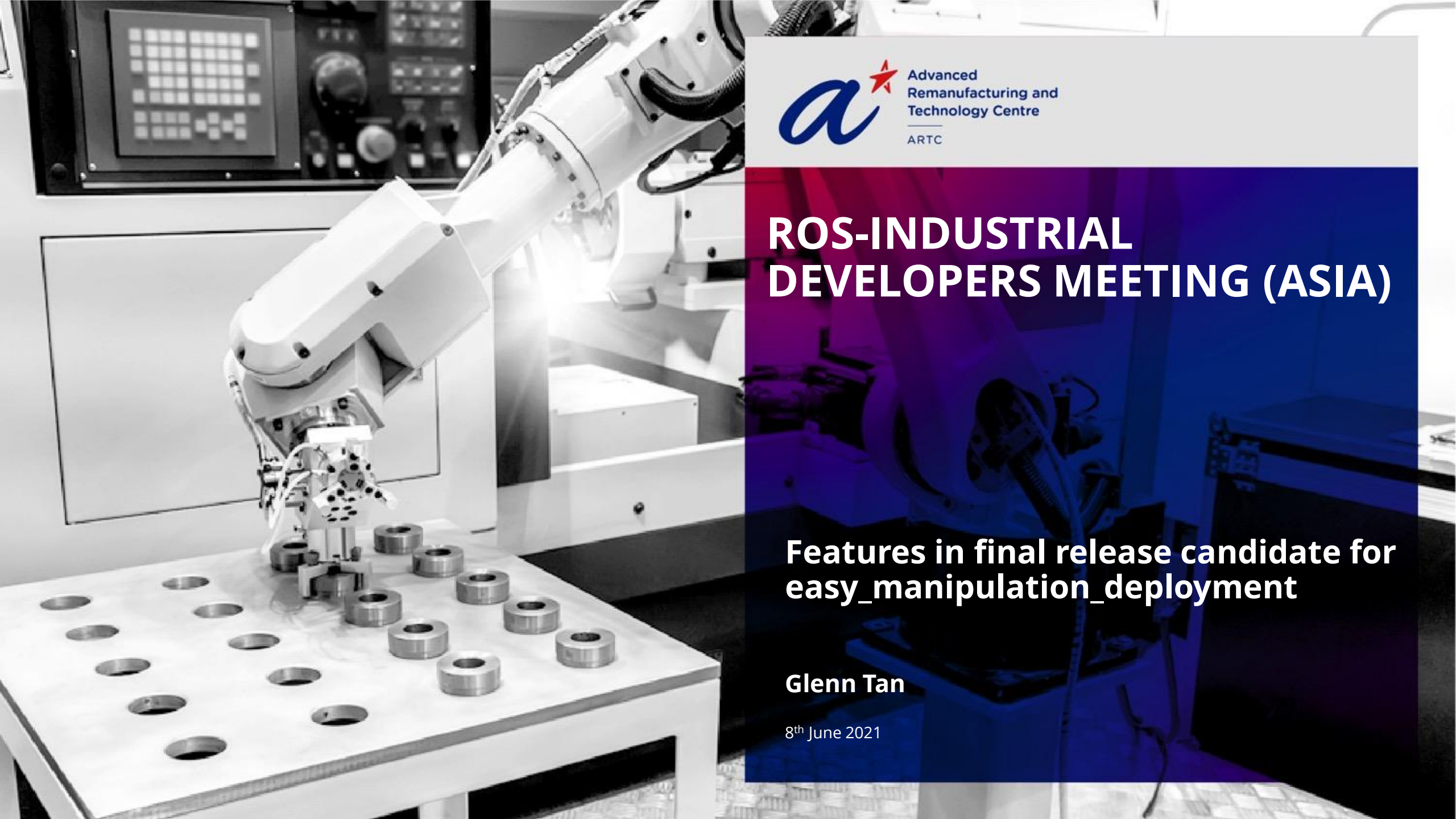


Advanced Remanufacturing and Technology Centre
ARTC

# ROS-INDUSTRIAL DEVELOPERS MEETING (ASIA)

## Features in final release candidate for easy_manipulation_deployment

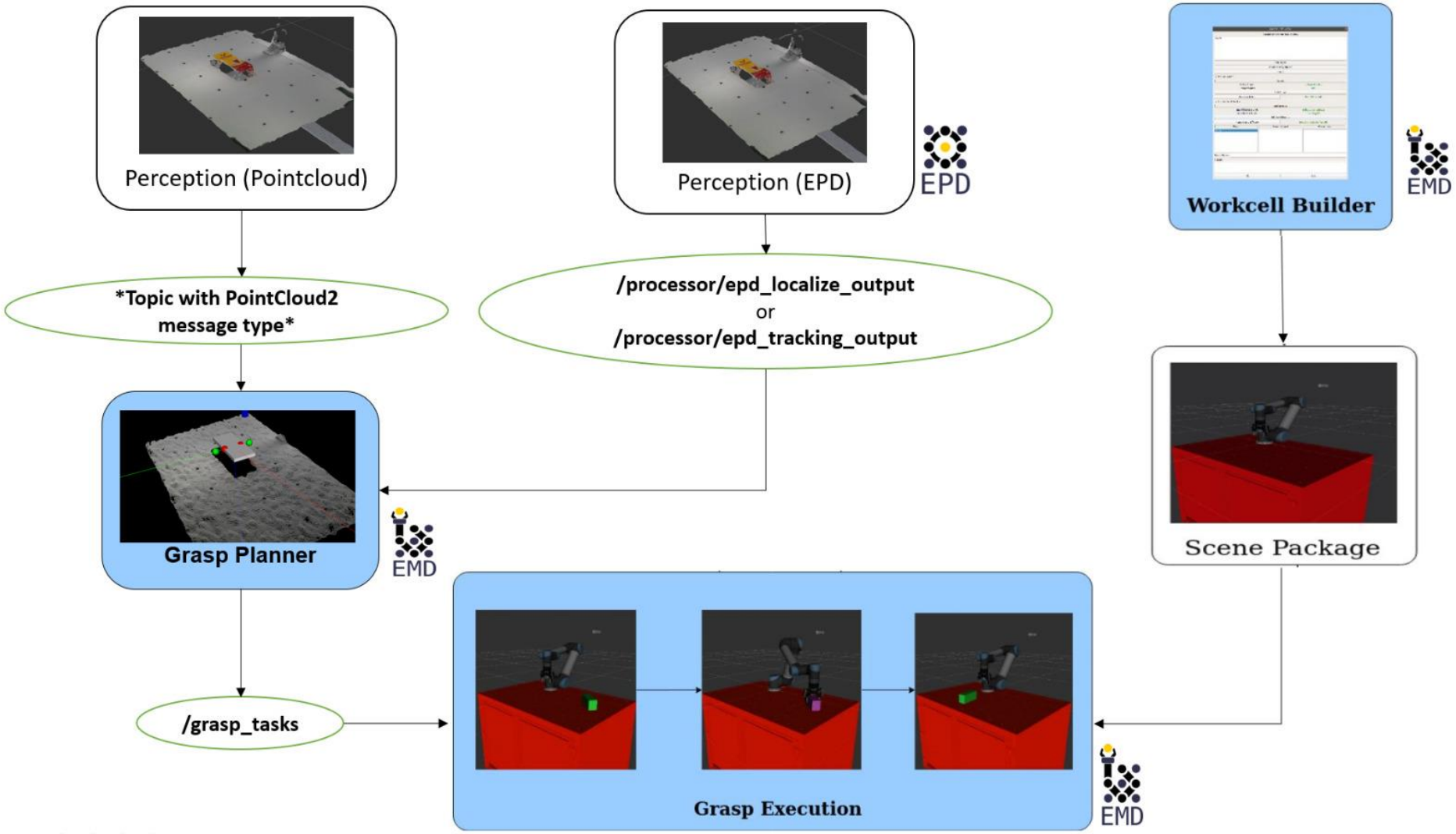
**Glenn Tan**

8th June 2021

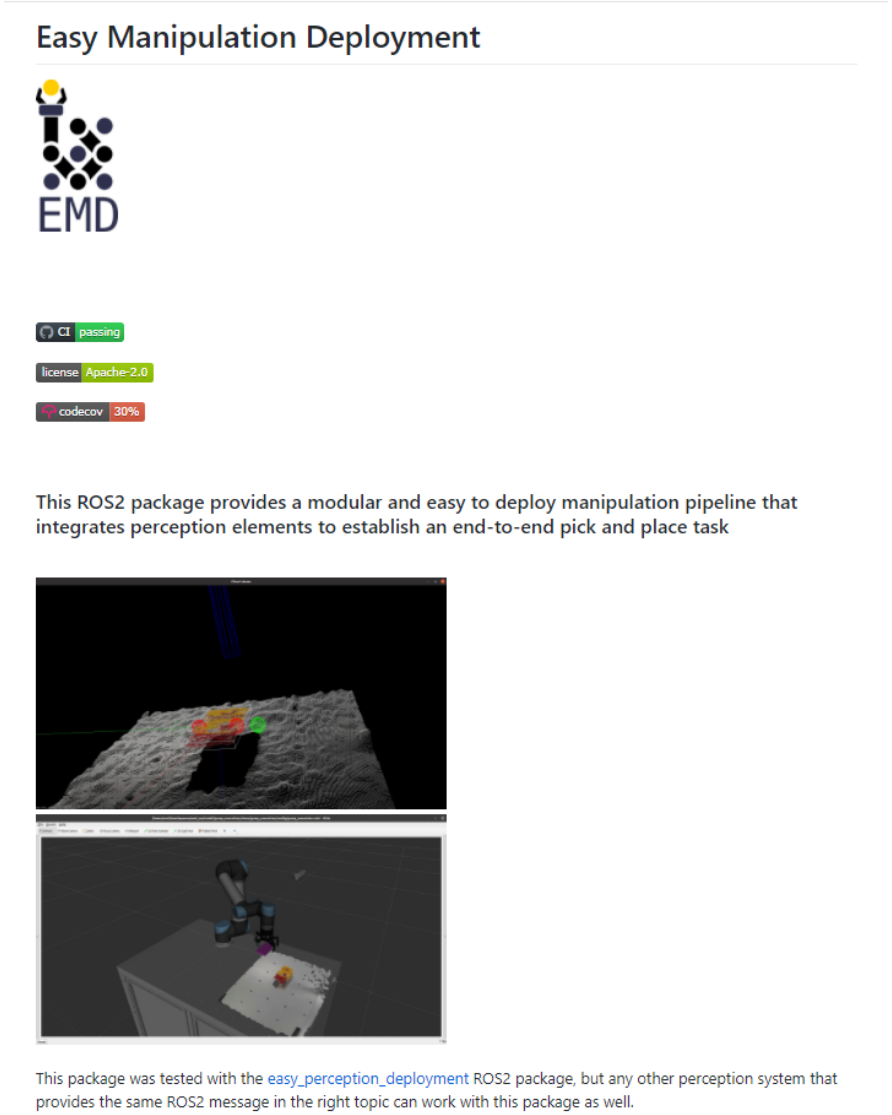
# easy_manipulation_deployment


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



