

Extended LTLvis Motion Planning Interface

by

Wei Wei

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved May 2016 by the
Graduate Supervisory Committee:

Georgios Fainekos, Chair
Hani Ben Amor
Yu Zhang

ARIZONA STATE UNIVERSITY

August 2016

ABSTRACT

Robots are becoming an important part of our life and industry. Although a lot of robot control interfaces have been developed to simplify the control method and improve user experience, users still cannot control robots comfortably. With the improvements of the robot functions, the requirements of universality and ease of use of robot control interfaces are also increasing. This research introduces a graphical interface for Linear Temporal Logic (LTL) specifications for mobile robots. It is a sketch based interface built on the Android platform which makes the LTL control interface more friendly to non-expert users. By predefining a set of areas of interest, this interface can quickly and efficiently create plans that satisfy extended plan goals in LTL. The interface can also allow users to customize the paths for this plan by sketching a set of reference trajectories. Given the custom paths by the user, the LTL specification and the environment, the interface generates a plan balancing the customized paths and the LTL specifications. We also show experimental results with the implemented interface.

ACKNOWLEDGEMENTS

I would like to thank my advisor and committee chair, Dr. Georgios Fainekos, for providing me an opportunity to work on this interesting research topic and many useful ideas and inspiring thoughts during the research. I would like to thank the committee members Dr. Hani Ben Amor and Dr. Yu Zhang for their valuable guidance.

I would also like to thank my labmate Kangjin Kim, who provided a lot support and useful resources. I would like to thank my other labmates Adel Donkachi, Bardh Hoxha and Erkan Tuncali who I have had a wonderful time to work together with.

TABLE OF CONTENTS

	Page
LIST OF TABLES.....	v
LIST OF FIGURES	vi
CHAPTER	
INTRODUCTION.....	1
CONTRIBUTIONS.....	4
PRELIMINARY.....	6
Linear Temporal Logic	6
Syntax.....	6
Semantics	8
LTL Planning.....	9
Graphical Language for LTL.....	11
Android Platform	14
Architecture.....	14
Activity lifecycle	15
LTLvis	16
PROBLEM DESCRIPTION	18
Problem Overview	18

CHAPTER	Page
Solution Overview	23
EXTENDED LTLvis GUIDE	26
Load Map and Create Roadmap	26
Sketch Path	29
Edit Specifications	44
PLANING USING E-LTLvis	46
Experiments	55
Experiment 1	56
Experiment 2	61
Extra Experiments	66
RELATED WORK	69
Control Interface	69
LTL Planner	70
Dynamic Time Warping	71
Others	71
CONCLUSIONS AND FUTURE WORK	69
REFERENCES	76
APPENDIX	
A PROCEDURE OF ALG. 1	80

LIST OF TABLES

Table	Page
1: LTL Semantics.....	8
2: The Meaning of Icons in Graphical Language for LTL	13
3: Path p_0 and its Possible BMP Candidates.....	34
4: All Cells in BMP are Set to Empty and All Cells in CWPD are Set to Infinite	37
5: Step 0-2 and 0-3	38
6: Step 1-2 to 1-5.....	39
7: Step 2-2 to 2-5.....	41
8: Final BMP and CWPD Tables.....	43
9: The Options in The Options Panel.....	46

LIST OF FIGURES

Figure	Page
1: A Scenario that Firefighters Control Robots to Search Fire Site in a Big Building.	2
2: A Sample Screenshot of Ltlvis	4
3: The Allowed Combinations of Boolean and Temporal Operators Over an Edge	12
4: Lifecycle of an Android Program	16
5: Sample Screenshot of Ltlvis	17
6: Data Collection Vehicle Have to Options (π_1, π_2) to Visit P3.	19
7: By Adding Node Q4, The V Can Visit Q4, Then Visit Q3.	19
8: A Complicated Oath π_2 Including Cycles.	20
9: π_2 Is Visiting a Blocked Area	21
10: The Vehicle Has Two Option Paths to Follow When Leaving Q3.	21
11: The Graphical Expression Of $q_1 \rightarrow X q_2 \vee q_4 \wedge GF(q_3 \wedge Gq_3 \rightarrow X q_2 \vee q_4)$	22
12: The Flowchat of the Interface Process.....	23
13: The Five Main Modules of the Interface	25
14: The User is Asked to Load a Map.	26
15: The Starting Interface of Roadmap Mode	27
16: Long Press to Add a Node	27
17: The Selected Node is Marked Green	28
18: When Adding Edges, the Nodes Got Selected will be Marked Red and Become the Green Node's Neighbors.....	28
19: The Staring Interface of Sketch Mode.....	29
20: In Sketch Mode, the Roadmap Data will be Hidden	30

Figure	Page
21: Black Nodes and Edges: the Graph of a Simple Environment. Green Path: pu . Blue Nodes: p_0	31
22: Distance dls of C to Edge(A, B) Equals to Distance dl of C to Line(A, B).	34
23: The CWPDs of p_1, p_2, p_3	35
24: Step 0	38
25: Step 1. $Dab1$ Denotes The Distance from $n1$ to Edge eab	39
26: Step 2	40
27: The User Sketched Path and its Bmp.....	44
28: The Default LTL Symbol Between Two Nodes	45
29: Long Press the Node to Display the Option Panel.....	45
30: A Sample LTL Specification	47
31: Left: TS. Right: Buchi Automaton	48
32: Local TS.....	49
33: The Product Automaton.....	50
34: The Path Generated Using Shortest Path.....	51
35: Extra Path Requirement for $q_0 \wedge GFq_1 \wedge Fq_2$	52
36: The Path Generated by the Extended Planner	52
37: The Experiment Environment.....	55
38: The Graymap for the above Environment	56
39: Two Required Inputs from E-Ltlvis.....	57
40: The Turtlebot Started from Q_0 and Went North.....	58
41: The Turtlebot Reached the Turn Point and Was Ready to go East.	58

Figure	Page
42: The Turtlebot Reached the Point in the North of Q2 and Continued to go East.	59
43: The Turtlebot Reached the Turn Point and Was Ready to go South.	59
44: The Turtlebot Reached Q1 and Was Ready to go to Q2.	60
45: The Turtlebot Reached Q2. Task Completed.	60
46: Two Required Inputs for E-Ltlvis.	61
47: The Turtlebot Started from Q0 and Headed East.	62
48: The Turtlebot Reached Q1.	62
49: The Turtlebot Continued to go East.	63
50: The Turtlebot Made a U Turn around The Legs of the Table.	63
51: The Turtlebot Headed West and Reached Q2.	64
52: The Turtlebot Headed North and Reached the Turn Point.	64
53: The Turtle Reached Q1 Again.	65
54: The Turtle Continued Above Actions Until we Manually Stopped the Planner.	66
55: Experiment Environment and Scanned Greymap.	66
56: Two Required Inputs from E-Ltlvis.	67
57: Robot Trajectory for Experiment 1.	67
58: Two Required Inputs for E-Ltlvis.	68
59: Robot Trajectory for Experiment 2.	68

Chapter1

INTRODUCTION

There is a constant progress in robotics every day. Humans are stepping into to a new society - a society in which robots play significant a role in Human's daily life. Tesla Factory uses robots to assemble car parts¹; Amazon deploys robots to manage their warehouses²; many people buy home cleaning robots to clean their carpets. From industrial machines to cleaning robots, the ability of robots to accomplish complex tasks is increasing at a high rate.

In the development of these robots, experts need to address not only the hardware design, but the control software as well. Robots designed for different purpose may require different control methods. Let us consider few control methods: To order a robot to move forward, a forward button is sufficient. If the robot is moving in a 2D plane, a joystick can be a good option. If the robot is required to reach a certain destination autonomously, a touch screen is a much easier option to locate the desired coordinates on a displayed map. Lastly, if there is also a specific path that the robot should follow, a sketch interface should be handy to define this requirement.

In order to fulfill control requirements, graphic control interfaces appear to be an effective way to control mobile robots [1, 2]. With a graphical interface, users can control multiple robots more conveniently [3] by clicking a predefined button instead of writing a robot control program. For example, consider a scenario that firefighters are searching for fire spots in a big building. Before going into some dangerous spots, they deploy an

¹ <http://www.fool.com/investing/general/2014/08/24/3-surprising-things-you-may-not-know-about-tesla-m.aspx>

² <http://www.chonday.com/Videos/how-the-amazon-warehouse-works>

unmanned robot using a graphical interface with the building blueprint to search this area. After quickly and concisely defining the task by sketching the path on the interface, firefighters can command the robot to follow such a path autonomously.

Even though a graphical interface is intuitive for non-experts and provides a convenient way to define each task for different robots, complex tasks can hardly be described only with a few buttons. Consider the following task: The robots should reach destination *B* by passing through *A* and avoid *C* only until *A* has been reached by at least one robot (in Figure 1). In this case, we need a formal representation to represent the task.

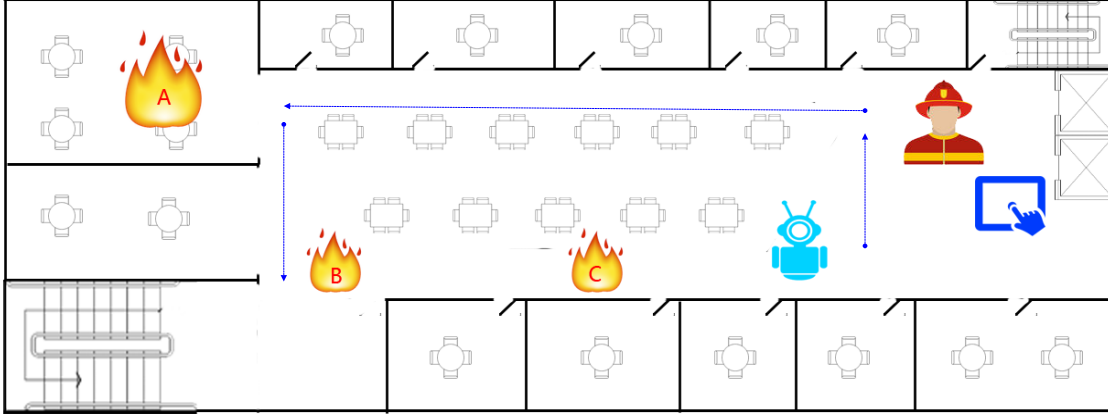


Figure 1: A scenario Firefighters control robots to search for fire sites in a big building.

Linear Temporal Logic (LTL) is a formal language that enables the specification of time related robot tasks. As it is concise, LTL has become a popular specification language [4, 5, 6, 7]. Furthermore, the logic is compact and can be used with a whole variety of tools available for planning. Therefore, LTL can easily capture human language intention. For example, in a firefighter scenario, firefighters need to control robots to explore the burning building. The robots have to sequentially visit locations *A*, *B*, *C*. This requirement can be translated into LTL specification as in the following formula.

$$F(A \wedge F(B \wedge F(C)))$$

The ‘F’ operator means that at some time point in the future, the value will become true. More temporal operators will be introduced in later sections.

However, writing planning tasks in LTL is a challenging prospect for people who do not have formal language expertise. There are only few works about graphically specifying temporal logic formulas (see [5, 8] and the references therein) or translating English language to temporal logic formulas (see [9, 10, 11] and the references therein).

In path planning, users may have extra requirements besides the starting point and final destination. Previous results from [12, 13] have provided solutions for such cases. There, using a sketch based interface, users can specify the exact path that the robot is required to follow. They also show that sketch-based interfaces are generally more efficient than button-based interfaces. In order to satisfy these extra requirements, the sketched path must be added to the LTL specification. We need to keep the interface user friendly while adding new features to it. This increases the challenge of designing a simple but multi-functional interface layout.

Fortunately, there is existing work which graphically visualizes the LTL specification [5]. By defining nodes and edges as atomic propositions and temporal operators, LTL specification can be transferred from a formula into a graph. Finding the optimal path between the nodes in this graph will be the one of the challenges of this research.

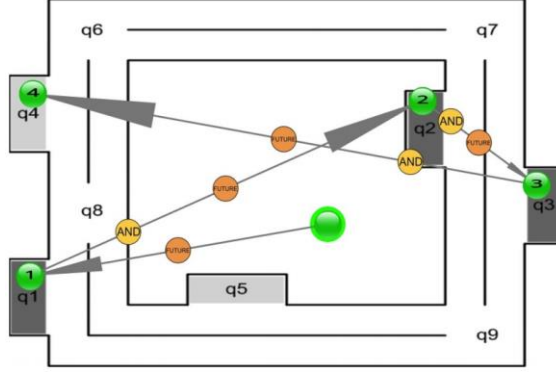


Figure 2: A sample screenshot of LTLvis

When using a graphic interface, it would be more efficient and useful to be able to draw preferred trajectories for the robot as for example in papers [12, 13]. In these works, using a sketch based interface, users can specify the exact path that the robot is required to follow by drawing a curve from start to an end point.

For the above firefighter example, if firefighters control robots using a sketch based interface, it will be more convenient and faster to describe the task than pressing a few buttons which is time consuming in such emergency scenarios. Besides, if the interface supports LTL specification, firefighters can describe the task more precisely than natural language.

Contributions

The main contribution in this research is to combine an easy-to-use sketch-based interface with the expressive power of LTL and to improve the LTL path planner provided by [14] for this hybrid interface. A secondary contribution is that we provide a greedy algorithm to identify the closest path on a directed topologically grounded graph to a hand Drawn curve. We remark that our algorithm allows the path to be cyclic.

