ДОДАТОК А

Апробація наукової статті

Selection of Sensors for Building a 3D Model of the Mobile Robot's Environment

Svitlana Maksymova[1], Mykhailo Akopov[1]
Department of CITAR, Kharkiv National University of Radioelectronics, UKRAINE, Kharkiv, Nauki Ave. 14, email: mykhailo.akopov@nure.ua

Abstract: In this material, methods for constructing a depth map were analyzed, and technical devices suitable for this task were considered..

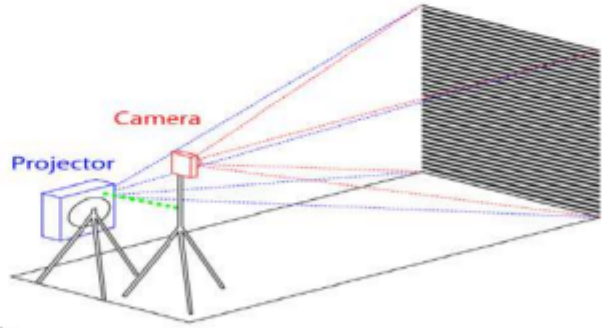
Key words: Mobile robot, Camera, Lidar, Stereo Camera

I. Introduction

Since the invention of the first unmanned aerial vehicles in 1933 until today, drones have become a part of everyday life. From film production to military applications, UAVs are more cost-effective than their manned counterparts. Cartography remains one of the current areas for the use of unmanned aerial vehicles. In my master's thesis we will talk specifically about the software necessary to perform cartographic work using UAVs. Airplane and helicopter drones are used to carry out cartographic work. Airplane-type UAVs are used to process large area (over 20 km). Such devices are capable of photographing hundreds of hectares in 1 hour in the air. A quadcopter is suitable for exploring local areas. This difference in application is due to the specific design of the above-described types of drones. Aircraft-type drones, due to the presence of wings, have an aerodynamic lift force, which in turn arises when the air flow flows. To create an oncoming air flow, it is enough to give the structure a certain speed relative to the air masses. Thus, the main energy consumption occurs during takeoff and landing of the aircraft. These features provide this type of UAV with better autonomy. On the other hand, aircraft-type UAVs have worse maneuverability and compactness compared to helicopter-type UAVs. Helicopter-type drones most often use a design with four rotors called a quadcopter. The quadcopter body does not have aerodynamic abilities, and lift into the air is carried out by electric motors and blades installed on them. Thus, different types of UAVs may be required to perform different tasks. Regardless of the type of drone you choose, additional electronics are required for mapping. Usually, these are a camera and GPS sensors. Using a regular camera, it is possible to obtain raw data in the form of video recording or a large number of photos. Such data must be processed after the UAV returns. An alternative would be to construct a depth map. On the basis of which, in real time, it is possible to construct a map taking into account the terrain. In this work, we will consider the hardware for constructing a depth map.

II. Methods for obtaining a depth map

Structured Light camera. Let's start with, perhaps, one of the simplest, oldest and relatively cheap ways to measure depth - structured light. This method appeared essentially as soon as digital cameras appeared, i.e. more than 40 years ago and greatly simplified a little later, with the advent of digital projectors. The basic idea is extremely simple.[1-3] We place next to a projector that creates, for example, horizontal (and then vertical) stripes and next to a camera that shoots a picture with stripes, as shown in this picture 1.1

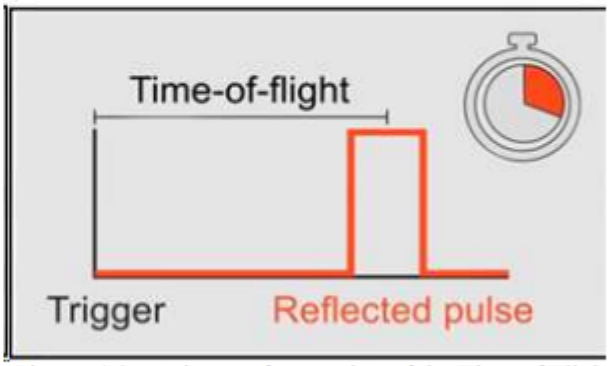


Picture 1.1 – Structured Light Camera Operating Scheme

Since the camera and projector are offset relative to each other, the stripes will also move in proportion to the distance to the object. By measuring this displacement we can calculate the distance to the object. This scheme is hardly suitable for use on UAVs since, due to sunlight, the designed grid will be illuminated and unreadable for sensors.[4-7]

Time of Flight camera

The next way to get depth is more interesting. It is based on measuring the round-trip delay of light (ToF - Time-of-Flight). As you know, the speed of modern processors is high, but the speed of light is low. In one clock cycle of a 3 GHz processor, light can travel only 10 centimeters. Or 10 clock cycles per meter. In fact, we need to measure the delay with which light returns to each point, Picture 1.2.

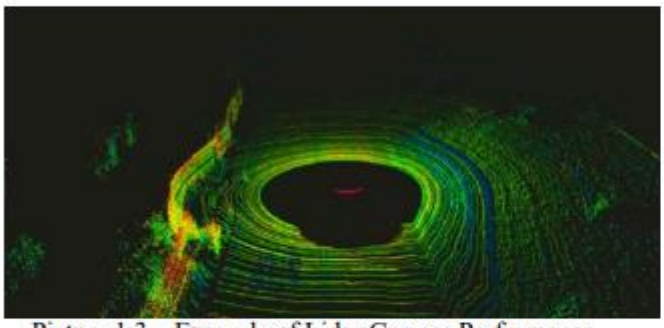


Picture 1.2 – Scheme of Operation of the Time of Flight Camera

The problems remain the same. There is a high probability of the image being exposed to bright light.

Camera based on lidar technology [8-11]

The first lidars (from LIDaR - Light Identification Detection and Ranging), built as bundles of similar devices rotating around a horizontal axis, were the first to be used by the militaries, then tested in car autopilots. They performed quite well there, which caused a powerful surge in investment in the region. Initially, the lidars rotated, giving a similar picture several times per second, Picture 1.3.

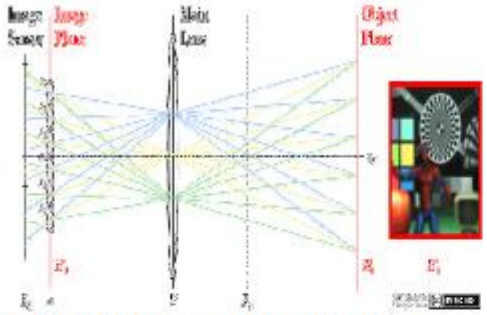


Picture 1.3 – Example of Lidar Camera Performance

The problem with these devices can be considered their relatively high cost. Putting together a compact and inexpensive device is quite problematic. In addition, bright sunlight reflected from surfaces may cause incorrect data reading.

