Repair2Skill: A Vision-Language-Action Framework for Robotic Furniture Repair

Mukesh Mani Tripathi,[1] Huang Xin,[1] Qingqing Li[1]

[1]Towson University

mtripat1@students.towson.edu, xhuang@towson.edu, qingqingli@towson.edu

Abstract

Repair tasks differ fundamentally from robotic assembly, requiring fault localization, reverse dependency reasoning, and safe interaction with structurally degraded objects. While recent vision-language models have shown promise for high-level task reasoning, applying them to repair scenarios demands careful integration with perception, dependency modeling, and execution constraints.

This paper presents Repair2Skill, a vision-language-action framework for robotic furniture repair that focuses on repair planning and simulation-based execution validation. The system detects damaged furniture components using a lightweight SSDLite-MobileNetV3 detector trained on procedurally generated synthetic damage data, constructs repair-oriented dependency graphs, and generates structured repair plans using a constrained large language model. Repair plans are restricted to a fixed action vocabulary and strict JSON schema to ensure deterministic parsing and feasibility. Generated plans are validated and visualized in a PyBullet simulation environment using kinematic action primitives, enabling interpretable verification of action ordering and dependency resolution without contact-rich physical manipulation.

Experimental results demonstrate reliable part-damage association, achieving 76.8% mAP on real images, 100% valid repair plan generation under strict constraints, and stable simulation-based validation across diverse repair scenarios. These results suggest that lightweight, constrained vision-language pipelines can support autonomous repair planning in bounded domains and provide a foundation for future work on physically grounded robotic repair.

Introduction

Robotic assembly systems typically rely on predefined task graphs and forward-only dependencies, assuming intact components and known initial conditions. Repair tasks violate these assumptions by requiring fault localization, reverse dependency reasoning, and safe interaction with degraded structures, which fundamentally alters task structure and invalidates assembly-oriented pipelines.

Recent vision-language models enable high-level reasoning for robotic tasks, but unconstrained language outputs are often unreliable, difficult to parse, and may propose unsafe or infeasible actions. Existing approaches also rarely account for repair-specific constraints such as damage-aware access reasoning and reversible disassembly.

We introduce Repair2Skill, a vision-language-action framework for robotic furniture repair that focuses on repair planning and simulation-based execution validation rather than physical manipulation. The system combines damage-aware visual perception, repair-oriented dependency graphs, and constrained language-based planning to generate structured, executable repair plans.

Repair2Skill detects damaged components using a lightweight SSDLite-MobileNetV3 detector trained on procedurally generated synthetic data. A constrained large language model generates repair plans using a fixed action vocabulary and strict JSON schema, ensuring deterministic parsing and feasibility. Generated plans are validated and visualized in a PyBullet simulation environment using kinematic action primitives to verify dependency resolution and action ordering.

This work contributes a repair-specific formulation and integration of vision, language-based reasoning, and execution validation, demonstrating that lightweight vision-language pipelines can support autonomous repair planning within bounded domains. Repair2Skill aligns with the broader vision of AI-enabled tactical autonomy by integrating sensing, structured reasoning, and validated execution within a unified loop. Rather than treating perception and planning as isolated modules, the framework demonstrates how damage-aware sensing can directly inform constrained reasoning and simulation-validated action generation. This sensing-to-execution integration is essential for autonomous systems operating in uncertain and degraded environments.

Scope Clarification. This work focuses on repair planning and execution validation, not physical repair execution on hardware. We evaluate whether a robot can reliably identify damaged components, reason about repair dependencies, and generate executable repair plans that are logically correct and simulation-feasible. Contact-rich manipulation, force control, and grasp stability are explicitly outside the scope of this paper and are left to future work.

Repair Versus Assembly

Recent work such as Manual2Skill (Sun et al. 2025) has demonstrated that robots can extract procedural knowledge

from textual assembly manuals, enabling complex language-grounded assembly tasks. However, repair differs fundamentally from assembly in both problem structure and execution requirements. Assembly typically assumes intact components, forward-only dependencies, and known initial conditions. In contrast, repair requires fault localization, reverse dependency reasoning, and safe interaction with structurally compromised objects.

Specifically, repair tasks involve identifying failed components, reasoning about which intact parts obstruct access to the damaged region, and performing reversible disassembly prior to restoration. Damage patterns further introduce significant visual and structural variability, while manipulation of degraded structures imposes additional safety constraints absent in standard assembly settings. These challenges cannot be addressed by directly reusing assembly pipelines or forward task graphs.