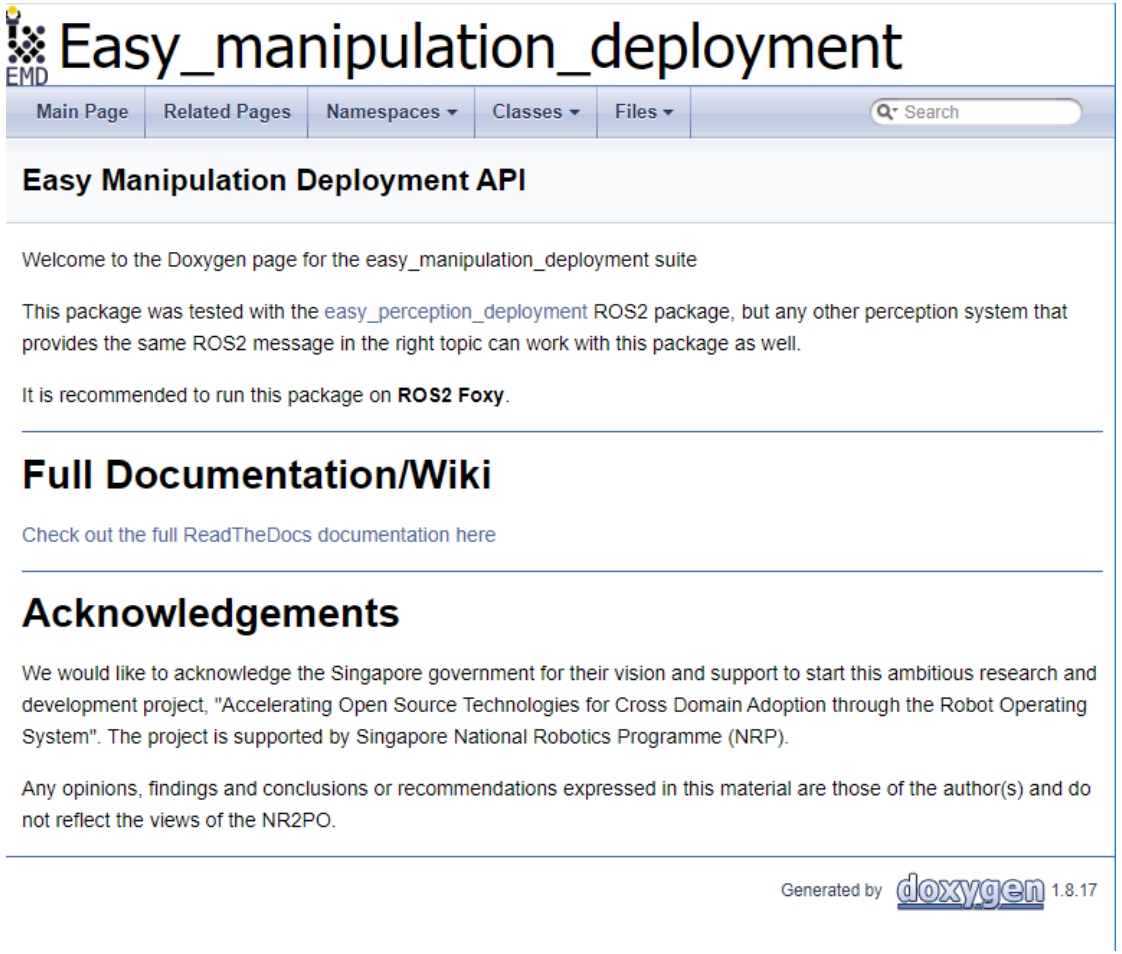
# Final Release Candidate

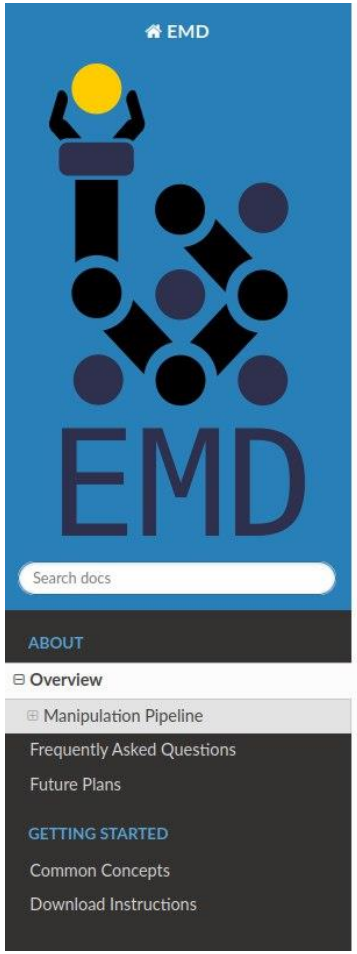


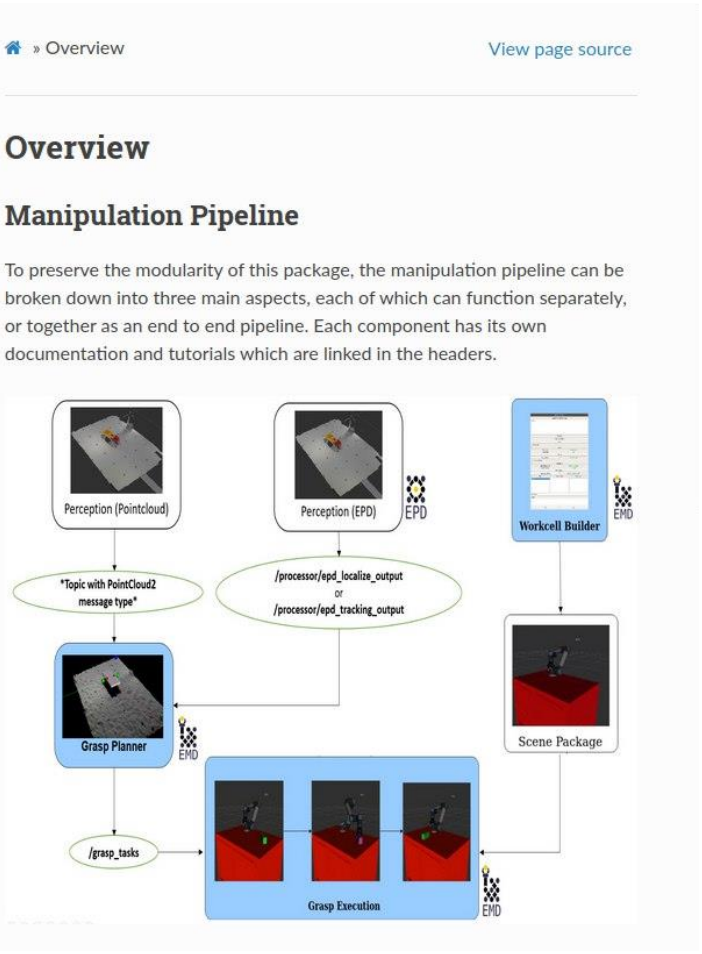

- Final Release Candidate has been released with all major new features added
  - Support for multifinger/suction array grasps
  - Support Dynamic Safety for grasp execution
  - Parameterization of entire pipeline

