AUTONOMOUS DISASTER RESPONSE & RECONNAISSANCE BOT

COURSE PROJECT FOR:

EECE 5550 - Mobile Robotics
Spring 2023 Semester

INSTRUCTOR:

Prof. Michael Everett

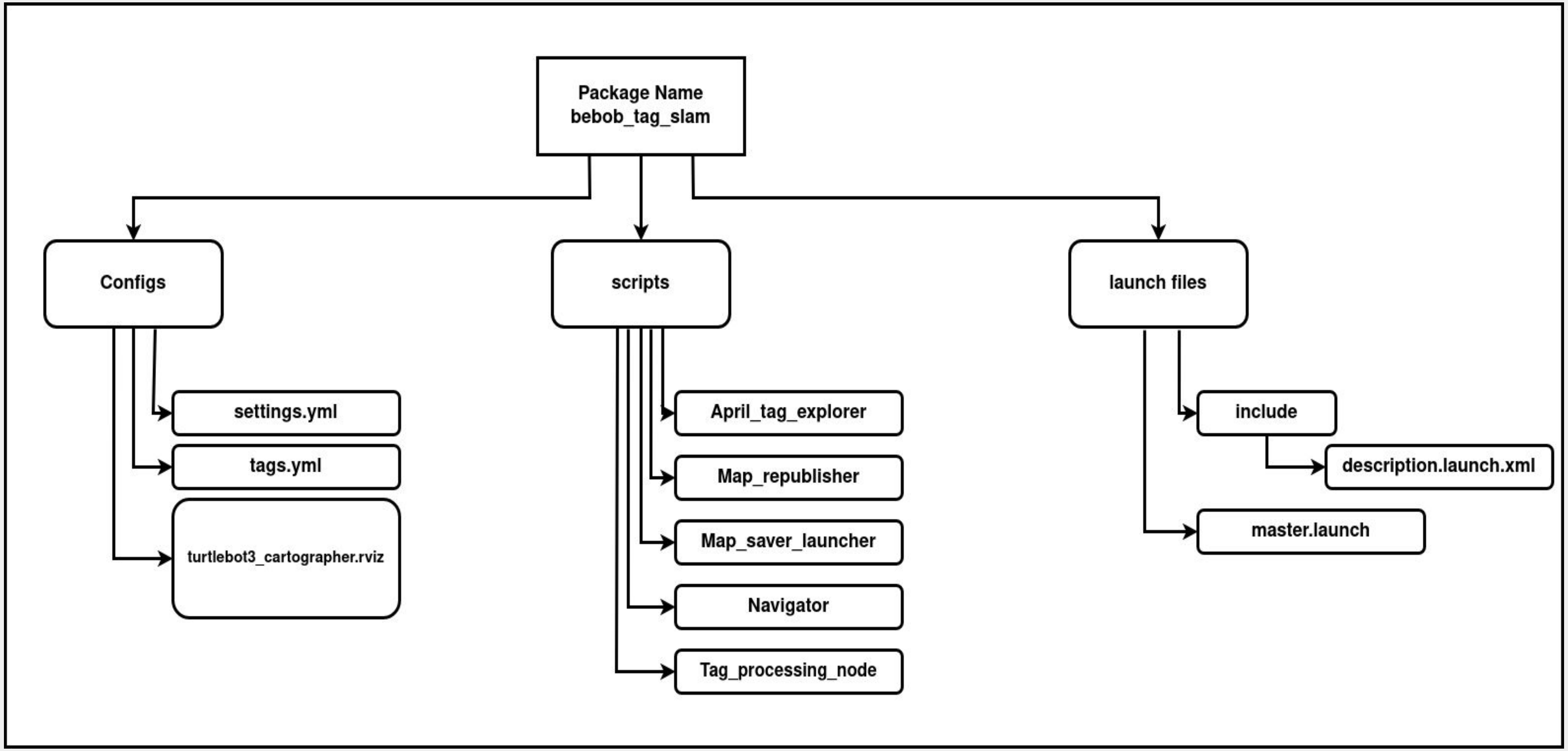
BOT USED:

TurtleBot 3 ‘BEBOP’

TEAM MEMBERS:

Deep Kotadiya
Vishnu Rohit Annadanam
Hussain Kanchwala
Nikhil Chowdary Gutlapalli
Aakash Singhal
Satvik Tyagi

