

ROS-INDUSTRIAL DEVELOPERS MEETING (ASIA)

**Features in final release candidate for
easy_manipulation_deployment**

Glenn Tan

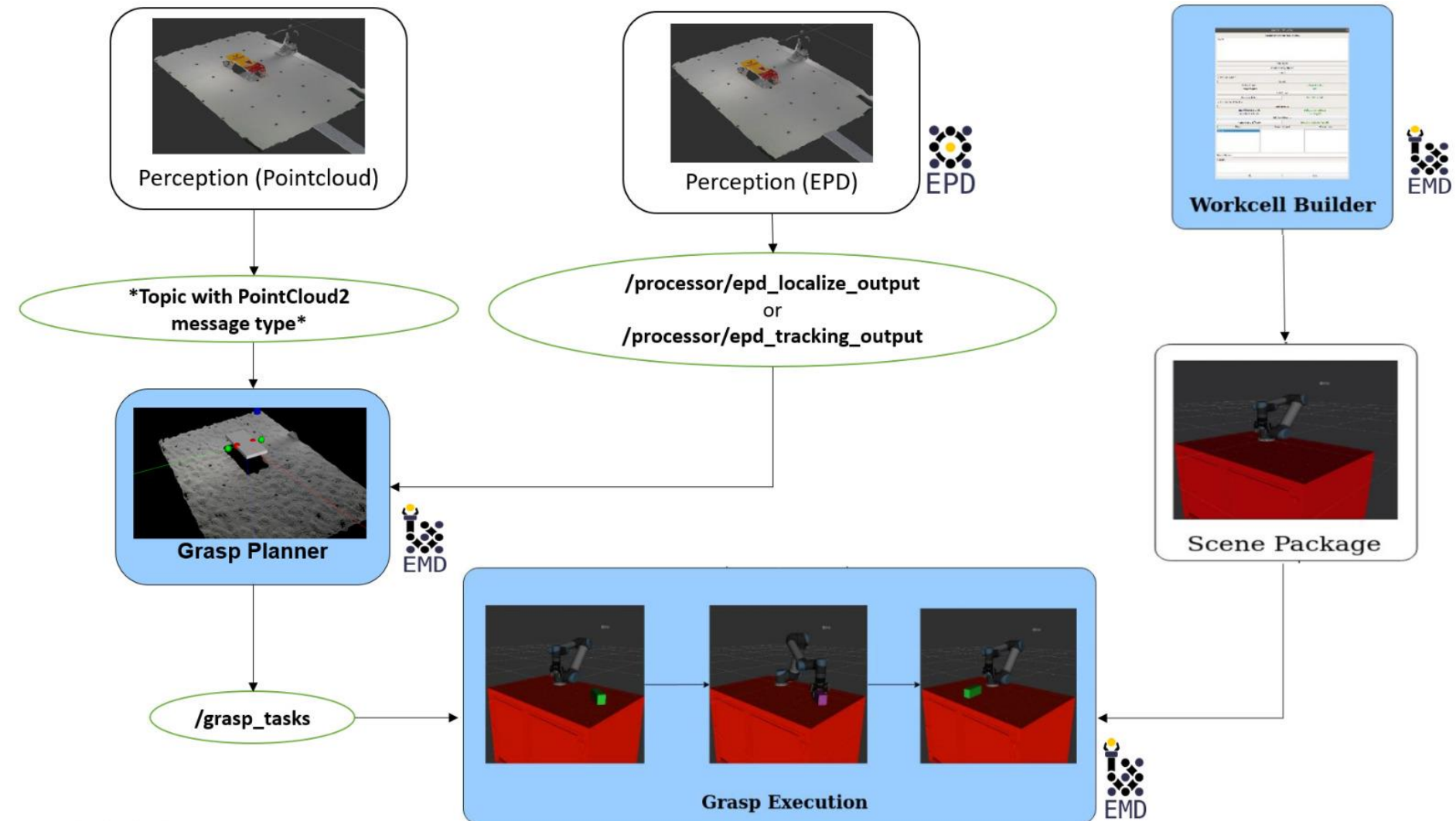
8th June 2021

easy_manipulation_deployment



An easy to use ROS2 manipulation package that uses the easy_perception_deployment output to provide a **modular** and **configurable** manipulation pipeline for pick and place tasks

Full Easy Manipulation Deployment Pipeline



Final Release Candidate

Easy Manipulation Deployment

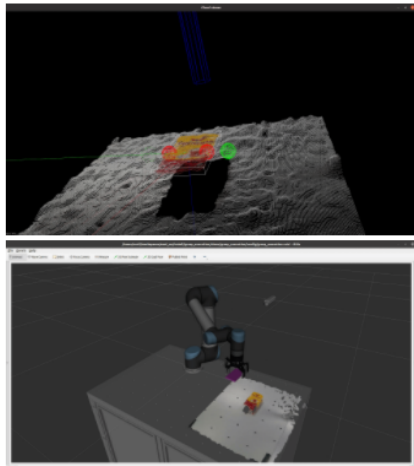


CI passing

license Apache-2.0

codecov 30%

This ROS2 package provides a modular and easy to deploy manipulation pipeline that integrates perception elements to establish an end-to-end pick and place task




This package was tested with the [easy_perception_deployment](https://github.com/ros-industrial/easy_perception_deployment) ROS2 package, but any other perception system that provides the same ROS2 message in the right topic can work with this package as well.

- Final Release Candidate has been released with all major new features added
 - Support for multifinger/suction array grasps
 - Support Dynamic Safety for grasp execution
 - Parameterization of entire pipeline

https://github.com/ros-industrial/easy_manipulation_deployment

Final Release Candidate Documentation



Easy_manipulation_deployment

[Main Page](#) [Related Pages](#) [Namespaces ▾](#) [Classes ▾](#) [Files ▾](#)

Easy Manipulation Deployment API

Welcome to the Doxygen page for the `easy_manipulation_deployment` suite

This package was tested with the `easy_perception_deployment` ROS2 package, but any other perception system that provides the same ROS2 message in the right topic can work with this package as well.

It is recommended to run this package on **ROS2 Foxy**.


Full Documentation/Wiki


[Check out the full ReadTheDocs documentation here](#)


Acknowledgements

We would like to acknowledge the Singapore government for their vision and support to start this ambitious research and development project, "Accelerating Open Source Technologies for Cross Domain Adoption through the Robot Operating System". The project is supported by Singapore National Robotics Programme (NRP).

Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of the NR2PO.

Generated by  1.8.17

 EMD



ABOUT

[Overview](#)

[Manipulation Pipeline](#)

[Frequently Asked Questions](#)

[Future Plans](#)

GETTING STARTED

[Common Concepts](#)

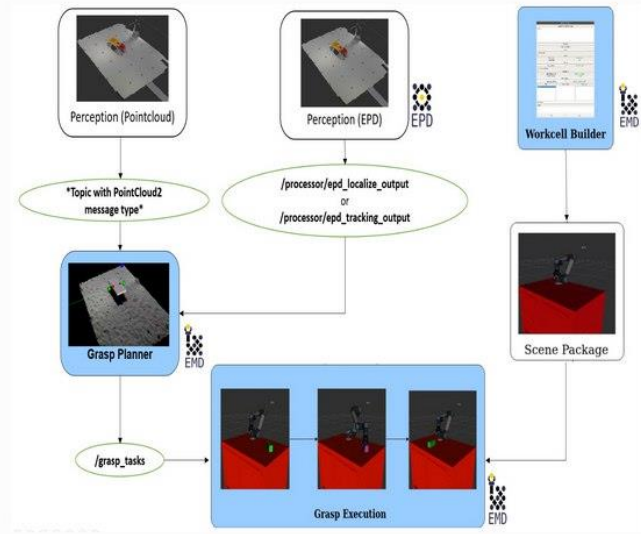
[Download Instructions](#)

[» Overview](#) [View page source](#)

Overview

Manipulation Pipeline

To preserve the modularity of this package, the manipulation pipeline can be broken down into three main aspects, each of which can function separately, or together as an end to end pipeline. Each component has its own documentation and tutorials which are linked in the headers.



```
graph TD
    subgraph Perception
        P1[Perception Pointcloud]
        P2[Perception EPD]
    end
    subgraph Processing
        P3["*Topic with PointCloud2 message type*"]
        P4["/processor/epd_localize_output or /processor/epd_tracking_output"]
    end
    subgraph Planning
        P5[Grasp Planner]
    end
    subgraph Execution
        P6[Grasp Execution]
    end
    subgraph Building
        P7[Workcell Builder]
    end
    subgraph Scene
        P8[Scene Package]
    end

    P1 --> P3
    P2 --> P4
    P3 --> P5
    P4 --> P5
    P5 --> P6
    P6 --> P7
    P7 --> P8
    P8 --> P6
```

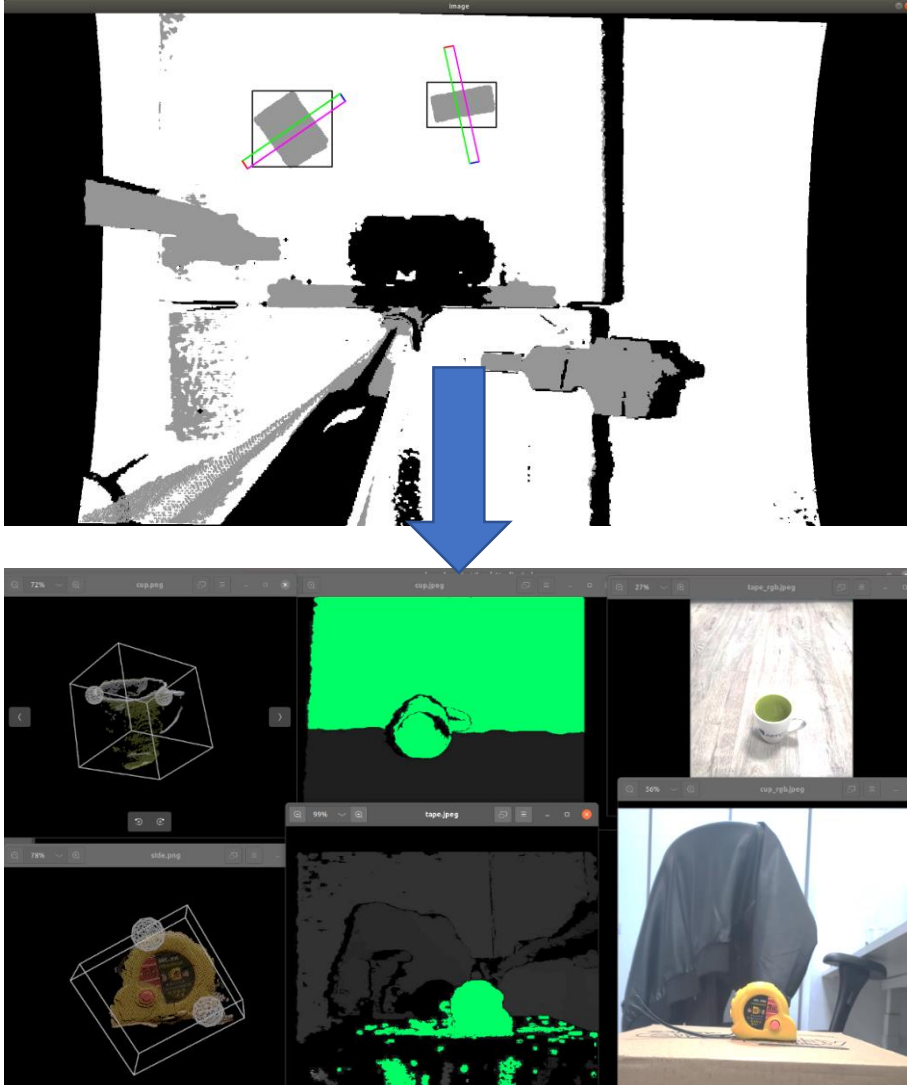
The diagram illustrates the Manipulation Pipeline. It starts with Perception (Pointcloud) and Perception (EPD). Perception (Pointcloud) leads to a topic with PointCloud2 message type, which then feeds into the Grasp Planner. Perception (EPD) leads to /processor/epd_localize_output or /processor/epd_tracking_output, which also feeds into the Grasp Planner. The Grasp Planner outputs /grasp_tasks to the Grasp Execution module. The Grasp Execution module feeds into the Workcell Builder, which then feeds into the Scene Package. The Scene Package also feeds into the Grasp Execution module.