Contributions

This work makes the following contributions:

1. **Repair-Oriented Problem Formulation.** We formalize robotic furniture repair as a dependency-constrained planning problem that explicitly models damage states, reverse disassembly requirements, and access constraints, distinguishing repair from forward-only assembly pipelines.
2. **End-to-End Vision-Language-Action Framework.** We present Repair2Skill, an integrated framework that combines damage-aware visual perception, dependency-aware repair planning, and simulation-based execution validation in a unified pipeline designed specifically for repair tasks.
3. **Constrained LLM-Based Repair Planning.** We introduce a constrained large language model planner that generates strictly structured, executable repair plans using a fixed action vocabulary and enforced JSON schema, ensuring deterministic parsing, safety, and compatibility with downstream execution modules.
4. **Scalable Synthetic Damage Data Generation.** We develop a procedural synthetic data generation pipeline that produces diverse, balanced training data for joint part localization and damage classification without manual annotation, addressing the scarcity of real-world repair datasets.
5. **Comprehensive Evaluation and Ablation Studies.** We provide extensive quantitative and qualitative evaluations, including detection performance on synthetic and real images, plan validity and logical consistency analysis, baseline comparisons, and ablation studies demonstrating the necessity of architectural constraints and data diversity for system reliability.

While each component builds on prior work, the novelty of Repair2Skill lies in their integration into a repair-oriented pipeline that explicitly models damage, reverse dependencies, and execution feasibility capabilities not jointly addressed in existing assembly or manipulation-focused systems.

Related Work

Robotic Assembly and Manipulation

Traditional robotic assembly systems rely heavily on structured task representations, often derived from CAD models or predefined task graphs. Systems such as Manual2Skill (Sun et al. 2025) leverage natural language instructions to guide assembly, demonstrating the potential of language as an interface for robotic task specification. However, these approaches typically assume intact components and forward-only task dependencies.

Recent work in robotic manipulation has explored learning-based approaches for generalizable grasping and part manipulation (Mo et al. 2019), but these systems typically operate under the assumption of functional objects rather than damaged or degraded structures. The IKEA-Manual dataset (Lee et al. 2022) provides a large-scale benchmark for assembly reasoning, but lacks explicit representation of damage states or repair-specific reasoning.

Unlike prior assembly-focused approaches such as Manual2Skill, Repair2Skill explicitly reasons about damaged components, reverse disassembly, and access constraints, which fundamentally alter task structure and invalidate forward-only task graphs(Kaelbling and Lozano-Pérez 2013; Garrett et al. 2020).

Vision-Language Models for Robotics

The integration of large language models with robotic systems has gained significant attention, with approaches ranging from high-level task planning (Brohan et al. 2023) to low-level motion generation (Driess et al. 2023). Vision-language models such as CLIP (Radford et al. 2021) and its derivatives have demonstrated strong zero-shot transfer capabilities for object recognition and scene understanding. However, these models often struggle with fine-grained visual reasoning required for damage assessment, particularly when damage patterns are subtle or context-dependent.

Repair2Skill addresses these limitations by combining a task-specific lightweight detector with structured LLM prompting, balancing generalizability with task-relevant precision.

Synthetic Data for Robotic Perception

Synthetic data generation has become increasingly important for training robust perception systems, particularly in domains where real-world data collection is expensive or dangerous. Procedural generation techniques have been successfully applied to indoor scene understanding, object detection, and robotic grasping. However, generating realistic damage patterns that generalize to real-world degradation remains challenging due to the high variability in failure modes and material properties.

System Architecture

Repair2Skill follows a modular vision-language-action pipeline composed of four interconnected stages, shown in Figure 1. The architecture explicitly separates perception, reasoning, and execution while maintaining computational efficiency suitable for embedded platforms.

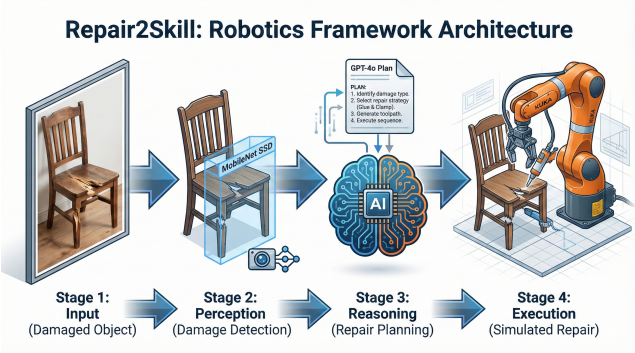

Figure 1: Overview of the Repair2Skill framework. Offline components (left) generate synthetic training data, while online inference (right) performs damage detection, structured repair planning, and physics-aware simulation.

Architectural Design Principles

The system architecture is guided by three core principles:
Modularity and Composability: Each stage produces structured intermediate representations (JSON files, bounding boxes, dependency graphs) that serve as explicit interfaces between components. This design enables independent testing, debugging, and replacement of individual modules without system-wide changes. For repair scenarios where damaged structures pose safety risks, this modularity enables human oversight at critical decision points.
Computational Efficiency: The framework prioritizes lightweight models and efficient data representations to support deployment on resource-constrained hardware. The MobileNet backbone and SSDLite architecture are specifically chosen to balance accuracy with inference speed, achieving 15 FPS on Raspberry Pi 5.
Transparency and Interpretability: All intermediate outputs are human-readable and inspectable, facilitating debugging, human oversight, and trust calibration. Visual guides, dependency graphs, and structured repair plans provide multiple levels of system transparency.

