METAS Project Report

Note: the code might not be working correctly due to the anonimization procedure.

Edits on the original iRotate Code and Added METAS Code

The edits are described according to the packages in src folder:

1) active_slam:

msg and srv:
Added new 3 messages for gathering path statistics like entropy, path length, number of waypoints etc. in the FSM.

- ***GroupPathInfo.msg***
- ***PathInfo.msg***
- ***SinglePath.msg***

Added these to ***get_best_path.srv*** as response values to service call.

scr:

- Created a copy of the original map_extract.cpp as **map_extract_orig.cpp**
- Made edits to **MapExtract::getBestPathServiceCallback** in the **map_extract.cpp** file to gather and return path statistics mentioned in the above messages.

2) grid_map

link: https://github.com/ANYbotics/grid_map
Added grid_map package for building costmaps. No changes were made to any underlying code in the grid_map package.

3) ipp_custom

The METAS package with the codes and files for building all the costmaps and communicating them with move_base and the FSM.

Srv:
getCostmap.srv is used in the service definition and call of **METAS::getCostmapService** in **transformation.cpp** and **default.cpp**. This service is used by the FSM to request a Costmap array from these nodes and send it to the move_base planner to calculate paths.

src:
transformation_cinf.cpp -> Builds the C_I informative Costmap.

transformation_transfer.py -> Entropy map service.
default.cpp -> Builds the C_{base} by transferring costmap_2d costmap to grid_map representation.

Other evaluatory and miscellaneous files:

getMaps.py -> debug code, to save a snapshot of all the relevant maps for pictures in the paper.
getPathLog.py -> evaluation code, to save the path statistics.

Launch files:
three_map_planner.launch -> used to run 3 global planners while evaluating for path statistics

4) navigation

navigation/global_planner

Srv:
GetPlan_.srv -> a service call added to make A* and Dijkstra planner compatible to work with costmaps of different map representations, like grid_map, outside of costmap2D. used in **GlobalPlanner::makePlanService** of planner_core.cpp

src:
planner_core.cpp ->

- Added a ros param "**use_grid_map**", to make use of grid_map costmap and services inside global_planner whenever set to true. It uses <planner>_gridmap.cpp planners whenever set to true.
- made changes to makePlanService to accommodate the same.

astar.cpp
astar_gridmap.cpp
dijkstra.cpp
dijkstra_gridmap.cpp
-> made similar changes to all the codes:
- for the <planner>gridmap.cpp variants, copied the exact codes of <planner>.cpp files but made changes (variable name changes) to ensure the "costs_grid" integer array variable (grid_map costmap) instead of "costs" char array variable (costmap2D variant).

5) robotino_fsm

src:

three_map_fsm_node.cpp -> runs **three** move_base planners for path statistics.

one_map_fsm_node_default.cpp -> runs **single** move_base planner, subscribes to "**getCostmap_default**" ros service, defined in **ipp_custom/default.cpp**. It is also used to run with costmap2D costmaps, simply by changing the "**use_grid_map**" param in config files.

one_map_fsm_node_transformation.cpp -> runs **single** move_base planner, subscribes to "**getCostmap_transformation**" ros service, defined in **ipp_custom/transformation.cpp**

All fsm **one_map_*.cpp** files have new **LogCurrentStats** / **LogDefaultStats** functions with appropriate global variables defined for logging in paths stats when needed.

Similarly, **three_map*.cpp** has **LogCostmap2DStats(), LogGridmapTransformationStats(), and LogGridmapDefaultStats()** to log path stats for all 3 maps. The path stats are published whenever new paths are created.

Launch Files:

Added three launch files for evaluations:

- **one_map_fsm_default.launch**
- **one_map_fsm_transformation.launch**
- **three_map_fsm.launch**

other files like **two_map_fsm** and **vs_two_fsm,** were created as testing codes and are not needed for evaluations.