Light Field depth camera

The topic of plenoptics (from the Latin plenus - complete and optikos - visual) or light fields is still relatively little known to the general public, although professionals have begun to study it extremely actively. The main idea is to try to capture not just light at each point, but a two-dimensional array of light rays, Picture 1.4, making each frame four-dimensional. In practice this is done using a microlens array

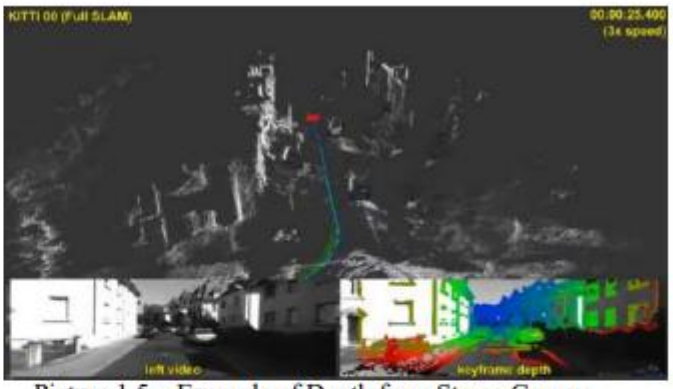


Picture 1.4 – Light Field Camera Operation Scheme

The main disadvantages of plenoptics can be considered the relatively low resolution and complexity of software implementation.

Depth from Stereo camera [12-14]

Of the 5 methods under consideration for constructing depth video, only two - this and the previous one (stereo and plenoptics) - do not interfere with the sun and are not interfered with by surface reflections. At the same time, plenoptics is many times more expensive and less accurate at long distances. Depth from stereo (Picture 1.4) - in terms of equipment cost - is the cheapest way to obtain depth, since cameras are now inexpensive and continue to become cheaper quickly.



Picture 1.5 – Example of Depth from Stereo Camera Operation

The difficulty is that further processing is much more resource-intensive than other methods. Despite the disadvantages discussed above, this technology was chosen as depth sensors for UAVs.

Waveshare IMX219-83

This stereo camera (Picture 1.5) was chosen due to its high resolution of 3280 x 2464, compactness, and the presence of additional sensors - accelerometer, gyroscope, magnetometer.

Picture 1.5 – Stereo Camera Waveshare IMX219-83

To create a stereo pair, two compatible Pi cameras would be suitable too, for example the Raspberry Pi Camera Module 3 cameras (Picture 1.6).

Figure 1.6 – Raspberry Pi Camera Module 3

But in this case, it would be necessary to synchronize the cameras, which would add extra load to the computing module. To perform the calculations,

Rasberry Pi CM4 and Carrier board were chosen for it.[15-17]

III. Conclusions

Thus, various types of sensors that may be suitable for the implementation of a system for building a 3D model of the environment of a mobile robot were considered.

References

[1] Shuttle-Based Storage and Retrieval System 3d Model Improvement and Development / I. Nevliudov, V. Yevsieiev, S. Maksymova, O. Klymenko, M. Vzhesniewski // V International Conference on Natural Science and Technologies (ICONAT 2023), 1st-3th June 2023. – Sunny Beach-Bulgaria. – P. 15.

[2] Yevsieiev, V. Comparative Analysis of the Characteristics of Mobile Robots and Collaboration Robots Within INDUSTRY 5.0. / V. Yevsieiev, D. Gurin // In the VI International Scientific and Theoretical Conference, September 8, 2023. Chicago, USA. P.92-94

[3] Nevliudov, I., Yevsieiev, V., Maksymova, S., Demska, N., Kolesnyk, K., & Miliutina, O. (2022, September). Object Recognition for a Humanoid Robot Based on a Microcontroller. In 2022 IEEE XVIII International Conference on the Perspective Technologies and Methods in MEMS Design (MEMSTECH) PP. 61-64. DOI: 10.1109/MEMSTECH55132.2022.10002906

[4] Attar, H., & et al.. (2022). Zoomorphic Mobile Robot Development for Vertical Movement Based on the Geometrical Family Caterpillar. Computational Intelligence and Neuroscience, 2022, Article ID 3046116, https://doi.org/10.1155/2022/3046116.

[5] Невлюдов І. Ш., Андрусевич А. О., Євсєєв В. В., Новоселов С. П., Демська Н. П. Проектування мобільних маніпуляційних роботів: Монографія. – Х. :, 2022. – 427 с.

[6] Євсєєв В.В. Проектування мобільних роботів на базі одноплатних комп'ютерів (Raspberry Pi и мови Python 3.6) // Невлюдов І. Ш., Андрусевич А. О., Євсєєв В. В. Підручник. – Харків : 2020. С. 257.

[7] Vladyslav Yevsieiev, Nikolaj Starodubcev (2023). Development of a control algorithm for a small-sized mobile manipulation robot. Scientific Collection «InterConf», (140), P. 648-651.

[8] Yevsieiev V. (2023) Development of a program for modeling the control of a mobile manipulation robot in the unity environment / Yevsieiev V., Starodubcev N. // Scientific Collection «InterConf», (141), P. 331-334.

[9] A Small-Scale Manipulation Robot a Laboratory Layout Development / Yevsieiev V., Starodubcev N., Maksymova S., Stetsenko K. // International independent scientific journal, №47, 2023. P.18-28.

[10] Yevsieiev V. Analysis of Crawler Robots / V. Yevsieiev, S. Shmatko // "Innovations Technologies in Science and Practice" : The VI International Scientific and Practical Conference, February 15-18, 2022. – Haifa, Israel, 2022. – P. 510-514.

[11] Yevsieiev, V. ., Maksymova, S. ., & Starodubcev, N. . (2022). A ROBOTIC PROSTHETIC A CONTROL SYSTEM AND A STRUCTURAL DIAGRAM DEVELOPMENT. Collection of Scientific Papers «ΛΌГОΣ», (August 12, 2022; Zurich, Switzerland), 113–114. https://doi.org/10.36074/logos-12.08.2022.33

[12] Yevsieiev V., Maksymova S., Starodubcev N. Software Implementation Concept Development for the Mobile Robot Control System on ESP-32CAM // Current issues of science, prospects and challenges: collection of scientific papers «SCIENTIA» with Proceedings of the II International Scientific and Theoretical Conference (Vol. 2), June 10, 2022. Sydney, Australia: European Scientific Platform., 2022. P. 54-56

[13] Yevsieiev V. Development of Architecture for Mobile Robot Control Based on Raspberry Pi Model 3 B+ / V. Yevsieiev, A. Skripkin // Scientific Horizon in the Context of Social Crises : The XI International Scientific and Practical Conference, April 6-8, 2022. – Tokyo, Japan, 2022. – P. 274–277.

[14] Yevsieiev V. Analysis of Crawler Robots / V. Yevsieiev, S. Shmatko // "Innovations Technologies in Science and Practice" : The VI International Scientific and Practical Conference, February 15-18, 2022. – Haifa, Israel, 2022. – P. 510-514.

[15] Розробка 3D-моделі зооморфного мобільного робота для вертикальних переміщень по металевим поверхням / І. Ш. Невлюдов, В. В. Євсєєв, Н. П. Демська, В. О. Руденко // Наука і техніка сьогодні. – 2022. – № 4(4). – С.163-174.

[16] Yevsieiev , . V., Maksymova, S. ., & Starodubcev, N. . (2022). DEVELOPMENT OF AN ALGORITHM FOR ESP32-CAM OPERATION IN HTTP SERVER MODE FOR STREAMING VIDEO. Collection of Scientific Papers «ΛΌГОΣ», (July 8, 2022; Paris, France), 177–179. https://doi.org/10.36074/logos-08.07.2022.049

[17] Attar, H., & et al.. (2022). Control System Development and Implementation of a CNC Laser Engraver for Environmental Use with Remote Imaging. Computational Intelligence and Neuroscience, 2022, Article ID 9140156, https://doi.org/10.1155/2022/9140156.