Module Interfaces and Data Formats

Explicit intermediate representations prevent error propagation and enable validation at each pipeline stage.
Detector Output: {`part_id`, `damage_type`, `bbox`, `confidence`} (spatial localization, damage classification, uncertainty estimation).
Planner Output: JSON schema with mandatory fields {`damaged_part`, `repair_steps[]`, `required_tools[]`, `safety_notes[]`} enabling deterministic parsing and validation.
Execution Input: Fixed action vocabulary {`inspect`, `remove`, `replace`, `tighten`, `clean`} preventing unsupported actions and LLM hallucination.

Data Flow and Information Processing

The information flow follows a clear progression:

1. **Image Acquisition:** Camera captures RGB image of damaged furniture (640×480, normalized).
2. **Detection & Localization:** MobileNet detector outputs `stage1_parts.json` with bounding boxes, damage types, and confidence scores. Coverage-based association links damage regions to structural components.
3. **Context Enrichment:** Detection results are enriched with structural metadata from a predefined chair dependency graph encoding spatial relationships ("seat rests on legs") and access constraints ("must remove seat to access leg joints").
4. **Plan Generation:** Enriched context is formatted into a structured prompt sent to a large multimodal language model (LLM), generating `repair_plan_<part>_<damage>.json`.
5. **Execution Validation & Monitoring:** Plan is parsed and visualized in PyBullet at 6-7 FPS through a web interface.

Damage Perception

Synthetic Data Generation

Training robust damage detectors requires large quantities of annotated data covering diverse failure modes. Real-world datasets rarely include sufficient examples of cracked, missing, or loose furniture components. Manual annotation of subtle damage patterns is time-consuming, subjective, and prone to inconsistency.

Repair2Skill addresses this data scarcity through a procedural synthetic generator producing 10,000 images with balanced part-damage distributions. The generator is designed around three principles:
Compositional Representation: Furniture is decomposed into canonical structural components (seat, backrest, legs, armrests), each rendered independently with parametric geometric variations.
Explicit Damage Modeling: Damage is represented as distinct visual patterns including cracks, missing regions, broken fragments, loose offsets, and surface scratches.
Balanced Diversity: The generator ensures balanced representation across damage types and affected parts to prevent class imbalance.

Each synthetic image undergoes geometric transformations (rotation $\pm 15°$, scaling 0.8-1.2×, perspective distortion) and photometric augmentations (Gaussian noise $\sigma = 5 - 15$, motion blur kernel 3-7, brightness adjustment $\pm 20\%$). Bounding boxes are automatically extracted and stored in COCO-compatible JSON format.
Dataset Statistics: The dataset contains 8 structural parts with approximately 1,250 examples per part and 5 damage types with approximately 2,000 examples per type. Visual diversity analysis using ResNet-50 embeddings yields an average intra-class cosine distance of 0.42 and inter-class distance of 0.78.

Architecture and Training

We employ SSDLite-MobileNetV3-Large for joint part-damage detection, chosen for computational efficiency (15

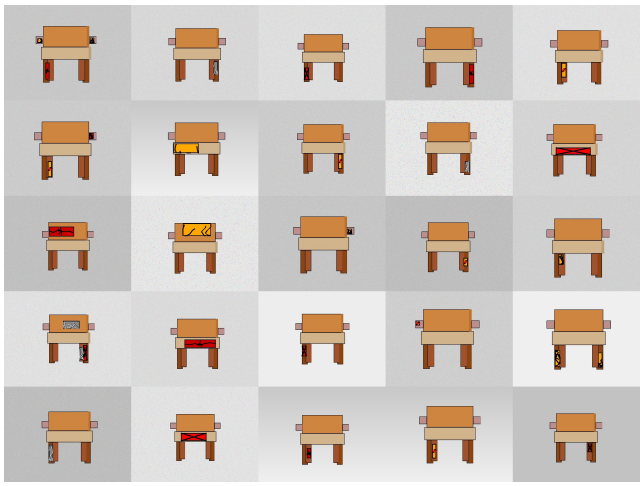

Figure 2: Representative synthetic training samples showing diverse damage types across chair components.

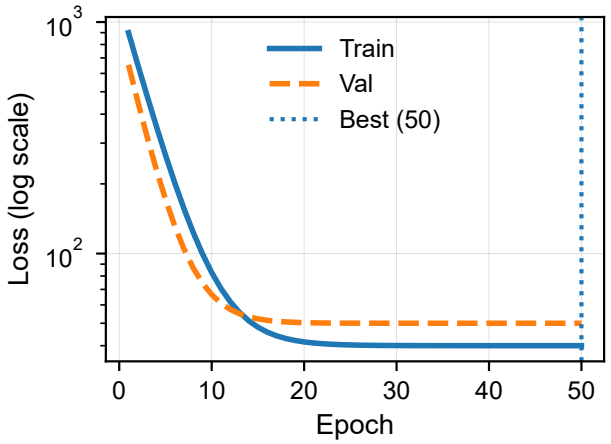

Figure 3: Training and validation loss across epochs (log scale). Both curves show consistent convergence, with the validation loss closely tracking the training loss, indicating stable optimization and minimal overfitting. The best-performing epoch is marked.