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

# Final Release Candidate Documentation

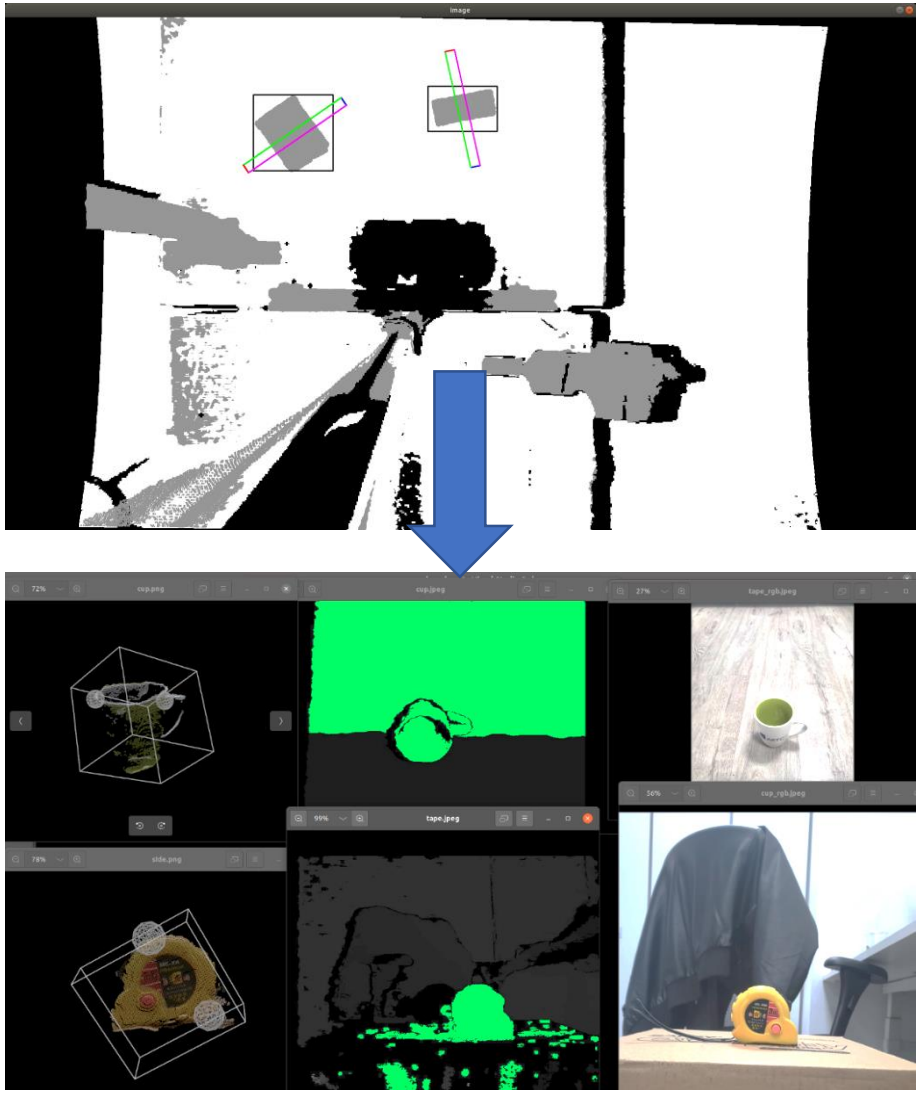


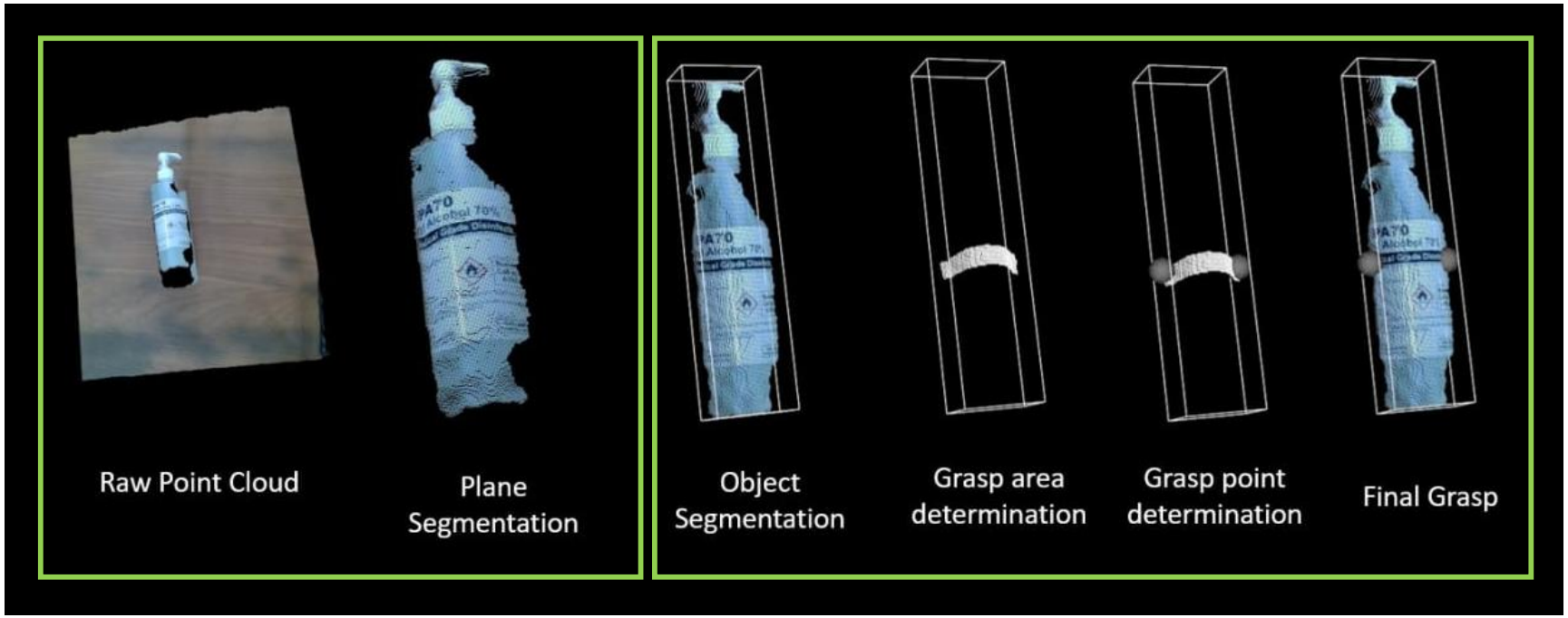
# Improved Grasp Planner

# 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

$$rank(q_1, q_2) = w1 \times r1(q_1, q_2) + w2 \times r2(q_1, q_2)$$
$$r1(q_1, q_2) = \Big(1.0 - dist(\gamma, q_1)\Big) + \Big(1.0 - dist(\gamma, q_2)\Big) - \Big(cos(\beta) - 0.2\Big) \times 10.0$$
$$r2(q_1, q_2) = (1.0 - \lambda_{q_1}) + (1.0 - \lambda_{q_2}) + \cos(\alpha_{q_1}) + \cos(\alpha_{q_2}) - ||\cos(\alpha_{q_1}) - \cos(\alpha_{q_2})||$$

$$Rank_{finger} = \omega_1 \times r_1 + \omega_2 \times r_2$$
$$r_1 = \Big(\sum_{i=0}^{n_1} 1.0 - \gamma_i\Big) + \Big(\sum_{j=0}^{n_2} 1.0 - \gamma_j\Big) - (cos\beta - 0.2) \times 10.0$$
$$r_2 = \sum_{i=0}^{n_1} 1.0 - \lambda_i + \sum_{j=0}^{n_2} 1.0 - \lambda_j$$
$$Rank_{suction} = 2.0 - (\lambda \times \omega_\lambda) - (\gamma \times \omega_\gamma) + (c \times \omega_c)$$

- 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**

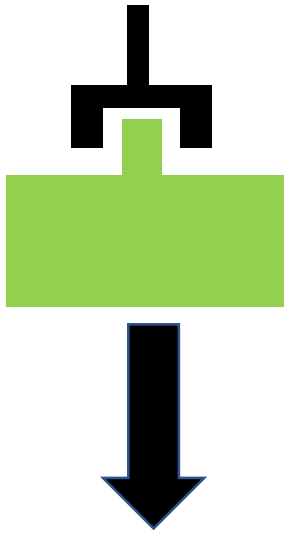
Suction contact points

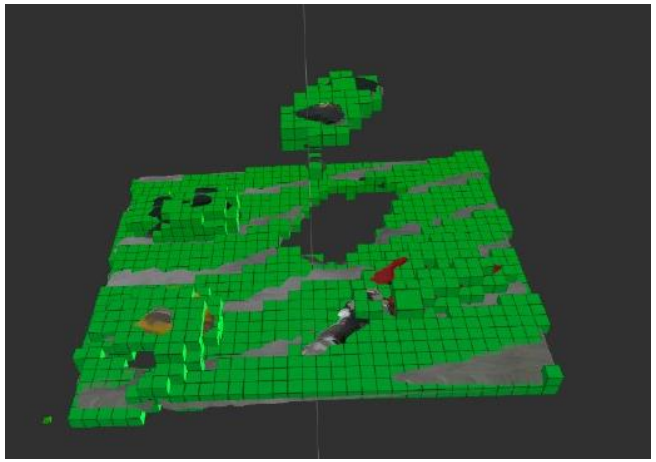
Distance from grasp to centroid of object

**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



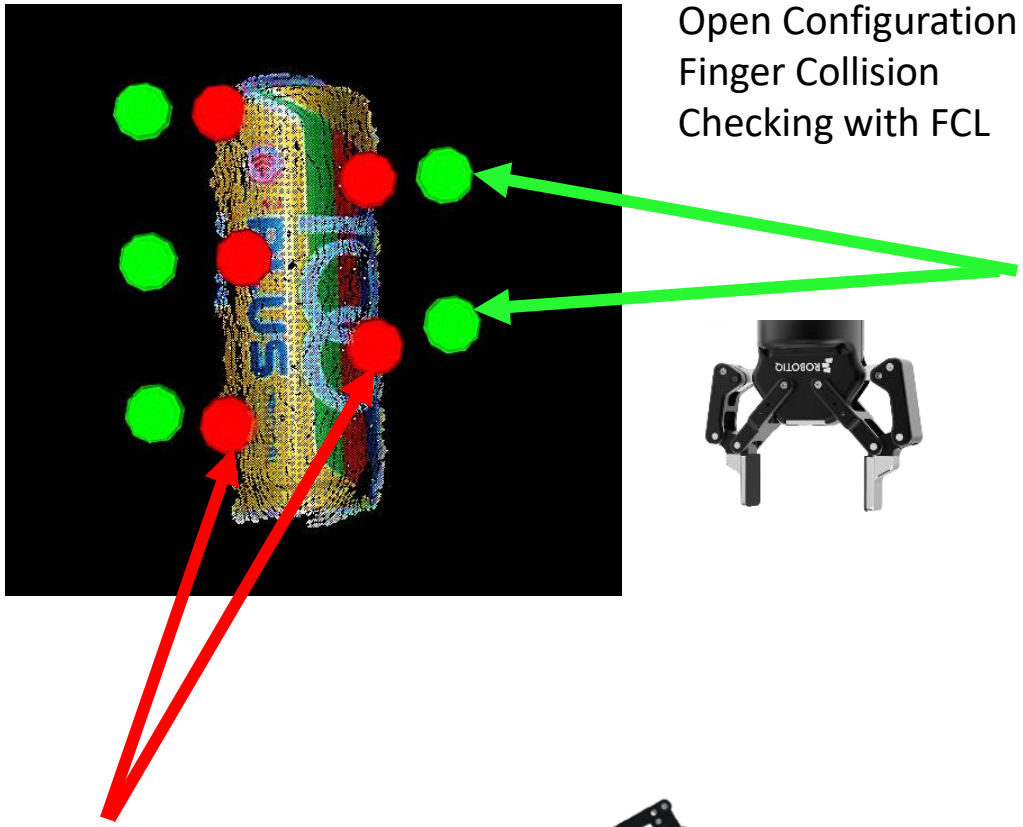


- Finger collision checks were initially based on checking depth values at the point of grasp and average values of the points representing the gripper
  - 2 Dimensional method does not provide extensive checks and is not truly reflective In a 3 dimensional environment

- 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.**
  - Colllision checks are now more robust and takes into account surrounding objects

CREATING GROWTH, ENHANCING LIVES

# Improved Finger Grasping Methodology



Open Configuration Finger Collision Checking with FCL



Closed Configuration Contact point ranking

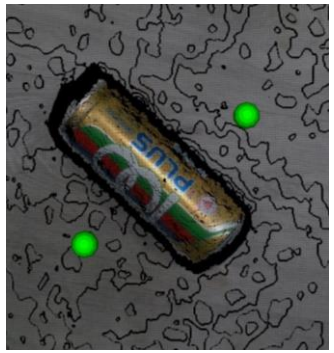
- Finger collision check is now not part the ranking system, but rather a first-pass criteria for each grasp option.
- Valid grasp candidates are now determined with a 2 step method. Closed and open configuration of the grasps
- Speeds up the grasp candidate generation by only ranking grasps that wont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

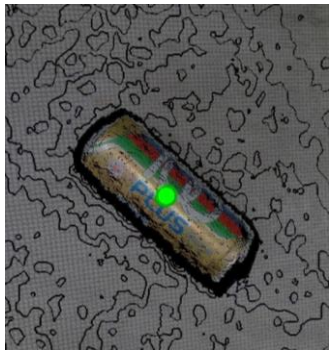
CREATING GROWTH, ENHANCING LIVES

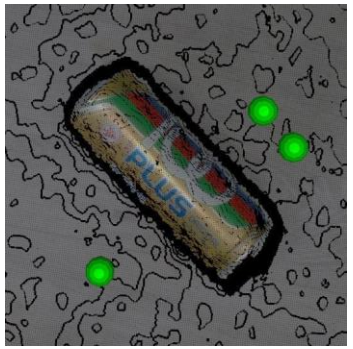
# Improved support for Multi-Finger/Suction array

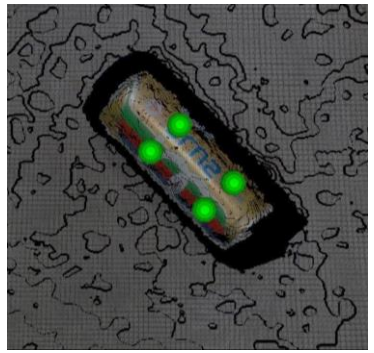












# Improved Grasp Planner – Improved Planning times

| Number of Fingers | Point Cloud Grasp Planning [1] Time/ ms* | Depth image Grasp Planning [2] Time/ ms* |
|---|---|---|
| 2 | 11 | 420 |
| 3 | 90 | - |
| 4 | 96 | - |
| 5 | 292 | - |
| 8 | 309 | - |
| 10 | 663 | - |

| Suction array | Point Cloud Grasp Planning [1] Time/ ms* | Depth image Grasp Planning [2] Time/ ms* |
|---|---|---|
| 1x1 | 195 | 250 |
| 2x1 | 216 | - |
| 2x2 | 272 | - |
| 3x3 | 416 | - |
| 4x4 | 658 | - |
| 5x5 | 874 | - |