Preliminary results of this thesis were published in [15]:

- W. Wei, K. Kim and G. Fainekos, "Extended LTLvis Motion Planning interface," in *Conference on Systems, Man, and Cybernetics (SMC 2016)*, 2016.

Chapter 2

PRELIMINARY

In this section, we will first cover the graphical language for LTL. Then, we will review LTL path planning.

Linear Temporal Logic

Temporal logic is the logic in terms of time. The statement “I am rich now” can be an instance of temporal logic. It is true in time range “now”. This true value may not hold if the time range is beyond “now”. Similarly, “I was rich until the great depression” and “I will be rich eventually” are true referring to “until the great depression” and “eventually” respectively.

Temporal logic is used to define properties of requirements of robotic system. It has similar form as structured English. As it can formulate the properties of the specification, it is much precise and unambiguous than structured English.

Linear temporal logic (LTL) is first proposed for the formal verification of computer programs by Amir Pnelli [16]. It is a modal temporal logic with infinite sequence of states. Each point in this sequence has unique successor.

Syntax

LTL formula consists of:

- A finite set of atomic propositions (AP)
- A set of logic operators
 - Negation (\neg)
 - Conjunction (\wedge)
 - Disjunction (\vee)

- Implication (\rightarrow)
- A set of temporal operators
 - always (G)
 - eventually (F)
 - next (X)
 - until (X)

With these atomic propositions and operators it is possible to formulate the truth values of propositions over time. By combining several operators together, it can generate more complicated operators. Because of this property, LTL can be extended to express high level specifications like task plan for robot navigation.

LTL formula can be defined as follows:

- If $p \in AP$ then p is an LTL formula;
- If ψ and φ are LTL formulas then $\neg \psi$, $\psi \vee \varphi$, $X \psi$ and $\psi \cup \varphi$ are LTL formulas.

Therefore, the grammar can be defined as:

$$\phi ::= true \mid \neg \psi \mid \psi \vee \varphi \mid X \psi \mid \psi \cup \varphi$$

These basic operators can be further combined into always (G) and eventually (F) operators.

$$F \psi = true \cup \psi$$

$$G \psi = \neg F \neg \psi$$

$F \psi$ means that ψ is true at some point of time in the future.

$G \psi$ means that ψ is always true.

Once F and G are formally defined, they can be combined with other operators to form composite operators.

- $\psi \rightarrow F \varphi$ means if ψ is true, φ will be true eventually.
- $GF \psi$ means ψ is true infinitely often.
- $FG \psi$ means ψ will be true eventually. And from that point of time, ψ is always true.
- $\psi \cup \varphi$ means ψ is true at current time and stays true until φ is true.

These composite propositions can also be joined with conjunction and disjunction operators to build higher level composite propositions.

Semantics

Let ψ be an LTL formula over AP. The LTL property induced by ψ is:

$$Words(\psi) = \{\sigma \in (2^{AP})^\omega \mid \sigma \models \psi\}$$

Where the satisfaction relation $\models \subseteq (2^{AP})^\omega \times LTL$ is the smallest relation with the properties in below table where $\sigma = A_0A_1A_2 \dots \in (2^{AP})^\omega$, $\sigma[j \dots] = A_jA_{j+1}A_{j+2} \dots$ is the suffix of σ starting in the (j+1) symbol A_j .

$\sigma \models true$	
$\sigma \models a$	Iff $a \in A_0$
$\sigma \models \varphi_1 \wedge \varphi_2$	Iff $\sigma \models \varphi_1$ and $\sigma \models \varphi_2$
$\sigma \models \neg \varphi$	Iff $\sigma \not\models \varphi$
$\sigma \models X \varphi$	Iff $\sigma[1 \dots] = A_1A_2A_3 \dots \models \varphi$
$\sigma \models \varphi_1 \cup \varphi_2$	Iff $\exists j \geq 0. \sigma[j \dots] \models \varphi_2$ and $\sigma[i \dots] \models \varphi_1$, for all $0 \leq i < j$

Table 1: LTL semantics

LTL Planning

Path planning is the problem of finding a path between a start position and an end position. Temporal logic path planning is the path planning problem whose result, i.e., path must satisfy a temporal logic requirement. The basic theory on temporal logic planning is described in [4] [17] [18]. First we need to represent the environment as a discrete graph. Here, we introduce the definition of transition system.

Definition 1. (TS) *A transition system is a tuple*

- Q_{TS} is a set of states. It represents the accessible area in the graph.
- $q_{init} \in Q_{TS}$ is the starting state.
- $\delta_{TS} \subseteq Q_{TS} \times Q_{TS}$ denotes the transition relation between two states.
- Π is a finite set of atomic propositions.
- $h: Q_{TS} \rightarrow 2^\Pi$ is a function labeling areas in the environment with atomic propositions.

$w_{TS}: \delta \rightarrow \mathbb{N}$ is the weight assigned to each transition.

We denote a finite path on the transition system as $p = q_0, q_1, \dots, q_n$, where $q_0 = q_{init}$, $q_k \in Q_{TS}$ and $(q_k, q_{k+1}) \in \delta$. The result generated from running this path is a word $v_0 v_1, \dots$ where $v_k = h(q_k)$ is the set of atomic propositions satisfied at q_k .

After discretizing the environment into a transition system, we also need to convert the specification into the same format. Thanks to the tool³ provided by [19], we can easily convert any LTL formula into a Büchi automaton. We introduce the definition of a Büchi automaton.

³ lt2ba is a tool accepting LTL formulas as input and returning a Büchi automaton as output

Definition 2. (BA) A Büchi automaton is a tuple

$B := (Q_{BA}, Q_{init}, \delta_{BA}, \Sigma, F_{BA})$, where

- Q_{BA} is a set of states.
- Q_{init} is a set of initial states.
- $\delta \subseteq Q_{BA} \times \Sigma \times Q_{BA}$ is a transition relation.
- Σ is the input alphabet.
- F_{BA} is a set of accepting states.

For a run of input word $W = \omega_0 \omega_1 \dots$ on the Büchi automaton where $w_i \in \Sigma$, the resulting sequence would be $r = s_0 s_1 \dots$, where $s_i \in Q_{BA}$ and $(s_i, \omega_i, s_{i+1}) \in \delta_{BA}$.

Now we have both TS and BA in a graph format. The goal is to find a resulting sequence $r = c_0 c_1 \dots$ where $c_i := (q_j, s_k)$ and $q_j \in Q_{TS}, s_k \in Q_{BA}$. The resulting sequence should be valid in TS and ending at one accepting state in BA. Hence, we need to construct a product automaton $P := TS \times BA$.

Definition 3. (PA) the product automaton $P = TS \times BA$ between the transition system

$TS := (Q_{TS}, q_{init}, \delta_{TS}, \Pi, h, w_{TS})$ and Büchi automaton $BA := (Q_{BA}, Q_{init}, \delta_{BA}, \Sigma, F_{BA})$ is a tuple

$P := (S_P, S_{P0}, \delta_P, w_P, F_P)$, where

- $S_P = Q_{TS} \times Q_{BA}$ is a finite set of states.
- $S_{P0} = \{q_{init}\} \times Q_{init}$ is the set of initial states.
- $\delta_P \subseteq \delta_{TS} \times \delta_{BA}$ is a transition relation and $((q_i, s_i), (q_j, s_j)) \in \delta_P$ if and only if $(q_i, q_j) \in \delta_{TS}$ and $(s_i, \omega_i, s_j) \in \delta_{BA}$.
- $w_P((q_i, s_i), (q_j, s_j)) = w_{TS}(q_i, q_j)$ is a weight function.

- $F_P = Q_{TS} \times F_{BA}$ is a set of accepting states.

The set of final states F_P of the product automaton represents the ultimate goal of the planning path. Then we can reduce the problem of LTL path planning into finding the optimal path on a graph given a starting position. At this level, many methods can be utilized such as A^* , DFS, Dijkstra etc. For example, if the resulting path is $(q_0, s_0), (q_1, s_1) \dots (q_n, s_n)$, then the actual path on the transition system (robot workspace) will be $q_0, q_1 \dots q_n$.

Graphical Language for LTL

Temporal logic is a logic that describes events in time. Linear Temporal Logic (LTL) is a modal temporal logic reasoning over an infinite sequence of states [17]. This section mainly introduces the research work by Srinivas, et al on defining a graphical language [5]. In their work, the authors provide a graphical representation of an LTL formula in a 2D space. The graph G is a tuple $(V, E, v_0, c, L, \Lambda, x)$ where,

- V is the set of nodes.
- $E \subseteq V \times V$ is the set of edges.
- $v_0 \in V$ is the start node.
- $c: V \rightarrow \{\text{green}, \text{red}\}$ is a function that colors each node either green or red, which corresponds to visiting or avoiding a node⁴.
- $L: V \rightarrow \Phi_{\mathbb{B}}(\tau)$ labels each node with an LTL formula over the set of propositions Π .
- $\Lambda: E \rightarrow BO_1 \times BO_2 \times TO_2 \times TO_1$ is a function that labels each edge on the graph with one or more Boolean or temporal operators. In detail:

⁴ Icons can be added to help people with color blindness.

- $BO_1 = \{\text{AND, OR}\}$
- $BO_2 = BO_1 \cup \{\varepsilon^5, \text{IMPLIES}\}$
- $TO_1 = \{\varepsilon, \text{FUTURE, ALWAYS}\}$
- $TO_2 = TO_1 \cup \{\text{NEXT, UNTIL}\}$
- $x: V \rightarrow \mathbb{R}^2$ is the position of the node on the map or on the image.

As BO_1 is always implicitly used to connect consecutive propositions, it is not included when forming the graph. Figure 3 below is the flowchart of possible values of Λ . Table 2 explains the icons shown in Figure 3.

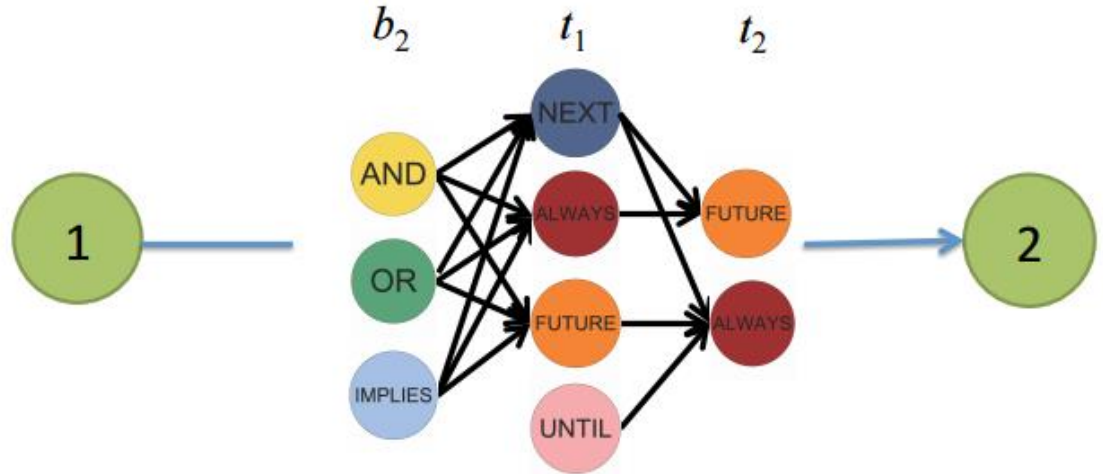





Figure 3: The allowed combinations of Boolean and temporal operators over an edge (reproduced from [5]).

	A node must be visited or an action must be performed
	A node must be avoided or an action must be forbidden
	Conjunctively connect two nodes or two actions

⁵ ε denotes an empty symbol.









	Disjunctively connect two nodes or two actions
	Conjunctively connect previous node and specification corresponding to the next node.
	Disjunctively connect previous node and specification corresponding to the next node.
	If the starting node or action is satisfied, the pointing node or action should be satisfied.
	The pointing node or action should be satisfied at next step
	The pointing node or action should be satisfied sometime in the future.
	The pointing node or action should be satisfied from now on.
	The starting node or action should be satisfied until the pointing node or action is satisfied.

Table 2: The meaning of icons in graphical language for LTL

For example:

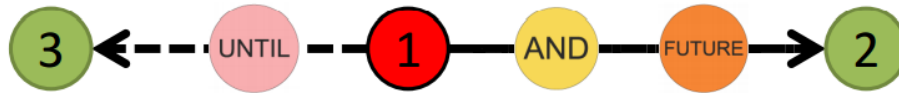
- $\neg q_1 \wedge F q_2 : BO_2 = \text{AND}, TO_2 = \text{FUTURE}, TO_3 = \varepsilon$



- $(\neg q_1 \wedge F q_2) \vee (\neg q_1)U q_3 :$

$$\neg q_1 \wedge F q_2 : BO_2 = \text{AND}, TO_2 = \text{FUTURE}, TO_3 = \varepsilon$$

$$(\neg q_1)U q_3 : BO_2 = \text{OR}, TO_2 = \text{UNTIL}, TO_3 = \varepsilon$$



More examples will be given in the chapter 4.

The theory has been implemented and tested in LTLvis which will be talked about later.