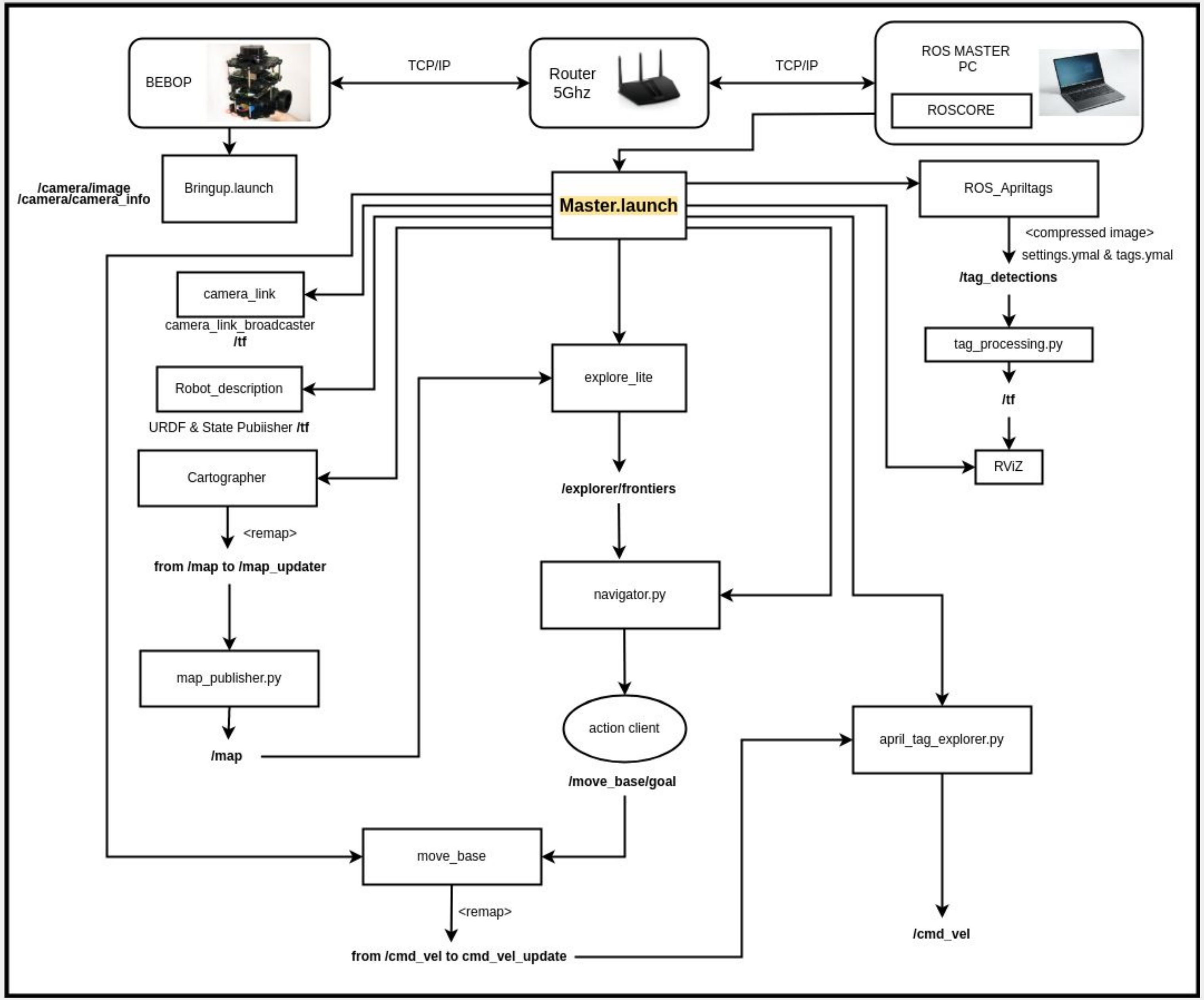
ROS Package Architecture



Package Name
bebob_tag_slam

- Configs
  - settings.yml
  - tags.yml
  - turtlebot3_cartographer.rviz
- scripts
  - April_tag_explorer
  - Map_republisher
  - Map_saver_launcher
  - Navigator
  - Tag_processing_node
- launch files
  - include
    - description.launch.xml
  - master.launch