ДОДАТОК Б

Лістинг програми

```
transform_tolerance: 0.5  # 0.2

always_send_full_costmap: false

footprint: [[0.4, 0.4], [0.4, -0.4], [-0.4, -0.4], [-0.4, 0.4]]

static_layer:
  unknown_cost_value: -1
  lethal_cost_threshold: 254
  first_map_only: true
  subscribe_to_updates: false
  track_unknown_space: true
  use_maximum: true
  trinary_costmap: true


inflation_layer:
  inflation_radius: 30.  # the bigger the better, defines where the gradient is
  cost_scaling_factor: 0.9  # 0.5 the bigger the steeper the gradient is


observation_sources: laser_scan_sensor point_cloud_sensor

laser_scan_sensor: {sensor_frame: laser_link, data_type: LaserScan, topic: /scan, marking:
true, clearing: true}

point_cloud_sensor: {sensor_frame: depth_camera_optical, data_type: PointCloud, topic:
/mark1/depth/points, marking: true, clearing: true}


global_costmap:
  global_frame: map
```

```
  robot_base_frame: base_link
  update_frequency: 5.0
  publish_frequency: 5.0
  static_map: true
  rolling_window: false
  transform_tolerance: 0.2


local_costmap:
  global_frame: odom
  robot_base_frame: base_link
  update_frequency: 5.0
  publish_frequency: 2.0
  static_map: false
  rolling_window: true
  width: 6.0
  height: 6.0
  resolution: 0.05


image: empty.bmp
resolution: 0.1
origin: [-1.0, -1.0, 0]
occupied_thresh: 0.5
free_thresh: 0.3
negate: 0


dictitems:
  acc_lim_theta: 15.0
  acc_lim_trans: 0.27356904308223723
  acc_lim_x: 6.4
```