Android Platform

Android is an open source operating system from Google. It is now wildly used in varies of devices including smart phones, table, smart watch, smart TV and smart car. This operating system are mainly running on ARM framework based processors and requires less RAM and processing power which tremendously reduces the hardware cost. As its low doorsill, many manufacturers are able to produce Android device nowadays. Low cost and mass production make these devices affordable to most of business, engineering and academic customers.

Architecture

Android operating system is based on Linux kernel and primarily designed for touchscreen devices such as phones and tablets. Although most Android devices are based on ARM architecture, it also officially supports x86 and MIPS architecture.

Today, tons of developers are building applications for Android device. Java is the preferred programming language. As the development of Android system, more and more programming languages are supported such as C/C++ and GO language. The research work of this thesis is based on Java as its programming environment is easiest to set up. It is similar to desktop java application; the source code of android application is first compiled

into bytecode. Instead of running these bytecode on Java Runtime Environment, Google provided their own runtime environment called Dalvik which is a process virtual machine. Android application was installed on Android platform as a form of bytecode until the release of Android 4.4. In this new release, users have options to further compile the code into machine code before the installation. These machine code will then run on the new runtime environment called Android Runtime (ART). This makes the program run much faster than before, but new runtime environment consumes more memories. As the development of hardware, memory cost is no more expensive and more devices are adopting large size of memory. From the release of Android 5.0, ART is the enabled by default and becomes the only runtime option.

Activity lifecycle

An activity is a single, focused thing that user can do. Every Android application contains at least one activity if the application need to interact with users. Figure 4 shows the lifecycle of an activity during the runtime.

Android manages activities in a stack. When a new activity is created or an existing activity is required, this activity will go to the top of the stack. If the activity (A_1) is new created, *onCreate* function will be called to initialize all necessary resource. After executing *onStart* and *onResume* function, A_1 is able to run in the foreground. When another activity (A_2) is requested, current activity A_1 will call *onPause* function and then be paused. If A_2 is running at a full screen and A_1 is no more visible, A_1 will call *onStop* to enter sleep cycle. If A_1 is requested while it is sleeping, it calls the *onRestart* and *onStart* function to go back to the foreground screen. Otherwise A_1 will be removed from memory or even shut down.

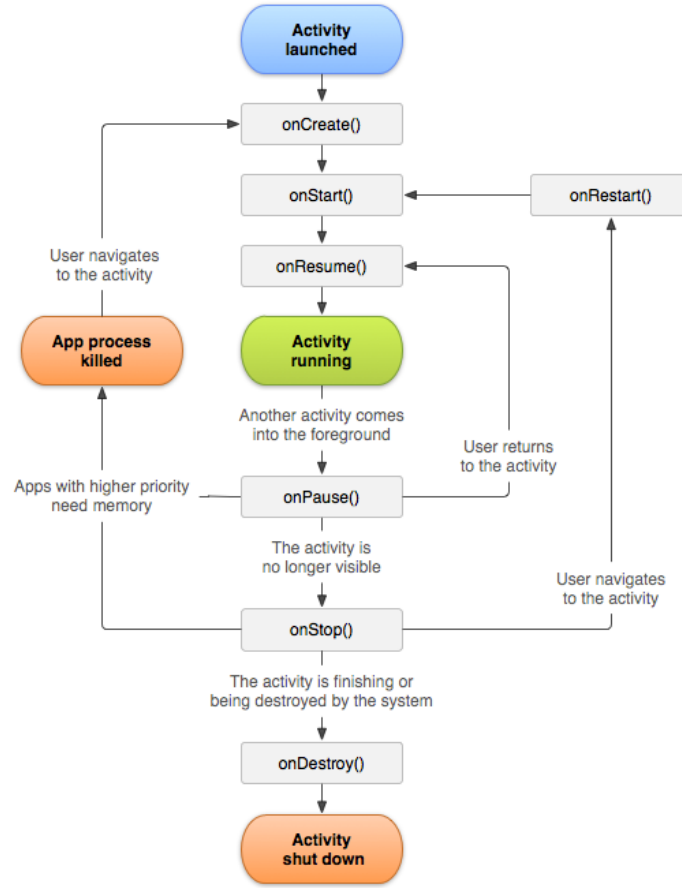


Figure 4: Lifecycle of an Android program

LTLvis

LTLvis is a graphical application running on android platform. It is based on the graphical language for LTL to express the specification of robot motion and mission plan. LTLvis combines the theory in [20] and [21], and realizes it on physical device. Motion gesture has been utilized in this application to define the actions to build the graphical language. Single tapping two different nodes will connect them with AND/OR operator and double tapping two different nodes will connect them with temporal operators. Long pressing a node will popup more option including renaming a node, toggling between logic

operators (BO_2) and temporal operators (TO_1 and TO_2). Author also implements a function to convert the graph to LTL specification formula. Once an action is performed, this function will be called to generate and display the formula in runtime. Below figure is the screenshot of LTLvis:

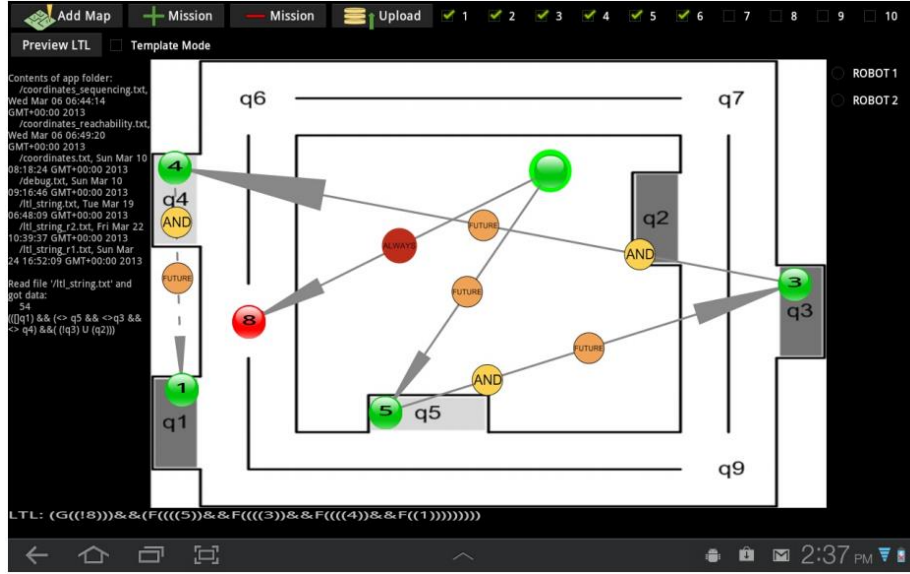


Figure 5: Sample screenshot of LTLvis

Chapter 3

PROBLEM DESCRIPTION

In this section, we describe our problem and then the expected solution

Problem Overview

This research mainly focuses on the problems of solving the path planning under a given LTL specification. Given an environment, a graphical LTL specification, and the user's preferred paths sketched on the environment, find the optimal path satisfying the LTL specification and maximally follows the user's path sketches. Once there exist conflicts between the user sketch path and the LTL specification, the interface should be able to regard the LTL specification as a higher priority requirement and find an alternative path not following the user sketch path. The rationale behind this choice is that the user may not be explicitly aware of important safety requirements and event dependencies when drawing the desired path. An alternative approach would be to recommend revisions to the mission requirements based on the path drawn by the user. This problem has been studied in the past by multiple authors. For example, see [22] and the references therein.

In addition, there are few more technical requirements and users may require robots operating in different environments. Thus, the control interface should be compatible to varieties of maps including laser scanned maps. Another necessary function for reducing users' mistakes and providing them convenience is the ability to undo and redo user's operations. When the user makes a mistake, the interface should have an option to reverse the incorrect operation instead of requiring the user to start over.

Scenario 1:

A data collection vehicle starts from location q_1 . It needs to visit location q_2 to collect data, then visit q_3 to upload data. But as the traffic along the path π_1 is unstable, the vehicle should adopt π_2 instead of π_1 even though π_1 has shorter geographical distance (in Figure 6).

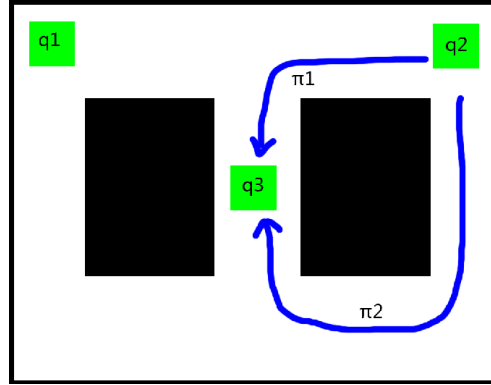


Figure 6: Data collection vehicle have to options (π_1, π_2) to visit p_3 .

Assuming the vehicle always adopts the shortest path from current location to its destination. This scenario can be solved by creating another location q_4 at lower right corner, then ask vehicle to visit q_4 before q_3 . The resulting path π_2 can be resolved as $\pi_2 = (q_2, q_4, q_3)$ (Figure 7).

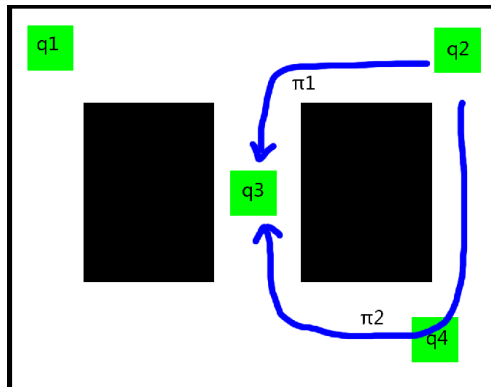


Figure 7: By adding node q_4 , the vehicle can visit q_4 , then visit q_3 .

However, if π_2 is complicated and even includes circles, the above solution may create a lot of temporary locations along π_2 (Figure 8). The resulting path π_2 can be resolved as $\pi_2 = (q_2, q_4, q_5, q_7, q_6, q_4, q_3)$. Even though location q_1 and q_7 are close, the vehicle is not required to visit q_1 . Thus q_7 is created just as a middle point for the vehicle to visit top-left corner. Same as q_6 and q_2 , q_6 is just a middle point for the vehicle to visit top-right corner.

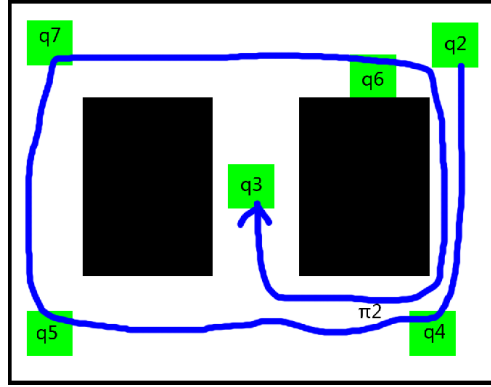


Figure 8: A complicated path π_2 including cycles.

Obviously, if the map is getting larger and π_2 is becoming more complicated, it will be much harder for users to figure out a correct expression for π_2 .

If the vehicle only adopts the assigned path instead of shortest path, it may be easier to solve above scenario. However, the user may not always know the real time condition of the roadmap. Once the whole area of q_4 is blocked, the vehicle will be confused as the next assigned destination is unreachable. But for the higher requirement, the vehicle is asked to visit q_3 from q_2 . If the vehicle knows the shortest path excluding the blocked area from q_2 to q_3 , then this scenario can be solved.

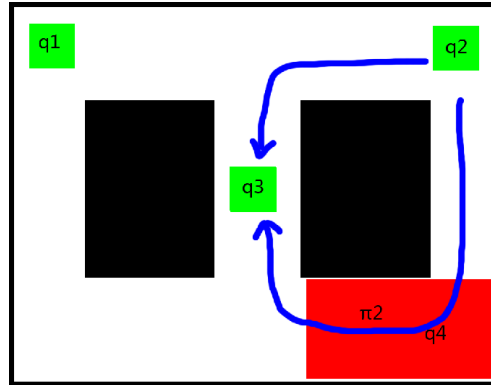


Figure 9: π_2 is visiting a blocked area

But, if the interface is smart enough to understand the meaning of path π_2 . It means that the vehicle are required to visit q_3 from q_2 via the preferred path π_2 . If π_2 is not applicable, find another applicable alternative. This is also one objective of the proposed interface.

Scenario 2:

If q_2 and q_4 are suppliers, then the vehicle may need to repetitively visit q_2 or q_4 to pick up machine parts and drop them off at q_3 .

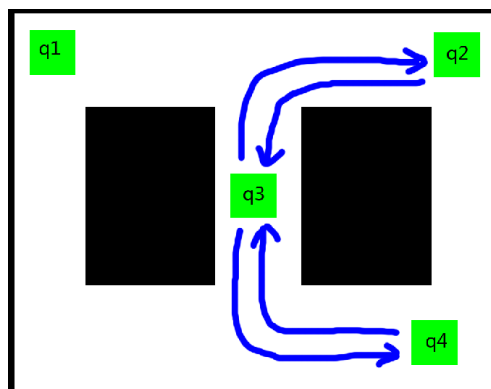


Figure 10: The vehicle has two option paths to follow when leaving q_3 .

If a user controls the vehicle via a sketched interface and assign a bi-directional path between q2 and q3, and between q3 and q4, this may confuse the vehicle when it reaches q3 and there are two assigned next destinations.