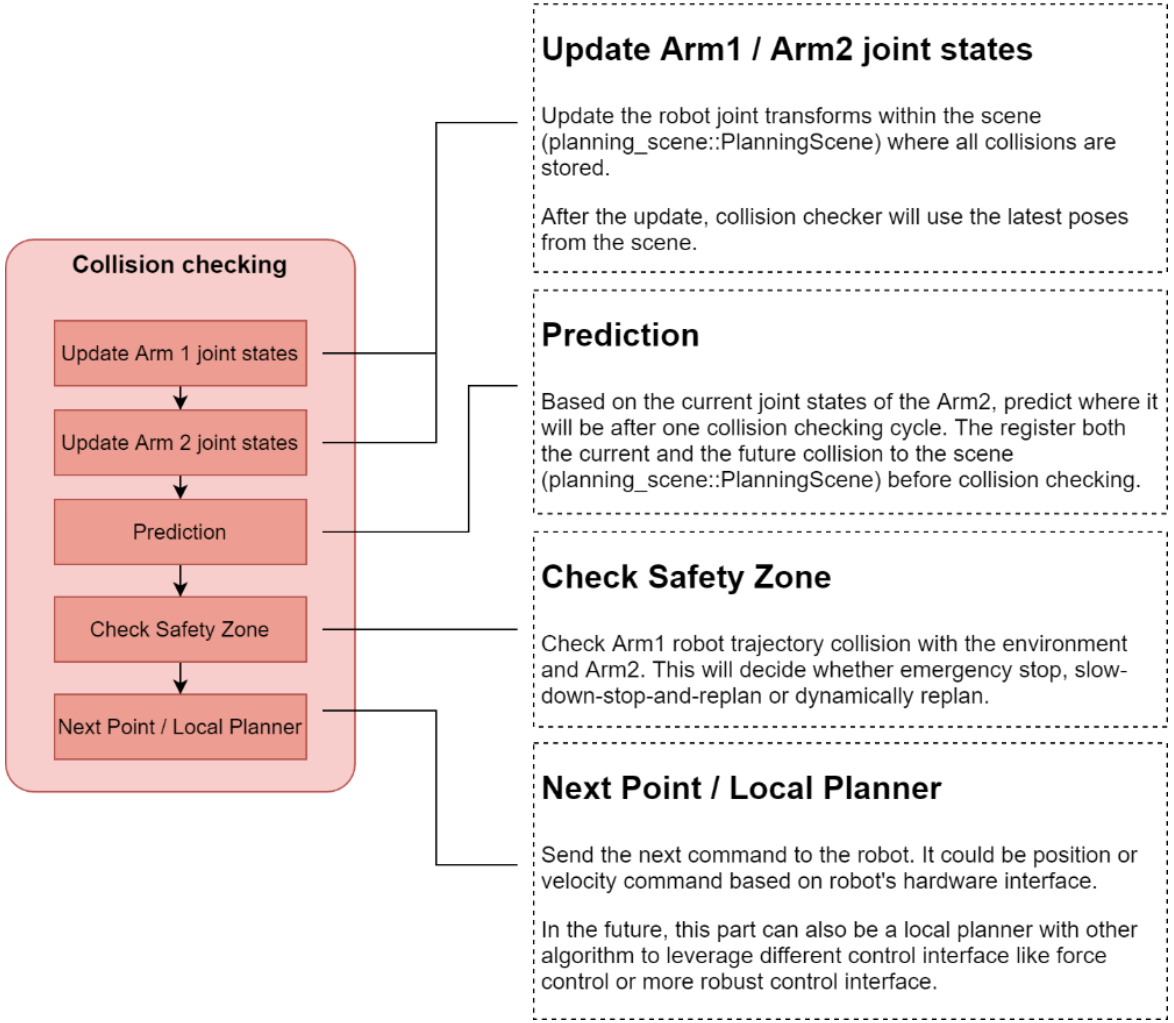
[1] https://doi.org/10.1177/1729881419831846

[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

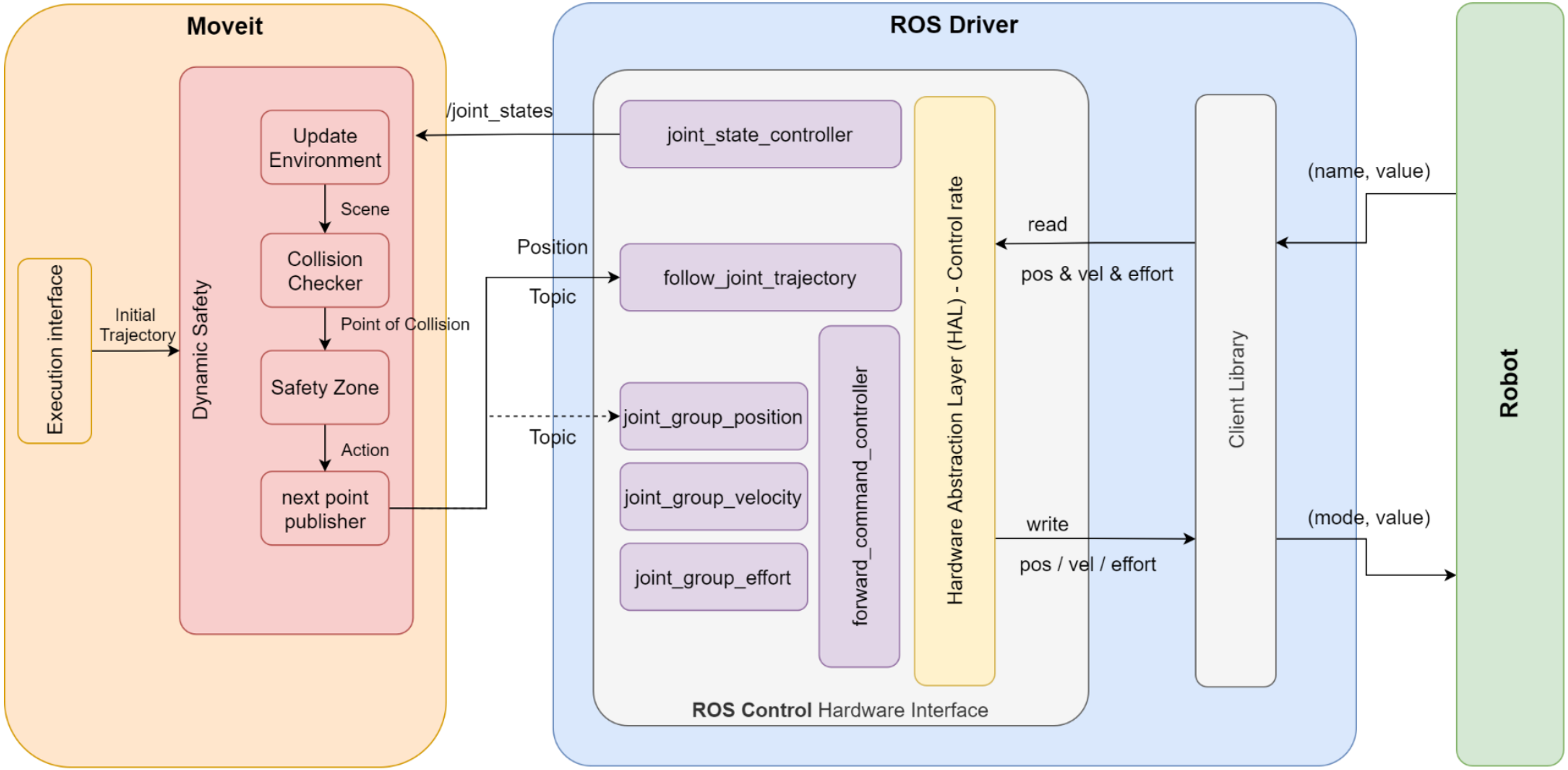






16

# Improved Grasp Execution – Dynamic Safety Architecture




CREATING GROWTH, ENHANCING LIVES


17

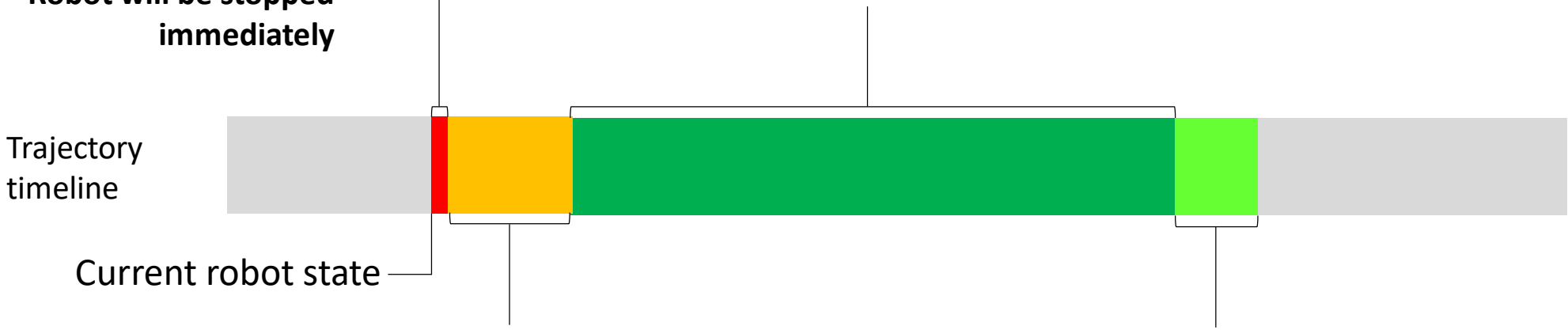
# Dynamic Safety Zones

## Blind Zone

Collision detected even before the collision check is complete. **Robot will be stopped immediately**

## Slow down and Re-plan zone

Collision detected but re-planning is possible. **Robot trajectory will slow down, and re-planning occurs**




## Emergency Zone

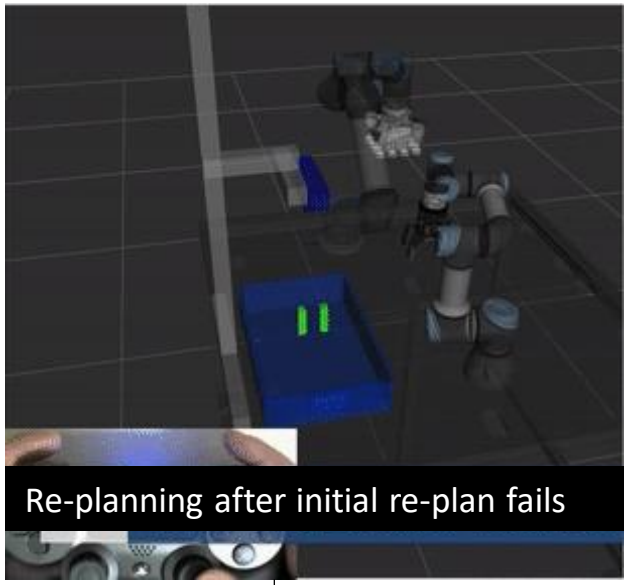
Collision detected but not enough time to slow down and stop safely. **Robot will be stopped immediately**