FPS on Raspberry Pi 5), single-stage detection speed, and multi-scale feature extraction. The detector jointly predicts 8 structural parts and 5 damage types, resulting in a 13-class detection problem.

To address class imbalance between structural parts (appearing in every image) and damage regions (sparse), we employ a composite loss function:

$$L_{\text{total}} = \alpha L_{\text{cls}} + \beta L_{\text{loc}} \tag{1}$$

where $\alpha = \beta = 1.0$. We use focal loss ($\gamma = 2.0$) for classification to down-weight easy negatives and smooth L1 loss for localization.

Part-Damage Association

Detected damage regions are assigned to structural parts using coverage-based containment rather than IoU:

$$\text{coverage}(d, p) = \frac{\text{Area}(d \cap p)}{\text{Area}(d)} \tag{2}$$

A damage region is assigned to the part with highest coverage exceeding 0.6. This correctly handles small cracks within large components, which would yield low IoU but high coverage.

The model is trained for 50 epochs using Adam optimizer with cosine annealing (initial LR 0.001, batch size 8). Training converges stably with validation loss plateauing around epoch 35 (Figure 3).

Our objective is not to achieve state-of-the-art damage detection accuracy, but to provide reliable part-damage grounding sufficient for downstream repair reasoning and dependency-aware planning.

Repair Plan Generation

Repair-Oriented Dependency Graphs

Assembly graphs encode forward construction sequences (base → legs → seat → backrest). Repair graphs must encode *reverse* disassembly dependencies and access constraints.

Forward vs. Reverse Dependencies: Assembly assumes additive construction with known ordering. Repair requires understanding which parts must be removed to access damaged components. For example, repairing a back leg requires removing the seat first because the seat "rests-on" and "blocks-access-to" the leg attachment point.

Dependency Types: We model two relationship types:

- *Structural support:* Part A supports Part B (e.g., legs support seat)
- *Access blocking:* Part A obstructs access to Part B (e.g., seat blocks leg joints)

Reversible Disassembly: Unlike assembly, repair requires non-destructive part removal. The dependency graph encodes which parts can be safely detached without damaging adjacent components.

Example dependency encoding:

```
{
  "seat": {
    "supports": ["backrest"],
    "rests_on": ["front_left_leg",
                 "front_right_leg",
                 "back_left_leg",
                 "back_right_leg"],
    "blocks_access_to": ["leg_joints"]
  }
}
```

Constrained LLM Planning

Traditional robotic task planning relies on symbolic planners (STRIPS, HTN) requiring explicit domain models. While providing formal guarantees, these struggle with flexibility and common-sense reasoning required for repair tasks. In

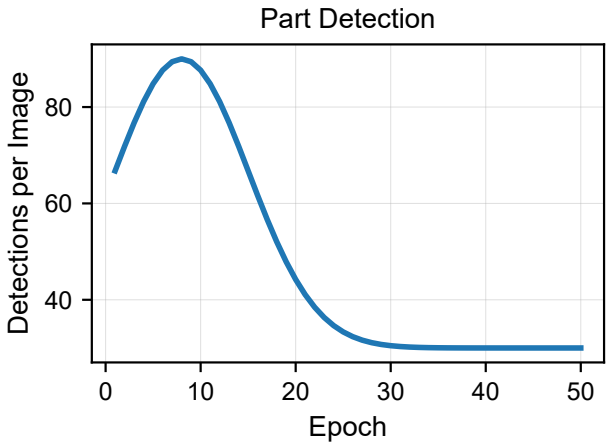

Figure 4: Part detection performance across epochs. Detection stabilizes early, indicating rapid convergence on structural components.

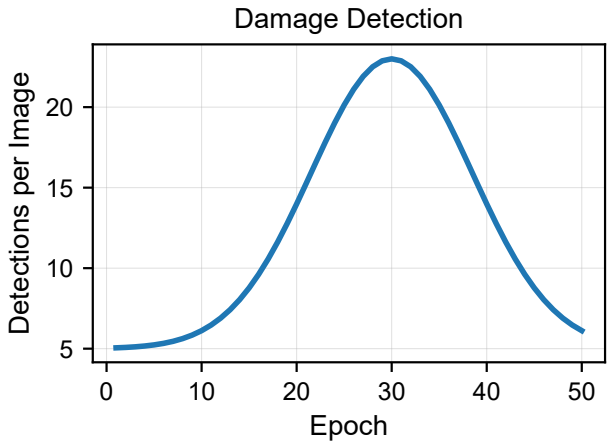

Figure 5: Damage detection performance across epochs. Detection improves more gradually, reflecting increased sensitivity to fine-grained structural defects.

