ROPEN: Robot Optimal Planner with ENvironmental-learning

[Yushan Chen](http://hyness.bu.edu/yushan)1, Jana Tumova2, [Alphan Ulusoy](http://www.alphanulusoy.com/alphan/index.html)3, Calin Belta4

1 Department of Electrical and Computer Engineering, Boston University, Boston, USA, Email: [yushanc@bu.edu](mailto:yushanc@bu.edu)  
2 Faculty of Informatics, Masaryk University, Brno, Czech Republic, Email: [tumova@bu.edu](mailto:tumova@bu.edu)  
3 Division of System Engineering, Boston University, Boston, USA, Email: [alphan@bu.edu](mailto:alphan@bu.edu)  
4 Department of Mechanical Engineering, Boston University, Boston, USA,

Robot Optimal Planner with ENvironmental-learning (ROPEN) is a software package for optimal path planning for a robot moving in a partitioned environment with partially unknown, stochastically changing elements. Specifically, we consider an indoor-like environment where the regions in the environment might be separated by doors, which are modeled as Markov Chains. We assume that the robot does not know the door models, on the other hand, it can observe the statuses of the doors (i.e., open or closed) at each time instance. The robot is equipped with a set of motion primitives and modeled as a game transition system. The robot needs to accomplish a persistent surveillance type of mission (e.g., keep monitoring room 1 and room 2 and gathering information, after gathering enough information, go back to base and upload the information, etc). ROPEN runs an online algorithm, which learns the door behavior models and synthesizes control policy for the robot to accomplish the mission. As the door models are incrementally learned, the performance of the robot keeps being improved and it will converge to the optimal one when the door models are fully learned. ROPEN is an implementation of the computational framework proposed in:

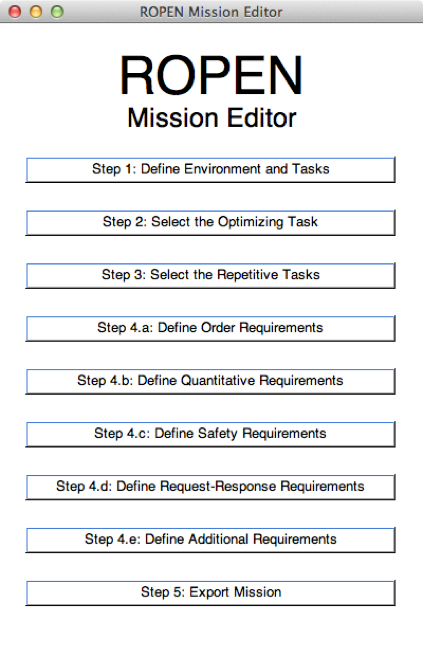
[1] Yushan Chen, Jana Tumova, and Calin Belta. "LTL Robot Motion Control based on Automata Learning of Environmental Dynamics", in IEEE International Conference on Robotics and Automation (ICRA), St. Paul, USA, 2011

[2] Yushan Chen, Jana Tumova, Alphan Ulusoy, and Calin Belta. "Temporal Logic Robot Control based on Automata Learning of Environmental Dynamics", IJRR

ROPEN contains a mission editor which allows the user to input the robot mission in a more convenient way. ROPEN utilizes the ltl2dstar tool [[3]](https://sites.bu.edu/hyness/ropen/#3) to generate the Deterministic Rabin Automaton from the LTL formula. In addition, ROPEN includes a simulator (implemented in Objective-C), which can be used to demonstrate runs of the robot.

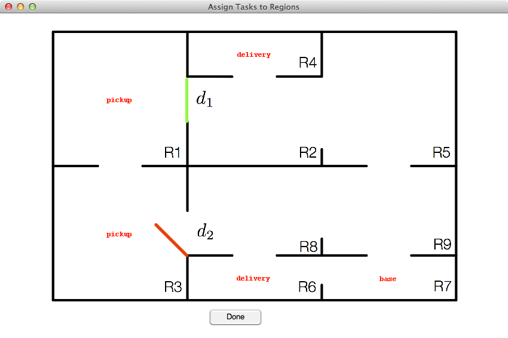
Ropen Mission Editor:

Our implementation also includes a graphical user interface, which can be used to define different missions with ease.

