BEN BENYAMIN

Chicago, IL, USA

872-946-5232 benbin52@gmail.com linkedin.com/in/Ben-Benyamin github.com/BenBenyamin benbenyamin.github.io

EDUCATION

Northwestern University, *Evanston, IL* - M.S. in Robotics **Sep 2024 - Dec 2025**
Tel Aviv University, *Tel Aviv, Israel* - B.S. in Mechanical Engineering **2016 - 2020**

SKILLS

Software Development: Python, C++, C, Git, Docker, Linux, Unit Testing, OpenMP, CMake, x86 Assembly

Robotics: ROS 2, LiDAR, YOLO, SLAM, MoveIt, Nav2, April Tags, RViz, PX4, OpenCV, Gazebo, Isaac Sim, Perception

Machine Learning: PyTorch, SAM 2, CLIP, Grounding DINO, VLM, FoundationPose, Segmentation, 6D Pose Estimation

Reinforcement Learning: Deep RL, Sim2Real, Gymnasium, PPO, Stable Baselines 3, Domain Randomization, Issac Lab

Mechanical and Electrical: SolidWorks, Milling, CNC, Sheet Metal, 3D Printing , PCBA , Arduino, FEA, PIC32 , Atmel Studio

Languages: English (Fluent), Mandarin Chinese (Proficient), Hebrew (Native)

EXPERIENCE

Loram Maintenance of the Way *Medina, Minnesota, USA* **July-September 2025**
Machine Learning / Artificial Intelligence Intern

- Trained an AI model in **Python** (**PyTorch**, **PyTorch Lightning**) for pixel wise segmentation of parts of railroad track.
- Built an automated data and training workflow in **Databricks** with **MLflow** for easy retraining on new data.
- Deployed the model with **NVIDIA Triton Inference Server** for fast GPU inference and edge deployment.

Automata – Advanced Automation Solutions *Hod Hasharon, Israel* **2022 – 2024**
Mechanical Engineer

- Designed and prototyped cost-efficient mechanical enclosures in **SolidWorks** for electronic automation systems.
- Supported integration of **PCB**s, sensors, and actuators into automated assemblies for industrial clients.

Automatica – Automation and Control Technologies Ltd. *Kfar Saba, Israel* **2020 – 2022**
Mechanical Engineer

- Designed automated factory machines in **SolidWorks** for the chemotherapy industry, including mechanical layouts.
- Collaborated with assemblers, electrical engineers, and controls engineers to ensure smooth integration.

PROJECTS

Robot Playing Real Life Arcade (*Python, PyTorch, Sim2Real, Stable Baselines 3, PPO, Domain Randomization*) **Apr 2025 - Dec 2025**

- Built a **Sim2Real Deep RL** setup where a physical **Hello Robot Stretch 3** plays Atari 2600, modeling real-world behavior.
- Modeled controller latency **stochastically** from measured timing distributions and discretized joint motion into frame-based actions, enabling a **Proximal Policy Optimization (PPO)** agent to reliably beat the game.
- The policy (**PPO**) found a "sweet spot" strategy where the policy learns a stable no-move winning position.

6D Pose Estimation and Object Grasping (*Python, PyTorch, Isaac Sim, SAM 2, MoveIt, ROS 2, Sim2Real*) **Dec 2024 - May 2025**

- Built an RGB-D pipeline for 3D grasping with point clouds, 6D pose estimation, and **MoveIt** motion planning.
- Implemented **PointNet** and **DenseFusion** in **Python**/**PyTorch** for RGB-D object classification and 6D pose estimation, robust to occlusions, unordered points, and noisy depth.
- Built an **Isaac Sim** pipeline generating labeled RGB-D data; trained on 16k+ samples to ∼70% test accuracy.
- Integrated **DenseFusion**, **Meta SAM 2**, and **ROS 2** motion planning into a unified RGB-D grasping stack with an Intel **RealSense D405** and Franka arm.

Autonomous Drone for LiDAR-Based SLAM (*C++, SLAM, PX4, Cartographer, LiDAR, ROS 2*) **Jan 2025 - Mar 2025**

- Built a drone to do **Simultaneous Localization and Mapping** using a 2D LiDAR of real life locations.
- Integrated **ROS 2** with **PX4** (Pixhawk 6C), RPLiDAR C1, and **Cartographer** for odometry and frame transforms.

Computer Graphics Renderer in C++(*C++, OpenMP*) **Jan 2025 – Mar 2025**

- Designed and implemented a full 3D rasterization pipeline in C++, built entirely without external graphics libraries.
- Used linear algebra and STL parsing modules to load, transform, and render complex triangle-mesh 3D models.
- Implemented projection, z-buffering, and per-triangle shading to simulate GPU-style rendering on the CPU, integrating **OpenMP** for concurrency.