acc_lim_y: 0.0
angular_sim_granularity: 0.1
forward_point_distance: 0.325
goal_distance_bias: 28.0
groups:
  dictitems:
    acc_lim_theta: 15.0
    acc_lim_trans: 0.27356904308223723
    acc_lim_x: 6.4
    acc_lim_y: 0.0
    angular_sim_granularity: 0.1
    forward_point_distance: 0.325
    goal_distance_bias: 28.0
    groups:
      state: []
    id: 0
    max_scaling_factor: 0.2
    max_vel_theta: 9.057886686238938
    max_vel_trans: 0.8816185923631891
    max_vel_x: 0.5497546521927701
    max_vel_y: 0.0
    min_vel_theta: 0.14232107570294295
    min_vel_trans: 0.1745279388943651
    min_vel_x: -3.4420225766692187
    min_vel_y: 0.0
    name: Default
    occdist_scale: 0.01
    oscillation_reset_angle: 10.0
    oscillation_reset_dist: 0.05
    parameters:
      state: []
    parent: 0

```
    path_distance_bias: 32.0
    prune_plan: true
    restore_defaults: false
    scaling_speed: 0.25
    sim_granularity: 0.025
    sim_time: 1.7
    state: true
    stop_time_buffer: 0.2
    theta_stopped_vel: 0.1
    trans_stopped_vel: 0.1
    twirling_scale: 0.0
    type: ''
    use_dwa: true
    vth_samples: 20
    vx_samples: 3
    vy_samples: 1
    xy_goal_tolerance: 0.3
    yaw_goal_tolerance: 0.3
  state: []
max_scaling_factor: 0.2
max_vel_theta: 9.057886686238938
max_vel_trans: 0.8816185923631891
max_vel_x: 0.5497546521927701
max_vel_y: 0.0
min_vel_theta: 0.14232107570294295
min_vel_trans: 0.1745279388943651
min_vel_x: -3.4420225766692187
min_vel_y: 0.0
occdist_scale: 0.01
oscillation_reset_angle: 10.0
oscillation_reset_dist: 0.05
path_distance_bias: 32.0
```

```
  prune_plan: true
  restore_defaults: false
  scaling_speed: 0.25
  sim_granularity: 0.025
  sim_time: 1.7
  stop_time_buffer: 0.2
  theta_stopped_vel: 0.1
  trans_stopped_vel: 0.1
  twirling_scale: 0.0
  use_dwa: true
  vth_samples: 20
  vx_samples: 3
  vy_samples: 1
  xy_goal_tolerance: 0.3
  yaw_goal_tolerance: 0.3
state: []


base_global_planner: "global_planner/GlobalPlanner"
base_local_planner: "dwa_local_planner/DWAPlannerROS"

recovery_behaviors: #[]
  - name: conservative_reset
    type: clear_costmap_recovery/ClearCostmapRecovery
  - name: agressive_reset
    type: clear_costmap_recovery/ClearCostmapRecovery

controller_frequency: 20.
planner_frequency: 0.  # 0. is only plan on a new goal or when local planner fails
planner_patience: 5.
controller_patience: 15.
recovery_behavior_enables: true
```

```
shutdown_costmaps: false
oscillation_timeout: 0.  # 0. is infinite
oscillation_distance: 0.5
map_planning_retries: -1.  # -1 is infinite

# Planners
GlobalPlanner:
  allow_unknown: true
  default_tolerance: 0.
  visualize_potential: true
  use_dijkstra: true  # although A* is faster, implementation pecularities here make it
undesirable
  use_quadratic: true
  use_grid_path: false
  old_navfn_behavior: false
  lethal_cost: 253 #253 # how far from obstacle
  neutral_cost: 66 #40
  cost_factor:  0.55 #3
  publish_potential: True
  orientation_mode: 0.
  orientation_window_size: 1


DWAPlannerROS:
  use_dwa: true
  global_frame_id: odom

  acc_lim_x: 50.5
  acc_lim_y: 50.
  acc_lim_th: 50.7
  max_vel_trans: 50.15
  min_vel_trans: 0.
```

```
max_vel_x: 50.15
min_vel_x: 0.0
max_vel_y: 50.
min_vel_y: 0.
max_vel_theta: 50.0  # 0.5
max_rot_vel: 50.0

yaw_goal_tolerance: 0.3
xy_goal_tolerance: 0.3
latch_xy_goal_tolerance: false

sim_time: 1.7
sim_granularity: 0.025
vx_samples: 3
vy_samples: 1
vth_samples: 20

path_distance_bias: 32.  # 32
goal_distance_bias: 28.
occdist_scale: 0.01  # 0.01
forward_point_distance: 0.325
stop_time_buffer: 0.2
scaling_speed: 0.25
max_scaling_factor: 0.2
publish_cost_grid: true  # kinetic or obsolete?
publish_cost_grid_pc: true
publish_traj_pc: true

oscillation_reset_dist: 0.05
oscillation_reset_angle: 10.

prune_plan: true
```

```
# Recovery behaviors
conservative_reset:
  reset_distance: 3.0

agressive_reset:
  reset_distance: 4.0
```

```
<?xml version="1.0" encoding="UTF-8"?>
<launch>

   <node pkg="move_base" type="move_base" respawn="false" name="move_base"
output="screen">

        <rosparam file="$(find
mark1)/src/mark1_2dnav/config/costmap_common_params.yaml" command="load"
ns="global_costmap" />

        <rosparam file="$(find
mark1)/src/mark1_2dnav/config/costmap_common_params.yaml" command="load"
ns="local_costmap" />

        <rosparam file="$(find mark1)/src/mark1_2dnav/config/local_costmap_params.yaml"
command="load" />

        <rosparam file="$(find mark1)/src/mark1_2dnav/config/global_costmap_params.yaml"
command="load" />

        <rosparam file="$(find mark1)/src/mark1_2dnav/config/mark1_param_1.yaml"
command="load" />
```

```
        <!--remap from="cmd_vel" to="cmd_vel"/>
        <remap from="odom" to="odom"/-->

 </node>

</launch>


image: mymap.pgm
resolution: 0.050000
origin: [-100.000000, -100.000000, 0.000000]
negate: 0
occupied_thresh: 0.65
free_thresh: 0.196


version="1.0" encoding="UTF-8"?>
<launch>

   <!-- global params-->
    <arg name="localization_type" default="AMCL"/>
    <!-- Possible variants
        FIXED_ODOM
        AMCL
        GMAPPING
    -->
    <arg name="map" default="my_map"/>
    <!-- Possible maps
        nothing
        my_map
    -->
```

```
  <!-- URDF description -->
  <param name="robot_description" command="$(find xacro)/xacro.py '$(find
mark1)/src/mark1_description/urdf/mark1.xacro'"/>

 <!-- TF stuff -->
    <!-- send fake joint values -->
  <node name="joint_state_publisher" pkg="joint_state_publisher"
type="joint_state_publisher">
    <param name="use_gui" value="False"/>
  </node>

    <!-- Combine joint values -->
   <node name="robot_state_publisher" pkg="robot_state_publisher"
type="state_publisher"/>

    <!-- fixed odom -->
  <!--node pkg="tf" type="static_transform_publisher" name="static_odom_broadcaster"
    args="0 0 0 0 0 0 map odom 100"/-->

  <!-- run gazebo and spawn mybot -->
  <include file="$(find mark1)/src/mark1_gazebo/launch/mark1_world.launch"/>

  <!-- software -->

  <include file="$(find mark1)/src/mark1_config/launch/mark1_software.launch">
      <arg name="localization_type" value="$(arg localization_type)"/>
    <arg name="map" value="$(arg map)"/>
  </include>


   <!-- visualization -->
```

```
    <node name="rviz" pkg="rviz" type="rviz"  args="-d $(find
mark1)/src/mark1_config/rviz/mark1_gazebo.rviz"/>
</launch>


<?xml version="1.0" encoding="UTF-8"?>
<launch>

    <arg name="localization_type" default="FIXED_ODOM"/>
    <!-- Possible variants
        FIXED_ODOM
        AMCL
        GMAPPING
    -->

          <arg name="map" default="nothing"/>
          <!-- Possible maps
        nothing
        my_map
                  -->
<!-- fixed odom -->
    <node if="$(eval localization_type == 'FIXED_ODOM')" pkg="tf"
type="static_transform_publisher" name="static_odom_broadcaster"
        args="0 0 0 0 0 0 map odom 100">
          <!--remap from="scan" to="/mark1/laser/scan"/-->
  </node>

    <!-- Run the map server -->
    <node if="$(eval localization_type == 'FIXED_ODOM')" name="map_server"
pkg="map_server" type="map_server" args="$(find
mark1)/src/mark1_2dnav/config/map_empty.yaml"/>
```

```
  <node if="$(eval localization_type == 'AMCL' and map == 'nothing')"
name="map_server" pkg="map_server" type="map_server" args="$(find
mark1)/src/mark1_2dnav/config/map_empty.yaml"/>

  <node if="$(eval localization_type == 'AMCL' and map == 'my_map')"
name="map_server" pkg="map_server" type="map_server" args="$(find
mark1)/src/mark1_2dnav/maps/my_map.yaml"/>

   <!--- Run AMCL -->
  <node if="$(eval localization_type == 'AMCL')" pkg="amcl" type="amcl" name="amcl"
output="screen">
    <!-- Publish scans from best pose at a max of 10 Hz -->
    <param name="odom_model_type" value="diff"/>
    <param name="odom_alpha5" value="0.1"/>
    <param name="transform_tolerance" value="0.2" />
    <param name="gui_publish_rate" value="10.0"/>
    <param name="laser_max_beams" value="30"/>
    <param name="min_particles" value="500"/>
    <param name="max_particles" value="5000"/>
    <param name="kld_err" value="0.05"/>
    <param name="kld_z" value="0.99"/>
    <param name="odom_alpha1" value="0.2"/>
    <param name="odom_alpha2" value="0.2"/>
    <!-- translation std dev, m -->
    <param name="odom_alpha3" value="0.8"/>
    <param name="odom_alpha4" value="0.2"/>
    <param name="laser_z_hit" value="0.5"/>
    <param name="laser_z_short" value="0.05"/>
    <param name="laser_z_max" value="0.05"/>
    <param name="laser_z_rand" value="0.5"/>
    <param name="laser_sigma_hit" value="0.2"/>
    <param name="laser_lambda_short" value="0.1"/>
```

```
        <param name="laser_lambda_short" value="0.1"/>
        <param name="laser_model_type" value="likelihood_field_prob"/>
        <!-- <param name="laser_model_type" value="beam"/> -->
        <param name="laser_likelihood_max_dist" value="2.0"/>
        <param name="update_min_d" value="0.2"/>
        <param name="update_min_a" value="0.5"/>
        <param name="odom_frame_id" value="odom"/>
        <param name="resample_interval" value="1"/>
        <param name="transform_tolerance" value="0.1"/>
        <param name="recovery_alpha_slow" value="0.0"/>
        <param name="recovery_alpha_fast" value="0.0"/>

                <param name="initial_pose_x" value="1"/>
        <param name="initial_pose_y" value="1"/>
        <param name="initial_pose_a" value="0.5"/>

        <param name="initial_cov_xx" value="10"/>
        <param name="initial_cov_yy" value="15"/>
        <param name="initial_cov_aa" value="10"/>

        <param name="use_map_topic" value="true"/>
        <param name="base_frame_id" value="base_link"/>
        <remap from="scan" to="/scan"/>

    </node>

     <!-- gmapping -->
    <node if="$(eval localization_type == 'GMAPPING')" pkg="gmapping"
type="slam_gmapping" name="gmapping" output="screen">
        <param name="base_frame" value="base_link"/>
        <param name="maxRange" value="29"/>
        <param name="maxRange" value="30"/>
```

```
      <!--remap from="scan" to="/scan"/-->
    </node>

    <include file="$(find mark1)/src/mark1_2dnav/launch/mark1_move_base.launch"/>

</launch>


mark1:
  # Publish all joint states -----------------------------------
  joint_state_controller:
    type: joint_state_controller/JointStateController
    publish_rate: 50

  mobile_base_controller:
    type: "diff_drive_controller/DiffDriveController"
    left_wheel: 'Left_Motor_Joint'
    right_wheel: 'Right_Motor_Joint'
    pose_covariance_diagonal: [0.001, 0.001, 1000000.0, 1000000.0, 1000000.0, 1000.0]
    twist_covariance_diagonal: [0.001, 0.001, 1000000.0, 1000000.0, 1000000.0, 1000.0]


<?xml version="1.0" encoding="UTF-8"?>
<launch>

  <!-- Load joint controller configurations from YAML file to parameter server -->