Improved Grasp Planner





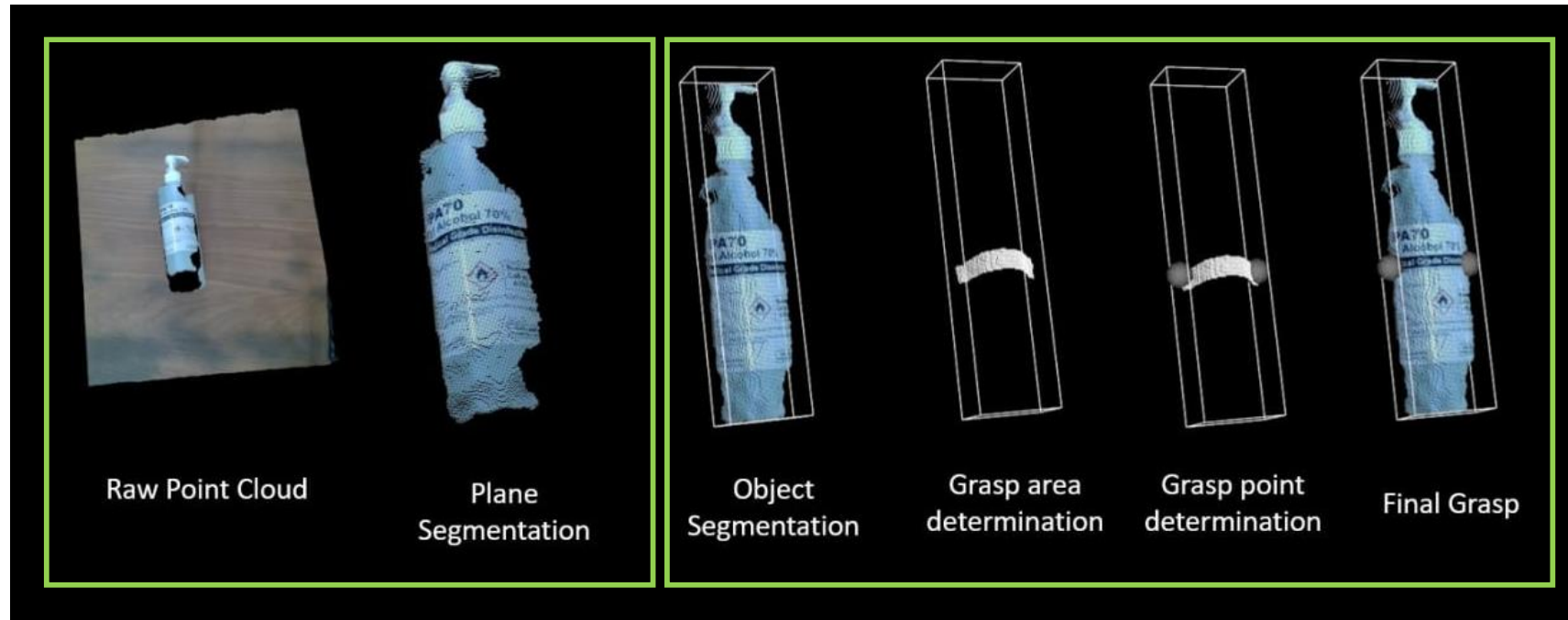
Grasp planner – Transition from 2D to 3D



- Provides a better representation of the grasp object, including surface characteristics
- Improves synergy with other existing useful libraries (Point Cloud Library, Flexible Collision Library)
- Grasp planning methodology has been improved from using Depth images to using 3D Point Clouds.



Improved Grasp Planner.





Improved grasp ranking support

$$\begin{aligned} rank(q_1, q_2) &= w1 \times r1(q_1, q_2) + w2 \times r2(q_1, q_2) \\ r1(q_1, q_2) &= \left(1.0 - dist(\gamma, q_1)\right) + \left(1.0 - dist(\gamma, q_2)\right) \\ &\quad - \left(\cos(\beta) - 0.2\right) \times 10.0 \\ r2(q_1, q_2) &= (1.0 - \lambda_{q_1}) + (1.0 - \lambda_{q_2}) + \cos(\alpha_{q_1}) \\ &\quad + \cos(\alpha_{q_2}) - ||\cos(\alpha_{q_1}) - \cos(\alpha_{q_2})|| \end{aligned}$$



$$Rank_{finger} = \omega_1 \times r_1 + \omega_2 \times r_2$$

$$r_1 = \left(\sum_{i=0}^{n_1} 1.0 - \gamma_i\right) + \left(\sum_{j=0}^{n_2} 1.0 - \gamma_j\right) - (\cos\beta - 0.2) \times 10.0$$

$$r_2 = \sum_{i=0}^{n_1} 1.0 - \lambda_i + \sum_{j=0}^{n_2} 1.0 - \lambda_j$$

$$Rank_{suction} = 2.0 - (\lambda \times \omega_\lambda) - (\gamma \times \omega_\gamma) + (c \times \omega_c)$$

- Initial 2 Finger Gripper ranking algorithm for **2 finger gripper** referenced from existing implementation*

- Algorithm was then **modified to support multi fingered grippers and suction arrays**
- With the use of Point Clouds, Grasp ranking algorithms have changed significantly to provide a **more comprehensive method of checking grasp stability** for both finger and suction grippers.

* Zapata-Impata, B. S., Gil, P., Pomares, J., & Torres, F. (2019). Fast geometry-based computation of grasping points on three-dimensional point clouds. International Journal of Advanced Robotic Systems, 16(1), 172988141983184. <https://doi.org/10.1177/1729881419831846>



Improved grasp ranking support

Finger collision avoidance

Distance from grasp to centroid of object

Curvature of surface at grasp points

Angle of grasp with respect to object axis

Suction contact points

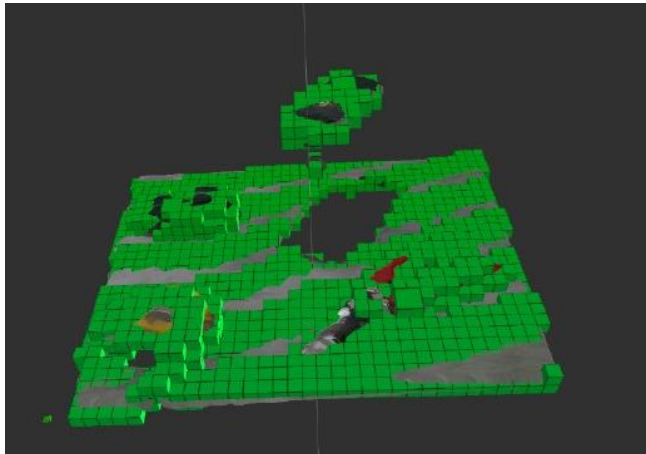
Distance from grasp to centroid of object

Curvature of surface at grasp point

- Additional parameters added into grasp ranking consideration (Highlighted in green)
- Generation of 3D grasp samples allows for more information to be derived from each sample
- Using of point clouds provides increased understanding of the grasp area and surface profiles



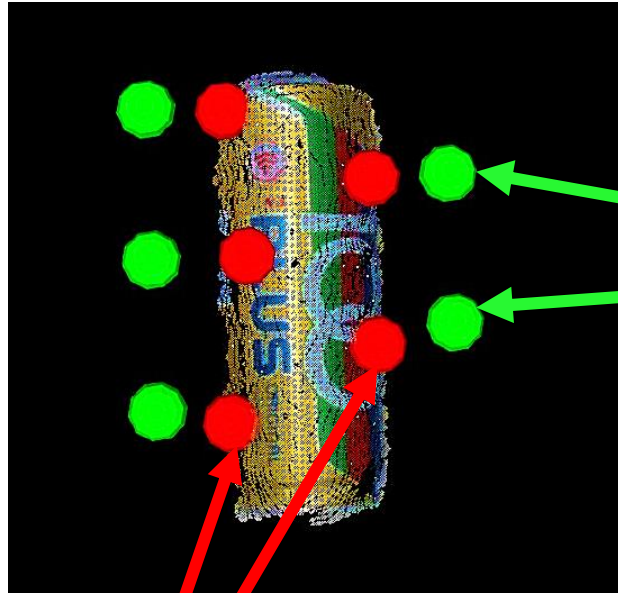
Improved Collision Detection using Flexible Collision Library



- Finger collision checks were initially based on checking depth values at the point of grasp and average values of the points representing the gripper
 - 2 Dimensional method does not provide extensive checks and is not truly reflective In a 3 dimensional environment
- Finger collision checks with point cloud now creates a 3D collision object representation of the end effector fingers and checks its collision with a 3D collision object representation of the environment, using the **Flexible Collision Library**.
 - Collision checks are now more robust and takes into account surrounding objects