High level Flowchart



BEBOP

TCP/IP

Router 5Ghz

TCP/IP

ROS MASTER PC

ROSCORE

/camera/image
/camera/camera_info

Bringup.launch

Master.launch

ROS_Apriltags

<compressed image>
settings.ymal & tags.ymal

/tag_detections

tag_processing.py

/tf

RViZ

camera_link

camera_link_broadcaster
/tf

Robot_description

URDF & State Pubiisher /tf

Cartographer

<remap>
from /map to /map_updater

map_publisher.py

/map

explore_lite

/explorer/frontiers

navigator.py

action client

/move_base/goal

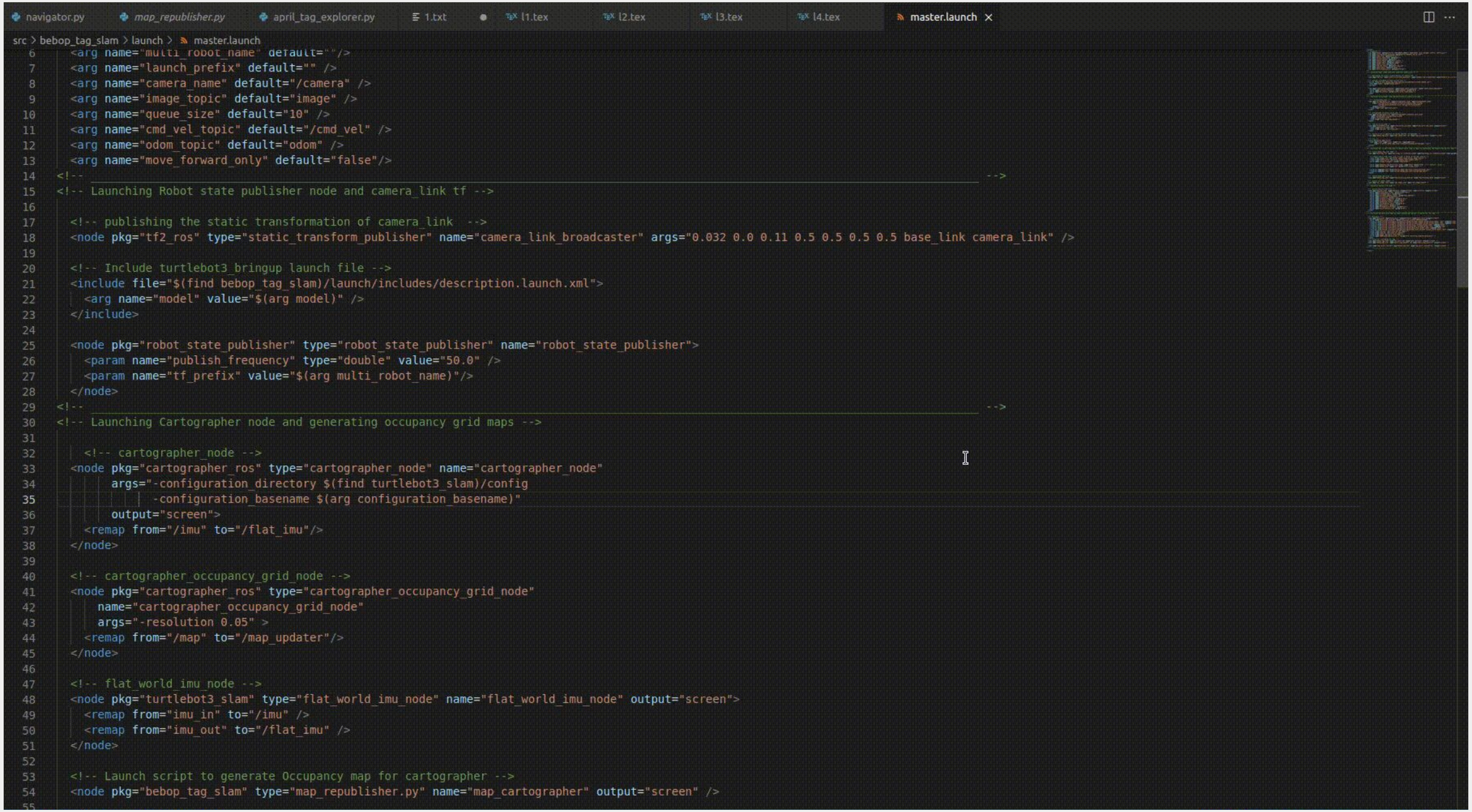
move_base

<remap>
from /cmd_vel to cmd_vel_update

april_tag_explorer.py

/cmd_vel

master.launch



```xml
    <arg name="multi_robot_name" default=""/>
    <arg name="launch_prefix" default="" />
    <arg name="camera_name" default="/camera" />
    <arg name="image_topic" default="image" />
    <arg name="queue_size" default="10" />
    <arg name="cmd_vel_topic" default="/cmd_vel" />
    <arg name="odom_topic" default="odom" />
    <arg name="move_forward_only" default="false"/>
  <!-- __________________________________________________________________________________ -->
  <!-- Launching Robot state publisher node and camera_link tf -->

    <!-- publishing the static transformation of camera_link  -->
    <node pkg="tf2_ros" type="static_transform_publisher" name="camera_link_broadcaster" args="0.032 0.0 0.11 0.5 0.5 0.5 0.5 base_link camera_link" />

    <!-- Include turtlebot3_bringup launch file -->
    <include file="$(find bebop_tag_slam)/launch/includes/description.launch.xml">
      <arg name="model" value="$(arg model)" />
    </include>

    <node pkg="robot_state_publisher" type="robot_state_publisher" name="robot_state_publisher">
      <param name="publish_frequency" type="double" value="50.0" />
      <param name="tf_prefix" value="$(arg multi_robot_name)"/>
    </node>
  <!-- __________________________________________________________________________________ -->
  <!-- Launching Cartographer node and generating occupancy grid maps -->

      <!-- cartographer_node -->
    <node pkg="cartographer_ros" type="cartographer_node" name="cartographer_node"
          args="-configuration_directory $(find turtlebot3_slam)/config
                -configuration_basename $(arg configuration_basename)"
          output="screen">
      <remap from="/imu" to="/flat_imu"/>
    </node>

    <!-- cartographer_occupancy_grid_node -->
    <node pkg="cartographer_ros" type="cartographer_occupancy_grid_node"
        name="cartographer_occupancy_grid_node"
        args="-resolution 0.05" >
      <remap from="/map" to="/map_updater"/>
    </node>

    <!-- flat_world_imu_node -->
    <node pkg="turtlebot3_slam" type="flat_world_imu_node" name="flat_world_imu_node" output="screen">
      <remap from="imu_in" to="/imu" />
      <remap from="imu_out" to="/flat_imu" />
    </node>

    <!-- Launch script to generate Occupancy map for cartographer -->
    <node pkg="bebop_tag_slam" type="map_republisher.py" name="map_cartographer" output="screen" />
```

Cartographer vs Gmapping

- Preferred system for mapping and localization.

- Subscribes to:
  - Laser Scan (/scan)
  - IMU data (/imu)
  - Odometry (/odom)

- Publishes:
  - Occupancy grid map with cell values in probability range: 0 - 100. (/map)

- Cartographer uses a global optimization approach to SLAM.
- The entire trajectory of the robot is corrected and optimized based on IMU data as well.

- GMapping, on the other hand, uses a local optimization approach to SLAM.
- Only the most recent sensor data is used to update the map and estimate the robot's position in GMapping

Exploration and Navigation

- Used explore_lite for exploration and move_base (from ROS navigation stack) for robot navigation.