  <rosparam file="$(find mark1)/src/mark1_control/config/mark1_control.yaml"
command="load"/>

  <!-- load the controllers -->
  <node name="controller_spawner"
```

```
    pkg="controller_manager"
    type="spawner" respawn="false"
    output="screen" ns="/mark1"
    args="joint_state_controller
      mobile_base_controller"
  />

  <!-- convert joint states to TF transforms for rviz, etc -->
  <node name="robot_state_publisher" pkg="robot_state_publisher" type="state_publisher"
respawn="false" output="screen">
    <param name="robot_description" command="$(find xacro)/xacro.py '$(find
mark1)/src/mark1_description/urdf/mark1.xacro'"/>
    <remap from="/joint_states" to="/mark1/joint_states" />
  </node>

</launch>


<?xml version="1.0"?>
<robot>


  <gazebo>
    <plugin name="differential_drive_controller" filename="libgazebo_ros_diff_drive.so">
      <legacyMode>false</legacyMode>
      <alwaysOn>true</alwaysOn>
      <updateRate>20</updateRate>
      <leftJoint>Left_Motor_Joint</leftJoint>
      <rightJoint>Right_Motor_Joint</rightJoint>
      <wheelSeparation>0.4</wheelSeparation>
      <wheelDiameter>0.3</wheelDiameter>
      <torque>20</torque>
```

```
    <commandTopic>cmd_vel</commandTopic>
    <odometryTopic>odom</odometryTopic>
    <odometryFrame>odom</odometryFrame>
    <robotBaseFrame>base_link</robotBaseFrame>
  </plugin>
</gazebo>


<gazebo reference="base_link">
  <material>Gazebo/Orange</material>
</gazebo>

<gazebo reference="Left_Motor_Link">
  <material>Gazebo/Blue</material>
</gazebo>

<gazebo reference="Right_Motor_Link">
  <material>Gazebo/Blue</material>
</gazebo>


<!-- hokuyo -->
<gazebo reference="laser_link">
  <sensor type="ray" name="head_hokuyo_sensor">
    <pose>0 0 0 0 0 0</pose>
    <visualize>true</visualize>
    <update_rate>40</update_rate>
    <ray>
      <scan>
        <horizontal>
          <samples>720</samples>
          <resolution>1</resolution>
```

```
          <min_angle>-3.1415</min_angle>
          <max_angle>3.1414</max_angle>
        </horizontal>
      </scan>
      <range>
        <min>0.10</min>
        <max>30.0</max>
        <resolution>0.01</resolution>
      </range>
      <noise>
        <type>gaussian</type>
        <!-- Noise parameters based on published spec for Hokuyo laser
             achieving "+-30mm" accuracy at range < 10m.  A mean of 0.0m and
             stddev of 0.01m will put 99.7% of samples within 0.03m of the true
             reading. -->
        <mean>0.0</mean>
        <stddev>0.01</stddev>
      </noise>
    </ray>
    <plugin name="gazebo_ros_head_hokuyo_controller"
filename="libgazebo_ros_laser.so">
      <topicName>/scan</topicName>
      <frameName>laser_link</frameName>
    </plugin>
  </sensor>
 </gazebo>



<!-- camera -->
 <gazebo reference="camera_link">
  <sensor type="camera" name="camera1">
    <update_rate>30.0</update_rate>
```

```
<camera name="head">
  <horizontal_fov>1.3962634</horizontal_fov>
  <image>
    <width>800</width>
    <height>800</height>
    <format>R8G8B8</format>
  </image>
  <clip>
    <near>0.02</near>
    <far>300</far>
  </clip>
  <noise>
    <type>gaussian</type>
    <!-- Noise is sampled independently per pixel on each frame.
         That pixel's noise value is added to each of its color
         channels, which at that point lie in the range [0,1]. -->
    <mean>0.0</mean>
    <stddev>0.007</stddev>
  </noise>
</camera>
<plugin name="camera_controller" filename="libgazebo_ros_camera.so">
  <alwaysOn>true</alwaysOn>
  <updateRate>0.0</updateRate>
  <cameraName>mark1/camera1</cameraName>
  <imageTopicName>image_raw</imageTopicName>
  <cameraInfoTopicName>camera_info</cameraInfoTopicName>
  <frameName>camera_link</frameName>
  <hackBaseline>0.07</hackBaseline>
  <distortionK1>0.0</distortionK1>
  <distortionK2>0.0</distortionK2>
  <distortionK3>0.0</distortionK3>
  <distortionT1>0.0</distortionT1>
```

```
      <distortionT2>0.0</distortionT2>
    </plugin>
  </sensor>
 </gazebo>


<!-- deth camera -->
 <gazebo reference="depth_camera_link">
 <sensor name="kinect" type="depth">
  <update_rate>20</update_rate>
  <camera>
   <horizontal_fov>1.047198</horizontal_fov>
   <image>
    <width>640</width>
    <height>480</height>
    <format>R8G8B8</format>
   </image>
   <clip>
    <near>0.05</near>
    <far>3</far>
   </clip>
  </camera>
  <plugin name="kinect_controller" filename="libgazebo_ros_openni_kinect.so">
   <baseline>0.2</baseline>
   <alwaysOn>true</alwaysOn>
   <updateRate>1.0</updateRate>
   <cameraName>/mark1/depth_camera</cameraName>
   <imageTopicName>/mark1/rgb/image_raw</imageTopicName>
   <cameraInfoTopicName>/mark1/rgb/camera_info</cameraInfoTopicName>
   <depthImageTopicName>/mark1/depth/image_raw</depthImageTopicName>
   <depthImageInfoTopicName>/mark1/depth/camera_info</depthImageInfoTopicName>
   <pointCloudTopicName>/mark1/depth/points</pointCloudTopicName>
```

```
      <frameName>/depth_camera_optical</frameName>
      <pointCloudCutoff>0.5</pointCloudCutoff>
      <pointCloudCutoffMax>3.0</pointCloudCutoffMax>
      <distortionK1>0.00000001</distortionK1>
      <distortionK2>0.00000001</distortionK2>
      <distortionK3>0.00000001</distortionK3>
      <distortionT1>0.00000001</distortionT1>
      <distortionT2>0.00000001</distortionT2>
      <CxPrime>0</CxPrime>
      <Cx>0</Cx>
      <Cy>0</Cy>
      <focalLength>0</focalLength>
      <hackBaseline>0</hackBaseline>
    </plugin>
  </sensor>
</gazebo>
</robot>


<?xml version='1.0'?>

<robot name="mark1" xmlns:xacro="http://www.ros.org/wiki/xacro">


  <xacro:include filename="$(find mark1)/src/mark1_description/urdf/mark1.gazebo" />
  <xacro:include filename="$(find mark1)/src/mark1_description/urdf/materials.xacro" />
  <xacro:include filename="$(find mark1)/src/mark1_description/urdf/macros.xacro" />


  <link name="base_link">
    <inertial>
      <origin
```

```
    xyz="0.00096237 -0.01836 0.0013951"
    rpy="0 0 0" />
  <mass
    value="10.207" />
  <inertia
     ixx="0.5" ixy="0" ixz="0"
      iyy="1.0" iyz="0"
      izz="0.1" />
</inertial>
<visual>
  <origin
    xyz="0 0 0"
    rpy="0 0 0" />
  <geometry>
    <mesh
      filename="package://mark1/src/mark1_description/meshes/base_link.STL" />
  </geometry>
  <material
    name="">
    <color
      rgba="1 1 1 1" />
  </material>
</visual>
<collision>
  <origin
    xyz="0 0 0"
    rpy="0 0 0" />
  <geometry>
    <mesh
      filename="package://mark1/src/mark1_description/meshes/base_link.STL" />
  </geometry>
</collision>
```

```
</link>
<link    name="Left_Motor_Link">
  <inertial>
    <origin
      xyz="-6.9389E-18 -0.029509 1.3878E-17"
      rpy="0 0 0" />
    <mass
      value="1.483" />
    <inertia
      ixx="0.0078938" ixy="1.3747E-18" ixz="-2.0644E-19"
      iyy="0.014347" iyz="-3.4936E-18"
      izz="0.0078938" />
  </inertial>
  <visual>
    <origin
      xyz="0 0 0"
      rpy="0 0 0" />
    <geometry>
      <mesh
        filename="package://mark1/src/mark1_description/meshes/Left_Motor_Link.STL" />
    </geometry>
    <material
      name="">
      <color
        rgba="0.29804 0.29804 0.29804 1" />
    </material>
  </visual>
  <collision>
    <origin
      xyz="0 0 0"
      rpy="0 0 0" />
    <geometry>
```

```
      <mesh
        filename="package://mark1/src/mark1_description/meshes/Left_Motor_Link.STL" />
    </geometry>
  </collision>
</link>
<joint name="Left_Motor_Joint" type="continuous">
  <origin
    xyz="-0.31944 0.20433 -0.05511"
    rpy="-3.1415 3.1415 3.141" />
  <parent
    link="base_link" />
  <child
    link="Left_Motor_Link" />
  <axis
    xyz="0 1 0" />
</joint>
<link name="Right_Motor_Link">
  <inertial>
    <origin
      xyz="0 -0.02951 7.5894E-18"
      rpy="0 0 0" />
    <mass
      value="1.483" />
    <inertia
      ixx="0.0078938"
      ixy="4.2444E-18"
      ixz="-7.123E-20"
      iyy="0.014347"
      iyz="-7.9282E-19"
      izz="0.0078938" />
  </inertial>
  <visual>
```

```
      <origin
        xyz="0 0 0"
        rpy="0 0 0" />
      <geometry>
        <mesh
          filename="package://mark1/src/mark1_description/meshes/Right_Motor_Link.STL"
/>
      </geometry>
      <material
        name="">
        <color
          rgba="0.29804 0.29804 0.29804 1" />
      </material>
    </visual>
    <collision>
      <origin
        xyz="0 0 0"
        rpy="0 0 0" />
      <geometry>
        <mesh
          filename="package://mark1/src/mark1_description/meshes/Right_Motor_Link.STL"
/>
      </geometry>
    </collision>
  </link>
  <joint  name="Right_Motor_Joint" type="continuous">
    <origin
      xyz="-0.3197 -0.23965 -0.05511"
      rpy="-3.14 3.1415 3.1395" />
    <parent
      link="base_link" />
    <child
```

```
    link="Right_Motor_Link" />
  <axis
    xyz="0 1 0" />
</joint>


<!-- Hokuyo Laser -->
<link name="laser_link">
  <collision>
    <origin xyz="0 0 0" rpy="0 0 0"/>
    <geometry>
  <box size="0.1 0.1 0.1"/>
    </geometry>
  </collision>

  <visual>
    <origin xyz="0 0 0" rpy="0 0 0"/>
    <geometry>
      <mesh filename="package://mark1/src/mark1_description/meshes/hokuyo.dae"/>
    </geometry>
  </visual>

  <inertial>
    <mass value="1e-5" />
    <origin xyz="0 0 0" rpy="0 0 0"/>
    <inertia ixx="1e-6" ixy="0" ixz="0" iyy="1e-6" iyz="0" izz="1e-6" />
  </inertial>
</link>

  <joint name="hokuyo_joint" type="fixed">
  <axis xyz="0 1 0" />
  <origin xyz="-0.1 0 0.19" rpy="0 0 0"/>
```

```
  <parent link="base_link"/>
  <child link="laser_link"/>
</joint>


<!-- Camera -->
<link name="camera_link">
  <collision>
    <origin xyz="0 0 0" rpy="0 0 0"/>
    <geometry>
  <box size="0 0 0"/>
    </geometry>
  </collision>

  <visual>
    <origin xyz="0 0 0" rpy="0 0 0"/>
    <geometry>
  <box size="0.05 0.05  0.05"/>
    </geometry>
    <material name="red"/>
  </visual>

  <inertial>
    <mass value="1e-5" />
    <origin xyz="0 0 0" rpy="0 0 0"/>
    <inertia ixx="1e-6" ixy="0" ixz="0" iyy="1e-6" iyz="0" izz="1e-6" />
  </inertial>
</link>

<joint name="camera_joint" type="fixed">
  <axis xyz="0 1 0" />
  <origin xyz="0.27 0 0.05" rpy="0 0  0"/>
```

```
    <parent link="base_link"/>
    <child link="camera_link"/>
  </joint>

    <!-- Depth Camera -->
<link name="depth_camera_link">

    <collision>
      <origin xyz="0 0 0" rpy="0 0 0"/>
      <geometry>
    <box size="0.05 0.05 0.05"/>
      </geometry>
    </collision>

    <visual>
      <origin xyz="0 0 0" rpy="0 0 0"/>
      <geometry>
    <box size="0.05 0.05  0.05"/>
      </geometry>
      <material name="blue"/>
    </visual>

    <inertial>
      <mass value="0.210" />
      <origin xyz="0 0 0" rpy="0 0 0"/>
      <inertia ixx="1e-6" ixy="0" ixz="0" iyy="1e-6" iyz="0" izz="1e-6" />
    </inertial>
  </link>

<joint name="depth_camera_joint" type="fixed">
    <axis xyz="0 1 0" />
    <origin xyz="-0.1 0 0.4" rpy="0 0  0"/>
```