Improved Finger Grasping Methodology



Open Configuration
Finger Collision
Checking with FCL



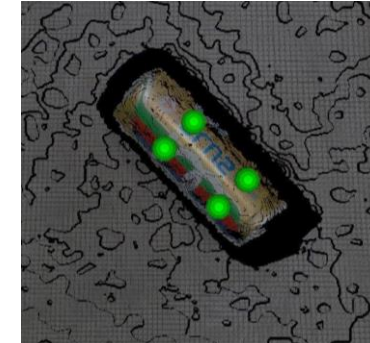
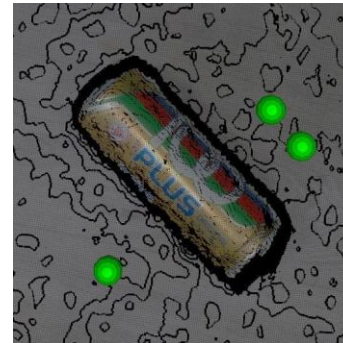
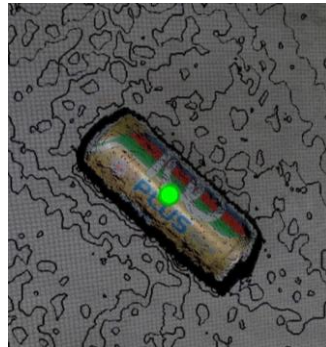
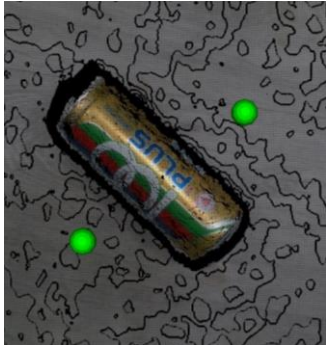
Closed Configuration Contact
point ranking



- Finger collision check is now not part the ranking system, but rather a first-pass criteria for each grasp option.
- Valid grasp candidates are now determined with a 2 step method. Closed and open configuration of the grasps
- Speeds up the grasp candidate generation by only ranking grasps that wont collide with the environment



Improved support for Multi-Finger/Suction array





Improved Grasp Planner – Improved Planning times

Number of Fingers	Point Cloud Grasp Planning [1] Time/ ms*	Depth image Grasp Planning [2] Time/ ms*	Suction array	Point Cloud Grasp Planning [1] Time/ ms*	Depth image Grasp Planning [2] Time/ ms*
2	11	420	1x1	195	250
3	90	-	2x1	216	-
4	96	-	2x2	272	-
5	292	-	3x3	416	-
8	309	-	4x4	658	-
10	663	-	5x5	874	-

[1] <https://doi.org/10.1177/1729881419831846>

[2] <https://arxiv.org/pdf/2001.05856.pdf>

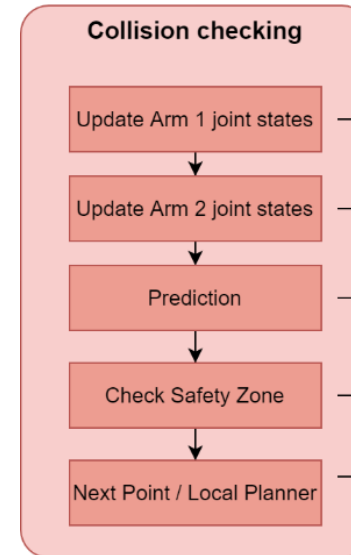
* Grasp planning times for a fixed selection of objects. Planning times may vary depending on object size

Improved Grasp Execution





Improved Grasp Execution – Dynamic Safety feature



Update Arm1 / Arm2 joint states

Update the robot joint transforms within the scene (planning_scene::PlanningScene) where all collisions are stored.

After the update, collision checker will use the latest poses from the scene.

Prediction

Based on the current joint states of the Arm2, predict where it will be after one collision checking cycle. The register both the current and the future collision to the scene (planning_scene::PlanningScene) before collision checking.

Check Safety Zone

Check Arm1 robot trajectory collision with the environment and Arm2. This will decide whether emergency stop, slow-down-stop-and-replan or dynamically replan.

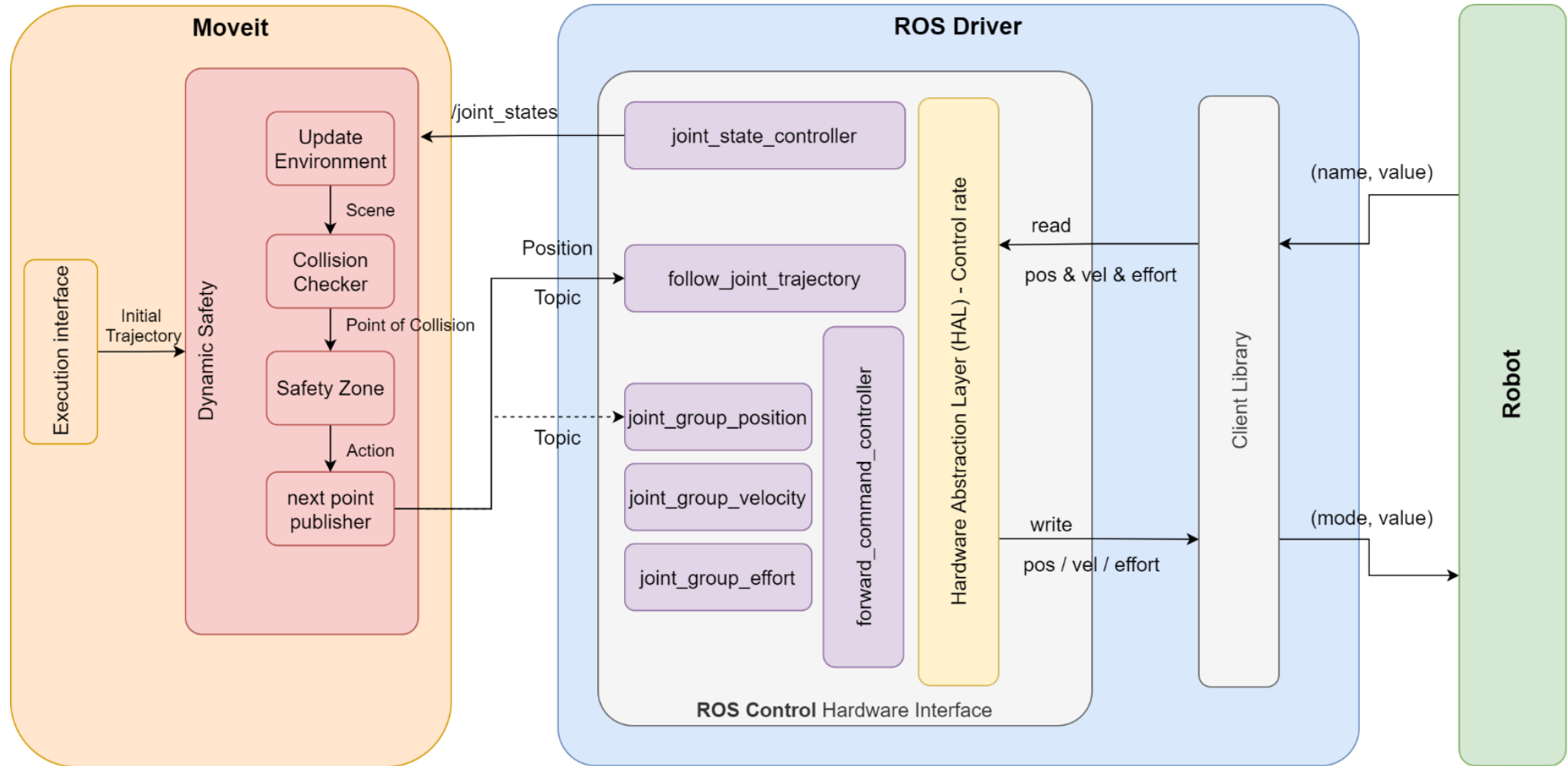
Next Point / Local Planner

Send the next command to the robot. It could be position or velocity command based on robot's hardware interface.

In the future, this part can also be a local planner with other algorithm to leverage different control interface like force control or more robust control interface.