Linear Temporal Logic (LTL) can easily solve this scenario. In LTL, q1, q2, q3 and q4 are regarded as atomic propositions. The edges between atomic propositions can be regarded as temporal or Boolean operators. Then above solution can be presented as:

$$(q1 \rightarrow X (q2 \vee q4)) \wedge GF(q3 \wedge G(q3 \rightarrow X (q2 \vee q4)))$$

And the graphically expression should look like below:

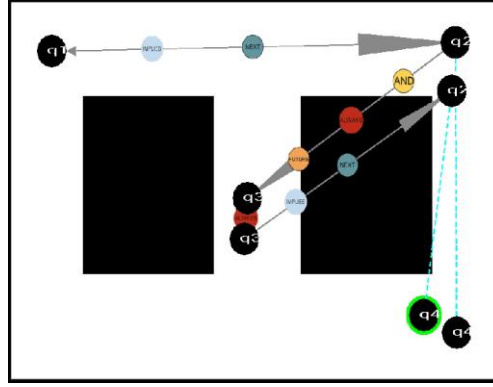


Figure 11: The graphical expression of $(q1 \rightarrow X (q2 \vee q4)) \wedge GF(q3 \wedge G(q3 \rightarrow X (q2 \vee q4)))$

From Scenario 1 and Scenario 2, we know that path based control interface and LTL control interface have their own advantages. A better solution will be the combination of both interfaces which allows users to specify their preferred path and high level specifications.

Solution Overview

The interface starts with an empty screen asking the user to input a map image. Then the user can sketch on the map using the interface. There are three different editing modes for planning, roadmap editing, and LTL editing.

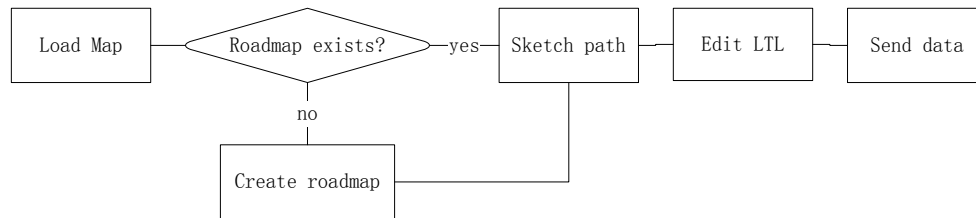


Figure 12: The flowchat of the interface process.

- Sketching Mode (Figure 21)
 - Create nodes
 - Move nodes
 - Draw a path from one node to another
 - Calculate the most suitable path according to the user drawing
 - Clear current drawing and planning path
- Roadmap Mode (Figure 15 to Figure 18)
 - Create nodes
 - Add or remove undirected edges between nodes
 - Automatically save once switching to another mode.
- LTL Mode (Figure 28)
 - Create nodes

- Add or remove edges with LTL attributes
- Edit LTL attributes

After loading the map, the interface enters the roadmap mode. A roadmap is an editable and storable transition system. For example, the roadmap in Figure 20 is the workspace of the robot [23]. The roadmap is stored locally as roadmap data. If the roadmap data exists, it will be loaded and then the sketch mode will be entered; otherwise, the interface will enter the roadmap mode and automatically create an empty roadmap for editing. When a user is done editing a roadmap, the interface will switch to sketching mode. The last step is to create an LTL specification. However, there is no restriction for the accessing order of each mode. A user can access any mode at any time. The basic order of each process is shown in Figure 12.

From the design aspect of view, the interface has one main activity. This activity will load several sub modules including roadmap editor (Figure 13), LTL formula generator, optimal path generator, TCP socket module, and status monitor. Roadmap editor, LTL formula generator and optimal path generator are deployed in roadmap mode, LTL mode and plan mode respectively. TCP socket module requires data of roadmap, LTL formula and optimal path, then sends the package to third party planner. Status monitor will keep tracking every action the user inputs to the interface and provide a jump point to recover the interface to a previous status which realizes the undo and redo functions.

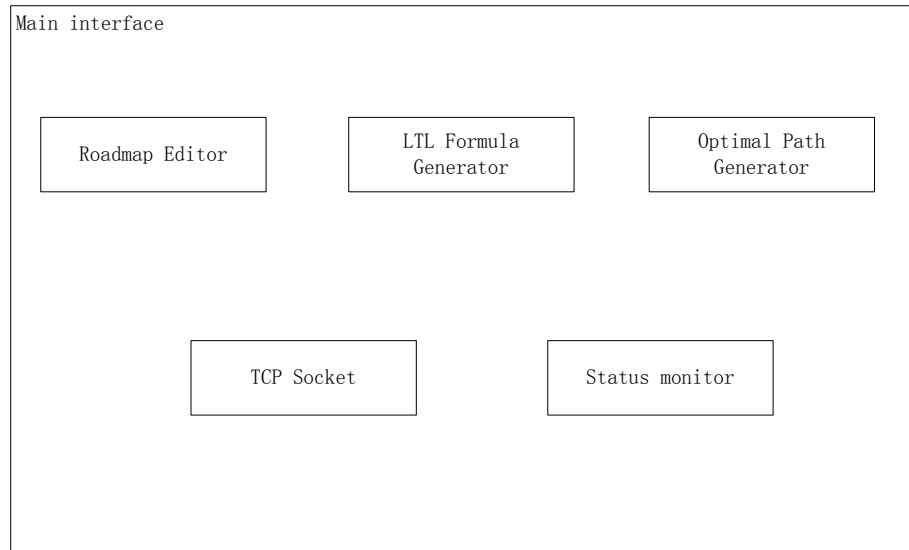


Figure 13: The five main modules of the interface

Chapter 4

EXTENDED LTLvis GUIDE

In this section, an extended LTLvis (E-LTLvis) will be introduced. It enables several drawing features and different interface layouts from the original LTLvis [5].

Load Map and Create Roadmap

The first goal of this interface is to develop the ability to be compatible to varieties of maps.

Roadmaps can be automatically generated using grid decomposition or a polyhedral decomposition of the environment [23]. In our interface, we require user to manually create their own roadmap. First, the interface requires the user to load the map image when the interface starts (Figure 14).

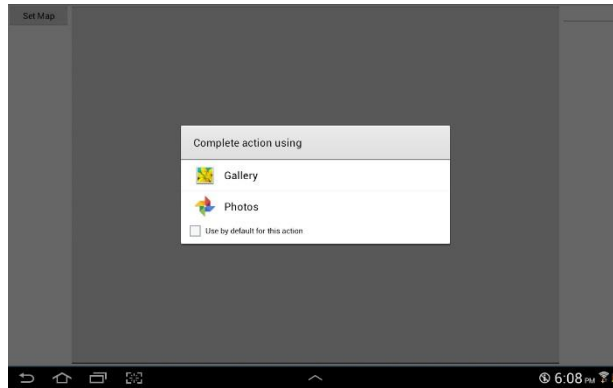


Figure 14: The user is asked to load a map.

After loading the map image, the interface will enter roadmap mode (Figure 15). In this mode, user is able to edit roadmap data.

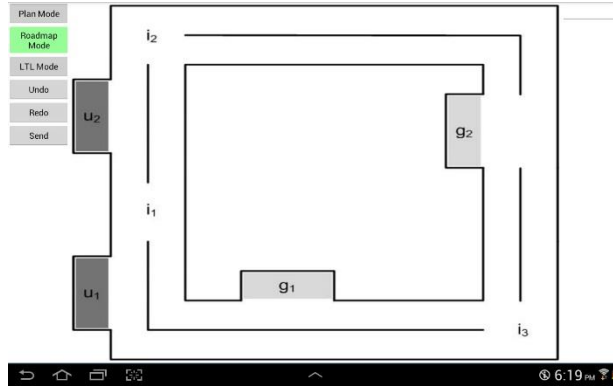


Figure 15: The starting interface of roadmap mode

Long press on the map to add a node at that location (Figure 16).

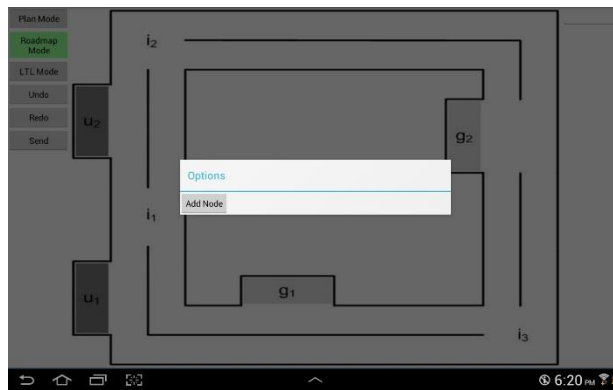


Figure 16: Long press to add a node

Single click the node to select it. Selected node will be surrounded by a green circle. After selecting the node, two actions will be provided, deleting the node and adding out-going edges for this node.

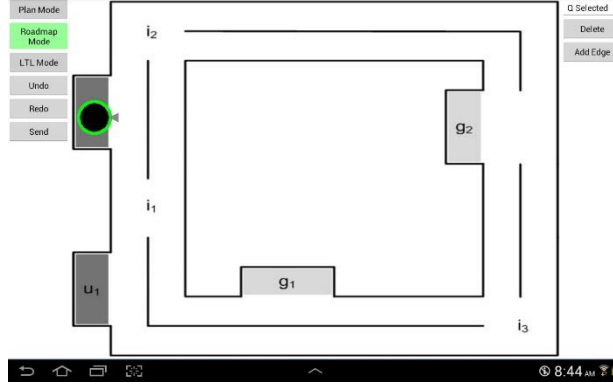


Figure 17: The selected node is marked green

When adding the node, first select the node, click the ‘Add Edge’ button, and then select other nodes as its neighbors. Once a neighbor is selected, it will be surrounded by a red circle and a line will be display between them (Figure 18).

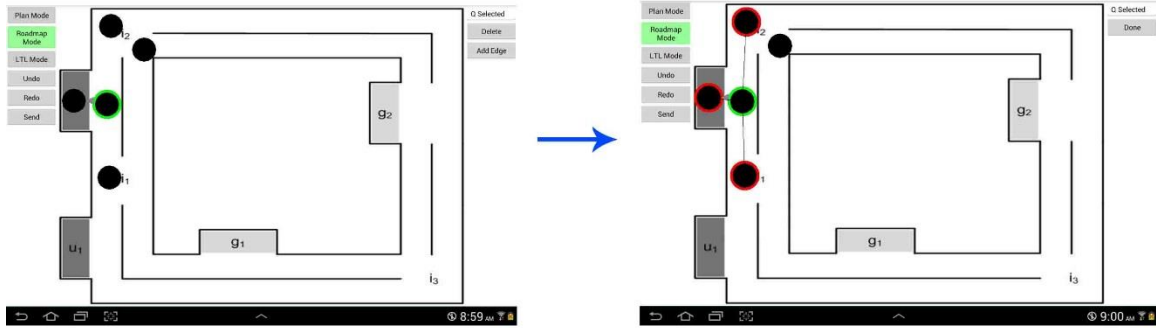


Figure 18: When adding edges, the nodes got selected will be marked red and become the green node’s neighbors.

After the image is loaded, the interface will search the corresponding roadmap file (.spc) which stores roadmap data⁶. If it exists, the data is loaded. If it does not exist, the interface will switch to roadmap mode and automatically create an empty roadmap data to allow the user to edit. When finishing editing the map, the roadmap file will be created to

⁶ For example, if the roadmap image is “simple.png”, the corresponding roadmap data related to this image is stored in “simple.spc” file.

store these nodes and edges locally. Next time, when the same map image is selected, this roadmap file will be loaded automatically. We also provide a video demo⁷ to show in more detail the procedure for creating a roadmap.

By manually creating the roadmap, the interface can apply to most environments as long as the environment can be described as a node-edge based roadmap.

Sketch Path

Users can customize the plan between two nodes in sketch mode, including adding nodes, renaming nodes, removing nodes, and defining paths.

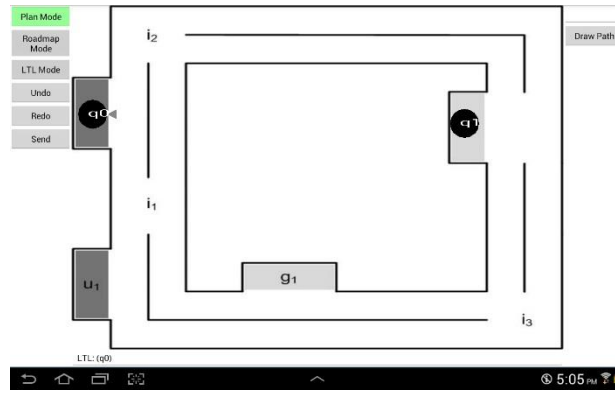


Figure 19: The staring interface of sketch mode

Same as roadmap mode, users can add a node by long pressing on the screen. In this mode, all roadmap information will be hidden, thus users can achieve the experience as if they are planning directly on the map (Figure 20).