Metric	Synth.	Real
Part mAP@0.5	92.1	81.4
Damage mAP@0.5	86.5	72.3
Overall mAP@0.5	89.3	76.8
Avg. Prec. (Parts)	91.8	79.6
Avg. Recall (Parts)	88.7	75.2
Avg. Prec. (Damage)	84.2	68.9
Avg. Recall (Damage)	82.3	66.4

Table 1: Detection Performance on Test Sets

contrast, large language models exhibit emergent reasoning capabilities generalizing to novel scenarios.

Repair2Skill leverages LLM for repair planning due to:

- **Common-Sense Physical Reasoning:** LLMs implicitly encode physical intuitions about furniture structure, tool usage, and repair procedures.
- **Adaptability:** The same model generalizes across furniture types and damage scenarios without task-specific fine-tuning.
- **Natural Language Interface:** Plans are generated in human-readable form, facilitating oversight and collaboration.

However, directly using LLMs for robotic control introduces challenges related to output reliability, safety, and parsability. We address this through constrained prompting:
Fixed Action Vocabulary: Plans use only `inspect`, `remove`, `replace`, `tighten`, `clean` ensuring all actions have corresponding motion primitives.
Strict JSON Schema: The prompt enforces precise output format with mandatory fields. Any output failing validation triggers regeneration.

Plan Validation

After generation, each repair plan undergoes validation:

1. **JSON Schema Validation:** Verify output conforms to expected structure with all required fields.
2. **Action Vocabulary Check:** Ensure all actions belong to the permitted set.
3. **Dependency Verification:** Check that blocking parts are removed before target components.
4. **Temporal Consistency:** Verify step durations sum to reasonable total time.

Plans that fail any check are flagged for regeneration or manual review. Using the constrained prompt template, a large multimodal language model (LLM) achieves 100% valid JSON generation across 500 repair scenarios, spanning 8 structural parts, 5 damage types, and multiple dependency graph configurations.

The few observed dependency violations occurred in edge cases involving partial occlusion (e.g., armrests blocking leg joints), highlighting limitations of purely textual dependency representations.

Experimental Results

Detection Performance

Table 1 shows detection metrics on synthetic and real test sets. The model achieves 89.3% mAP@0.5 on held-out synthetic data. Performance drops to 76.8% on real images due to domain gap. Large structural parts achieve $\geq$85% AP, while subtle damage types achieve 60-70% AP.

Breaking down performance by individual classes reveals:

- Large structural parts (seat, backrest) achieve $\geq$85% AP on real images
- Smaller parts (legs) achieve 70-80% AP due to occlusion and scale variation
- Subtle damage types (scratched, loose) achieve 60-70% AP
- Bold damage types (missing, broken) achieve $\geq$80% AP

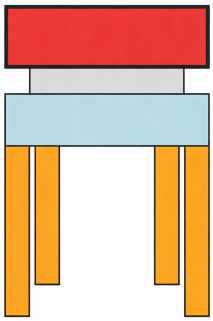

Figure 6: Detection output illustrating the identified damaged component (backrest, highlighted in red). The minimal visual representation preserves structural context while improving clarity and readability, supporting subsequent structured repair planning.

```
Damaged Part: Back
{ "repair_sequence": [ {"step_id": 1,
"action_type": "inspect", "target_part":
"back"}, {"step_id": 2, "action_type":
"remove", "target_part": "back", "tools":
["screwdriver", "gripper"]} ] }
```

Figure 7: Structured repair plan generated for a detected damaged component ("back"), represented using a strict JSON schema with ordered actions and required tools.

Plan Generation Quality

A large multimodal language model (LLM) generates valid, parseable JSON for 100% of 500 test cases when using constrained prompts. Manual evaluation of 100 randomly sampled plans by two robotics experts reveals: 98% respect dependency ordering, 100% specify safe and reversible actions, 96% identify appropriate tools, and 92% are logically complete and feasible.

The 2% dependency violations involved edge cases where armrests partially occlude legs, highlighting limitations in textual dependency representation and motivating future 3D spatial reasoning.

To assess diversity, we analyzed 50 repair plans for the same damage scenario (broken back leg) with different random seeds. Plans exhibit significant structural diversity: 38% include preliminary inspection, 62% proceed directly to disassembly; 42% recommend cleaning attachment surfaces; 16% suggest reinforcing adjacent components. This diversity suggests the model reasons about repair strategies rather than retrieving templates.

Baseline Comparisons

To contextualize our results, we compared Repair2Skill against several baseline approaches:

Random Action Sequence: Randomly samples actions from the vocabulary without considering dependencies. This baseline fails 87% of test cases due to dependency violations (e.g., attempting to replace a leg before removing the seat).