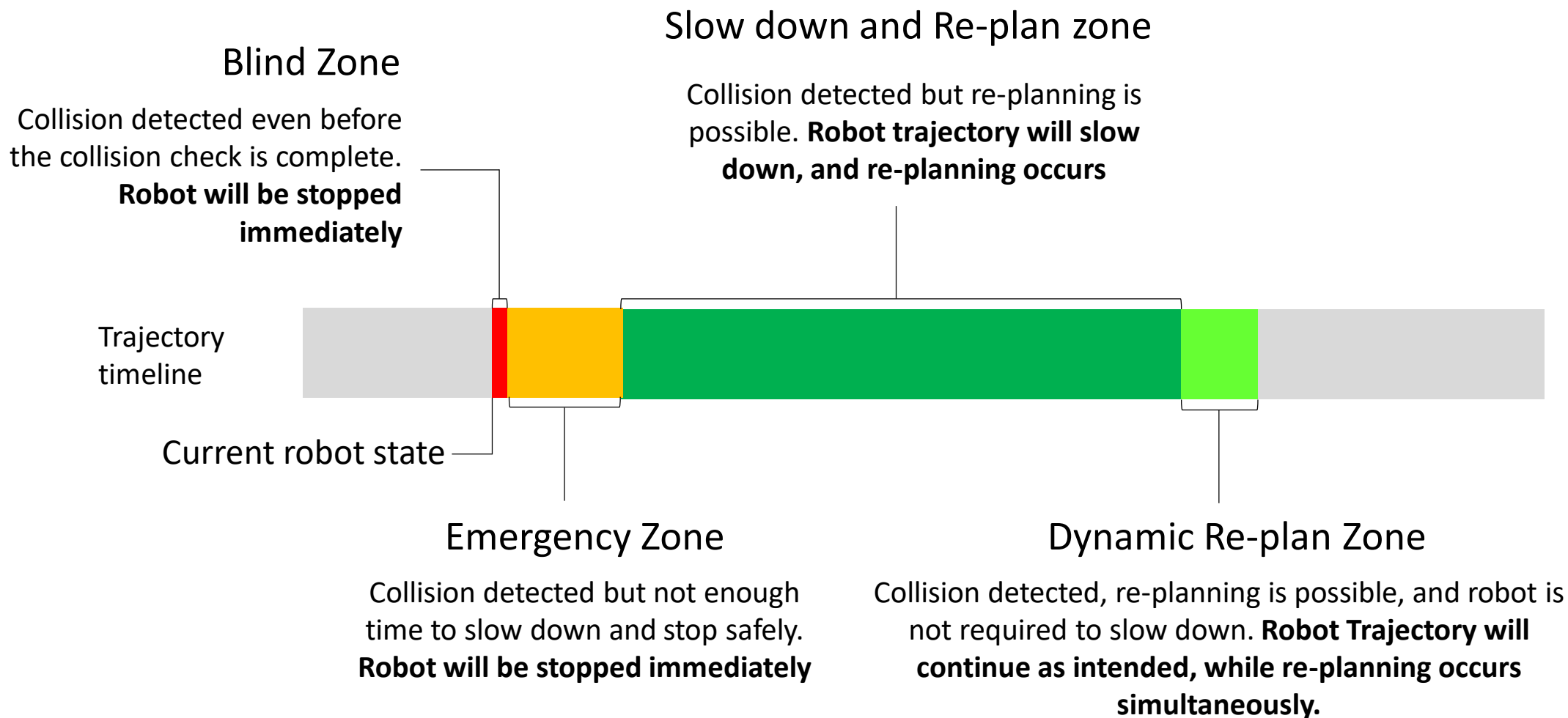


Improved Grasp Execution – Dynamic Safety Architecture



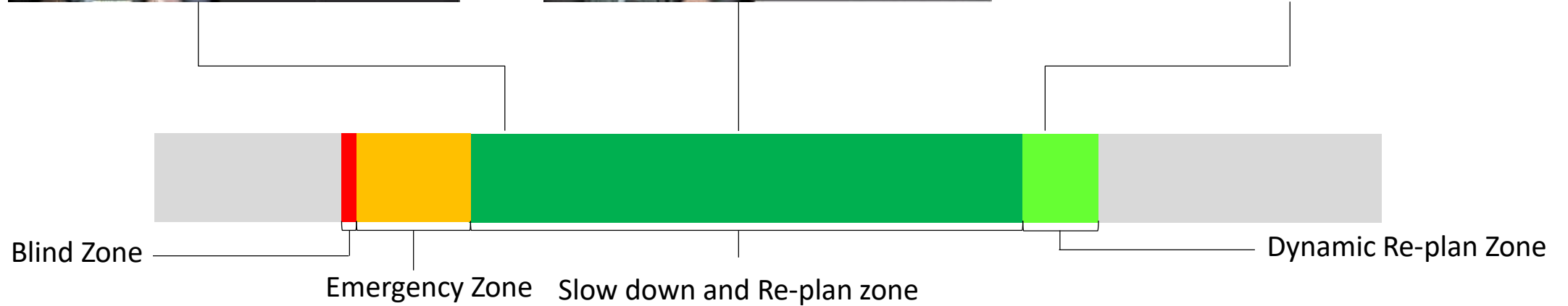
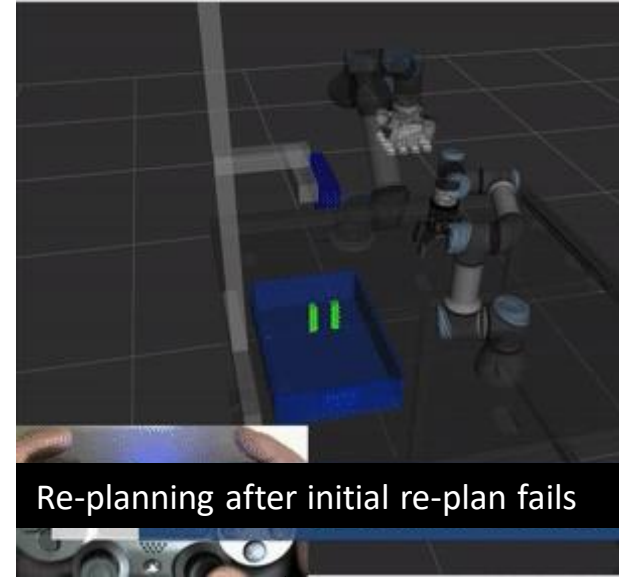


Dynamic Safety Zones





Improved Grasp Execution – Dynamic Zone feature





Increased Configurability for Grasp Execution

```
# Load octomap
load_octomap: true

# Dynamic safety parameters
rate: 20
allow_replan: true
visualize: true

safety_zone:
  manual: true
  unit_type: second
  collision_checking_deadline: 0.05
  slow_down_time: 0.2
  replan_deadline: 1.2
  look_ahead_time: 1.65

collision_checker:
  distance: false
  continuous: false
  step: 0.1
  thread_count: 8
  realtime: false

next_point_publisher:
  command_out_type: "trajectory_msgs/JointTrajectory"
  publish_joint_position: true
  publish_joint_velocity: false
  publish_joint_effort: false

replanner:
  planner_name: ompl

workcell:
  - group_name: manipulator
    executors:
      default:
        plugin: grasp_execution/DefaultExecutor
      ds_async:
        plugin: grasp_execution/DynamicSafetyAsyncExecutor
        controller: ur5_arm_controller
    end_effectors:
      robotiq_2f0:
        brand: robotiq_2f
        link: ee_palm
        clearance: 0.1
        driver:
          plugin: grasp_execution/DummyGripperDriver
          controller: ""
```

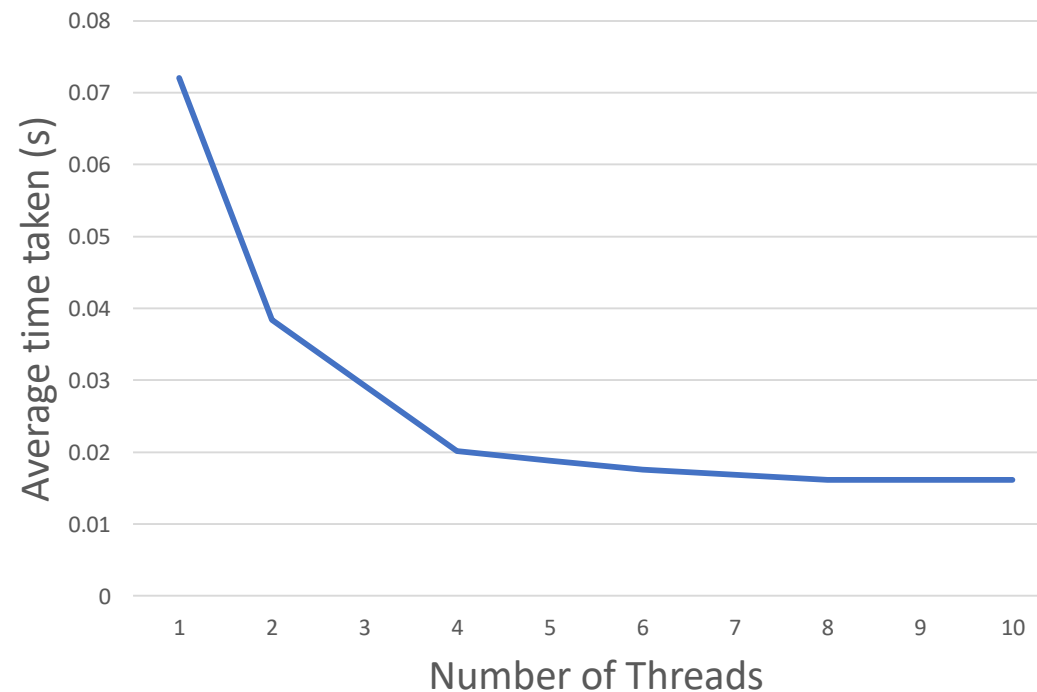
User defined parameters are stored in an easy to understand configuration file for increased customizability



Dynamic Safety Multithreading

- Check collision for 20 times (one loop of dynamic safety, average over 200 samples)
- 8 thread can reliably reach 50Hz
- **OS:** Ubuntu 20.04, **Kernel:** 5.8.0-48-generic,
- **Specs:** Ryzen 3900X 12-core 24 thread 3.8Hz,
GeForce RTX 2070 SUPER, 32GB RAM