⁷ <https://www.assembla.com/spaces/Itlvis/wiki/E-LTLvis>

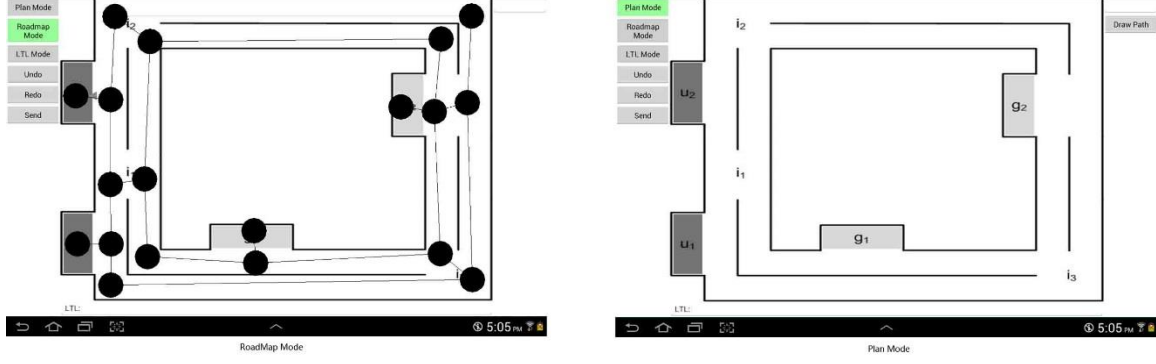


Figure 20: In sketch mode, the roadmap data will be hidden

When customizing the path, you can first select the starting node, drag the path along the map, and end the path at another node. This path is denoted as **user sketched path** p^u . Then, we find the node in the environment closest to the first node of p^u , and denote it as q_{start} . Also, we find the node in the environment closest to the last node of p^u , and denote it as q_{end} . Because the user sketched path may be drawn by curves which consist of too many nodes, to reduce the computation workload, the path is sampled by distance d_m and angle θ_m into a list of (blue in the figure) nodes (n_1, n_2, \dots). The sample algorithm is shown in Alg. 0 below. After appending q_{start} to the beginning of the list and q_{end} to the end of the list, we get a new list of nodes. This list of nodes is denoted as **sampled user sketched path** p^0 . For example, in Figure 21, the green curve is the user sketched path.

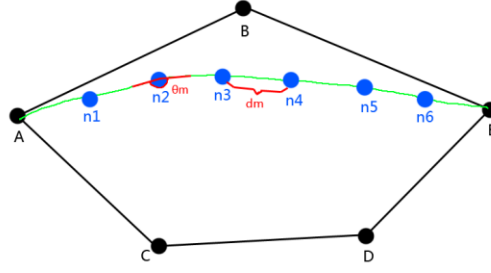


Figure 21: Black nodes and edges: the roadmap of a simple environment. Green path:

p^u . Blue nodes: p^0

Algorithm0 sampling

Input: p^u , TS

Output: p^0

1. $d_m \leftarrow$ distance threshold
 2. $\theta_m \leftarrow$ angle threshold
 3. $p^u \leftarrow []$
 4. $p^0 \leftarrow []$
 5. $q_{start} \leftarrow$ the node closet to $p^u[1]$ in Q_{TS}
 6. $q_{end} \leftarrow$ the node closet to $p^u[-1]$ in Q_{TS}
 7. **for** n_i^u as i_{th} point in p^u **do**:
 8. flag \leftarrow false
 9. **if** $i = 0$ **then**:
 10. flag \leftarrow true
 11. **else**:
 12. $n_{-1}^0 \leftarrow p^0[-1]$
 13. **if** distance(n_i^u, n_{-1}^0) $> D$ **then**:
 14. flag \leftarrow true
 15. **if** \neg flag & size(p^0) > 1 **then**:
 16. $n_{-2}^0 \leftarrow p^0[-2]$
 17. **if** angle($n_i^u, n_{-1}^0, n_{-2}^0$) $< \theta_m$ **then**:
 18. flag \leftarrow true
 19. **if** flag **then**:
 20. append p_i to p^0
 21. append q_{start} to the beginning of p^0
 22. append q_{end} to the end of p^0
 23. **return** p^0
-

In this algorithm, before putting a point into sampling set, we need to check if it satisfies either of the two requirements:

1. The distance of current point and last sampled point should be greater than the predefined threshold d_m .
2. The angle of the current point, last sampled point and last second sampled point should be less than the predefined threshold θ_m .

This algorithm will tremendously reduce the number of point in the sampling set. To balance the accuracy and computation workload, the distance threshold is set to 50 pixels and the angle threshold is set to 145 degree.

After the sampling process, the input path will be represented as a list of nodes. Then, the touch up event will be triggered and the computed best matching path will be displayed. Since p^u may stretch to areas undefined in the roadmap, this best match path may not be the same as p^u (Figure 27). As we need to compare the similarity of two paths, the best approach is to calculate the volume between two paths. But this approach has heavy workload, so that we define a new heuristic, CWPD, to compare two paths.

Definition 4. (CWPD) *component-wise path distance is distance of two paths $p^0 = (n_0^0, n_1^0 \dots n_{N-1}^0)$ and $p^x = (n_0^x, n_1^x \dots n_{N-1}^x)$.*

$$EQ1: CWPD(p^0, p^x) = \sum_{i=1}^{N-1} distance\left(n_i^0, edge_{(n_j^x, n_i^x)}\right),$$

Where $N = length(p^0)$, $n_i^0 \in p^0$, $n_i^x, n_j^x \in p^x$, and n_j^x is previous node which differs from n_i^x . If n_i^x is the first node, n_j^x equals to n_i^x .

From above EQ1, we can also derive:

$$\begin{aligned}
 \text{EQ2: } CWPDP((n_0^0, n_1^0 \dots n_{N-1}^0), (n_0^x, n_1^x \dots n_{N-1}^x)) &= \sum_{i=1}^{N-1} \text{distance}(n_i^0, \text{edge}_{(n_j^x, n_i^x)}) = \\
 &\left(\sum_{i=1}^{N-2} \text{distance}(n_i^0, \text{edge}_{(n_j^x, n_i^x)}) \right) + \text{distance}(n_{N-1}^0, \text{edge}_{(n_{j-1}^x, n_{N-1}^x)}) \\
 &= CWPDP((n_0^0, n_1^0 \dots n_{N-2}^0), (n_0^x, n_1^x \dots n_{N-2}^x)) + \text{distance}(n_{N-1}^0, \text{edge}_{(n_{j-1}^x, n_{N-1}^x)})
 \end{aligned}$$

Then, we definite the best match path in order to compare it in terms of distance.

Definition 5. (BMP) *Best Matching Path p_{bmp} is a feasible path on the transition system TS with the same starting q_{init} and ending position q_{end} as p^0 . It also has the properties:*

- $\text{length}(p_{bmp}) = \text{length}(p^0)$
- p_{bmp} can be cyclic on TS
- The **component-wise path distance** between p_{bmp} and p^0 should be minimal.

We use distance to line segment ($\text{edge}_{(n_j^x, n_i^x)}$) instead of line to avoid the situation where n_i^0 is very far from $\text{edge}_{(n_j^x, n_i^x)}$ but close to the line(n_j^x, n_i^x) (Figure 22). The distance can be defined as following.

Definition 6. (DISTANCE) The distance of a point to a line segment is the shortest distance from a point to another point in the line segment in Euclidean geometry.



Figure 22: Distance d_{ls} of C to edge(A, B) equals to distance d_l of C to line(A, B). But distance d_{ls} of D to edge(A, B) does not equal to distance d_l of D to line(A, B).

For example in Figure 22, d_l is the distance from node C to line AB which is equal to the distance d_{ls} from node C to line segment AB. But d_l and d_{ls} do not always equal each other. For node D, d_{ls} is greater than d_l .

We can reduce the sample distance d_m and angle θ_m to increase N . Thus p^0 can always have more nodes than p_{bmp} so that the size of p_{bmp} can be extended to N by adding copies of nodes in between. For the example (Figure 21), some possible BMP candidates are listed in Table 3 for the candidate path set (p^1, p^2, p^3) :

p^1	A	B	B	B	B	E	E	E
p^2	A	C	C	D	D	E	E	E
p^3	A	B	B	B	B	B	E	E
p^0	A	n1	n2	n3	n4	n5	n6	E

Table 3: Path p^0 and its possible BMP candidates.

As p^0 has more nodes than p^x , we can extend the path (A, B, C) to path (A, B, B... C) or (A, B, C... C) to make their number of nodes equal to N .

To achieve the minimum CWPDP, we need to compare the CWPDPs (shown in Figure 23) between p^0 and each p^x . In this example, the path p^3 minimizes the CWPDP.

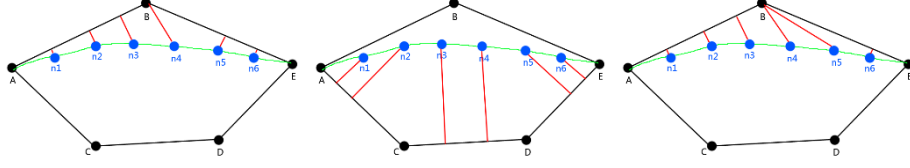


Figure 23: The CWPDs of p^1, p^2, p^3

As the number of candidate path in the worst case is N^M , where M is the number of nodes in TS, it is impractical to list all of them before searching the minimum CWPD. Instead, we create a matrix to store a BMP ending at n_i^x for each node in the roadmap. When looping through each node in p^0 , the path stored in the matrix will be updated. The pseudo code of this greedy algorithm is provided in Alg. 1.

Because the user sketched path may contain cycles intentionally, standard shortest path algorithm [24] will not return results with cycles. This is why we cannot modify and utilize standard shortest path algorithm. Algorithm 1 solves the problem also with cycles on the graph. It takes p^0 and TS as input. It proceeds sequentially through all nodes in p^0 (line 7). In each iteration of this ourter loop, it calculates M BMPs for each q_j (line 8) according to current user input path $(n_0^0, n_1^0 \dots n_{i-1}^0)$. These BMPs start from q_{start} and end at q_j .

Algorithm 2 calculates the new CWPD and BMP by utilizing the results from the previous BMPs and CWPDs using EQ2. For each node q_j in Q_{TS} , it first checks if q_j 's previous BMP $bmp[i-1, j]$ for $(p^0[1] \dots p^0[i-1])$ exists. If it exists, it calculates the distance between $p^0[i]$ and the last edge of the path $(bmp[i-1, j], q_j)$. Then, it stores the result in $bmp[i, j]$ and $cwpd[i, j]$ if the new $cwpd[i, j]$ is smaller than the existing

value. Then, it repeats the process for all paths($bmp[i - 1, j], q_k$), where $q_k \in neighbors(q_j)$. Note that we can get q_j 's previous BMP and CWPd directly from $bmp[i - 1, j]$ and $cwpd[i - 1, j]$, respectively, without recomputing the results. The process will repeat at most M times; thus, the run time of Alg. 2 is $O(M)$.

Algorithm 1 FIND_BMP

Input: p^0, TS

Output: p_{bmp}

1. $M \leftarrow |Q_{TS}|$
 2. $N \leftarrow |p^0|$
 3. $cwpd[:, :] \leftarrow \infty$ //for N x M matrix
 4. $bmp[:, :] \leftarrow \emptyset$ //for N x M matrix
 5. $\langle start, end \rangle \leftarrow \langle index(p^0[1]), index(p^0[N]) \rangle$ //index of nodes in Q_{TS}
 6. $\langle cwpd[1, start], bmp[1, start] \rangle \leftarrow \langle 0, \{p^0[1]\} \rangle$
 7. for i in range (2, N) do:
 8. for j in range (1, M) do:
 9. UPDATE($cwpd, bmp, i, j, p^0, TS$)
 10. $p_{bmp} \leftarrow bmp[N, end]$
 11. Return p_{bmp}
-

where $Q_{TS} = \{q_0, q_1 \dots q_{M-1}\}$ is the set of nodes in the TS
 $|Q_{TS}|$ returns the number of elements in Q_{TS}

Algorithm 2 UPDATE

Input: $cwpd, bmp, i, j, p^0, TS$

1. if $bmp[i - 1, j] \neq \emptyset$ then: //previous bmp ending at this node
 2. $q_j \leftarrow index^{-1}(j, Q_{TS})$ //index⁻¹() returns a node of Q_{TS}
 3. $n_i^0 \leftarrow index^{-1}(i, p^0)$ //index⁻¹() returns a node of p^0
 4. $edge_{prev} \leftarrow GetLastEdge(bmp[i - 1, j])$
 5. $cwpd_{candi} \leftarrow cwpd[i - 1, j] + distance(n_i^0, edge_{self})$ //EQ2
 6. if $cwpd_{candi} < cwpd[i, j]$ then:
 7. $cwpd[i, j] \leftarrow cwpd_{candi}$
 8. $bmp[i, j] \leftarrow bmp[i - 1, j] + q_j$ //concatenate q_j to the end
 9. for q_k in $neighbors(q_j)$ do:
 10. $edge_{curr} \leftarrow \langle q_j, q_k \rangle$
 11. $k \leftarrow index(q_k)$ //index of nodes in Q_{TS}
 12. if $edge_{curr} \neq edge_{prev}$ then:
 13. $cwpd_{candi} \leftarrow cwpd[i - 1, k] + distance(n_i^0, edge_{curr})$
 14. if $cwpd_{candi} < cwpd[i, k]$ then:
-

15.	$cwpd[i, k] \leftarrow cwpd_{candi}$
16.	$bmp[i, k] \leftarrow bmp[i - 1, k] + q_k$
<hr/> <i>GetLastEdge()</i> returns the last edge of a given path or an edge with the same two nodes if there is no last edge e.g., <i>GetLastEdge</i> ([ABCDE]) returns ⟨DE⟩ and <i>GetLastEdge</i> ([A]) returns ⟨AA⟩. <hr/>	

In each step, the minimum CWPDP ending at each node in TS will be stored. Thus, this algorithm finds the BMP with the minimum CWPDP eventually. Let us proceed this algorithm over the example in Table 3.

