**Problem**

First of all, we investigated through discussion whether we can implement RL inputs based on combining OBJECTIVE and SEGMENTATION implementations, and we encountered some problems during practice, such as trying to implement the judgment of walls, floors, people and so on. However, the segmentation method is not applicable in the gazebo world because we tested 2 combined OBJECTIVE DETECTION + SEGMENTATION models this time:

* yolo+detectron2
* yolo+self-segmentation

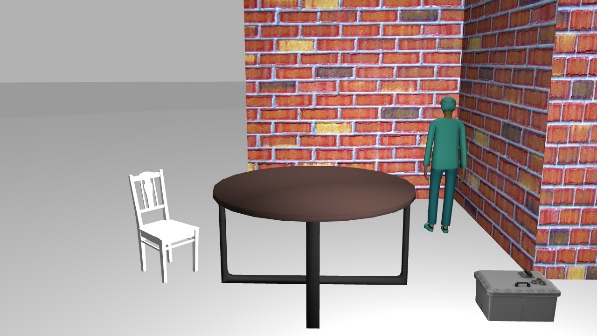


Figure original image

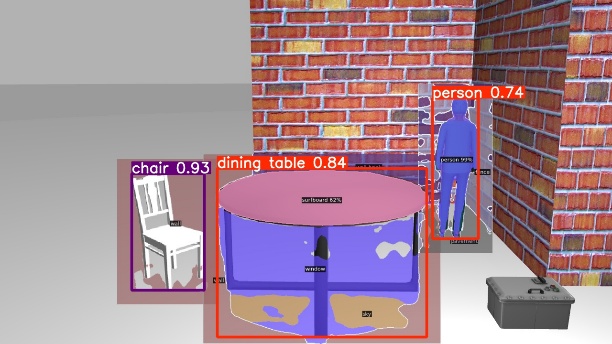


Figure Object detection with Detectron2

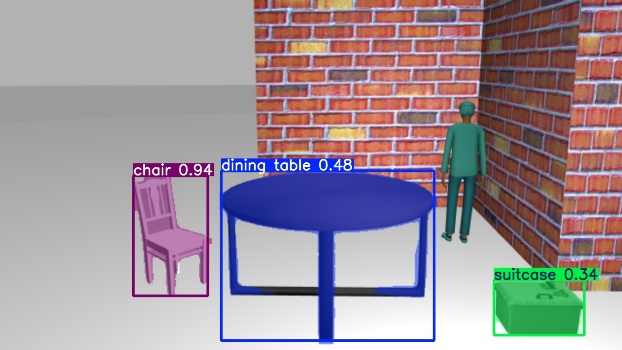
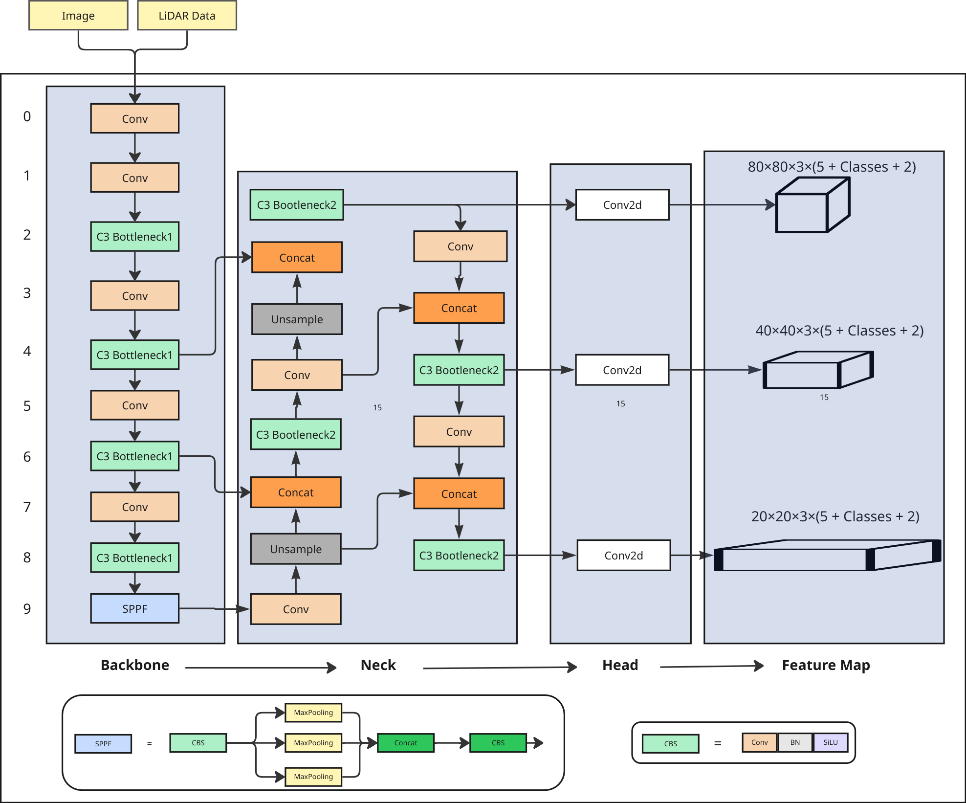