Graph of Average Time Taken vs Thread Count

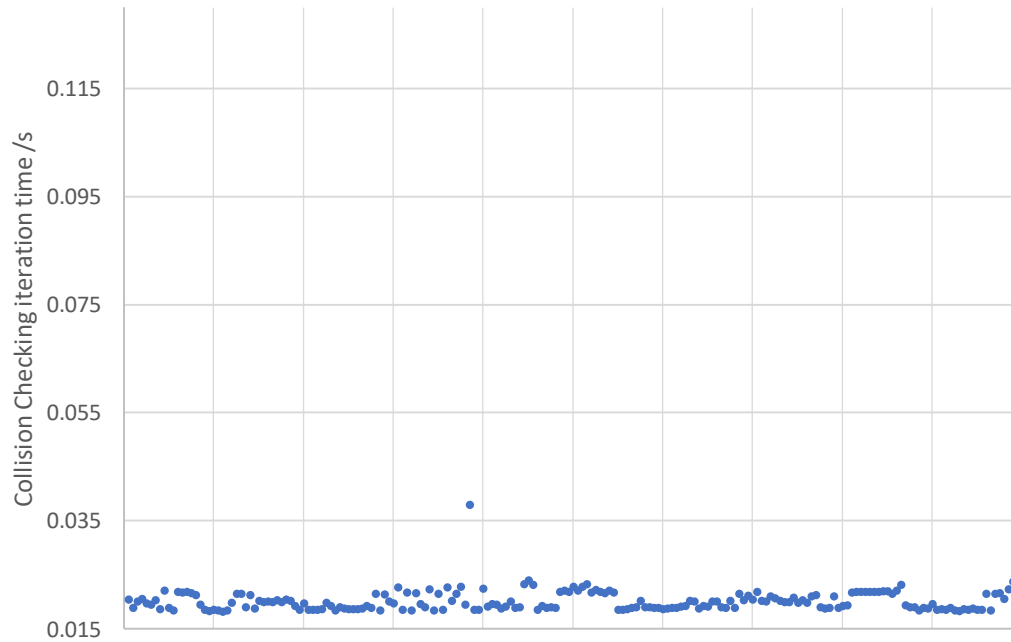




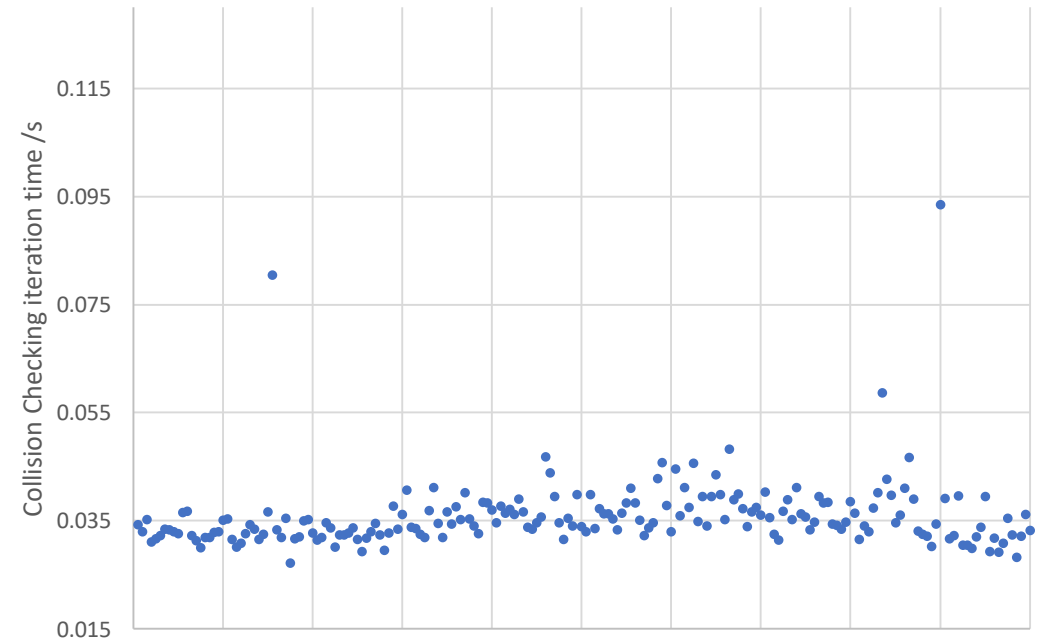
Dynamic Safety CPU Benchmarking

- **Ryzen Specs:** Ryzen 3900X 12-core 24 thread 3.8Hz,
GeForce RTX 2070 SUPER, 32GB RAM
- **Intel Specs:** Intel® Core™ i7-7700HQ CPU @ 2.80GHz × 8
NVIDIA Quadro M1200 Mobile, 32 GB RAM

Ryzen CPU with 4 Threads



Intel CPU with 4 Threads





Contact Us



Glenn Tan

Research Engineer

ARTC, ROS-Industrial Consortium Asia Pacific

Email: glenn_tan@artc.a-star.edu.sg

GitHub: tanjpg





CREATING GROWTH, ENHANCING LIVES



THANK YOU

www.a-star.edu.sg

EPD FINAL RELEASE CANDIDATE FEATURES

Bey Hao Yun (Gary)
ROS-I

8th June 2021

Final Release Candidate



cardboardcode Merge pull request #11 from cardboardcode/master ... ✓ affd46d 9 days ago 🔄 43 commits		
📁 .github/workflows	✓ Verified GitHub Action workflow with code-coverage QA.	23 days ago
📁 Dockerfiles	🔥 Included higher EMD-compatibility variant of EPD for Object Localiz...	23 days ago
📁 docs	📄 Updated documentation.	17 days ago
📁 easy_perception_deployment	🔧 Rectified EMD-compatibility flaws.	17 days ago
📁 epd_msgs	🔥 Included higher EMD-compatibility variant of EPD for Object Localiz...	23 days ago
📄 .gitignore	🔧 Rectified EMD-compatibility flaws.	17 days ago
📄 .gitlab-ci.yml	🔥 Included higher EMD-compatibility variant of EPD for Object Localiz...	23 days ago
📄 .readthedocs.yaml	🔧 Fix attempt for failing ReadtheDocs build.	last month
📄 CONTRIBUTING.md	Added Milestone 1 alpha release prototype. Tested and peer-verified.	7 months ago
📄 LICENSE	Added Milestone 1 alpha release prototype. Tested and peer-verified.	7 months ago
📄 QUALITY_DECLARATION.md	Added Milestone 1 alpha release prototype. Tested and peer-verified.	7 months ago
📄 README.md	🔧 Rectified EMD-compatibility flaws.	17 days ago



easy_perception_deployment

🔄 CI passing 📄 codecov 89% 📄 License Apache 2.0 📄 docs passing

Final Release Candidate has been released with the following CV features:

- Object Detection
- Object Localization
- **Object Tracking**

https://github.com/ros-industrial/easy_perception_deployment

What is EPD [1/6]?



A **ROS2** package that accelerates the training and deployment of custom-trained Computer Vision model for industries.

What is EPD [2/6]?

- Permissively Licensed and Open Source.
- Reduces time needed in training and deploying robotic vision systems by use of **transfer-learning**.
- Reduces knowledge barrier with the use of GUI to guide users. Targeted mainly at **users with no programming background**.
- Relies on **open-standard ONNX AI models**. Removes overreliance on any one given Machine Learning library (Eg. Tensorflow, PyTorch, MXNet).

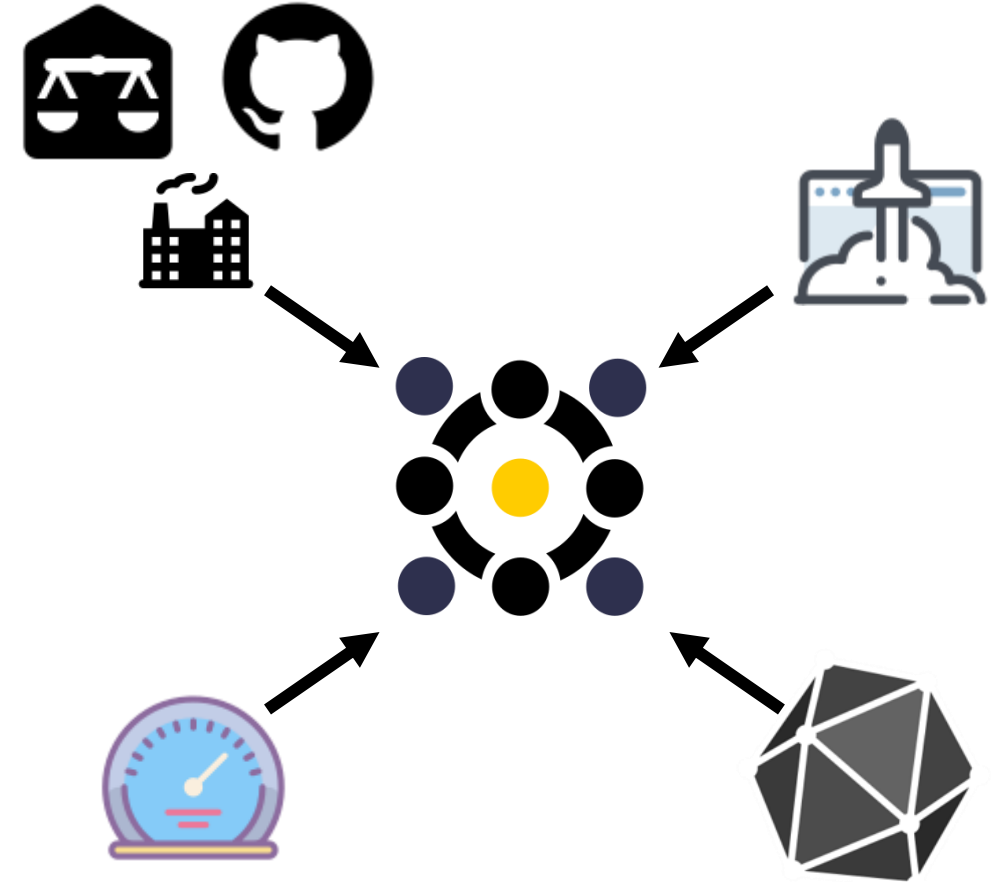


Image courtesy of icons8.com and onnx.com

What is EPD? [3/6] - Training

To train a model for custom object detection, a user need only prepare inputs:

- *.jpgs/.pngs* Image Dataset of custom objects.
 - (Approximately 30 images for each object)
- *.txt* Class Labels List

The expected output will be:

- *.onnx* trained AI model

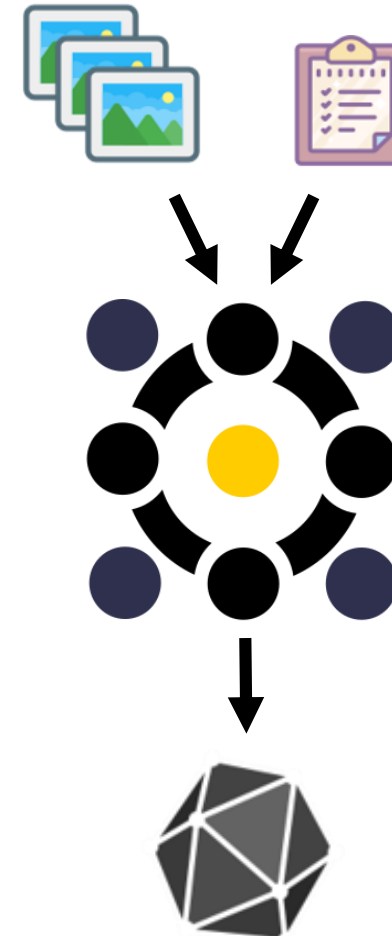


Image courtesy of icons8.com and onnx.com

What is EPD? [4/6] – Training GUI

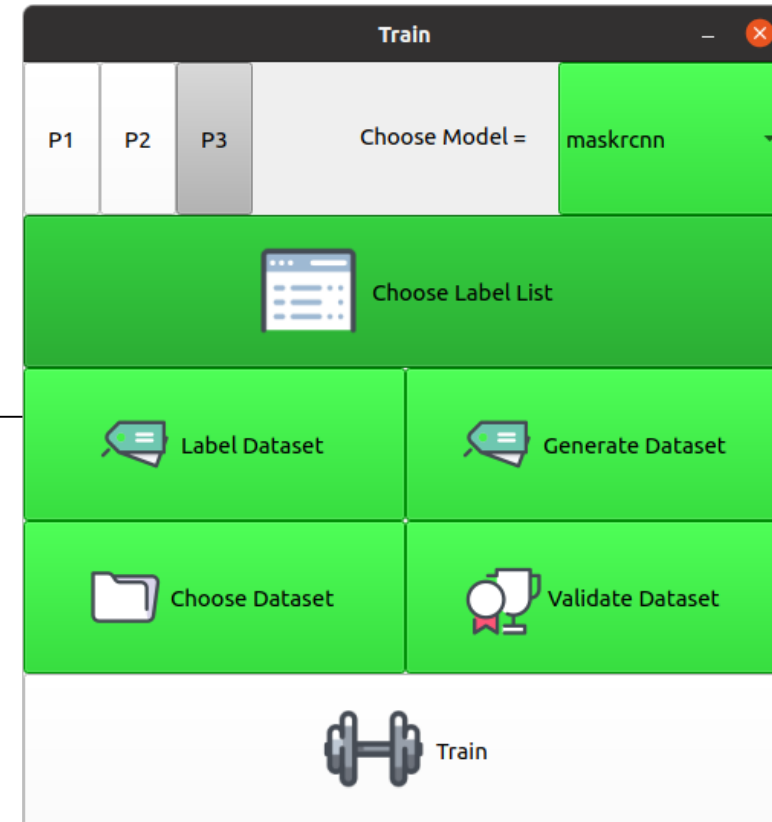
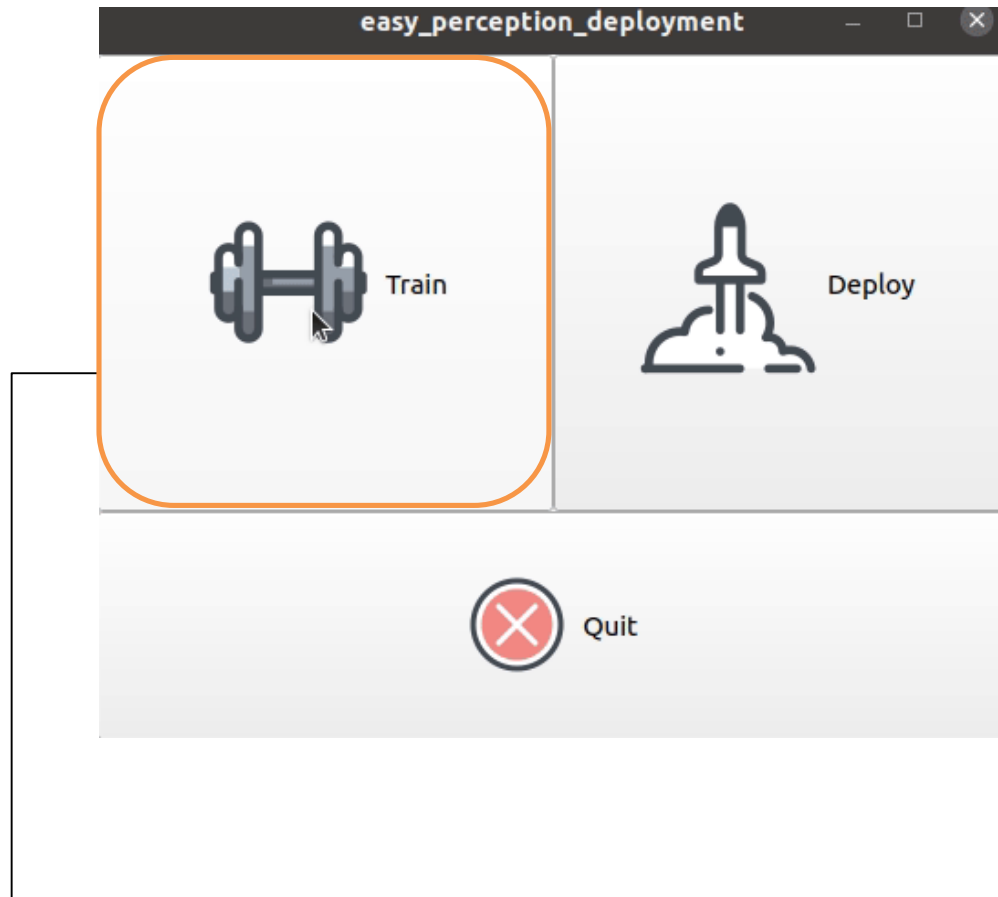


Image courtesy of icons8.com and onnx.com

What is EPD? [5/6] - Deployment

To deploy a model for custom object detection, a user need only prepare inputs:

- `.onnx` trained AI model
- `.txt` Class Labels List

The **expected output** will be:

- A *ROS2 package* that runs inference using the model and classifies images provided by a video stream from a camera.

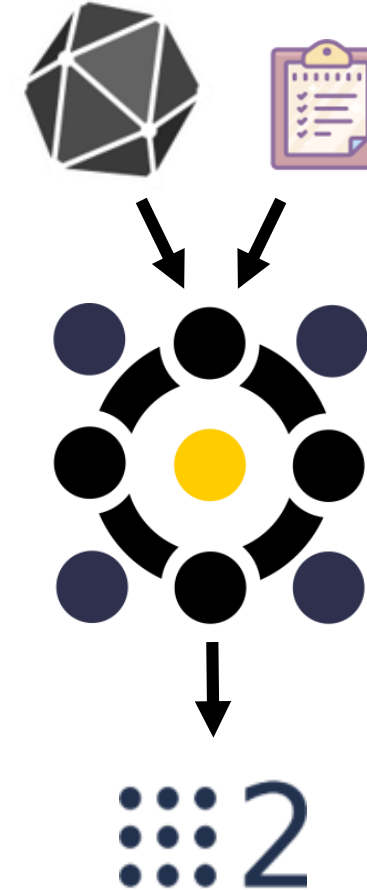
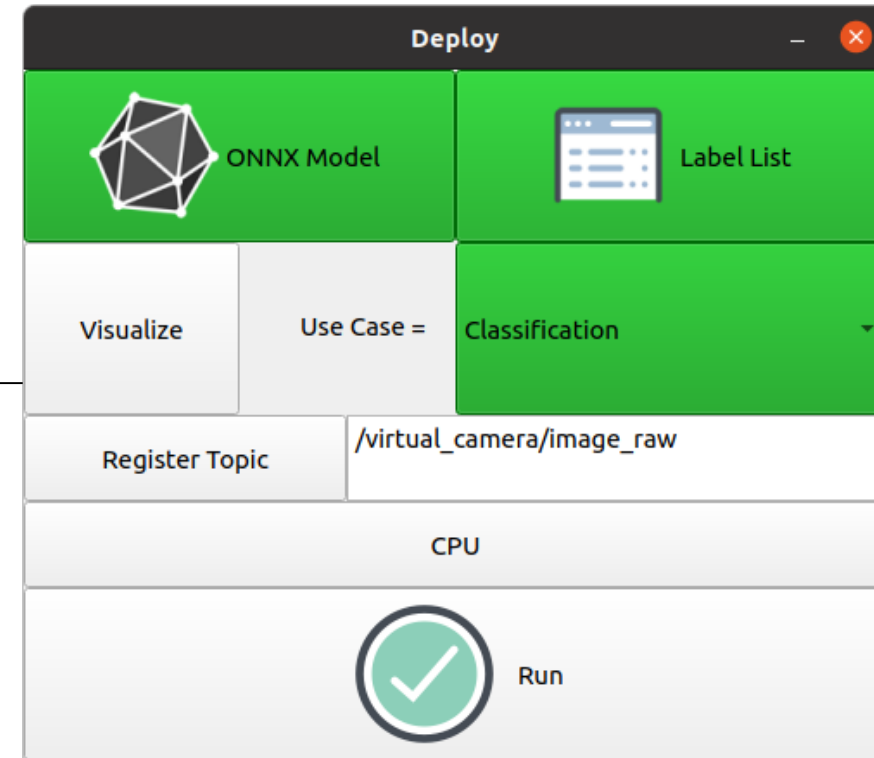
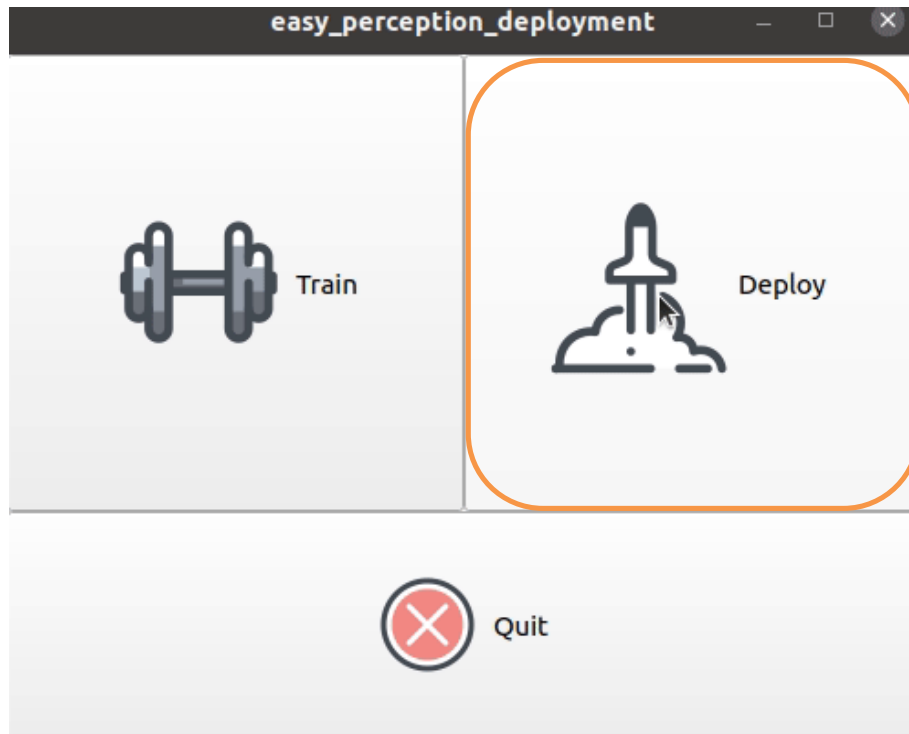


Image courtesy of icons8.com and onnx.com

What is EPD? [6/6] – Deployment GUI

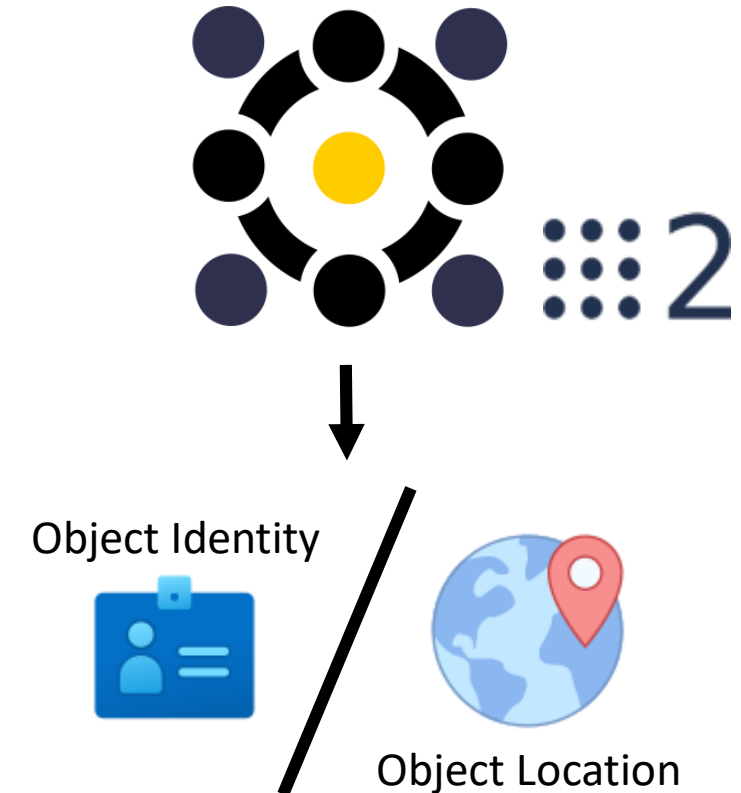


Built-In Use Case Configurations

EPD runs a deep-learning model as a ROS2 inference engine.

