
Tree height	2.4 m	*Sim. choice* (arm reach); modern dwarf/semi-dwarf ∼3–3.5 m [37]
Trunk radius	0.035 m	Mature M.9 trunk ∼5–7.5 cm dia. (nominal)
Fruit		
Stem detachment force	14–23 N	Within reported 9–40 N spread [38–40]
Apple mass	0.16 kg ([0.09,0.40])	Commercial dessert apple ∼150–180 g (nominal grade weight)
Apple diameter	4.8–7 cm	*Sim. choice* (Franka 80 mm gripper); real ∼58–92 mm
Pedicel length	25 mm	Pomological descriptions (nominal)
Leaf blade	8.5×5 cm	Elliptic-ovate, L:W≈1.7 (nominal)
Platform & environment		
Base max speed	1.1 m s^{-1}	Clearpath Ridgeback datasheet [41]
Base drive effort limit	900 N	Traction μmg, 135 kg, $\mu \approx 0.7$ (datasheet [41] + *solver*)
Depth camera FOV / range / rate	75° / 0.3–3 m / 5 Hz	Consumer RGB-D, Intel RealSense D435i class [42]
Terrain roughness	3 cm (up to 20 cm)	Agricultural soil random-roughness [43]