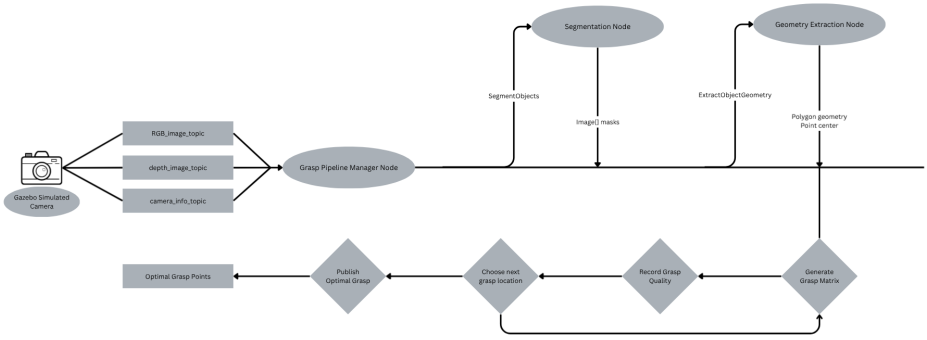
RBE 4540 — Group Assignment: Top Surface Grasping

Brent Weiffenbach, Isabella Rosenstein, Evan Carmody

October 9, 2025

The goal of this project is to identfy the top surface of any number of objects, and estimate the shape of the surface as a 2D polygon. The polygon will be used to plan a grasp with the best quality metrics to pick up the object.



Step 1: Enviornment Setup

This project contains 5 ROS packages. A launch file at the root of the **TopSurface-Grasping** folder which is not part of a specific ROS2 package launches all the necessary nodes.

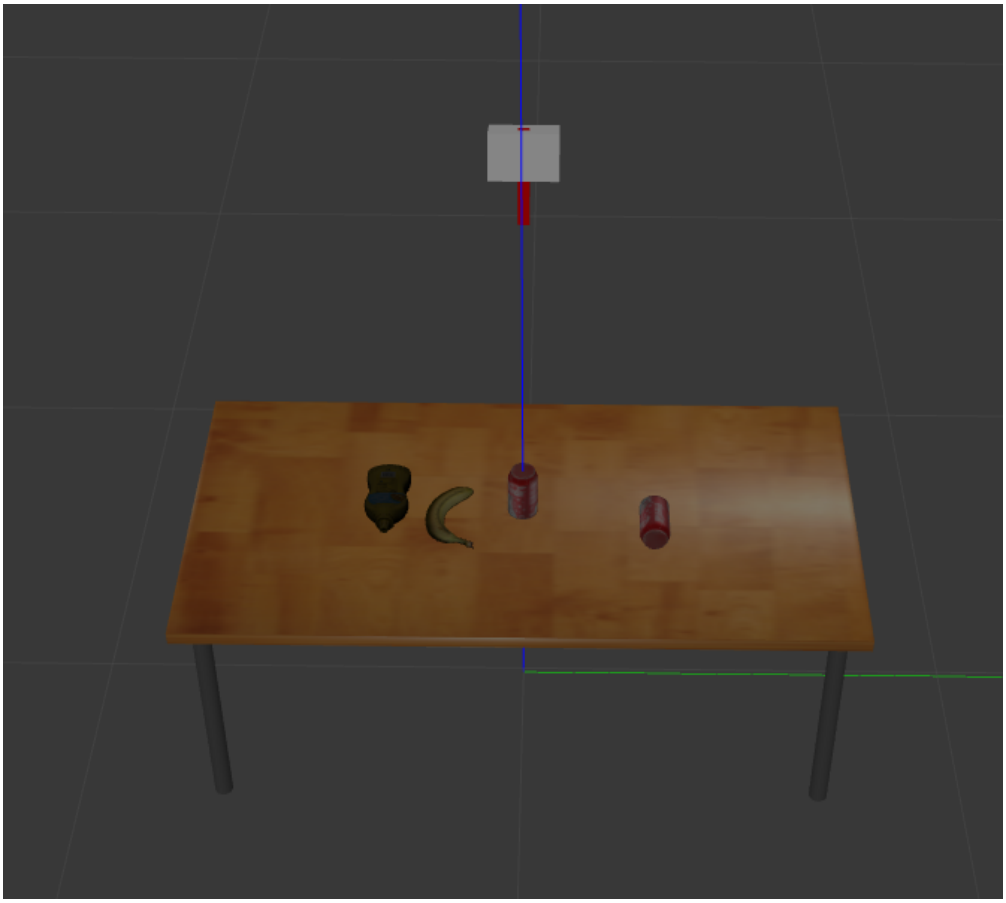
First we open a gazebo world with a table, 2 coke cans, banana, and mustard bottle in it. Then we spawn in the camera above the table looking at an angle towards the objects. To keep a global world frame we can use to identify what the 'top' of surfaces is, we use a static transform publisher to publish a transform from the camera frame to the world frame.

```python
def camera_to_world(x, y, z, roll, pitch, yaw):
    cr = math.cos(roll)
    sr = math.sin(roll)
    cp = math.cos(pitch)
    sp = math.sin(pitch)
    cy = math.cos(yaw)
    sy = math.sin(yaw)

    R = [
        [cy * cp, cy * sp * sr - sy * cr, cy * sp * cr + sy * sr],
        [sy * cp, sy * sp * sr + cy * cr, sy * sp * cr - cy * sr],
        [-sp,     cp * sr,                cp * cr]
    ]

    inv_R = [
        [R[0][0], R[1][0], R[2][0]],
        [R[0][1], R[1][1], R[2][1]],
        [R[0][2], R[1][2], R[2][2]],
    ]
    orig_x = inv_R[0][0]*x + inv_R[0][1]*y + inv_R[0][2]*z
    orig_y = inv_R[1][0]*x + inv_R[1][1]*y + inv_R[1][2]*z
    orig_z = inv_R[2][0]*x + inv_R[2][1]*y + inv_R[2][2]*z
    world_z = -orig_x
    world_y = orig_z
    world_x = orig_y

    return world_x, world_y, world_z

def rpy_to_matrix(roll, pitch, yaw):
    cr = math.cos(roll)
    sr = math.sin(roll)
    cp = math.cos(pitch)
    sp = math.sin(pitch)
    cy = math.cos(yaw)
    sy = math.sin(yaw)
    # R = Rz(yaw) * Ry(pitch) * Rx(roll)
    R = [
        [cy*cp, cy*sp*sr - sy*cr, cy*sp*cr + sy*sr],
        [sy*cp, sy*sp*sr + cy*cr, sy*sp*cr - cy*sr],
        [-sp,   cp*sr,            cp*cr]
    ]
    return R
```