Figure Object detection with YOLO-seg

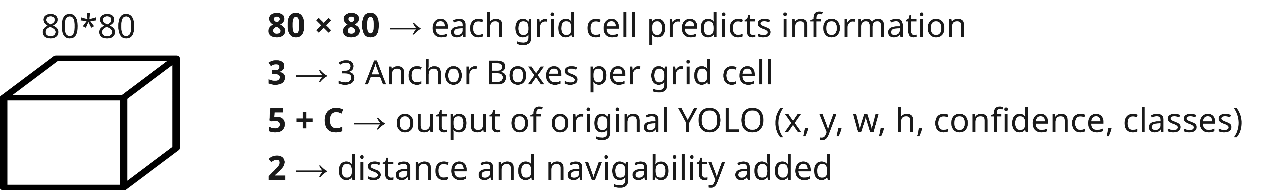
Seeing that detectron2 doesn't work well in gazebo, since detectron2 only has segmentation, I tested using yolo object recognition first, and then using detectron2 segmentation for the object box and peripheral parts, and the results I got weren't as good as with yolo+best possible segmentation. segmentation, and the two models for wall recognition is not good, the dataset is not easy to obtain (can only be achieved with manually labeled images, but this requires a lot of time) that would not be able to carry out the next step in the design of the logic code for obstacle avoidance judgement, but based on the segmentation that comes with the yolo + can be very good recognition of the target of the existing categories, one of our The idea is that we can consider the wall obstacle avoidance using radar to realize, and obstacles by the camera to realize obstacle avoidance.

We design a modified network structure based on YOLO, which is characterized by simultaneous prediction of target class, distance and navigability. Inputs include **RGB images** and **LiDAR data**, key features are extracted by Backbone, and multi-scale information is integrated by Neck.



**The entire feature network is divided into four main parts:**

* **Backbone** is responsible for extracting features from the input images and LiDAR data.
  + **Conv**: Extract basic image features, such as edges, texture.
  + **C3 Bottleneck**: optimize feature expression through residual connection to improve detection effect.
  + **SPPF:** increase the sensory field to enhance the network's ability to adapt to targets at different scales.
* **Neck** performs multi-scale feature fusion in order to detect targets of different sizes.
  + **Upsample**: Increases the resolution of the feature map for better detection of small targets.
  + **Concat** (stitching): fuses features from different layers to enhance multi-scale information.
  + **C3 Bottleneck2** (residual module): optimizes the fused features again to improve detection.
* **Head** detects small, medium and large targets by 80×80, 40×40 and 20×20 feature maps and predicts x, y, w, h, confidence, class, distance, navigability, and so on.
* **Post-processing** parses the feature maps, calculates the optimal target distance, and determines whether the environment is passable based on the LiDAR data.



Finally, the Head generates 80×80, 40×40, 20×20 feature maps, and each grid cell predicts 3 Anchor Boxes, which are dedicated to detecting targets of different sizes. The number of output channels is expanded from (5 + C) in YOLO to (5 + C + 2), with the addition of target distance and navigability. In the post-processing stage, distance (based on monocular camera, LiDAR, depth camera to calculate the optimal distance) and navigability (based on LiDAR to determine whether the path is feasible or not) are resolved to generate the target detection information available to the robot. This enables both object recognition and environment navigation information, providing usable input for reinforcement learning (RL).