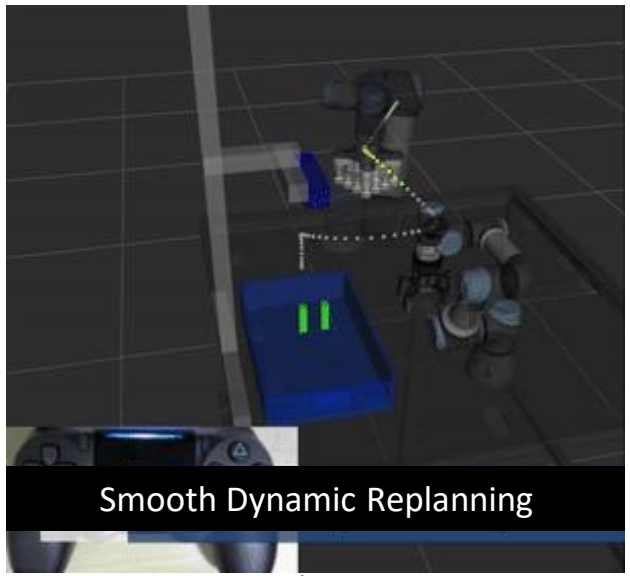
## Dynamic Re-plan Zone

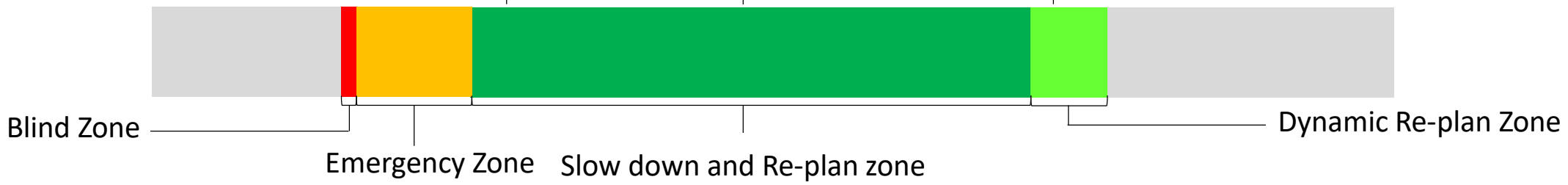
Collision detected, re-planning is possible, and robot is not required to slow down. **Robot Trajectory will continue as intended, while re-planning occurs simultaneously.**