0. Step 0: Initialize the tables.

BMP	A	B	C	D	E
A	∅	∅	∅	∅	∅
n1	∅	∅	∅	∅	∅
n2	∅	∅	∅	∅	∅
n3	∅	∅	∅	∅	∅
n4	∅	∅	∅	∅	∅
n5	∅	∅	∅	∅	∅
n6	∅	∅	∅	∅	∅
E	∅	∅	∅	∅	∅

CWPDP	A	B	C	D	E
A	∞	∞	∞	∞	∞
n1	∞	∞	∞	∞	∞
n2	∞	∞	∞	∞	∞
n3	∞	∞	∞	∞	∞
n4	∞	∞	∞	∞	∞
n5	∞	∞	∞	∞	∞
n6	∞	∞	∞	∞	∞
E	∞	∞	∞	∞	∞

Table 4: All cells in BMP are set to empty and all cells in CWPDP are set to infinite

- 1) Navigate to the current node in p^0 , which is the start point A.
- 2) Add node A as the current best matching path ending at node A in $bmp[A, A]$.
- 3) Add 0 to as the current minimum CWPDP whose path ends at node A in $cwpd[A, A]$.

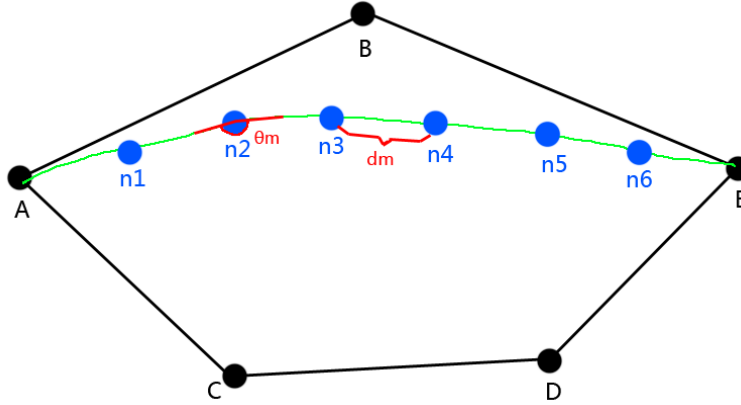


Figure 24: Step 0

BMP	A	B	C	D	E
A	A				
n1					
n2					
n3...E					

CWPD	A	B	C	D	E
A	0				
n1					
n2					
n3...E					

Table 5: Step 0-2 and 0-3

1. Step 1:

- 1) Navigate to the current node in p^0 , which is the n1.
- 2) For cell $\text{bmp}[n1, A]$, find all possible best matching paths ending at A by adding one node to all paths in $\text{bmp}[A, :]$, which is AA^8 .
- 3) Calculate the CWPD by adding daa1 to the previous cwpd which is $\text{cwpd}[A, A]$. The new CWPD is daa1 .

⁸ Possible BMPs for cell $\text{bmp}[n1, A] = (\text{all BMP}[A, :] \text{ that can access A within one step}) + A$. These BMPs can be ended at A or A's neighbors. For above row n1, previous BMP is A. By adding a node to A, we can get AA, AB, AC.

- 4) Compare all CWPD calculated in $\text{cwpd}[n1, A]$ and only save the minimum one as $\text{cwpd}[n1, A]$.
- 5) Save the corresponding BMP as $\text{bmp}[n1, A]$.
- 6) Repeat above 2) to 5) for node B, C, D and E respectively.

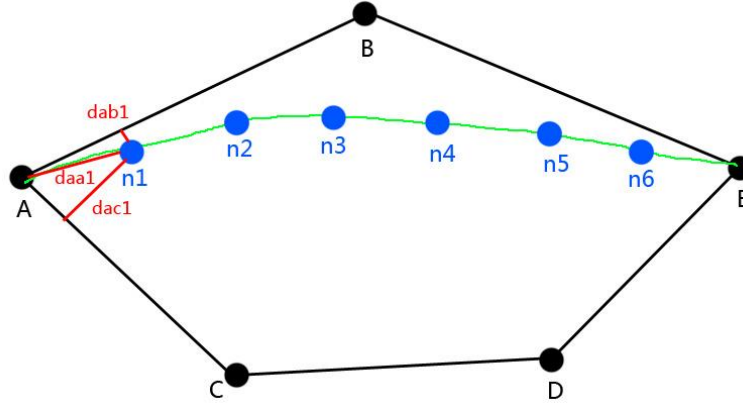


Figure 25: Step 1. dab1 denotes the distance from n_1 to edge e_{ab}

BMP	A	B	C	D	E
A	A				
n1	AA	AB	AC		
n2					
n3...E					

CWPD	A	B	C	D	E
A	0				
n1	daa1	dab1	dac1		
n2					
n3...E					

Table 6: Step 1-2 to 1-5.

As node D and E cannot be access by adding one node to all the path in $\text{bmp}[A, :]$, there will be no value in $\text{bmp}[n1, D]$ and $\text{bmp}[n1, E]$, thus $\text{cwpd}[n1, D]$ and $\text{cwpd}[n1, E]$ will also be empty.

2. Step 2:

- 1) Navigate to the current node in p^0 , which is the $n2$.

- 2) For cell $\text{bmp}[n2, A]$, find all possible best matching paths ending at A by adding one node to all paths in $\text{bmp}[n1, :]$, which is AAA, ABA and ACA.
- 3) Calculate the CWPd of AAA by adding daa2 to the previous cwpd which is $\text{cwpd}[n1, A]$. Calculate the CWPd of ABA by adding dab2 to the previous cwpd which is $\text{cwpd}[n1, B]$. Calculate the CWPd of ACA by adding dac2 to the previous cwpd which is $\text{cwpd}[n1, C]$.
- 4) Compare all CWPd calculated above and only save the minimum one as $\text{cwpd}[n2, A]$.
- 5) Save the corresponding BMP as $\text{bmp}[n2, A]$.
- 6) Repeat above 2) to 5) for node B, C, D and E respectively.

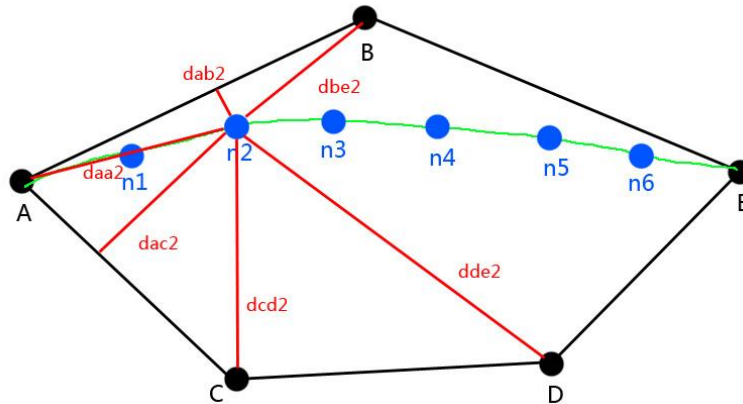


Figure 26: Step 2

BMP	A	B	C	D	E
n1	AA	AB	AC		
	$\text{bmp}[n1][A]+A$ -> AAA	$\text{bmp}[n1][B]+B$ -> ABB	$\text{bmp}[n1][C]+C$ -> ACC	$\text{bmp}[n1][C]+D$ > ACD	$\text{bmp}[n1][B]+D$ -> ABE
n2	$\text{bmp}[n1][B]+A$ -> ABA $\text{bmp}[n1][C]+A$ -> ACA	$\text{bmp}[n1][A]+B$ -> AAB	$\text{bmp}[n1][A]+C$ -> AAC		
n3...E					

CWP D	A	B	C	D	E
n1	daa1	dab1	dac1		
n2	cwpd[n1][A]+d aa2	cwpd[n1][B]+d ab2	cwpd[n1][C]+d ac2	cwpd[n1][C]+d cd2	cwpd[n1][B]+d be2
	cwpd[n1][B]+d ab2	cwpd[n1][A]+d ab2	cwpd[n1][A]+d ac2		
	cwpd[n1][C]+d ac2				
n3...E					

Table 7: Step 2-2 to 2-5

- Repeat step 2 for rest of the note in p^0 .
- Find the minimum CWPd in cwpd[E, E] and its corresponding BMP will be the final path.

BMP	A	B	C	D	E
A	A				
n1	AA	AB	AC		
n2	bmp[n1][A]+A -> AAA	bmp[n1][B]+B -> ABB	bmp[n1][C]+C -> ACC	bmp[n1][C]+D-> ACD	bmp[n1][B]+D -> ABE
	bmp[n1][B]+A -> ABA	bmp[n1][B] -> bmp[n2][B]	bmp[n1][C] -> bmp[n2][C]	bmp[n1][D] -> bmp[n2][D]	bmp[n1][E] -> bmp[n2][E]
	bmp[n1][A]+B -> bmp[n2][A]	bmp[n1][A]+B -> AAB	bmp[n1][A]+C -> AAC		
n3	bmp[n1][C]+A -> ACA				
	bmp[n2][A]+A -> ABAA	bmp[n2][B]+B -> ABBA	bmp[n2][C]+C -> ABAC	bmp[n2][D]+D -> ABED	bmp[n2][E]+E -> ABEE
	bmp[n3][A] -> ABBA	bmp[n3][B] -> ABAB	bmp[n2][A]+C -> ABAC	bmp[n2][C]+D -> ABED	bmp[n3][E] -> ABEE
n4	bmp[n2][B]+A -> ABBA	bmp[n2][A]+B -> ABAB	bmp[n3][C] -> ABAC	bmp[n2][E]+D -> ABED	bmp[n2][B]+E -> ABEE
	bmp[n2][C]+A -> ABBA	bmp[n2][E]+B -> ABEB	bmp[n2][D]+C -> ABED	bmp[n3][D] -> ABED	bmp[n2][D]+E -> ABEE
	bmp[n3][A]+A -> ABAAA	bmp[n3][B]+B -> ABBA	bmp[n3][C]+C -> ABBA	bmp[n3][D]+D -> ABBA	bmp[n3][E]+E -> ABBA
n4	bmp[n4][A] -> ABBA	bmp[n3][A]+B -> ABBA	bmp[n3][A]+C -> ABBA	bmp[n3][C]+D -> ABBA	bmp[n3][B]+E -> ABBA
	bmp[n3][B]+A -> ABBA	bmp[n3][E]+B -> ABBA	bmp[n4][C] -> ABBA	bmp[n3][E]+D -> ABBA	bmp[n4][E] -> ABBA
	bmp[n3][C]+A -> ABBA	bmp[n4][B] -> ABBA	bmp[n3][D]+C -> ABBA	bmp[n4][D] -> ABBA	bmp[n3][D]+E -> ABBA

n5	bmp[n4][A]+A -> ABAAAA	bmp[n4][B]+B -> ABEEBB	bmp[n4][C]+C -> ABAACC	bmp[n4][D]+D -> ABEEDD	bmp[n4][E]+E -> ABBBEE ->
	bmp[n4][B]+A -> ABEEBA->	bmp[n4][A]+B -> ABAAAB	bmp[n4][A]+C -> ABAAAC ->	bmp[n4][C]+D -> ABAACD	bmp[n5][E] bmp[n4][B]+E
	bmp[n5][A] bmp[n4][C]+A -> ABAAACA	bmp[n4][E]+B -> ABBBEB ->	bmp[n5][C] bmp[n4][D]+C -> ABEEDC	bmp[n4][E]+D -> ABBBED ->	-> ABEEBE bmp[n4][D]+E -> ABEEDE
	bmp[n5][B]+A -> ABBBEBA->	bmp[n5][A]+B -> ABEEBAB	bmp[n5][A]+C -> ABEEBAC	bmp[n5][C]+D -> ABAAACD	bmp[n5][B]+E -> ABBBEBA
	bmp[n6][A] bmp[n5][C]+A -> ABAAACA	bmp[n5][E]+B -> ABBBEEB ->	bmp[n5][D]+C ABBBEDC-> bmp[n6][C]	bmp[n5][E]+D -> ABBBEED-> bmp[n6][D]	bmp[n5][D]+E -> ABBBEDE
n6	bmp[n5][A]+A -> ABEEBAA	bmp[n5][B]+B -> ABBBEBB	bmp[n5][C]+C -> ABAAACC	bmp[n5][D]+D -> ABBBEDD	bmp[n5][E]+E -> ABBBEEE
	bmp[n5][B]+A -> ABBBEBA->	bmp[n5][A]+B -> ABEEBAB	bmp[n5][A]+C -> ABEEBAC	bmp[n5][C]+D -> ABAAACD	-> bmp[n6][E] bmp[n5][B]+E
	bmp[n6][A] bmp[n5][C]+A -> ABAAACA	bmp[n6][B] -> ABBBEEB ->	ABBBEDC-> bmp[n6][C]	bmp[n6][D] -> ABBBEED->	bmp[n5][D]+E -> ABBBEDE
	bmp[n6][B]+A -> ABBBEBA->	bmp[n6][A]+B -> ABBBEBA	bmp[n6][A]+C -> ABBBEBA	bmp[n6][E]+D -> ABBBEED->	bmp[n6][B]+E -> ABBBEBA
	bmp[E][A] bmp[n6][C]+A -> ABBBEDCA	ABBBEBAB bmp[n6][E]+B -> ABBBEBA	bmp[n6][D]+C -> ABBBEEDC-> bmp[E][C]	ABBBEED-> bmp[E][D]	ABBBEED-> ABBBEED
E	bmp[n6][A]+A -> ABBBEBA	bmp[n6][B]+B -> ABBBEBA	bmp[n6][C]+C -> ABBBEDC	bmp[n6][D]+D -> ABBBEED	bmp[n6][E]+E -> ABBBEEE
	ABBBEBAA	ABBBEEBB	ABBBEDCC	bmp[n6][C]+D -> ABBBEDCD	bmp[n6][B]+E -> ABBBEBA
	bmp[n6][B]+A -> ABBBEBA->	bmp[n6][A]+B -> ABBBEBA	ABBBEBAC	bmp[n6][E]+D -> ABBBEED->	bmp[n6][D]+E -> ABBBEED
	ABBBEEBA->	ABBBEBAB	bmp[n6][D]+C -> ABBBEEDC->	ABBBEED-> bmp[E][D]	ABBBEED-> ABBBEED
	ABBBEDCA	ABBBEBAB	bmp[E][C]		