  
Fig 1: ROPEN Mission Editor Main Window

**A typical use of the ROPEN mission editor is as follows:**

1. the user defines which propositions are satisfied at each region in the environment

  
Fig 2: ROPEN GUI: Proposition Input Window

1. then, the user defines the robot tasks:

A screenshot of a computer

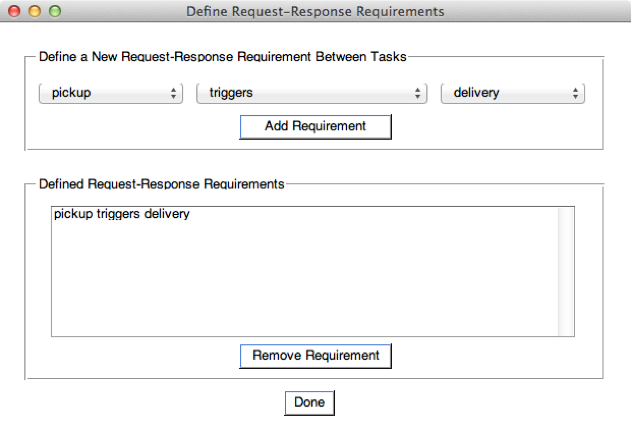
Description automatically generated  
Fig 3: ROPEN GUI: Optimizing Task Input Window

A screenshot of a computer

Description automatically generated  
Fig 4: ROPEN GUI: Repetitive Tasks Input Window

A screenshot of a computer

Description automatically generated  
Fig 5: ROPEN GUI: Additional Requirements Input Window

  
Fig 6: ROPEN GUI: Request-Response Requirements Input Window

1. finally, the user saves the mission specification in a file, which is an input to our Matlab implementation.

How to define new environment:

We must note that ROPEN mission editor is not limited to the environment that we consider in our paper. The mission editor can easily accommodate different environments as long as the corresponding environment definition files are provided. Locations and contents of these files are as follows:  
data/env\_def.png: A bird-eye view of the environment.  
data/env\_def.txt: Definition of each region in the environment. Each line defines a rectangular region in the environment and the format of the lines is: REGION\_NAME LOWER\_LEFT\_X\_COORD LOWER\_LEFT\_Y\_COORD UPPER\_RIGHT\_X\_COORD UPPER\_RIGHT\_Y\_COORD.

How to Run:

1. Extract the contents of the archive (ropen\_mission\_editor.zip & Ropen\_implementation.zip) to a directory and navigate to that directory in Matlab
2. Run the mission editor by typing "main\_window" and hitting enter
3. Define the mission following the steps defined in the GUI (Please see the sample video for defining the specification that we used in our IJRR paper)
4. Click "Export Mission" when done (the output files that can be used by our ROPEN implementation is now generated, these files are phi.ltl: Mission specification in prefix notation, data/phi\_infix.ltl: Mission specification in infix notation, and data/region\_assignments.txt: Assignment of propositions to regions)
5. Define the game transition system and the door models in an M-file
6. Run "main.m" script in the ROPEN Implementation (Matlab), which imports the M-files and the text file described above and then executes the algorithms online to generate the control policy.
7. The output of the "run\_parser.m" script can be used to define the robot trajectories to be run in the Simulator or in the robotic test-bed (currently hardcoded in robot.m).

Source Download:

You can download the source code of ROPEN along with the Xcode simulator if you want to examine and/or modify the code. ROPEN is a Matlab implementation of the algorithms present in. Make sure that your system meets the requirements given below before attempting to run any of the programs.

**Requirements**

Simulator:

1. Mac OS X 10.6 or higher,
2. For Xcode projects: Xcode 4

The algorithms implemented in Matlab requires:  
Matlab 2010a or later

**Sources:**

[ROPEN\_implementation.zip](https://calinbelta.com/wp-content/uploads/2024/01/ROPEN_implementation.zip)

[ropen\_mission\_editor.zip](https://calinbelta.com/wp-content/uploads/2024/01/ropen_mission_editor.zip)

[learning\_simulator.zip](https://calinbelta.com/wp-content/uploads/2024/01/learning_simulator.zip)

**References:**

[1] Yushan Chen, Jana Tumova, and Calin Belta. " LTL Robot Motion Control based on Automata Learning of Environmental Dynamics ", in IEEE International Conference on Robotics and Automation (ICRA), St. Paul, USA, 2011

[2] Yushan Chen, Jana Tumova, and Calin Belta. " Temporal Logic Robot Control based on Automata Learning of Environmental Dynamics " (Conditionally Accepted by IJRR)

[3] Klein, J. (2012), " ltl2dstar – LTL to deterministic Streett and Rabin automata tool ". <http://www.ltl2dstar.de/>