CREATING GROWTH, ENHANCING LIVES

# Improved Grasp Execution – Dynamic Zone feature


CREATING GROWTH, ENHANCING LIVES

# Increased Configurability for Grasp Execution

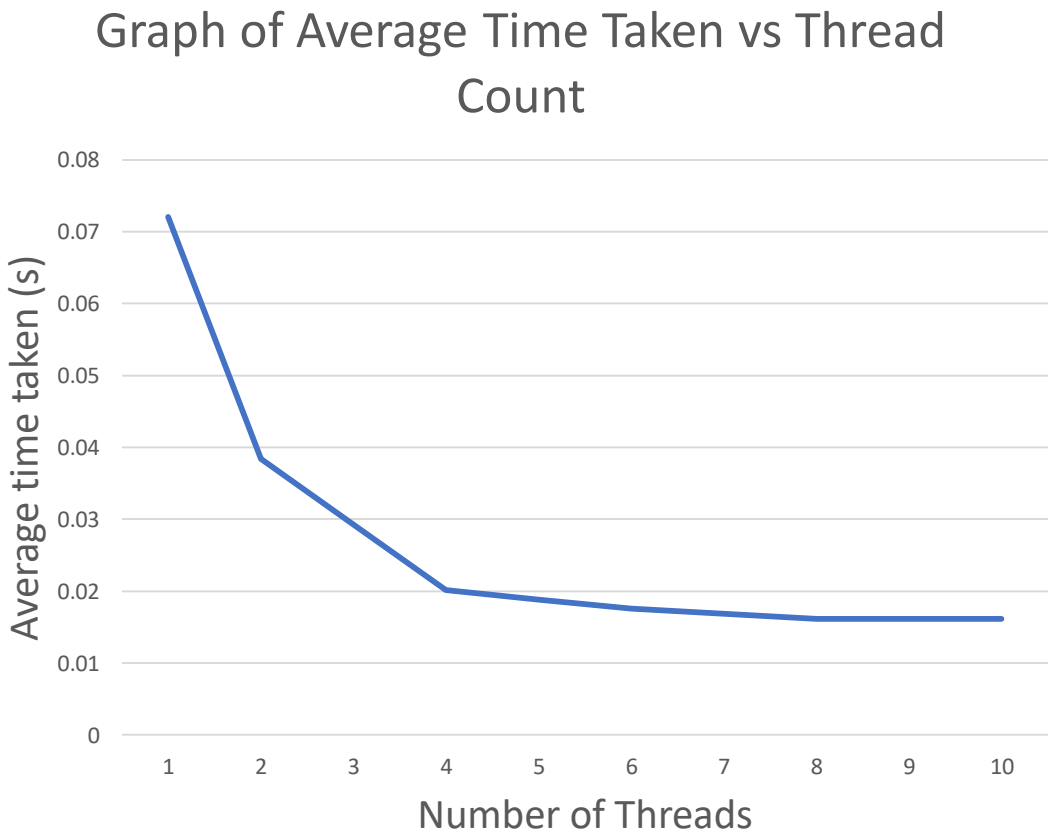
```yaml
# Load octomap
load_octomap: true

# 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
```

```yaml
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



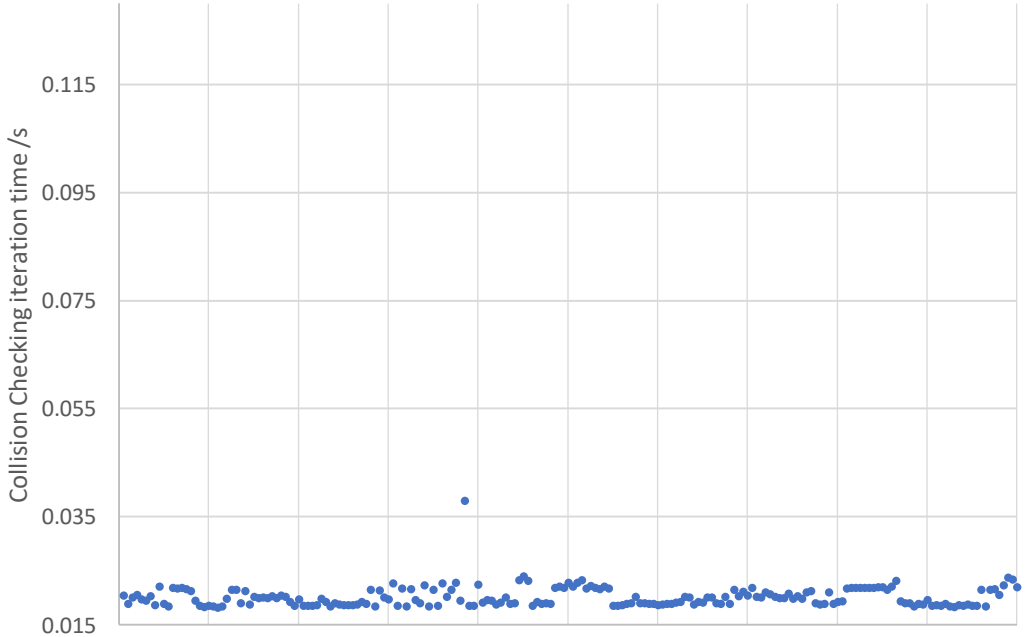

CREATING GROWTH, ENHANCING LIVES


21

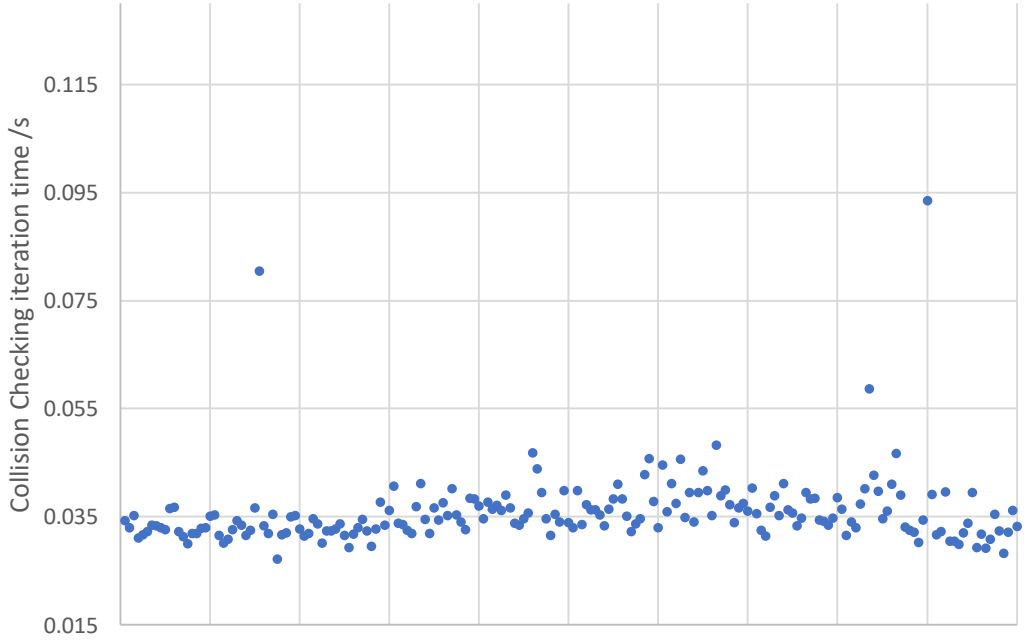
# Dynamic Safety CPU Benchmarking

- **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




CREATING GROWTH, ENHANCING LIVES


22

# 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


23

CREATING GROWTH, ENHANCING LIVES

Advanced Remanufacturing and Technology Centre
ARTC

# THANK YOU

www.a-star.edu.sg

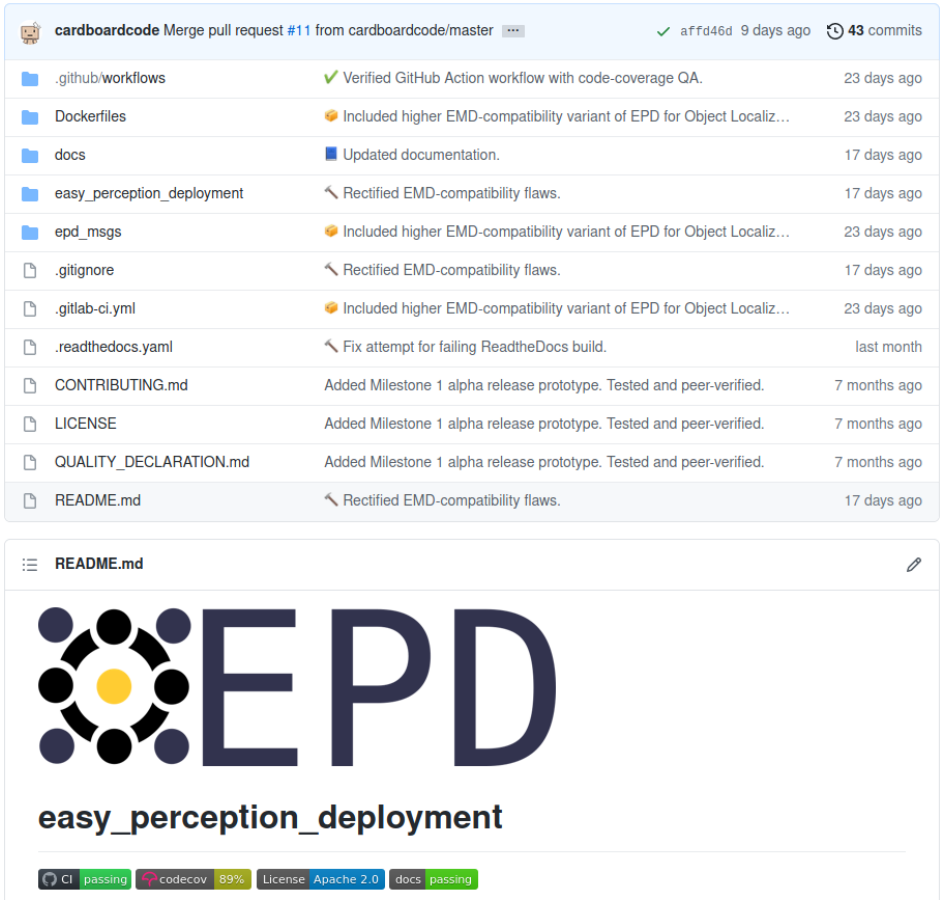
Advanced Remanufacturing and Technology Centre
ARTC

# EPD FINAL RELEASE CANDIDATE FEATURES

**Bey Hao Yun (Gary)**
ROS-I

8th June 2021

# Final Release Candidate




**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).

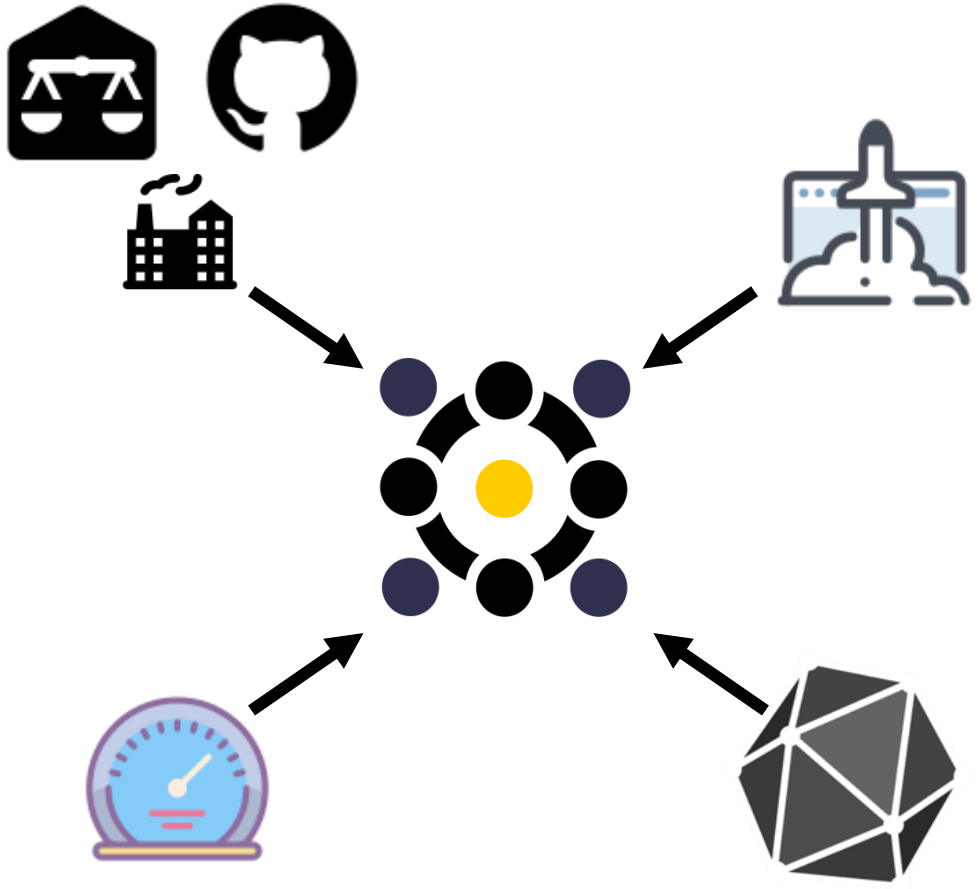


Image courtesy of icons8.com and onnx.com

# 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

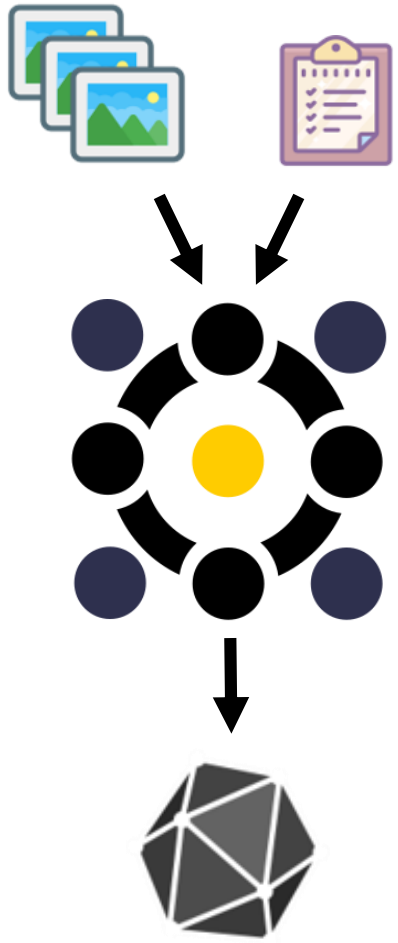


Image courtesy of icons8.com and onnx.com

# What is EPD? [4/6] – Training GUI

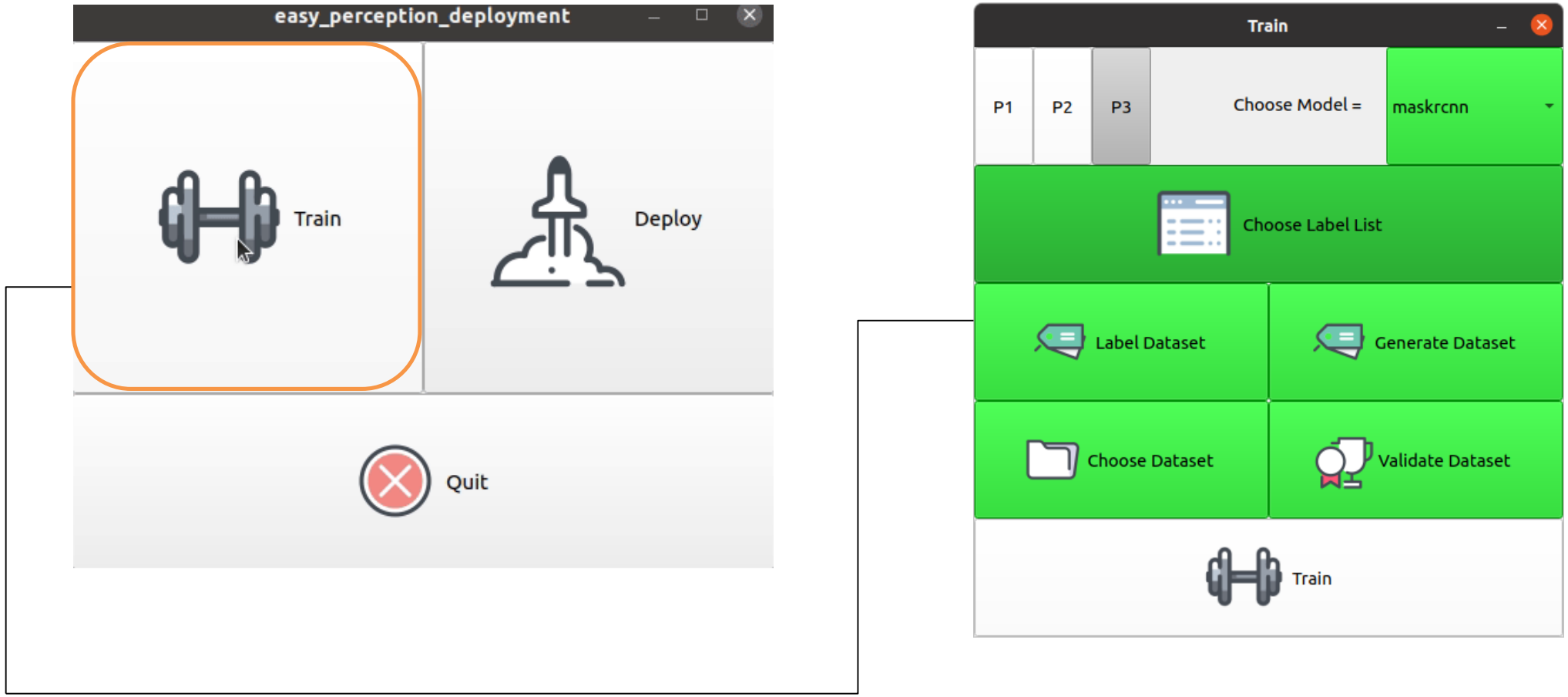