It outputs the following object information in the form of **custom ROS2 messages** that caters to common Computer Vision demands.

1. What is the object? (Object Classification)
2. Where is the object? (Object Localization/Tracking)



Customizable Speed-Accuracy Tradeoffs - [1/2]



There is **no one-size-fits-all** deep learning model in Computer Vision.

There are a huge variety of deep-learning models but there are only a handful that is suitable for specific use cases.

EPD can be configured to run at **3 different Precision Levels**.

Eg. **1** being the **fastest** but **least accurate**. **3** being the **slowest** but **most precise**.

Each level **determines** different model and label list for **inputs** as well as **outputs** given the selected model.

Customizable Speed-Accuracy Tradeoffs - [2/2]



Precision Level	Inputs	Outputs
1	Model: squeezeNet.onnx Label List: imagenet_classes.txt	EPDImageClassification ROS2 Message Format: string[] object_names
2	Model: FasterRCNN.onnx Label List: coco_classes.txt	EPDObjectDetection ROS2 Message Format: uint64[] class_indices float64[] scores sensor_msgs/RegionOfInterest[] bboxes sensor_msgs/Image[] masks – <Left empty>
3	Model: maskRCNN.onnx Label List: coco_classes.txt	EPDObjectDetection ROS2 Message Format: uint64[] class_indices float64[] scores sensor_msgs/RegionOfInterest[] bboxes sensor_msgs/Image[] masks

Customizable Use-Case Configurations - [1/6]



EPD caters to 5 common industrial tasks achievable via Computer Vision.

1. **Classification** (P1, P2, P3)
2. **Counting** (P2, P3)
3. **Color-Matching** (P2, P3)
4. **Localization/Measurement** (P3)
5. **Localization/Measurement/Tracking** (P3)

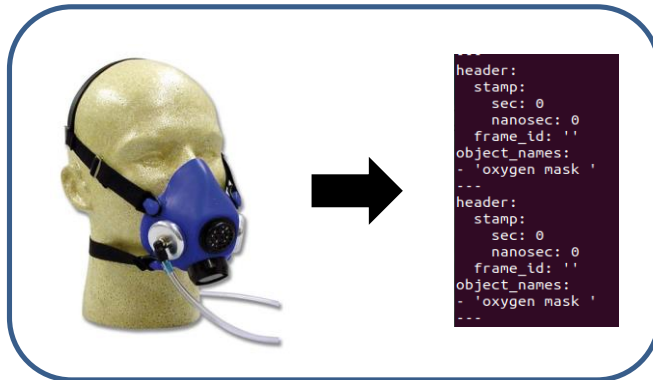
Localization is combined with Measurement since these use cases often go hand-in-hand for common industrial task like Pick-N-Place.

Tracking is also coupled together as well for the same reason.

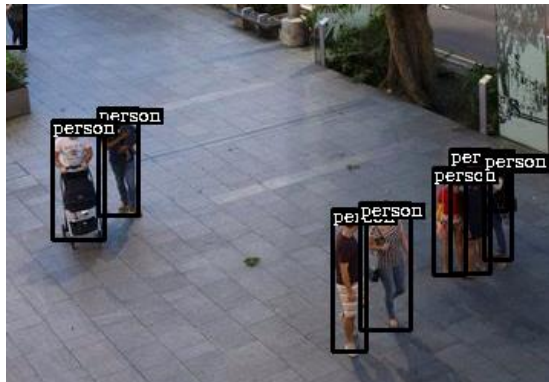
Customizable Use-Case Configurations - [2/6]

Classification (P1, P2, P3) – EPDObjectClassification.msg

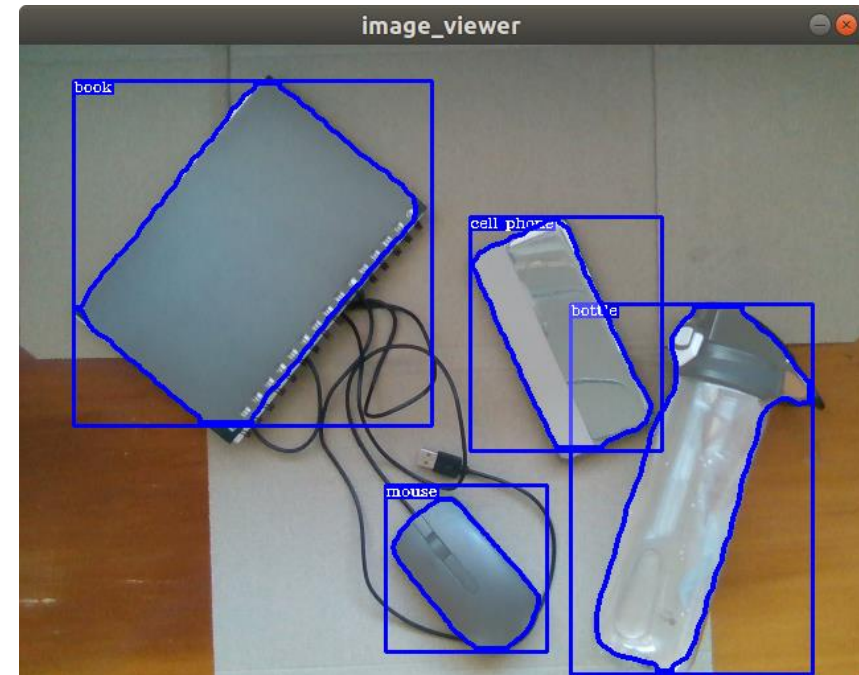
P1



P2

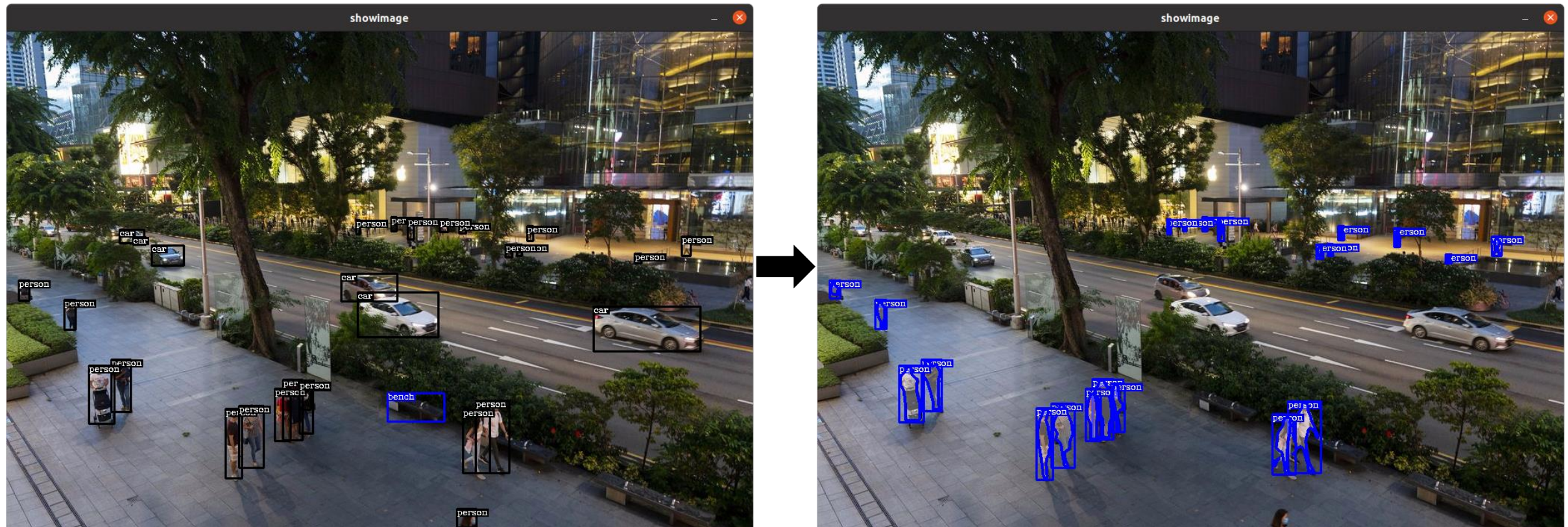


P3



Customizable Use-Case Configurations - [3/6]

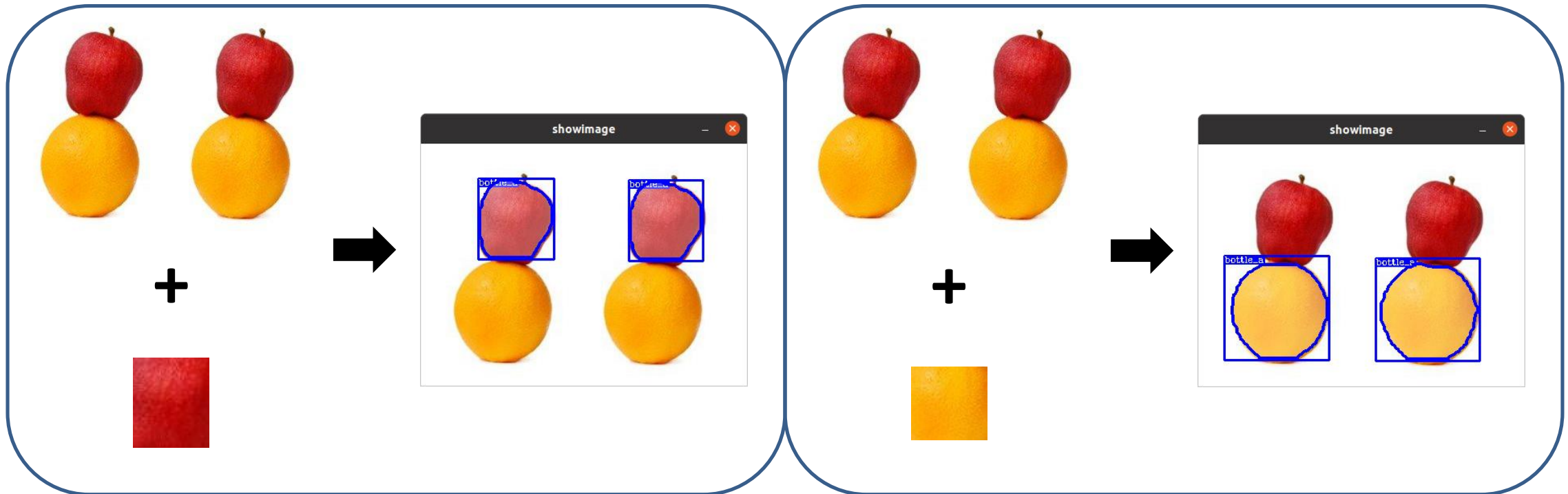
Counting (P2, P3) – EPDObjectDetection.msg



Automatically filter out unwanted detected objects without having to retrain your model.