- explore_lite:
  - Subscribes to: /map (occupancy grid map)
  - Publishes: /explorer/frontiers
  - To action client: move_base (provides goal position /move_base/goal)

- move_base:
  - Subscribes to: goal position (/move_base/goal)
  - Publishes: /cmd_vel

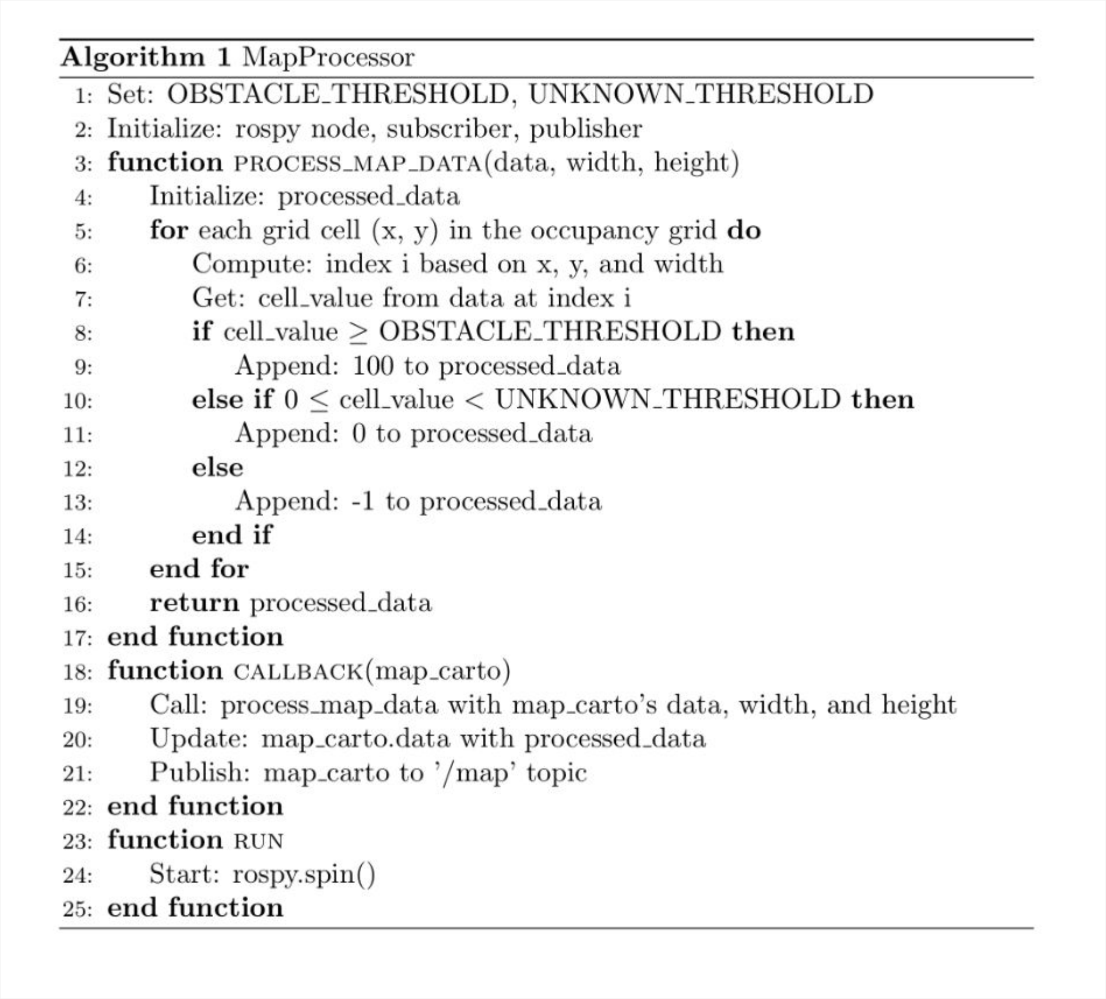
Problem:

explore_lite expects occupancy grid map with cell values [100,0,-1],

while Cartographer publishes an occupancy grid map with cell values in probability range: 0 - 100.

Solution:

Wrote a custom node that converts Cartographer’s map and publishes back to /map..



Algorithm 1 MapProcessor

```
Set: OBSTACLE_THRESHOLD, UNKNOWN_THRESHOLD
Initialize: rospy node, subscriber, publisher
function PROCESS_MAP_DATA(data, width, height)
    Initialize: processed_data
    for each grid cell (x, y) in the occupancy grid do
        Compute: index i based on x, y, and width
        Get: cell_value from data at index i
        if cell_value $\geq$ OBSTACLE_THRESHOLD then
            Append: 100 to processed_data
        else if $0 \leq$ cell_value $<$ UNKNOWN_THRESHOLD then
            Append: 0 to processed_data
        else
            Append: -1 to processed_data
        end if
    end for
    return processed_data
end function
function CALLBACK(map_carto)
    Call: process_map_data with map_carto’s data, width, and height
    Update: map_carto.data with processed_data
    Publish: map_carto to ’/map’ topic
end function
function RUN
    Start: rospy.spin()
end function
```

Apriltags Detection & Transformation

- Used Apriltag_ros package to detect apriltags.

- Subscribes to:
  - /camera/image
  - /camera/camera_info

- Publishes:
  - Transforms of Apriltag w.r.t. Camera.
  - Info about the detected tag.