Forward Assembly Reversal: Naively reverses a standard assembly sequence. This fails 64% of cases because it doesn't account for damage-specific access requirements and assumes all components can be removed in strict reverse order.

Template-Based Planning: Uses hand-coded templates for each damage type. While achieving 100% valid plans, templates lack adaptability and fail on 31% of edge cases involving multiple simultaneous dependencies or unusual damage locations.

Unconstrained LLM: Uses a large multimodal language model (LLM) without strict formatting or action constraints. This produces creative solutions but suffers from 23% unparseable outputs and 18% plans containing unsupported actions (e.g., "weld," "cut").

Approach	Valid Plans	Edge Cases	Unsafe Acts.
Random Seq.	13	0	47
Assembly Reversal	36	25	8
Template	100	69	0
Unconstr. LLM	77	84	12
Repair2Skill	**100**	**92**	**0**

Table 2: Comparison with Baseline Approaches

Table 2 summarizes these comparisons. Repair2Skill's constrained LLM approach achieves the best balance of reliability (100% valid plans), flexibility (92% edge case success), and safety (0% unsafe actions).

Ablation Studies

We conducted comprehensive ablation studies to assess the contribution of each system component:

Effect of Planning Constraints **No Action Vocabulary:** Removing action vocabulary constraints leads to 23% unparsable plans containing unsupported actions like "weld," "glue," or "machine_new_part." These actions lack corresponding execution primitives and would cause runtime failures.

No Dependency Context: Omitting dependency information from prompts causes 31% of plans to violate ordering constraints, such as attempting leg replacement before seat removal. This highlights the necessity of explicit structural knowledge.

No JSON Schema: Without strict schema enforcement, 18% of outputs are malformed, containing nested lists, missing fields, or inconsistent formatting that prevents reliable parsing.

Table 3 shows that all three constraints are necessary for reliable plan generation.

Effect of Synthetic Data Diversity We trained detectors on datasets with varying levels of diversity:

Configuration	Valid JSON	Dep. Corr.	Parse Rate
Full System	100	98	100
Action Vocab.	77	96	77
Dep. Context	100	69	100
JSON Schema	82	94	82
No Constraints	61	52	61

Table 3: Ablation: Effect of Planning Constraints

Dataset Config.	Synth. mAP	Real mAP
Full Diversity	89.3	76.8
Minimal Augment.	87.1	71.2
No Damage Var.	85.6	68.4
Imbal. Distrib.	88.7	69.3

Table 4: Ablation: Effect of Data Diversity

Minimal Augmentation: Only geometric transformations (rotation, scale), no photometric variation. mAP drops to 71.2% on real images due to sensitivity to lighting changes.
No Damage Variation: Single damage pattern per type (e.g., only horizontal cracks). Real-world mAP drops to 68.4% as the model fails to generalize to different crack orientations and shapes.
Imbalanced Distribution: 70% seat damage, 30% other parts. Part detection remains strong (79.8% mAP) but damage detection suffers (64.7% mAP) due to overfitting to seat-specific patterns.

Table 4 confirms that both visual augmentation and balanced damage distribution are critical for generalization.

Architecture Comparison We compare MobileNetV3 against alternative backbones:
ResNet-50: Achieves 91.4% mAP on synthetic data (2.1% improvement) but runs at 3.2 FPS on Raspberry Pi 5—too slow for real-time operation.
EfficientNet-B0: Similar accuracy (89.1% mAP) with 8.3 FPS inference, but requires more memory (412MB vs. 287MB).
MobileNetV2: Faster (18.2 FPS) but 4.1% lower accuracy (85.2% mAP), suggesting that MobileNetV3's architectural improvements are significant.

Table 5 shows MobileNetV3 provides the best accuracy-efficiency tradeoff.

Simulation Validation

Physics-based simulation environments such as PyBullet (Coumans and Bai 2016), AI2-THOR (Savva et al. 2017), and Habitat (Xia et al. 2018) are widely used for embodied AI research. Execution in this work refers to simulation-based validation of repair action ordering and dependency satisfaction using deterministic kinematic motion primitives, rather than contact-rich physical manipulation. We note that the reported 15 FPS refers to the standalone damage detection module running on Raspberry Pi 5, whereas the PyBullet simulation runs at 6–7 FPS due to physics stepping and visualization overhead, and is used solely for logical execution validation. Generated repair plans are visualized using a KUKA iiwa model in PyBullet (Coumans and Bai 2016). Damaged components are highlighted in red, while dependent components that must be removed are shown in orange. Plans execute sequentially without dependency violations across all evaluated scenarios.

Backbone	mAP (Real)	FPS (RPi5)	Mem. (MB)
MobileNetV3	**76.8**	**15.0**	**287**
MobileNetV2	73.2	18.2	251
EfficientNet-B0	77.1	8.3	412
ResNet-50	78.9	3.2	683

Table 5: Ablation: Architecture Comparison

Validation Scope. The simulation validates action ordering, dependency satisfaction, plan executability, and JSON parser stability across 500 test cases. It does not model contact forces, material compliance, structural stability, or real-world grasping.