6) robotino_simulations

configs:

global_planner_params.yaml -> added a param "**use_grid_map**", explained above. Similar copies: **global_planner_params_orig, global_planner_params_default, global_planner_params_transformation** were created where the only difference is the value of use_grid_map (true or false)

Evaluation Codes:

a) To Run the Experiments:

added configs in **tmux_configs** folder for running experiments using **tmuxinator**.

environment: small_house, cafe

<environment>_costmap2d.yml -> runs for $C_{costmap}$.
<environment>_default.yml -> runs for $C_{gridmap}$.
<environment>_transformation.yml -> runs for C_{l}.

three_map_costmaps.yml -> runs the three_map_fsm along with the code to compare paths.

b) To Get the Results:

results.py -> added a little change to shift the small_house **occupancy.txt** by **2m**. The code for legends was changed to handle **subscript**. This notebook was used to gather the final results. Changes were also added to display the confidence interval in the final plots and new BAC score on only known and occupied cells + output latex friendly results to use.

FSM Management

Much of the FSM management remains the same except some parts which make sure to call the getCostmap srv from default.cpp/transformation.cpp and saving paths stats.

one map (*_default.cpp/*_transformation.cpp)

- the new_goal(), replan_goal() and general_reset() remain the same as irotate.
- the changes in fsm are in:
    - **getBestPath()** -> added lines to save the path statistics (line 790 - 825).
    - **getPlan()** -> added a service call to client **“getCostmapClient”** calling “**getCostmatSrv_***” (described above) to get costmap from default.cpp or transformation.cpp

three maps fsm

- Similar to one map’s, the new_goal(), replan_goal() and general_reset() remain the same.
- Every time the currentPlan is to be calculated, they have been replaced with 3 different calculations of plans named: “**paths_costmap2D**”, “**paths_gridmap_transformation**” and “**paths_gridmap_default**”. which calculates paths on the three different maps using functions call “**getPlan_costmap2D**”, “**getPlan_gridmap_transformation**” and “**getPlan_gridmap_default**”, respectively.
- Added a ros param “**world_path_param**”, which selects which path is to be followed between “**paths_costmap2D**”, “**paths_gridmap_transformation**” and “**paths_gridmap_default**”, and assigns it to **currentPlan**.

Map Building

ipp_custom/transformation.cpp

- It subscribes to Probability Occupancy Map and Costmap2D costmap.
- It has two grid_map objects, ***map*** (with size and origin equal to Probability Occupancy Map) and ***Grid_costmap*** (with size and origin equal to Costmap2D costmap)**.**
- The node begins with a spinner with **METAS::Update**, which updates all the maps every time they are received and normalizes their value between [0, 1].
- Whenever the **METAS::getCostmapService** is called, it calls **METAS::Calculate()**, which begins building the costmap.
- First, it calculates the Entropy Map (**map[“transfer_map”]**) in through a service call to transformation_transfer.py
- After which, it transfers the entropy map from ***map to Grid_costmap*** in Grid_costmap[“Entropy_Map”], according to the origin of both the maps in the **METAS::Transfer()** function.
- Next it calculated the obstacle map in Grid_costmap[“Obstacle_Map”] and combines both maps in the **METAS::Combine()** function.

- After this, the final costmap is stored in the "**costs_grid.data**" integer array variable and saved as a response to the initial service call.

ipp_custom/default.cpp
The order of service and function calls remains the same, except the METAS::averageEntropy() is never called.

- In the **METAS::Calculate** only METAS::Combine is called.
- In **METAS::Combine**, only **Gridmap_costmap["Inflation_Map"]** from costmap2D is used to build the costmap $\mathbf{C}_{\mathbf{gridmap}}$.

Additional Details:

- Change the directory locations in evaluation codes and tmuxinator configs so they run properly.
- There are some unnecessary variables or functions like costs_grid[1000*1000] in fsm nodes or LogCurrentStats in three_map_fsm, which are never called, but dont disturb the flow of fsm.
- for **ipp_custom/transformation_cinf.cpp** or **ipp_custom/default.cpp** , the costmap2D subscriber needs to ensure its subscribing to the correct move_base costmap topic.