Image courtesy of icons8.com and onnx.com

# 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.

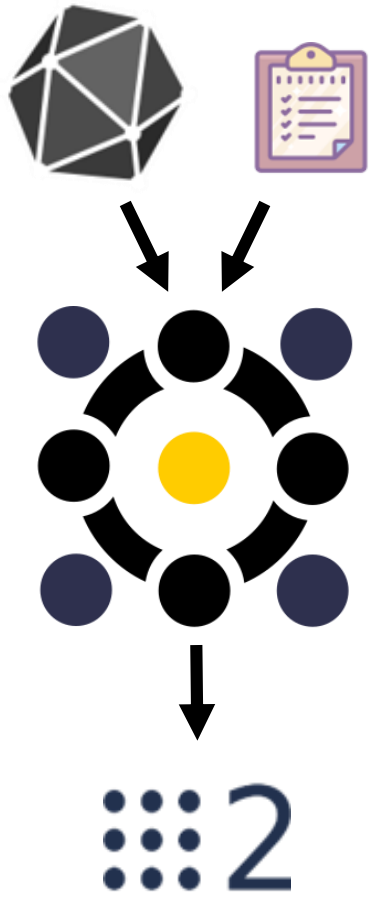


Image courtesy of icons8.com and onnx.com

# What is EPD? [6/6] – Deployment GUI

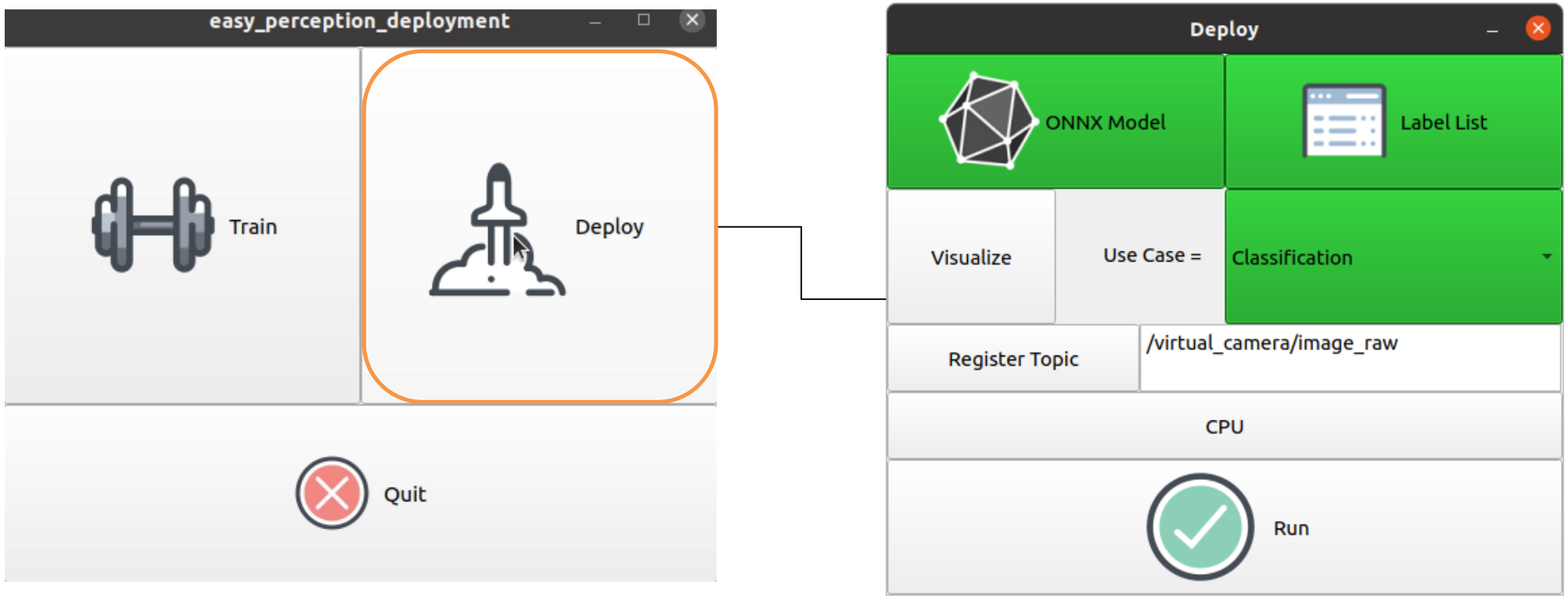



Image courtesy of icons8.com and onnx.com

# 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)

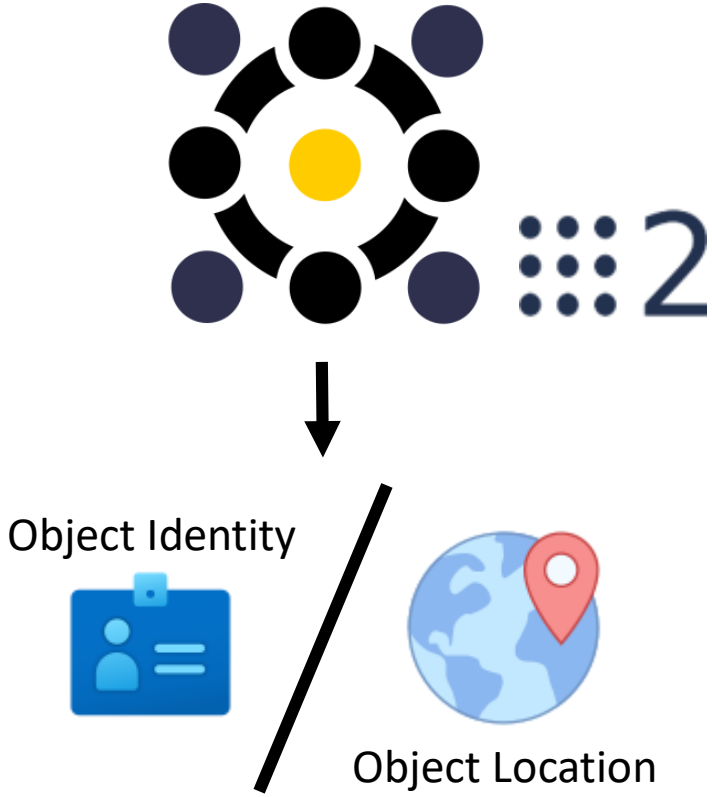



Image courtesy of icons8.com and onnx.com

# 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<br>**Label List**: imagenet_classes.txt | **EPDImageClassification** ROS2 Message Format:<br><br>**string**[] object_names |
| 2 | **Model**: FasterRCNN.onnx<br>**Label List**: coco_classes.txt | **EPDObjectDetection** ROS2 Message Format:<br><br>**uint64**[] class_indices<br>**float64**[] scores<br>**sensor_msgs/RegionOfInterest**[] bboxes<br>**sensor_msgs/Image**[] masks – <Left empty> |
| 3 | **Model**: maskRCNN.onnx<br>**Label List**: coco_classes.txt | **EPDObjectDetection** ROS2 Message Format:<br><br>**uint64**[] class_indices<br>**float64**[] scores<br>**sensor_msgs/RegionOfInterest**[] bboxes<br>**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.*

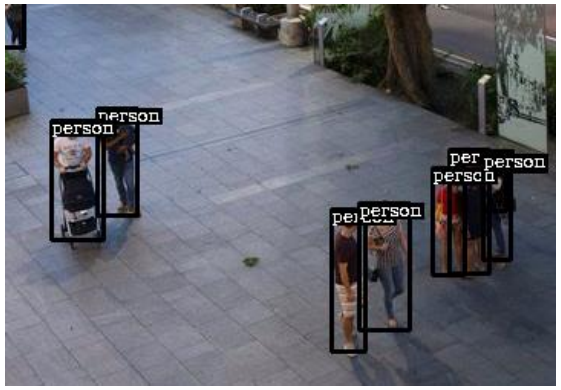
# Customizable Use-Case Configurations - [2/6]

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

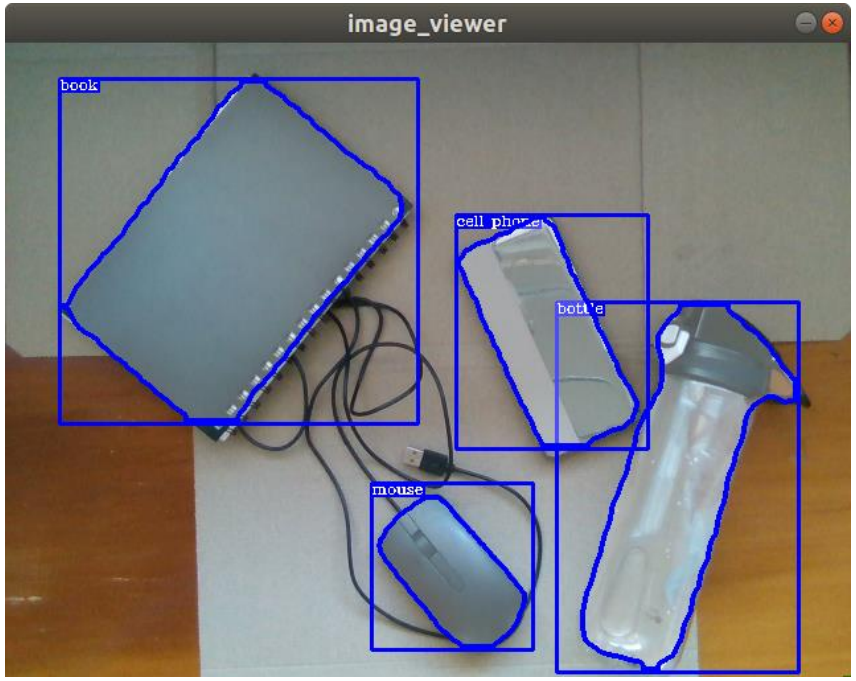
**P1**