CW PD	A	B	C	D	E
A	0				
n1	daa1	dab1	dac1		
n2	cwpd[n1][A] +daa2	cwpd[n1][B] +dab2->	cwpd[n1][C] +dac2->	cwpd[n1][C] +dcd2->	cwpd[n1][B] +dbe2->cwpd[n2][E]
	cwpd[n1][B] +dab2	cwpd[n2][B] ->cwpd[n1][A] +dab2	cwpd[n2][C] +cwpd[n1][A] +dac2	cwpd[n2][D]	
n3	cwpd[n1][C] +dac2				
	cwpd[n2][A] +dab3	cwpd[n2][B] +dab3	cwpd[n2][C] +dac3	cwpd[n2][D] +dcd3	cwpd[n2][E] +dbe3
	cwpd[n3][A] +dab3	cwpd[n3][B] +dab3	cwpd[n2][A] +dac3	cwpd[n2][C] +dcd3	cwpd[n2][B] +dbe3
	cwpd[n2][B] +dac3	cwpd[n2][A] +dab3	cwpd[n3][C] +dcd3	cwpd[n2][E] +dde3	cwpd[n2][D] +dde3
	cwpd[n2][C] +dac3	cwpd[n2][E] +dbe3	cwpd[n2][D] +dcd3	cwpd[n3][D]	

n4	cwpd[n3][A]+ dab4	cwpd[n3][B]+ ->dab4	cwpd[n3][C]+ dac4	cwpd[n3][D]+ dde4	cwpd[n3][E]+dbe4 ->cwpd[n3][B]+dbe4->
	cwpd[n4][A]	cwpd[n3][A]+	cwpd[n3][A]+	cwpd[n3][C]+	cwpd[n4][E]
	cwpd[n3][B]+ dab4	dab4	dac4	->dcd4	cwpd[n3][D]+dde4
	cwpd[n3][E]+	cwpd[n4][C]	cwpd[n3][E]+		
	cwpd[n3][C]+ dac4	dbbe4	->cwpd[n3][D]+ dcd4	dde4	->
n5	cwpd[n4][A]+ dab5	cwpd[n4][B]+ dbe5	cwpd[n4][C]+ dac5	cwpd[n4][D]+ dde5	cwpd[n4][E]+dbe5 ->cwpd[n5][E]
	cwpd[n4][B]+ dab5->	cwpd[n4][A]+ dab5	cwpd[n4][A]+ dac5	cwpd[n4][C]+ ->dcd5	cwpd[n4][B]+dbe5 ->cwpd[n4][D]+dde5
	cwpd[n5][A]	cwpd[n4][E]+	cwpd[n5][C]	cwpd[n4][E]+	
	cwpd[n4][C]+ dac5	dbbe5	->cwpd[n4][D]+ dcd5	dde5	->
		cwpd[n5][B]		cwpd[n5][D]	
n6	cwpd[n5][A]+ dab6	cwpd[n5][B]+ dbe6	cwpd[n5][C]+ dac6	cwpd[n5][D]+ dde6	cwpd[n5][E]+dbe6 ->cwpd[n6][E]
	cwpd[n5][B]+ dab6->	cwpd[n5][A]+ dab6	cwpd[n5][A]+ dac6	cwpd[n5][C]+ ->dcd6	cwpd[n5][B]+dbe6 ->cwpd[n5][D]+dde6
	cwpd[n6][A]	cwpd[n5][E]+	cwpd[n5][D]+	cwpd[n5][E]+	
	cwpd[n5][C]+ dac6	dbbe6	->dcd6	->dde6	->
		cwpd[n6][B]	cwpd[n6][C]	cwpd[n6][D]	
E	cwpd[n6][A]+ dabe	cwpd[n6][B]+ dbee	cwpd[n6][C]+ ->dcde	cwpd[n6][D]+ ddee	cwpd[n6][E]+dbee ->cwpd[E][E]
	cwpd[n6][B]+ dabe->	cwpd[E][B]	cwpd[n6][A]+ dace	cwpd[n6][C]+ dcde	cwpd[n6][B]+dbee ->cwpd[n6][D]+ddee
	cwpd[E][A]	dabe	cwpd[n6][D]+	cwpd[n6][E]+	
	cwpd[n6][C]+ dace	cwpd[n6][E]+ dabe	dcde	->ddee	->
			cwpd[E][C]	cwpd[E][D]	

Table 8: Final BMP and CWPDP tables. The final path is ABBBEEEE which is equivalent to ABE.

More detail of the step by step run of Alg. 1 over the example in Table 3 can be found in appendix.

The algorithm only creates two global matrices of size NM. Thus, the space complexity of this algorithm is $O(NM)$ and the runtime complexity is $O(NMM)$. Hence, this algorithm can be implemented on a mobile device.

After applying the algorithm to the scenario in Figure 19, we can get the result in Figure 27.

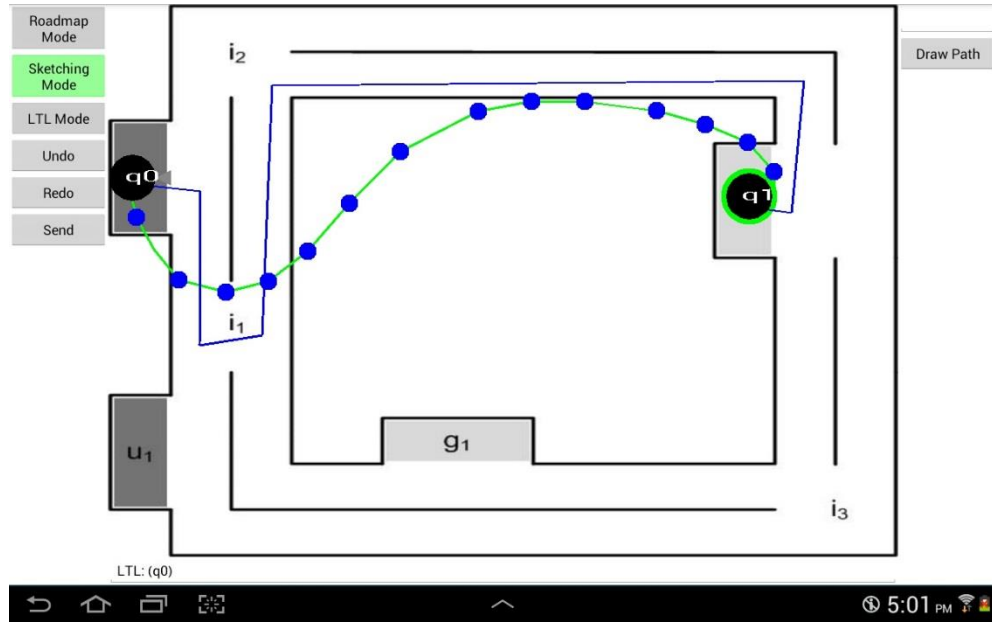


Figure 27: The user sketched path (arc with dots) and its BMP (solid blue line). Since the user sketches in areas undefined in the roadmap, the resulting BMP is much different as the sketched path.

Usually, users may need to specify the paths between multiple pairs of nodes. Our algorithm will generate multiple best matching paths for all user sketched paths. This set of best matching paths is called the **preferred path set**.

Edit Specifications

After a path is customized in the Sketching mode, there should be a default LTL specification displayed in the LTL Mode (for an example, see Figure 28).

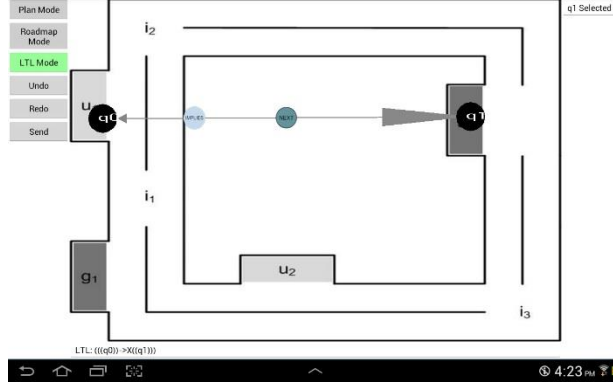


Figure 28: The basic LTL specification that corresponds to the sketched path in above figure.

Users can also skip the Sketching mode to directly edit the LTL specification. In this mode, the editing gestures are identical to LTLvis [5]:

- Single tap two consecutive nodes to connect them with AND or OR edge.
- Double tap two consecutive nodes to connect them with ALWAYS, EVENTUALLY, NEXT or UNTIL edge.
- Long press a node to display extra action options including renaming, toggling visit and avoid, toggling AND and OR, and changing operators between ALWAYS, EVENTUALLY, NEXT, and UNTIL (Figure 29).

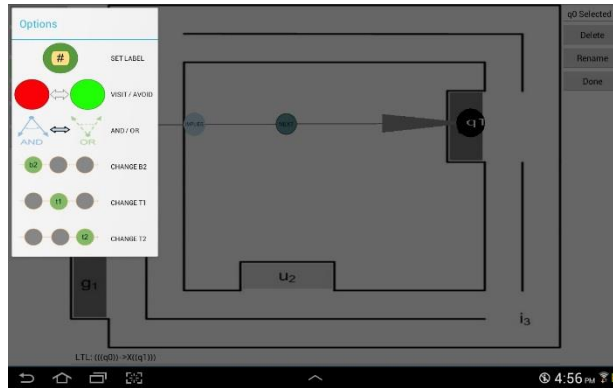


Figure 29: Long press the node to display the option panel.

For each icon in the popup option panel, there is a unique handler linked to it. Once the button down event is triggered, the option panel will be dismissed and the corresponding action will be performed. Simply touching anywhere outside the panel can also dismiss the options panel. The meaning of each action in the option panel is listed below (Table 9):

SET LABEL	Change the label of a node on the graph.
VISIT / AVOID	This option tells the robot whether to visit the location or avoid it.
AND / OR	Cycle through the boolean operator b_1 between AND and OR.
Change operator b_2	Cycle through the boolean operator b_2 between AND, OR and IMPLIES.
Change operator t_1	Cycle through the temporal operator t_1 between ALWAYS, EVENTUALLY, NEXT and UNTIL.
Change operator t_2	Cycle through t_2 between ALWAYS and EVENTUALLY.

Table 9: The options in the options panel are identical to LTLvis

Send Data

When all the data is ready, users can send the data to the LTL planner. The LTL planner used in this work is modified from the RHTL package [14]. By adding path preference logic (Alg. 3) in the traditional LTL planner, the resulting path generated from the new planner will attempt to satisfy both the LTL specification and the user input requirement. The details of this planner are explained in the next section.

PLANING USING E-LTLvis

The topic of this section will focus on how to do task planning using the proposed interface (E-LTLvis). As the interface provides additional information besides the LTL specification, regular LTL planner won't generate intended result if these information is not utilized.

Consider the following scenario. Node q_0, q_1, q_2 are atomic propositions (Figure 30). In LTL mode, q_0, q_1, q_2 consists of the specification $q_0 \wedge GF(q_1 \wedge F(q_2))$.

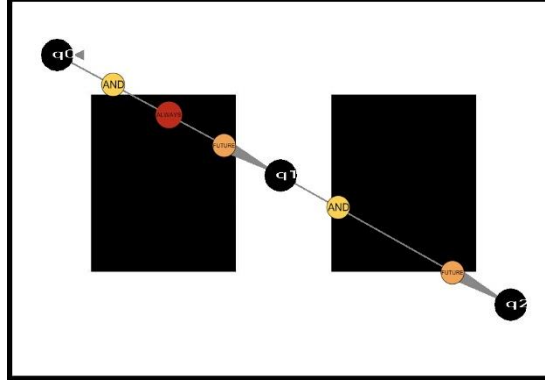


Figure 30: A sample LTL specification

After processing this specification through a tool such as lomap⁹ [14], two Büchi automata will be generated (Figure 31).

The first one in Figure 31 is transition system and second one is LTL automaton. To separate this transition system and the roadmap, we denote it as global transition system and the roadmap as local transition system. Global TS only has atomic propositions and the transitions between atomic propositions, while local TS represents real roadmap information and the transitions between all the nodes in the roadmap. In the above example, the local transition system is presented in Figure 32.