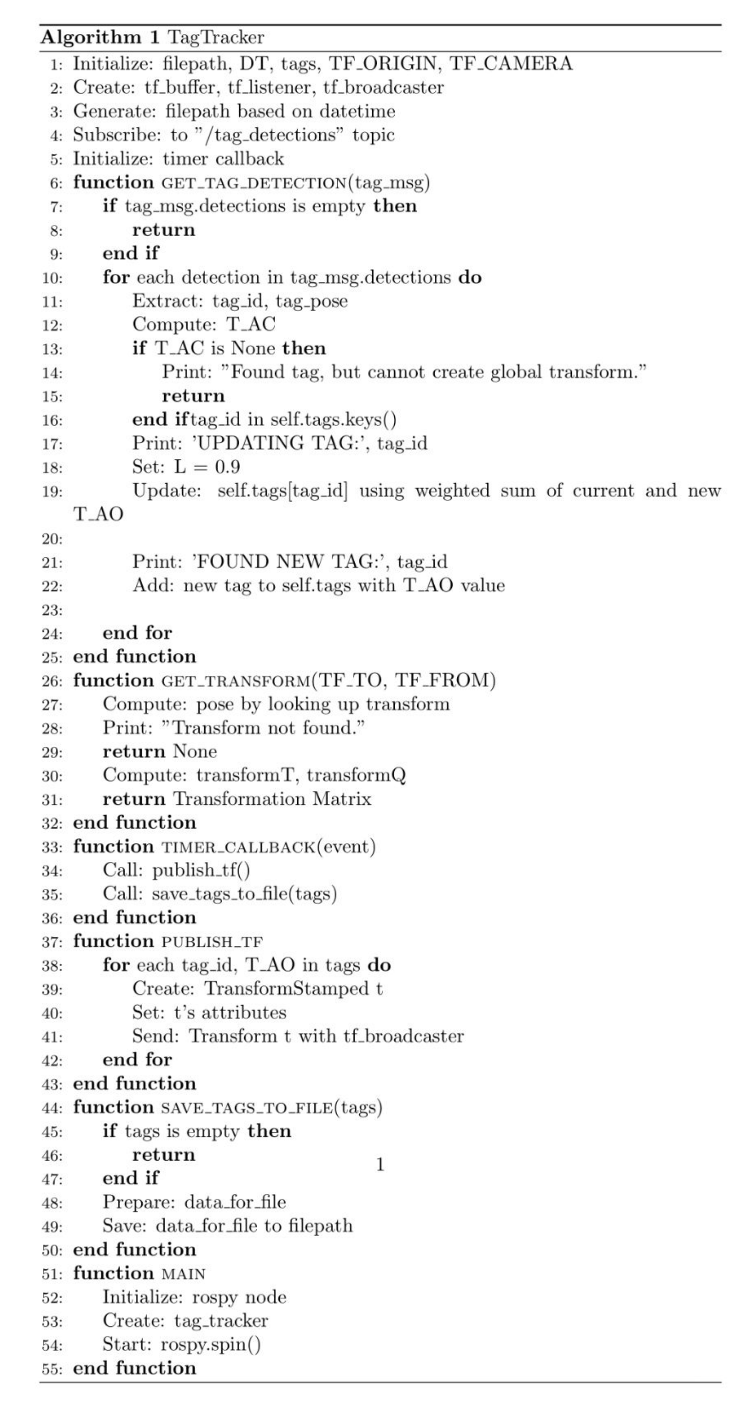
Problem:

/tf topic provides the relative pose b/w camera pose and tag.

But we need position w.r.t. world map frame (origin).

Solution:

Wrote the custom node to convert the position relative to origin frame, using transformation matrix.



Algorithm 1 TagTracker

```
Initialize: filepath, DT, tags, TF_ORIGIN, TF_CAMERA
Create: tf_buffer, tf_listener, tf_broadcaster
Generate: filepath based on datetime
Subscribe: to "/tag_detections" topic
Initialize: timer callback
function GET_TAG_DETECTION(tag_msg)
    if tag_msg.detections is empty then
        return
    end if
    for each detection in tag_msg.detections do
        Extract: tag_id, tag_pose
        Compute: T_AC
        if T_AC is None then
            Print: "Found tag, but cannot create global transform."
            return
        end iftag_id in self.tags.keys()
        Print: 'UPDATING TAG:', tag_id
        Set: L = 0.9
        Update: self.tags[tag_id] using weighted sum of current and new
T_AO

        Print: 'FOUND NEW TAG:', tag_id
        Add: new tag to self.tags with T_AO value

    end for
end function
function GET_TRANSFORM(TF_TO, TF_FROM)
    Compute: pose by looking up transform
    Print: "Transform not found."
    return None
    Compute: transformT, transformQ
    return Transformation Matrix
end function
function TIMER_CALLBACK(event)
    Call: publish_tf()
    Call: save_tags_to_file(tags)
end function
function PUBLISH_TF
    for each tag_id, T_AO in tags do
        Create: TransformStamped t
        Set: t's attributes
        Send: Transform t with tf_broadcaster
    end for
end function
function SAVE_TAGS_TO_FILE(tags)
    if tags is empty then
        return
    end if
    Prepare: data_for_file
    Save: data_for_file to filepath
end function
function MAIN
    Initialize: rospy node
    Create: tag_tracker
    Start: rospy.spin()
end function
```