```
---
header:
  stamp:
    sec: 0
    nanosec: 0
  frame_id: ''
object_names:
- 'oxygen mask '
---
header:
  stamp:
    sec: 0
    nanosec: 0
  frame_id: ''
object_names:
- 'oxygen mask '
---
```

**P2**

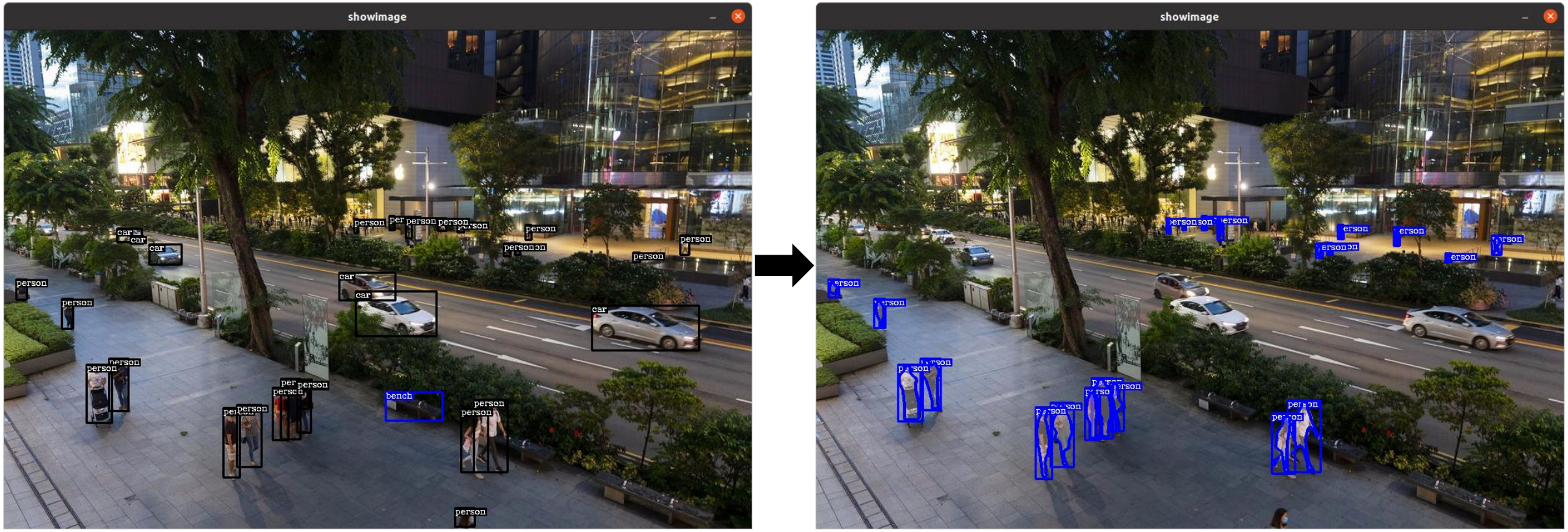



**P3**



# Customizable Use-Case Configurations - [3/6]

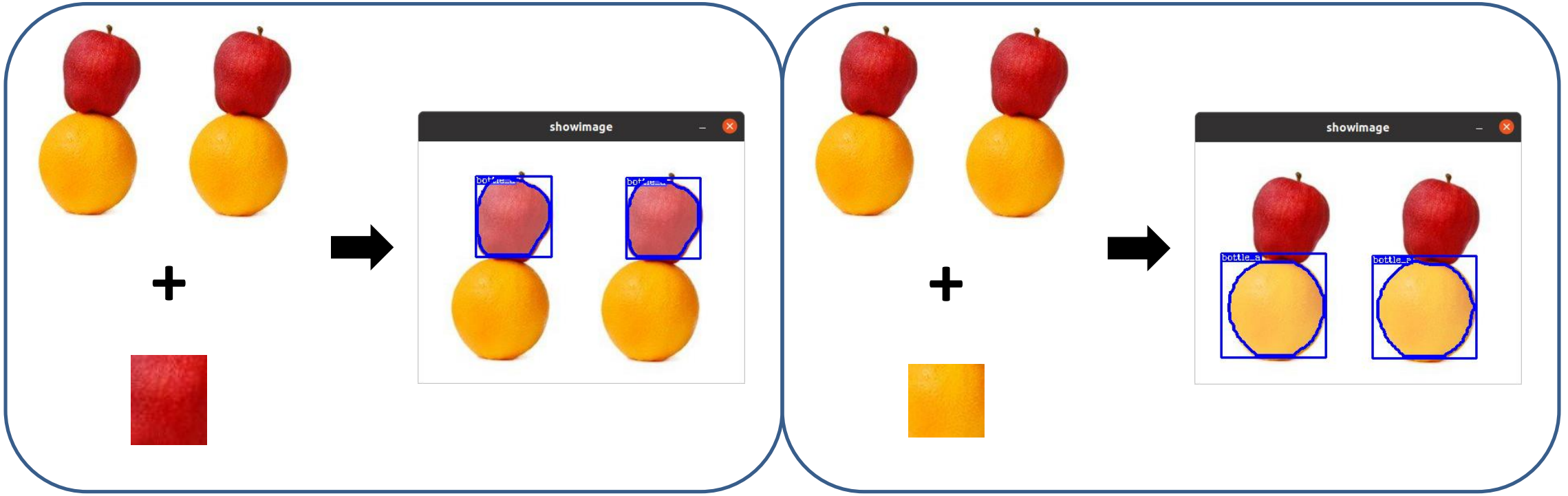
**Counting** (P2, P3) – **EPDObjectDetection**.msg




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

# Customizable Use-Case Configurations - [4/6]

## **Color-Matching** (P2, P3)



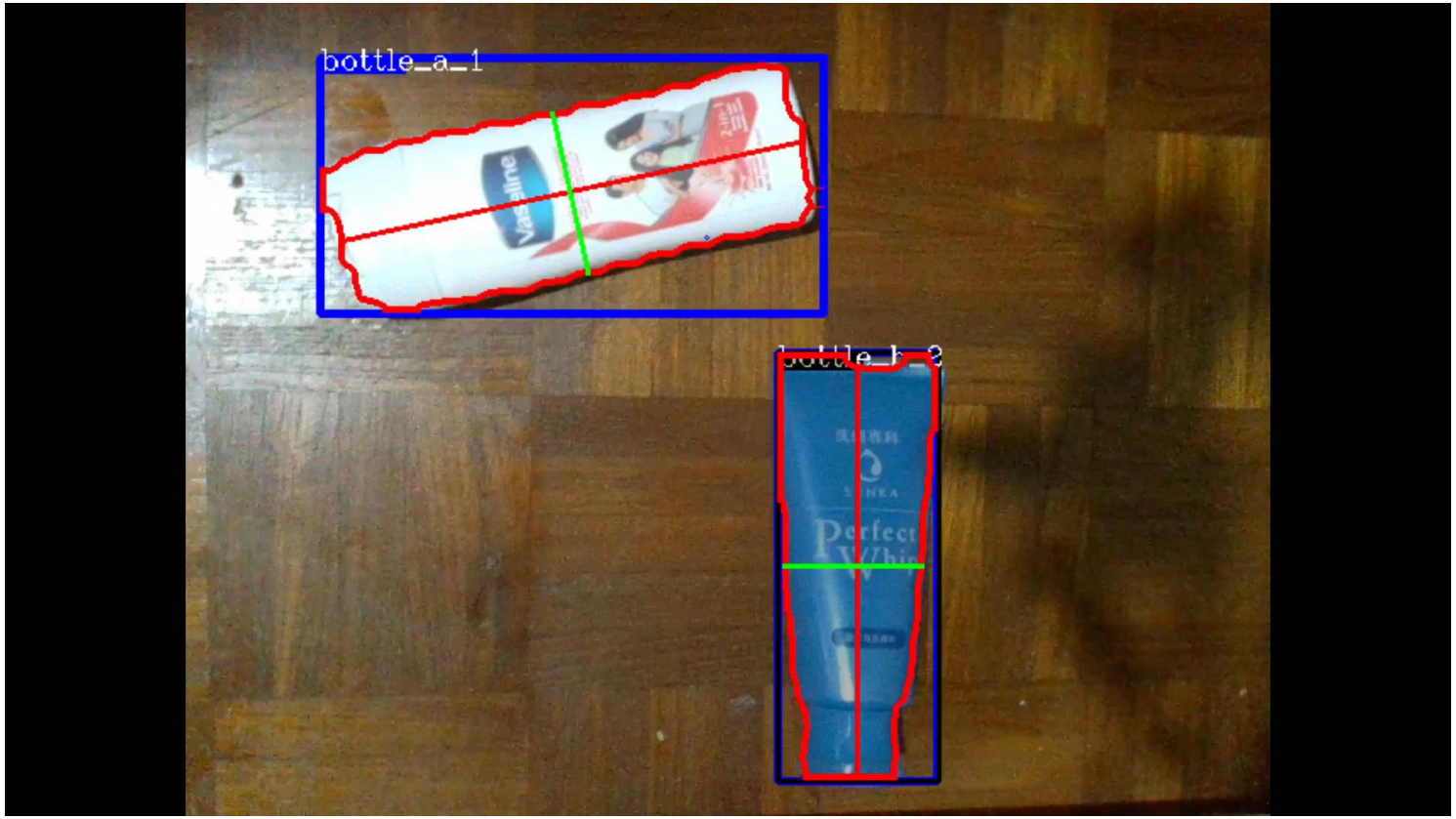
# Customizable Use-Case Configurations - [5/6]

## **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]

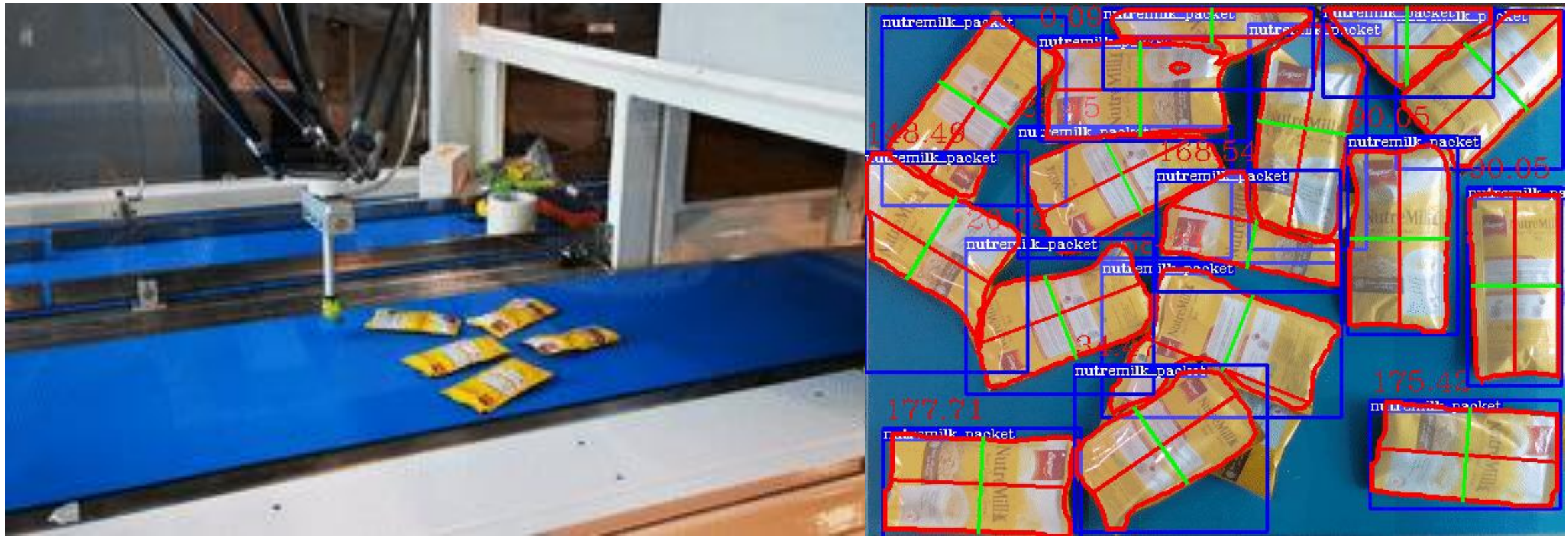
Localization/Measurement/**Tracking** (P3)



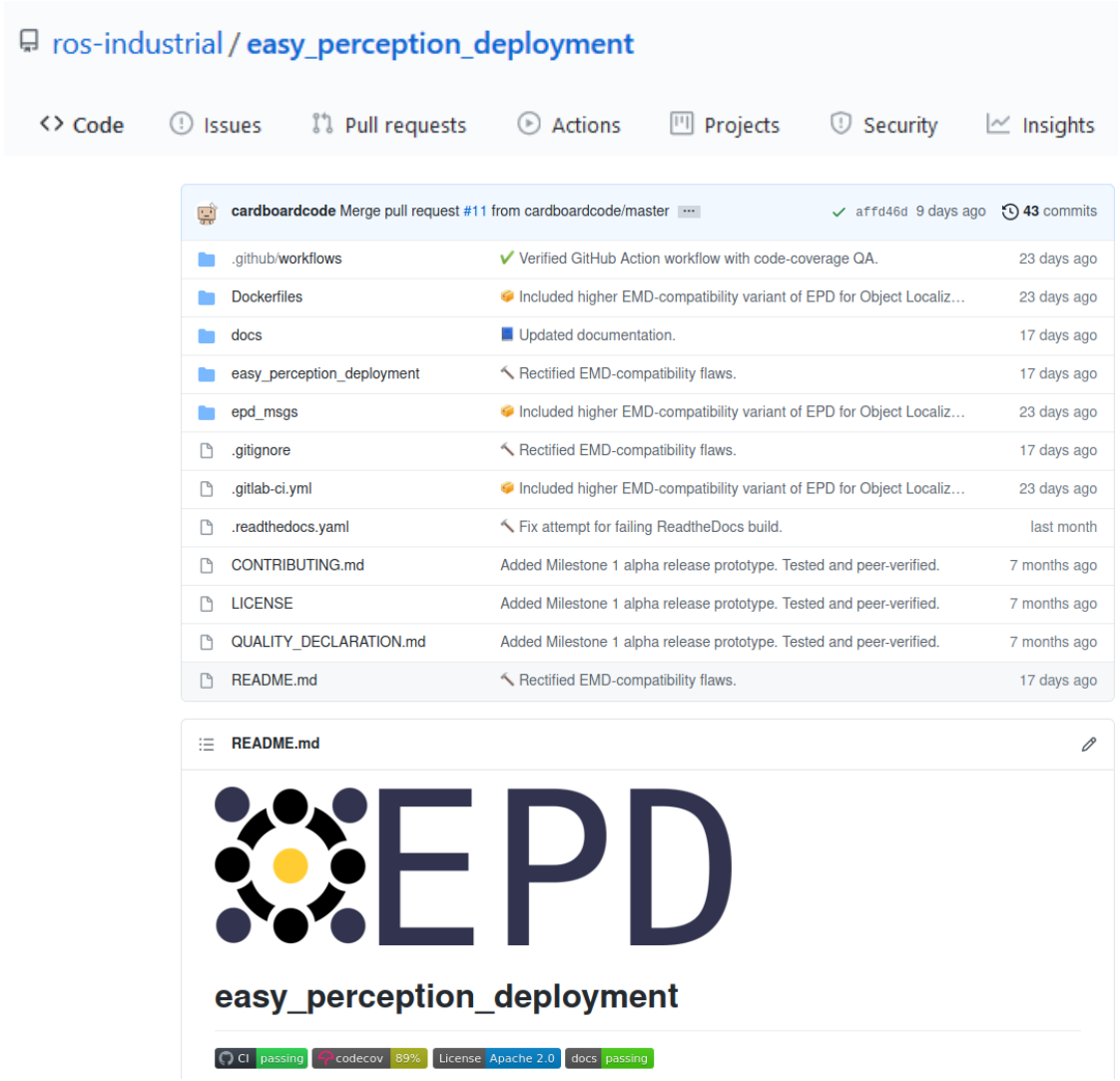
# Tested for Industrial Use

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

**Use Case Description**: Industrial Conveyor Tracking and Automated Picking.



# Get Started with EPD




SCAN ME

# Get In Touch

**Bey Hao Yun (Gary)**
Research Engineer
ARTC, ROS-Industrial Consortium Asia Pacific
**Email**: Bey_Hao_Yun@artc.a-star.edu.sg
**GitHub**: cardboardcode