Customizable Use-Case Configurations - [4/6]

Color-Matching (P2, P3)



Customizable Use-Case Configurations - [5/6]

Localization/Measurement/Tracking (P3)

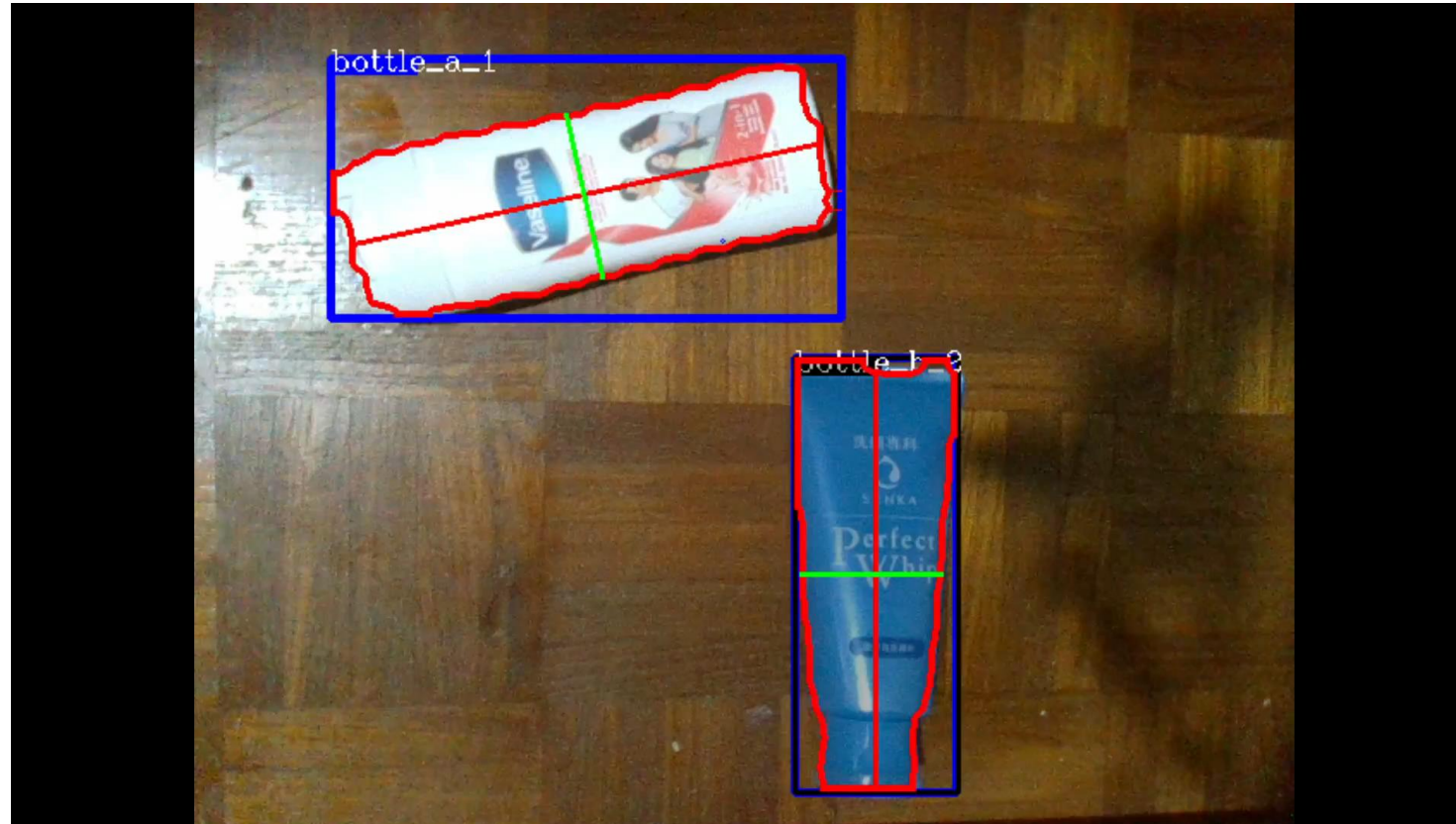


```
table_depth = 0.402
obj_surface_depth = 0.256
[-OBJ centroid x-] = 0.0246614
[-OBJ centroid y-] = 0.00464559
[-OBJ centroid z-] = 0.329
[-OBJ Name-] = bottle_a
[-OBJ Length-] = 0.112287
[-OBJ Breadth-] = 0.0386248
[-OBJ Height-] = 0.146
[INFO] [1622868349.677692128] [processor]: [-FPS-]= 4.950495
```

```
table_depth = 0.402
obj_surface_depth = 0.374
[-OBJ centroid x-] = 0.0341889
[-OBJ centroid y-] = 0.00555816
[-OBJ centroid z-] = 0.388
[-OBJ Name-] = bottle_a
[-OBJ Length-] = 0.165901
[-OBJ Breadth-] = 0.0564384
[-OBJ Height-] = 0.028
[INFO] [1622868349.881522108] [processor]: [-FPS-]= 4.950495
```

Customizable Use-Case Configurations - [6/6]

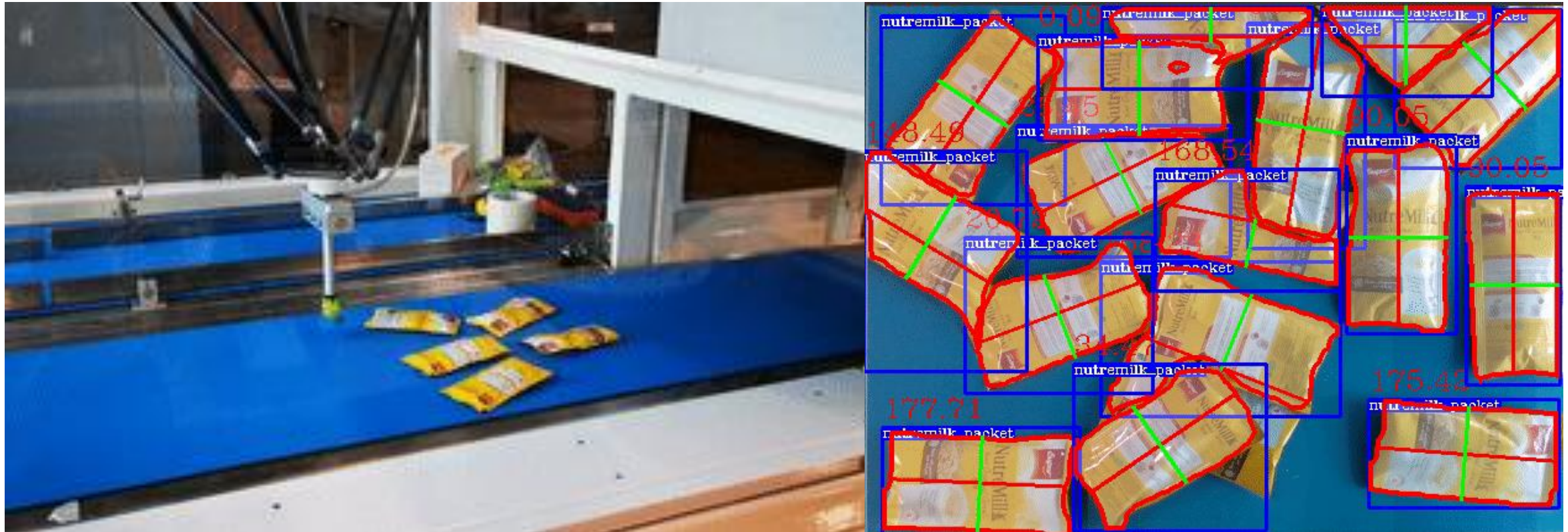
Localization/Measurement/Tracking (P3)



Tested for Industrial Use

EPD Configuration: Precision Level 3, Object Localization, operating at 2 FPS


Use Case Description: Industrial Conveyor Tracking and Automated Picking.



Get Started with EPD

ros-industrial / [easy_perception_deployment](#)

<> Code ⓘ Issues 🔗 Pull requests ⏮ Actions 📁 Projects 🛡 Security 📈 Insights

 cardboardcode	Merge pull request #11 from cardboardcode/master	✓ aafd46d 9 days ago	🕒 43 commits
📁 .github/workflows	✓ Verified GitHub Action workflow with code-coverage QA.	23 days ago	
📁 Dockerfiles	🔥 Included higher EMD-compatibility variant of EPD for Object Localiz...	23 days ago	
📁 docs	📄 Updated documentation.	17 days ago	
📁 easy_perception_deployment	🔗 Rectified EMD-compatibility flaws.	17 days ago	
📁 epd_msgs	🔥 Included higher EMD-compatibility variant of EPD for Object Localiz...	23 days ago	
📄 .gitignore	🔗 Rectified EMD-compatibility flaws.	17 days ago	
📄 .gitlab-ci.yml	🔥 Included higher EMD-compatibility variant of EPD for Object Localiz...	23 days ago	
📄 .readthedocs.yaml	🔗 Fix attempt for failing ReadtheDocs build.	last month	
📄 CONTRIBUTING.md	Added Milestone 1 alpha release prototype. Tested and peer-verified.	7 months ago	
📄 LICENSE	Added Milestone 1 alpha release prototype. Tested and peer-verified.	7 months ago	
📄 QUALITY_DECLARATION.md	Added Milestone 1 alpha release prototype. Tested and peer-verified.	7 months ago	
📄 README.md	🔗 Rectified EMD-compatibility flaws.	17 days ago	

☰ README.md ✎



easy_perception_deployment

🔄 CI passing 📄 codecov 89% 📄 License Apache 2.0 📄 docs passing



Bey Hao Yun (Gary)

Research Engineer

ARTC, ROS-Industrial Consortium Asia Pacific

Email: Bey_Hao_Yun@artc.a-star.edu.sg

GitHub: cardboardcode