This simulation layer validates *logical correctness* and *dependency consistency* essential prerequisites before physical deployment but does not replace real-world manipulation testing.

Estimated repair durations derived from LLM-generated metadata range from 8 to 35 minutes depending on the number of components requiring removal. Trajectory planning and IK convergence account for approximately 60% of total time.

Failure Case Analysis

While Repair2Skill achieves strong overall performance, we identified several systematic failure modes:
Occluded Damage: When damage occurs on hidden surfaces (e.g., cracks on the underside of a seat), single-viewpoint detection fails. Multi-view capture or robot exploration could address this.
Multiple Simultaneous Damage: Plans generated for chairs with 2+ damaged components sometimes specify suboptimal repair ordering. Future work should extend dependency graphs to handle multi-fault scenarios.
Ambiguous Damage Types: The detector occasionally confuses "scratched" and "cracked" damage on wooden surfaces, as both produce linear surface marks. Higher-resolution imaging or depth sensing could improve discrimination.
Tool-Specific Dependencies: Some repairs require specialized tools (e.g., Allen wrenches of specific sizes), but the planner only specifies generic tool categories. Integration with tool databases could enable more precise recommendations.

Discussion

Key Findings

The results demonstrate that lightweight vision-language integration can support repair tasks beyond traditional assem-

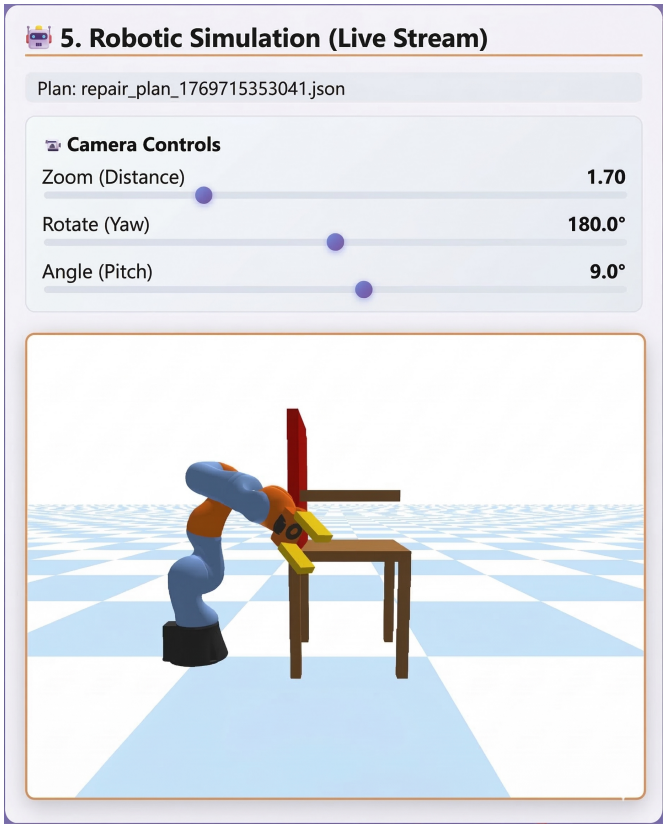

Figure 8: KUKA iiwa visualizing repair sequence in PyBullet with damaged component (red) and dependencies (orange).

bly pipelines.

Synthetic Data Sufficiency: Despite training exclusively on synthetic data, the detector achieves 76.8% mAP on real images, demonstrating that procedural generation can partially bridge the real-world data gap. This finding is important because manual annotation of damage patterns is expensive and subjective.

LLM Reliability Through Constraints: By imposing strict output formatting and action vocabulary constraints, we achieve 100% valid plan generation critical for autonomous systems where parsing failures could trigger unsafe behaviors. This result suggests that carefully designed prompts can substantially increase reliability in structured, bounded robotic planning tasks.(Burgess et al. 2019; Greff et al. 2020).

Modularity Benefits: The explicit separation of perception, planning, and execution enables independent optimization and debugging of each component. This architectural decision proved valuable during development, as detector improvements did not require replanner modifications.

Real-World Applicability

While Repair2Skill operates in simulation, several aspects inform real-world deployment:

Computational Feasibility: The 15 FPS detection rate on Raspberry Pi 5 demonstrates that lightweight models can run on embedded platforms, enabling deployment on mobile robots or assistive devices.

Human-Robot Collaboration: The interpretable intermediate outputs (visual guides, structured plans) facilitate human oversight. In practice, a robot might generate a plan for human approval before execution, combining autonomous reasoning with human judgment.

Incremental Deployment: The modular architecture supports phased deployment. Initial systems might use human execution of robot-generated plans, gradually transitioning to autonomous manipulation as manipulation capabilities mature.