We use these helper functions in the launch file to convert from camera coordinates to world coordinates, and keep the world frame aligned with the table surface.

With that we have a gazebo world with objects in it, and a camera looking at the objects, where the camera publishes RGB and depth information to ROS topics.

Finally, we have a grasp pipeline manager node which will manage the flow of data and publish debug information to topics we can view in rviz. The pipeline manager will call services on the segmentation and geometry extraction nodes.

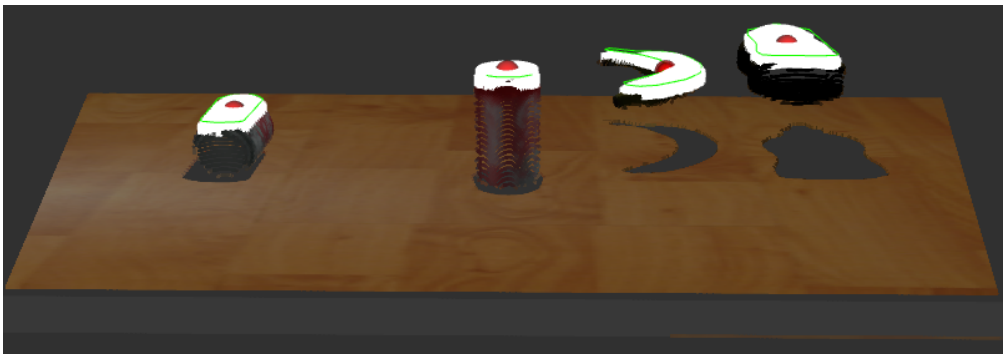


Step 2: Image Processing

Given an RGB image, we need to detect the objects in the image, and create a mask for each object. The `SegmentationNode` has a service `segment_objects` which receives an RGB image from the pipeline manager, and returns a list of masks for each detected object. It processes the image by masking out background regions like the table and sky to isolate the objects. Then contours are found in the masked image, and if the area is large enough, a mask is created for the object.

Step 3: Geometry Extraction

The `GeometryExtractionNode` has a service `extract_object_geometry` which receives a list of masks and a point cloud and returns a polygon representing the top surface and the center of mass projected onto the top surface. The node first masks the point cloud to isolate only relevant points of the object, then selects all the points with the minimum y-value (world coordinate system makes this the top surface), and projects these points into a 2D plane. Then using OpenCV contour detection, the top surface polygon is found. The center of mass is calculated by averaging the x and z coordinates of the top surface point cloud points, and using the minimum y-value for the y coordinate.



Step 4: Finding a Grasp

Once we have a polygon and center of mass, we can preform similar steps to previous homework assignments to find a grasp. We generate a list of grasp candiates on the object, and evaluate each candidate using grasp quality metrics. We find the minimum singular value, isotropy, and volume of ellipsoid metrics for each candidate, and select the candidate with the best metrics (if it has 2 metrics that are the best). Then these contact points can be visualized as markers in rviz.

Discussion

We had resulting grasps with the following metrics:

Object 1 (Standing Coke Can): Object 2 (Lying Down Coke Can): Object 3 (Banana): Object 4 (Mustard Bottle):

As seen in the images, the grasps are reasonably placed and could sufficently pick up the objects.