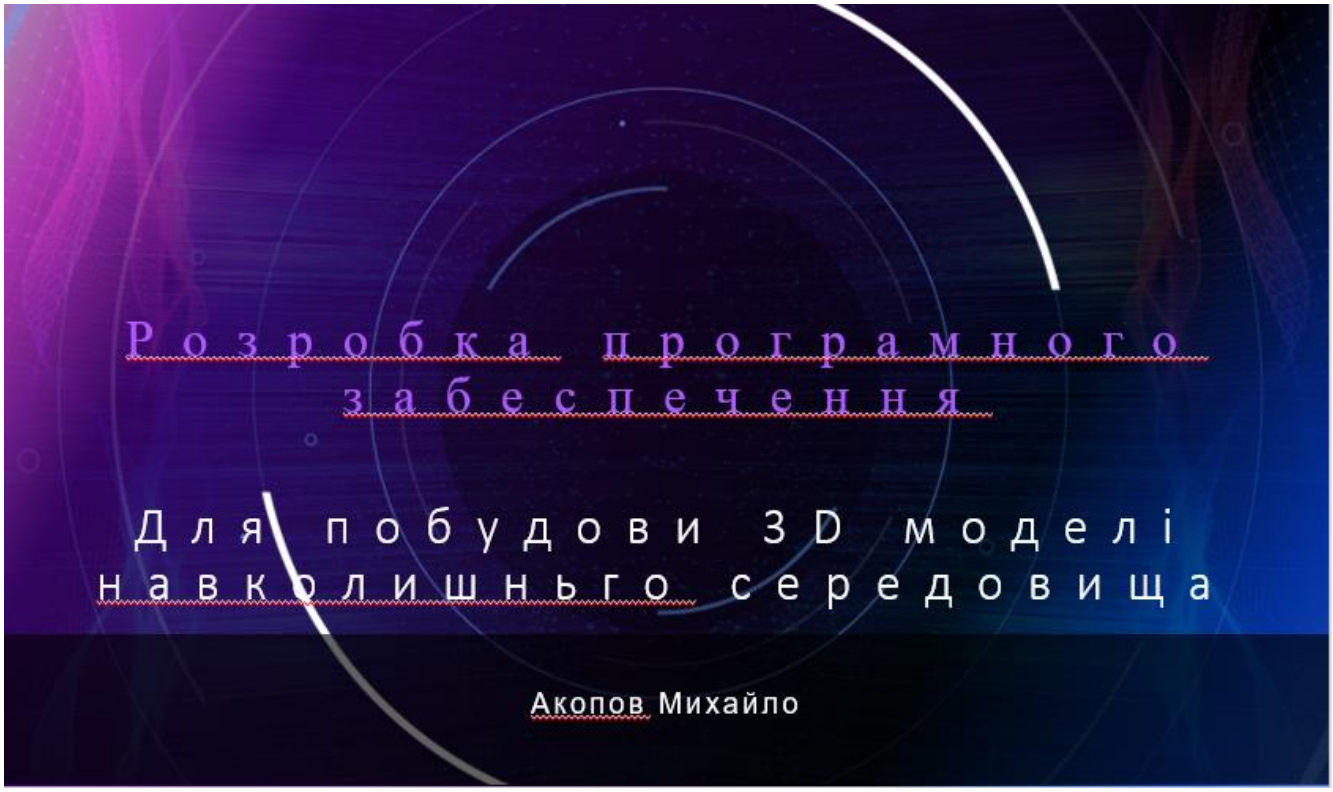
```
    <parent link="base_link"/>
    <child link="depth_camera_link"/>
  </joint>
 <link name="depth_camera_optical">
  </link>
 <joint name="depth_camera_optical_joint" type="fixed">
    <axis xyz="0 1 0" />
    <origin xyz="0 0 0" rpy="-1.57 0 -1.57"/>
    <parent link="depth_camera_link"/>
    <child link="depth_camera_optical"/>
  </joint>

</robot>



<?xml version="1.0"?>
<robot>

  <material name="black">
    <color rgba="0.0 0.0 0.0 1.0"/>
  </material>

  <material name="blue">
    <color rgba="0.0 0.0 0.8 1.0"/>
  </material>

  <material name="green">
    <color rgba="0.0 0.8 0.0 1.0"/>
  </material>

  <material name="grey">
    <color rgba="0.2 0.2 0.2 1.0"/>
```