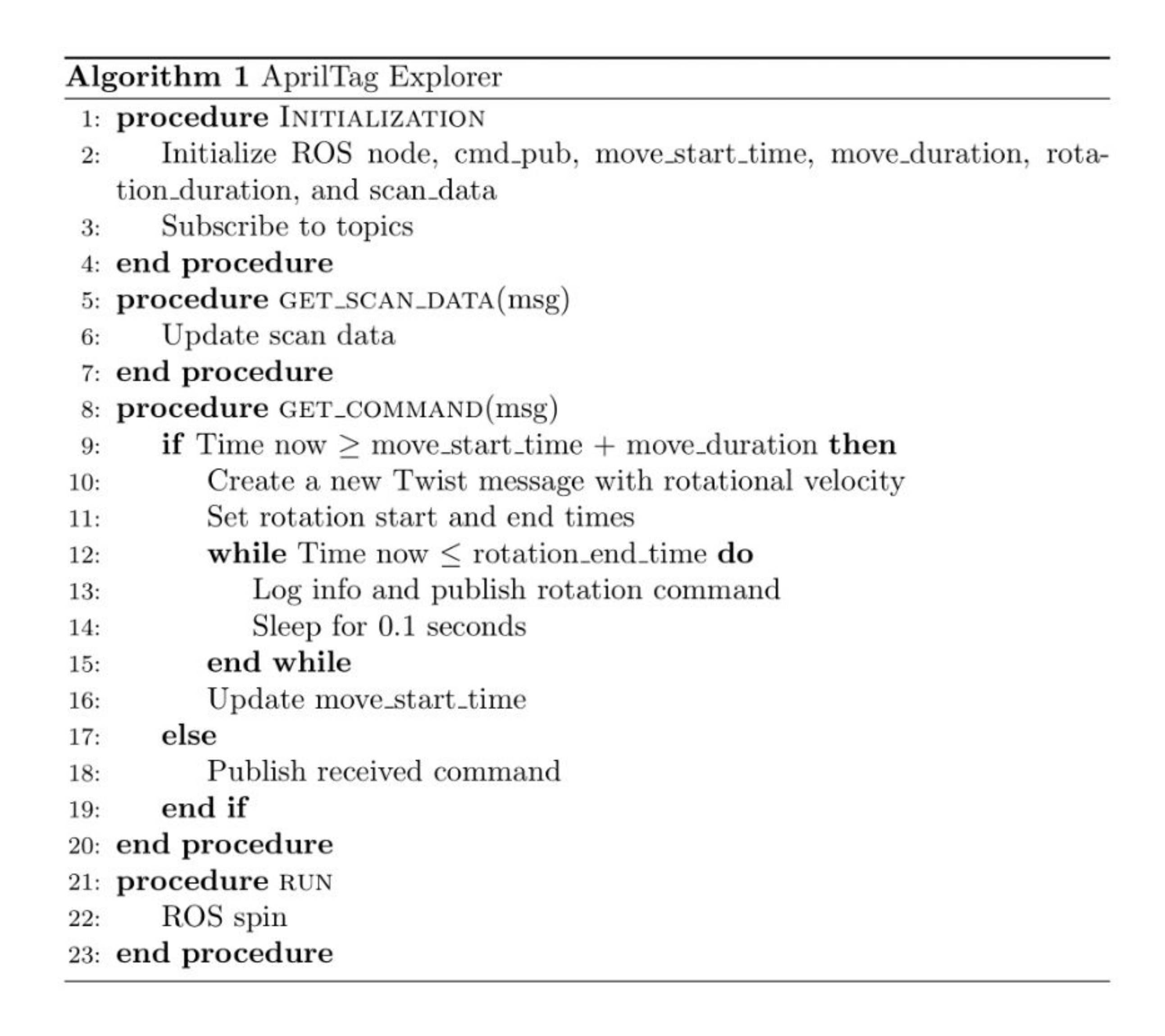
AprilTag Explorer

Problem:

/cmd_vel generated by move_base is not enough for finding all april tags in the unknown world.

Solution:

Wrote the custom node that subscribes to /cmd_vel_update, performs 360° rotation at periodic times and publishes to /cmd_vel



Algorithm 1 AprilTag Explorer

```
procedure INITIALIZATION
    Initialize ROS node, cmd_pub, move_start_time, move_duration, rotation_duration, and scan_data
    Subscribe to topics
end procedure
procedure GET_SCAN_DATA(msg)
    Update scan data
end procedure
procedure GET_COMMAND(msg)
    if Time now $\geq$ move_start_time + move_duration then
        Create a new Twist message with rotational velocity
        Set rotation start and end times
        while Time now $\leq$ rotation_end_time do
            Log info and publish rotation command
            Sleep for 0.1 seconds
        end while
        Update move_start_time
    else
        Publish received command
    end if
end procedure
procedure RUN
    ROS spin
end procedure
```