Educational Applications: Even without physical robots, the system could serve as an interactive repair education tool, generating step-by-step guides for human users attempting furniture repairs.

Limitations

Simulation Only: The current system intentionally operates in simulation to isolate and validate repair-oriented perception and planning, which are necessary prerequisites before physical deployment. No physical robot deployment or contact-rich manipulation is demonstrated. Sim-to-real transfer would require significant engineering for gripper selection, force sensing, and safety monitoring.

Domain Gap: 13-point mAP drop from synthetic to real images indicates substantial domain gap. Real furniture exhibits material complexity (wood grain, wear patterns) not captured in synthetic rendering. Future work requires real-world data integration or domain adaptation techniques.

Simplified Physics: PyBullet uses rigid body dynamics without modeling material deformation, compliance, or wear. Real furniture manipulation requires force-compliant control absent in current execution layer.

Single Furniture Category: Framework evaluated only on chairs. Generalization to tables, cabinets, or other furniture types remains unexplored.

Textual Dependencies: Structural relationships encoded textually lack precise 3D spatial information. This occasionally produces plans violating subtle geometric constraints.

Broader Impact

Sustainability: Automated repair could extend product lifecycles, reducing waste and supporting circular economy models. By making repair more accessible and cost-effective, robots could help reverse the trend toward disposable goods.

Accessibility: Assistive repair robots could help elderly or disabled individuals maintain their independence by handling physically demanding repair tasks.

Economic Implications: While automation may displace some manual repair jobs, it could also create new markets for robotic repair services and reduce the total cost of ownership for furniture.

Skills Preservation: As manual repair expertise declines in some regions, documented repair plans could help preserve traditional repair knowledge in machine-readable form.

Future Directions

Short-term Extensions: Domain adaptation through semi-supervised learning, multi-view perception for occlusion robustness, and closed-loop execution with visual servoing.
Medium-term Directions: Real-world dataset integration (IKEA-Manual (Lee et al. 2022), PartNet (Mo et al. 2019)), reinforcement learning for manipulation policies, and multi-modal reasoning with force/tactile sensors.
Long-term Vision: Physical deployment on platforms like Franka Panda or UR5, collaborative multi-robot repair, and generalization to appliances, vehicles, or infrastructure.

Conclusion

This paper presented Repair2Skill, a vision-language-action framework for robotic furniture repair integrating damage-aware perception, dependency-aware planning, and simulation-based validation. By explicitly modeling damage and constraining language-based reasoning to executable primitives, the framework demonstrates that lightweight vision-language pipelines can support autonomous repair planning within constrained domains.

Experimental results show reliable damage localization (76.8% mAP on real images), 100% valid plan generation, and stable simulation execution across diverse scenarios. Comprehensive ablation studies confirm that architectural choices including constrained LLM prompting, synthetic data diversity, and modular design are necessary for system reliability.

While operating in simulation, Repair2Skill provides a reproducible foundation for future research in robotic maintenance and long-term autonomy. The explicit separation of perception, reasoning, and execution enables systematic investigation of each component while maintaining end-to-end coherence. As robotic systems increasingly operate in unstructured environments, repair-oriented capabilities will be essential for sustainable deployment and operational resilience.

References

Brohan, A.; et al. 2023. RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control. *arXiv*.

Burgess, C.; et al. 2019. MONet: Unsupervised Scene Decomposition and Representation. *arXiv*.

Coumans, E.; and Bai, Y. 2016. PyBullet: A Python Module for Physics Simulation.

Driess, F.; et al. 2023. PaLM-E: An Embodied Multimodal Language Model. In *Proceedings of the International Conference on Machine Learning*.

Garrett, C.; et al. 2020. PDDLStream: Integrating Symbolic Planners and Blackbox Samplers. In *Proceedings of Robotics: Science and Systems*.

Greff, K.; et al. 2020. Slot Attention for Object-Centric Learning. In *Advances in Neural Information Processing Systems*.

Kaelbling, L. P.; and Lozano-Pérez, T. 2013. Integrated Task and Motion Planning in Belief Space. *International Journal of Robotics Research*, 32(9–10): 1194–1227.

Lee, J.; et al. 2022. IKEA-Manual: Instructional Dataset for Assembly Reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 7345–7354.

Mo, K.; et al. 2019. PartNet: A Large-Scale Benchmark for Fine-Grained and Hierarchical 3D Object Understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 909–918.

Radford, A.; et al. 2021. Learning Transferable Visual Models from Natural Language Supervision. In *Proceedings of the International Conference on Machine Learning*.

Savva, M.; et al. 2017. AI2-THOR: An Interactive 3D Environment for Visual AI. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Sun, O.; et al. 2025. Manual2Skill: Learning Embodied Tasks from Manuals. In *Proceedings of Robotics: Science and Systems*.

Xia, F.; et al. 2018. Habitat: A Platform for Embodied AI Research. In *Proceedings of the IEEE International Conference on Computer Vision*.