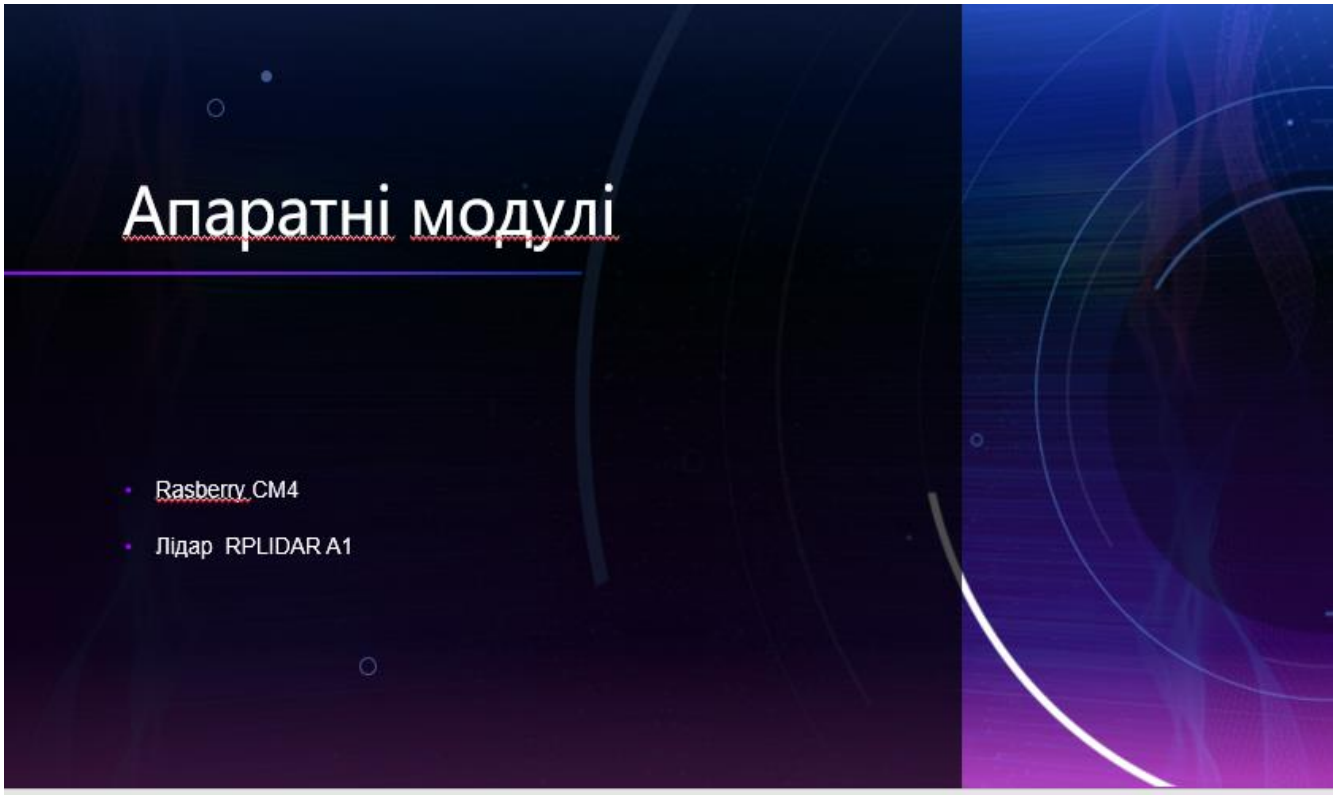
```
  </material>

  <material name="orange">
    <color rgba="${255/255} ${108/255} ${10/255} 1.0"/>
  </material>

  <material name="brown">
    <color rgba="${222/255} ${207/255} ${195/255} 1.0"/>
  </material>

  <material name="red">
    <color rgba="0.8 0.0 0.0 1.0"/>
  </material>

  <material name="white">
    <color rgba="1.0 1.0 1.0 1.0"/>
  </material>

</robot>



<?xml version="1.0" encoding="UTF-8"?>
<launch>

  <arg name="world" default="empty"/>
  <arg name="paused" default="false"/>
  <arg name="use_sim_time" default="true"/>
  <arg name="gui" default="true"/>
  <arg name="headless" default="false"/>
  <arg name="debug" default="false"/>

  <include file="$(find gazebo_ros)/launch/empty_world.launch">
```

```
    <arg name="world_name" value="$(find
mark1)/src/mark1_gazebo/worlds/mark1.world"/>
    <arg name="paused" value="$(arg paused)"/>
    <arg name="use_sim_time" value="$(arg use_sim_time)"/>
    <arg name="gui" value="$(arg gui)"/>
    <arg name="headless" value="$(arg headless)"/>
    <arg name="debug" value="$(arg debug)"/>
  </include>

  <node name="mark1_spawn" pkg="gazebo_ros" type="spawn_model" output="screen"
   args="-urdf -param robot_description -model mark1" />

</launch>
```