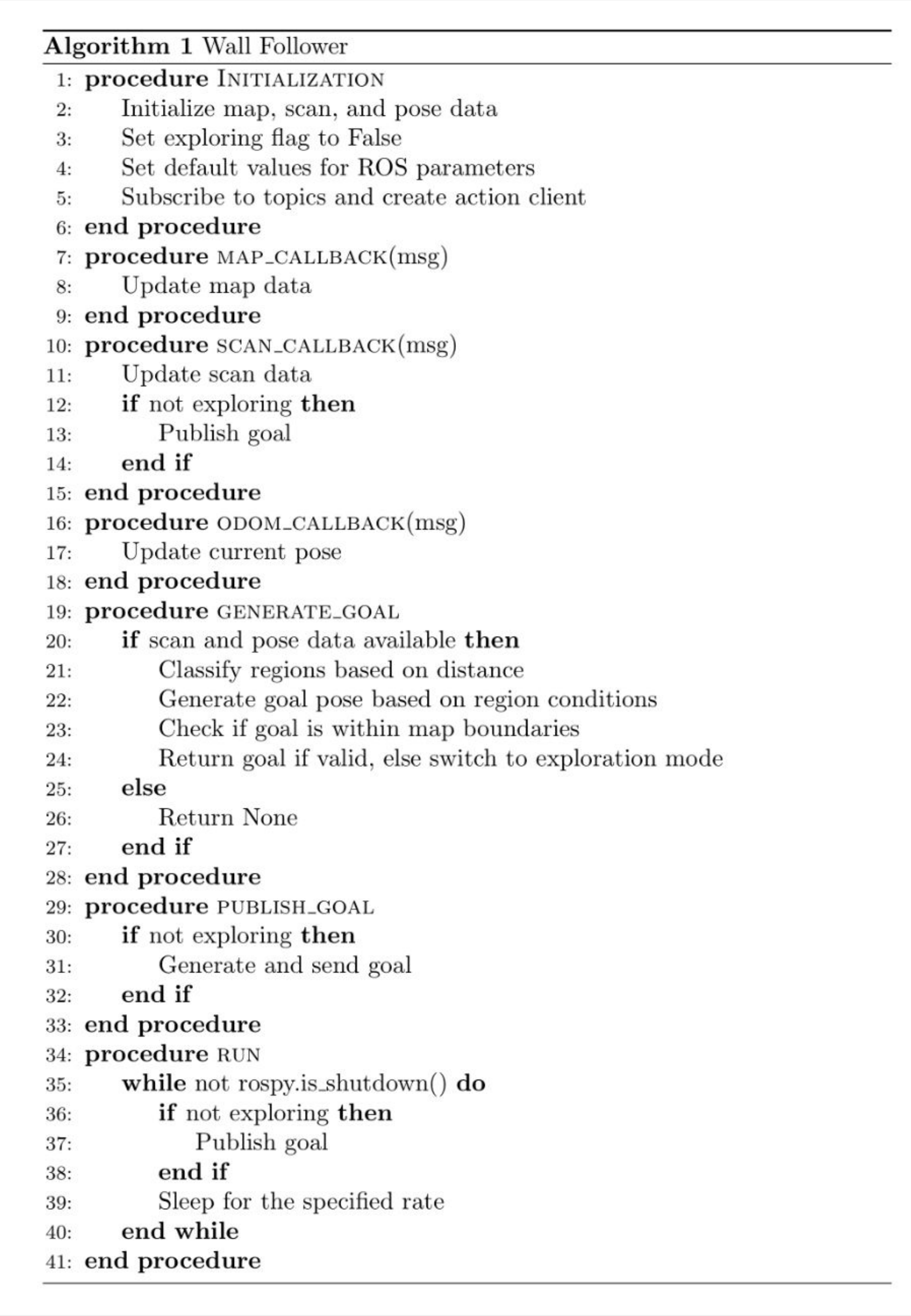
Problem:

Exploration needs to be optimized in order to find April tags while mapping an unknown world.

Solution:

Wrote the custom node that generates goal position for move_base action client.

Used wall following algorithm & explore_lite pkg with proper conditions to generate goal.

