Rheinische Friedrich-Wilhelms-Universität Bonn
Prof. Dr. Maren Bennewitz

Institut für Informatik
Abteilung VI
Humanoid Robots Lab
Adresse:
Friedrich-Hirzebruch-Allee 8
53115 Bonn

Humanoid Robotics

Assignment 2 (Ungraded)

Discussion in Tutorial on 05.05.2026.

3D World Representations:

1. **Task Oriented Mapping Methods**
   Roboticists at the University of Bonn are developing several robots for different applications:
   - **UBoButler**: A humanoid robot tasked with sorting objects on a kitchen counter. The robot must identify and manipulate items like 'cups', 'plates', and 'utensils' and place them in designated locations. It does not wander out of the kitchen.
   - **UBoWanderDog**: UBoWanderDog is a quadruped navigating a hilly terrain with obstacles such as 'rocks', 'trees', and 'bushes'. The robot's primary task is to explore the area and map potential paths.
   - **UBoAssist**: UBoAssist is a humanoid assisting elderly and visually challenged people for navigation in dense urban areas with pedestrians, cyclists, other vehicles, traffic lights, and crosswalks.

   These robots rely on accurate 3D perception to function effectively. Their perception systems process visual data to create 3D maps of their environments. The team is focused on using pixel-level classification from deep learning models to label elements in the scene. To understand the challenges involved, consider the following computational cost considerations:
   - **Panoptic Segmentation**: Highest computational cost.
   - **Semantic Segmentation**: Provides pixel-level classification of object categories
   - **Instance Segmentation**: Similar to Semantic Segmentation but requires additional computation for tracking individual instances.

   Determine the most suitable scene understanding approach for the mapping for each of the three scenarios (Kitchen, Hills, and Downtown Area). Justify your choice for each scenario. Note, do not assume a one-one relation between mapping methods and scenarios

   (3 points)

2. **Semantic Point Cloud Creation and Stitching**
   In this task you are provided with a set of depth images, semantic segmentation masks, camera poses, and intrinsics. Each depth image should be converted to a camera frame point cloud, colored using the semantic mask and a provided CSV file that maps 41 semantic classes (including background) to RGB colors, and then transformed into world coordinates using the camera pose. Finally, the per-image point clouds should be stitched together and visualized as one global colored point cloud.

   a. Convert each depth image into a 3D point cloud using the camera intrinsics.

   (1 point)

   b. Modify above function to color the point cloud. For each depth, use the corresponding semantic segmentation mask and the provided CSV file mapping semantic classes to colors, to assign an RGB color to each point. Use nearest point interpolation for aligning depth image with segmentation image. Assume segmentation image has same instrinsics as color image. Assume their optical centers are co-incident.

   (2 points)

c. Transform the colored point cloud into world coordinates using the provided camera pose.
(1 point)

d. Stitch the transformed point clouds from all images into a single global point cloud and visualize it using a 3D visualization tool such as Open3D.
(1 point)

3. **2D TSDF Grid Generation, Weighted Update and Occupancy Computation**

In this task you are provided with a defined parabola given by the equation $y = 4x^2$, that represents a curved surface in the first quadrant. The scene is defined over a 1m × 1m area with a user-defined grid cell size i.e. your method should be general to the size. You are also provided with multiple camera poses that indicate the camera's position and orientation in the scene. Your goal is to generate a 2D Truncated Signed Distance Function (TSDF) grid by performing raycasting from the camera pose(s) into the scene. The TSDF should be computed as follows: along each ray cast from the camera, determine the intersection with the surface defined by $y = 4x^2$ (the measured surface). For a given grid cell along that ray:

- If the cell lies between the camera and the measured surface, assign a positive TSDF value.
- If the cell is beyond the measured surface, assign a negative TSDF.
- Use TSDF updating principles in the lecture including weight drop-off beyond -0.1m.

You can refer to state-of-the-art TSDF weight and distance calculation and update methods in the repository https://github.com/ethz-asl/voxblox.

a. Generate the grid covering the 1m × 1m domain with user defined resolution (Use 0.1m for initial testing). Initialize a maps with two values for the grid: TSDF distance and the weight. Also perform suitable initialization of the weights and distances.
(1 point)

b. Using the provided camera pose(s), perform raycasting into the scene. You are given the helper function *get_intersection_point()*. Compute the TSDF value based on the distance from the cell to the measured surface along that ray using the following convention: cells between the camera and the surface have positive TSDF values, and cells beyond the surface have negative TSDF values. Update the cell's weight only if its absolute TSDF value is within 0.1m. Document your raycasting method and detail how you use the *get_intersection_point()* function to determine the surface intersection.
(2 points)

c. Once the TSDF weights and distances are updated for all poses, generate an occupancy map from the two maps.
(2 point)

d. Visualize the TSDF distance, weights and the occupancy maps for two different gride cell sizes 0.1m and 0.01m.
(1 point)

e. Explain in text the trade-off between occupancy and TSDF maps on one hand and different grid sizes on the other based on the visualizations generated
(1 point)