ДОДАТОК В

Презентація

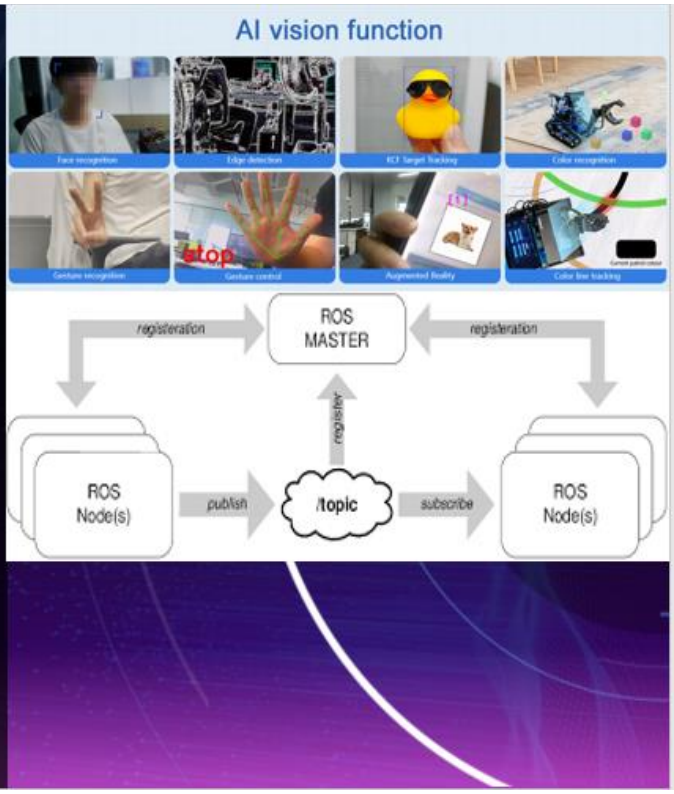

Розробка програмного забезпечення
Для побудови 3D моделі навколишнього середовища
Акопов Михайло


Апаратні модулі
Rasberry CM4
Лідар RPLIDAR A1

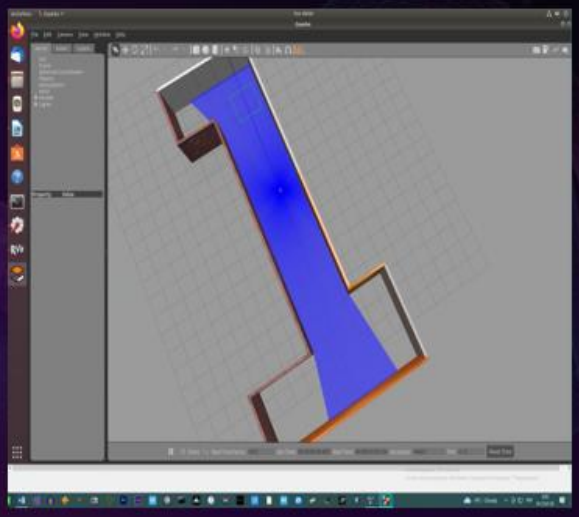
ROS

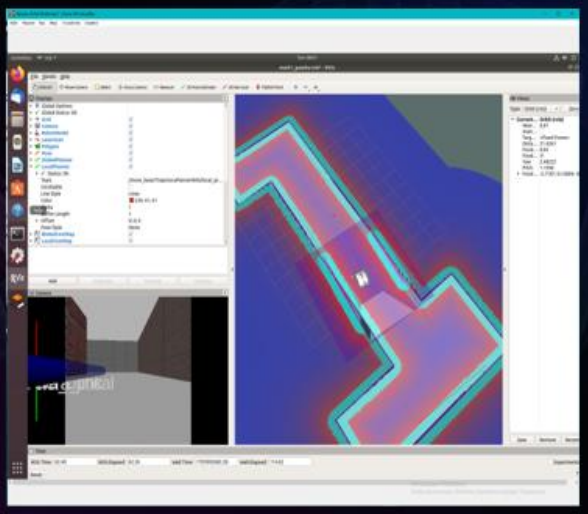
ROS або robot operating system – фреймворк що базується на архітектурі графів

Базові одиниці – ноди та графи

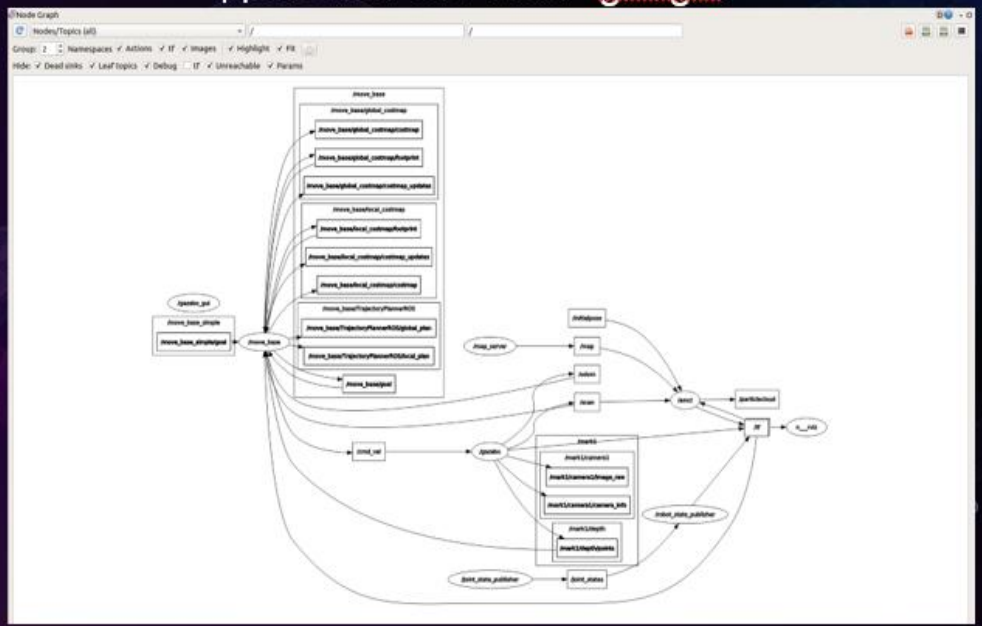


- Gazebo та Rwiz





- Візуалізація процесів за допомогою RQT gragh



№	Позначення	Найменування	Дод. відомості
		Текстові документи	
1	ГЮИК. 505400ПЗ	Пояснювальна записка	41 с., ф. А4
		Додаткові матеріали	
2		Апробація наукової статті	3 с. ф. А4
3		Лістинг програми побудови 3D моделі навколинього середовища мобільного робота	19 с. ф. А4
4		Презентація	3 с. у форматі с. *.ppt

					ГЮИК. 505400ПЗ			
Зм.	Арк.	№ докум.	Підп.	Дата				
Розроб.		Акопов М.Г.			Розроблення автоматизованого модуля ідентифікації об’єктів для багатоцільової платформи Відомість кваліфікаційної роботи	Літ.	Аркуш	Аркушів
Перевір.		Максимова С.С.						1
						ХНУРЕ Кафедра КІТАМ		
Н. контр.								
Затв.		Невлюдов І.Ш.						