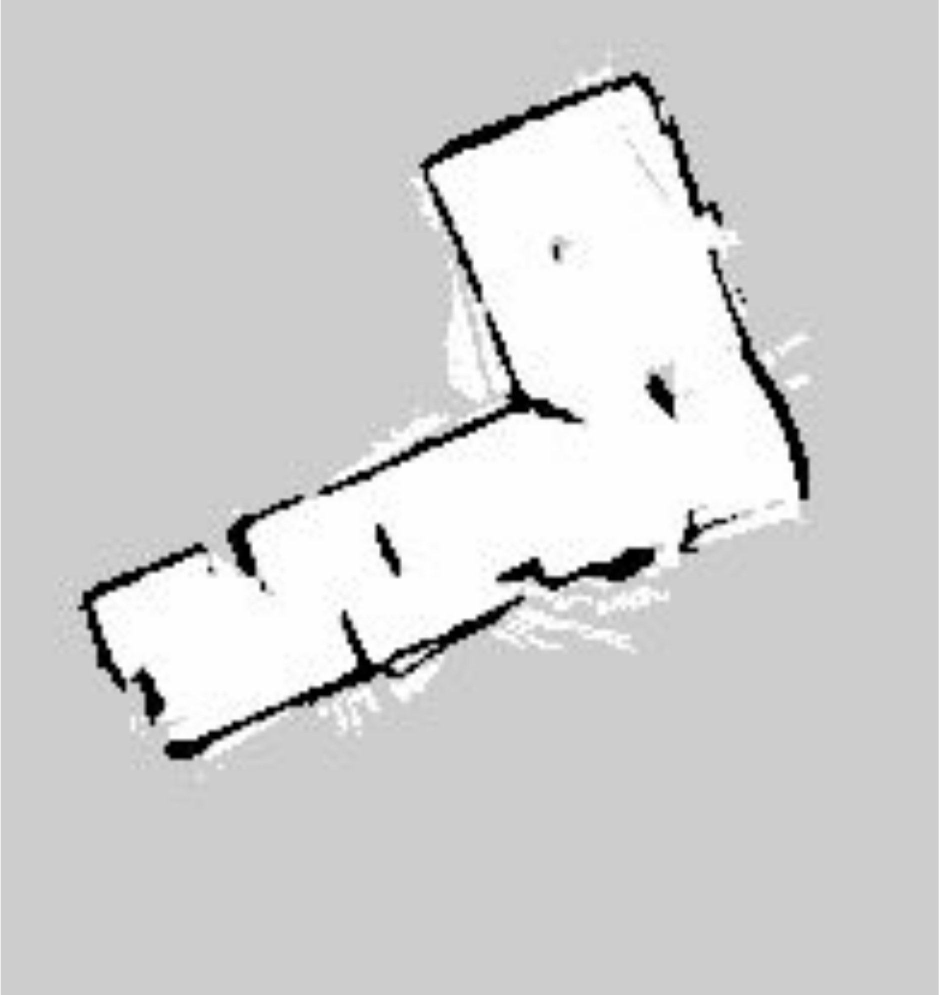


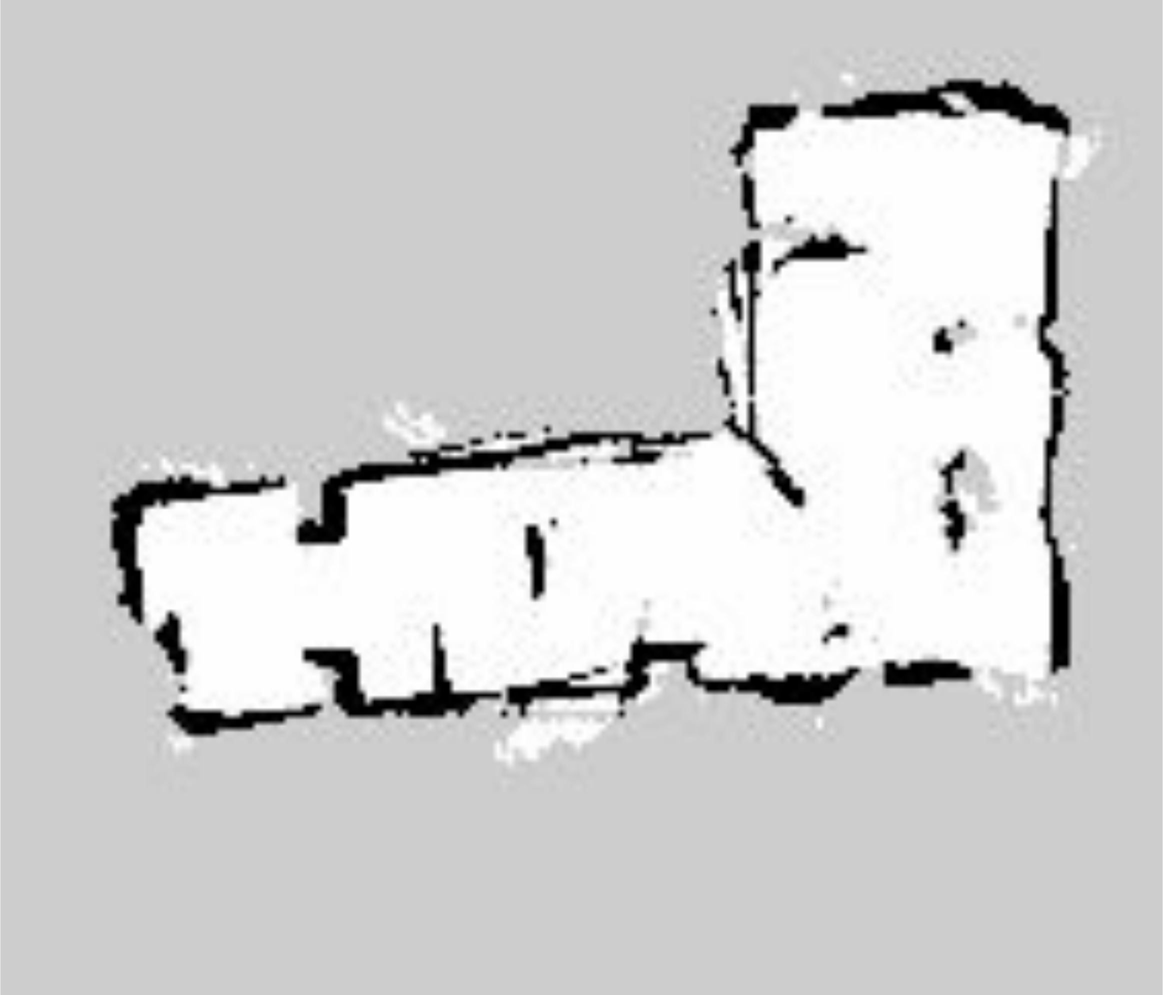
Algorithm 1 Wall Follower

```
procedure INITIALIZATION
    Initialize map, scan, and pose data
    Set exploring flag to False
    Set default values for ROS parameters
    Subscribe to topics and create action client
end procedure
procedure MAP_CALLBACK(msg)
    Update map data
end procedure
procedure SCAN_CALLBACK(msg)
    Update scan data
    if not exploring then
        Publish goal
    end if
end procedure
procedure ODOM_CALLBACK(msg)
    Update current pose
end procedure
procedure GENERATE_GOAL
    if scan and pose data available then
        Classify regions based on distance
        Generate goal pose based on region conditions
        Check if goal is within map boundaries
        Return goal if valid, else switch to exploration mode
    else
        Return None
    end if
end procedure
procedure PUBLISH_GOAL
    if not exploring then
        Generate and send goal
    end if
end procedure
procedure RUN
    while not rospy.is_shutdown() do
        if not exploring then
            Publish goal
        end if
        Sleep for the specified rate
    end while
end procedure
```

Turtlebot in Action

Final Maps





Future Goals

Before 25th April:

- Implementation of GTSAM library to improve mapping and pose accuracy.

Long Term Plan:

- Implement swarm robots to improve efficiency in disaster response.
- Provides scalability.
- Communicate and share information.

Long
Term
Goal

Questions...?