
PSEUDO CODE: DepthToMap click-to-distance pipeline

IMPORT ROS2, OpenCV, NumPy, threading, message types

CLASS: DepthToMap Node

INIT FUNCTION

- Initialize ROS2 Node named 'depth_to_map_node'
- Create CvBridge for ROS ↔ OpenCV conversions
- Initialize camera intrinsic matrix K as None
- Create a thread lock for shared resources
- Define topic names for depth image, RGB image, camera info
- Initialize variables for depth image, RGB image, clicked point, and shutdown flag
- Initialize a placeholder for display image
- Create threading event for GUI thread control
- Start GUI thread to show images and handle clicks
- Initialize TurtleBot4 navigator
    - If robot is undocked:
        - Log info about docking first
        - Dock the robot to set initial pose
    - Undock robot to be ready for navigation
- Flags for logging shapes only once
- Create ROS subscriptions:
    - CameraInfo → camera_info_callback
    - Depth Image → depth_callback
    - RGB Image (compressed) → rgb_callback
- Start periodic timer every 0.2s → display_images function

CALLBACK: camera_info_callback(msg)

- Lock shared state
- Extract camera intrinsic matrix K from incoming message
- If not already logged:
    - Log fx, fy, cx, cy

CALLBACK: depth_callback(msg)

- Try to convert ROS depth image to OpenCV format
- If empty → Log error
- If shape not yet logged → Log shape
- Lock and store depth image

CALLBACK: rgb_callback(msg)

- Try to decode compressed RGB image to OpenCV format
- If empty → Log error
- If shape not yet logged → Log shape
- Lock and store RGB image

CALLBACK: mouse_callback(event, x, y)

- If left mouse click:
  - Lock and store clicked pixel coordinate (x,y)
  - Log the clicked pixel

FUNCTION: display_images()

- Lock and copy current RGB, depth, and clicked point
- If both RGB and depth exist:
  - Copy RGB and depth images for local use
  - Normalize depth to [0,255] range and apply a color map
  - If a click exists:
    - Extract (x,y)
    - Check bounds for RGB and depth
    - Get depth z-value at (x,y) in meters
    - If valid (0.2–5.0m):
      - Annotate RGB with crosshair and circle at click
      - Annotate depth with distance text
  - Horizontally stack RGB and depth display images
  - Lock and store combined display image

THREAD: gui_loop()

- Create OpenCV window for display
- Set mouse callback for window
- Loop until GUI stop signal set:
    - Lock and copy display image
    - If image exists:
        - Show image in window
        - If 'q' pressed:
            - Log shutdown request
            - Dock the robot for safety
            - Set shutdown flag and GUI stop event
            - Shutdown ROS2
    - Else:
        - Wait for short time

MAIN FUNCTION

- Initialize ROS2
- Create DepthToMap node
- Create multi-threaded executor
- Add node to executor
- Spin executor to process callbacks and timers
- On KeyboardInterrupt:
    - Signal GUI thread to stop
    - Join GUI thread
    - Destroy node and close OpenCV windows

NODE: DepthToMap

INPUT TOPICS

- /oakd/rgb/camera_info → CameraInfo → 3×3 intrinsics matrix K
- /oakd/stereo/image_raw → Image → depth image (CV 16-bit)
- /oakd/rgb/image_raw/compressed → CompressedImage → RGB image (JPEG)

OUTPUT

- On-screen GUI window: side-by-side RGB and depth image, with overlayed distance for clicked pixel
- Logs:
    - Intrinsics (fx, fy, cx, cy)
    - Image shapes
    - Clicked pixel coordinates
    - Distance at clicked point (meters)

MOUSE INPUT

- User clicks on RGB image → (x,y) pixel coordinates stored

PROCESSING

- For each timer cycle:
    - Normalize depth to 0–255 range
    - Colorize depth for display
    - Extract depth z at (x,y) → convert mm → meters
    - If z valid (0.2–5m), overlay distance text & marker

GUI OUTPUT

- OpenCV window: 'RGB (left) | Depth (right)'
- Shows combined RGB + depth image
- Markers for clicked pixel, with distance annotation

SHUTDOWN

- Press q → dock robot → shutdown ROS → close windows

DATA FLOW

```
Camera → ROS Topics
  /oakd/rgb/camera_info --> K
  /oakd/stereo/image_raw --> depth
  /oakd/rgb/image_raw/compressed --> RGB

Mouse Click --> (x,y)
```

```
Processing:
  RGB + depth --> overlay --> GUI Window
  (x,y) + depth[y,x] --> z (meters)

Output:
  Visual display + log distance
```