⁹ lomap is one sub-package of RHTL. It provides functions to visualize automata generated from LTL2BA [19].

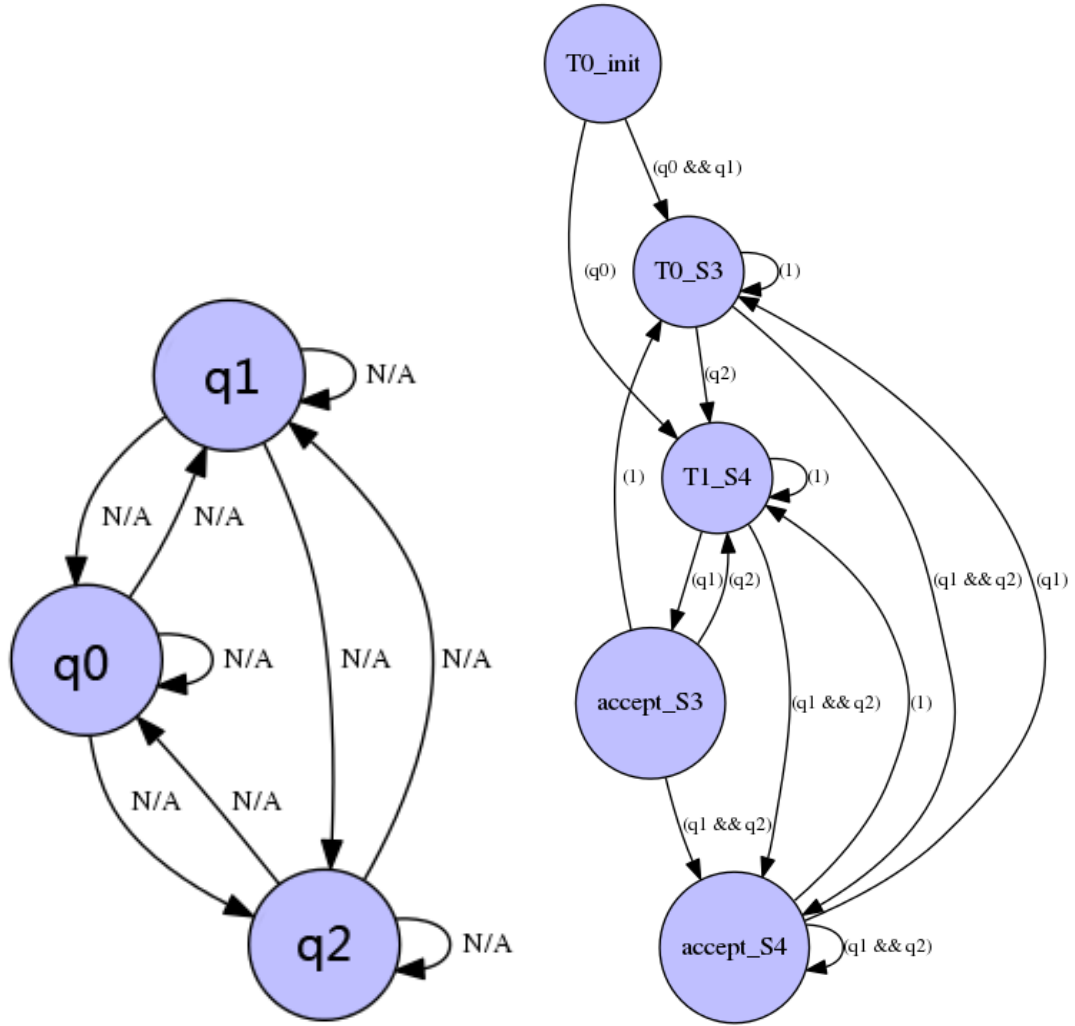


Figure 31: Left: TS. Right: Buchi automaton which corresponds to LTL formula $\mathbf{q_0} \wedge \mathbf{GF(q_1 \wedge F(q_2))}$

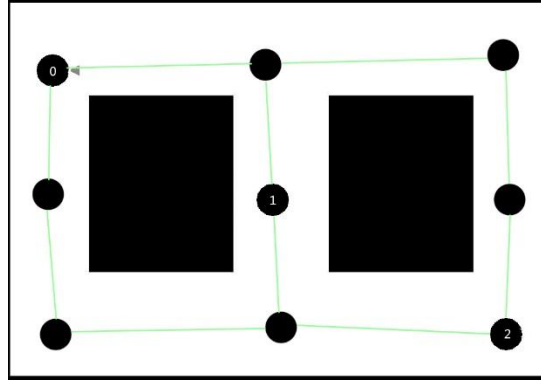


Figure 32: Local TS

Among all the nodes from the local transition system, node 0, node 1, node 2 share the same coordinates with labels q_0 , q_1 , q_2 respectively in the global transition system. The simplified product¹⁰ of the above two automata is a finite state automaton A as in Figure 33. For example, the state $(q_0, T0_S3)$ is a paired state of q_0 in global TS and $T0_S3$ in Büchi automaton.

It is a finite state automaton $A := (\Psi, q_{init}, \delta, \Sigma, F)$ where

- $\Psi = \{T0_S3[q_0], T1_S4[q_0], T0_S3[q_1], T1_S4[q_1], accept_S3[q_1], T1_S4[q_2], T0_S3[q_2]\}$
- $q_{init} = T0_S3[q_0]$
- $\Sigma = 2^\Pi$ accepting all finite words over 2^Π
- $\delta: \Psi \times \Sigma \times \Psi$ is the deterministic transition relation in Figure 33.
- $F = \{accept_S3[q_1]\}$

¹⁰ The number of states in product automaton is $N1 \times N2$, where $N1$, $N2$ are the number of state in there two the automaton respectively. However, there may exist some unreachable states.

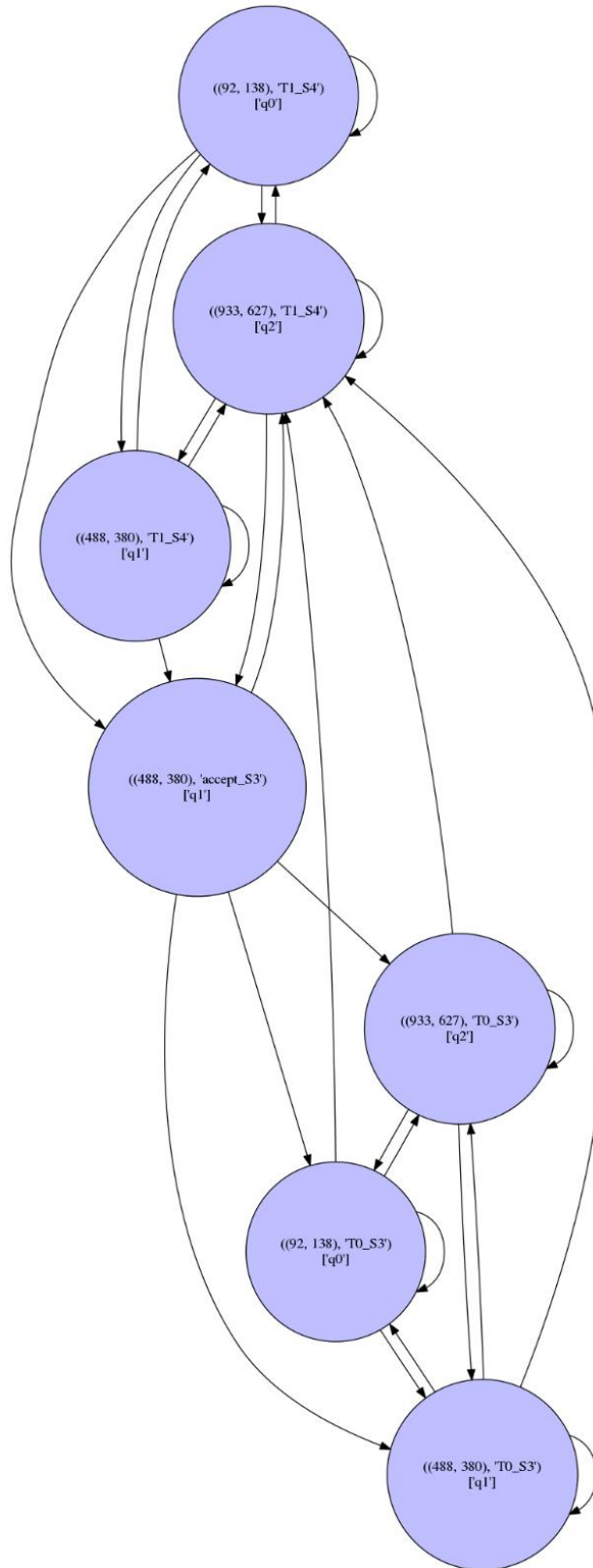


Figure 33: The product automaton

Planning under the LTL constraint $q_0 \wedge GF(q_1 \wedge F(q_2))$ is equivalent to finding a path from q_{init} to any accepting states in A . Depending on the weight of each transition, different planner may adopt different path finding algorithm, then generate different path plan. If the shortest distance is preferred,

$$w = \{w_{e_{q_i q_j}} \mid e_{q_i q_j} \in \delta, w_{q_1 q_2} \text{ is the shortest distance from } q_1 \text{ to } q_2\}$$

The solution can be generated using Dijkstra's algorithm [9]. The expected path will be:

$$\pi =$$

$$(T0_S3[q_0], T0_S3[q_1], T1_S4[q_2], accept_S3[q_1], T1_S4[q_2], accept_S3[q_1] \dots)$$

Namely, the robot starts from q_0 then repeatedly visits q_1 and q_2 . As the shortest path is preferred. The resulted path should be (Figure 34):

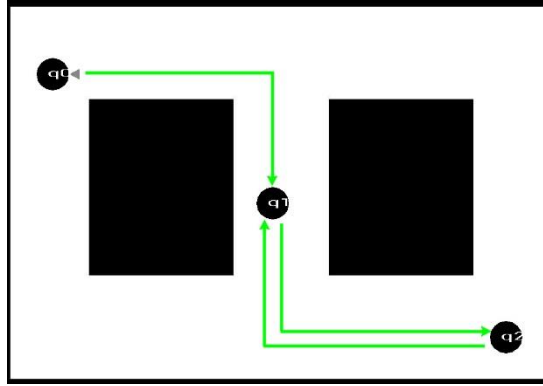


Figure 34: The path generated using shortest path

Now, let's introduce another constraint as:

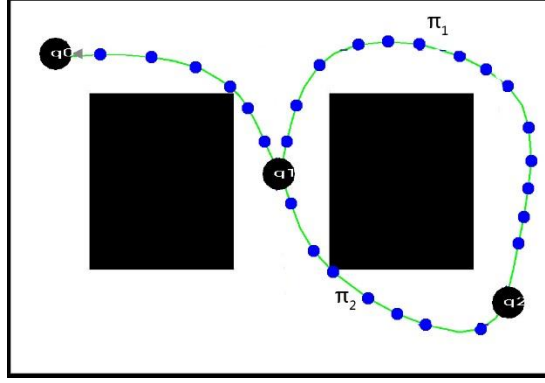


Figure 35: Extra path requirement for $q_0 \wedge GF(q_1 \wedge F(q_2))$

The other requirement provided in Figure 35 specifies the preferred path set between two assigned nodes. The expected result (shown in Figure 36) should satisfy both LTL specification and preferred path constraints.

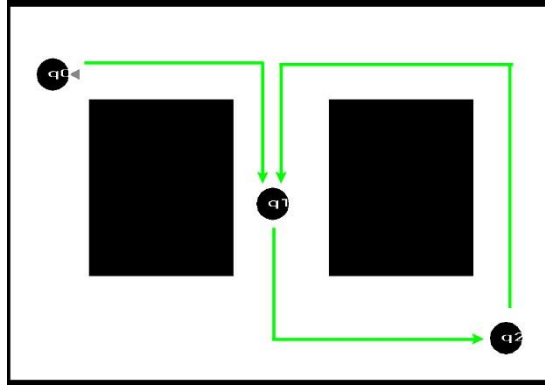


Figure 36: The path generated by the extended planner

This research proposes an extended planner (Alg. 3) based on the general LTL planner [14].

It takes the product automaton A , the local transition system TS and the preferred path set D as inputs. Here, We denote a preferred path set as D , where

$$D = \{\pi_{uv} | \pi_{uv} = (q_u, n_{a1}, n_{a2}, \dots, n_{am}, q_v), q_u, q_v \in \Psi, n_{a1}, n_{a2}, \dots, n_{am} \in Q_{TS}\}$$

Algorithm 3 EXTENDED-PLANNER

Input: A, TS, D **Output:** π_{ltl}

1. create an empty list π_{ltl}
 2. **for each** π_{ij} **in** D **do:** //set all the preferred paths to highest priority to be chosen.
 3. $\langle q_i, q_j \rangle \leftarrow \langle \pi_{ij}[1], \pi_{ij}[|\pi_{ij}|] \rangle$
 4. **if** $(q_i, q_j) \in \delta$ **then:**
 5. change the weight $w(q_i, q_j)$ to α
 6. find the shortest path π_{A0} with minimum sum of edge weight from q_{init} to q_{accept} in A //based on the modified priorities above
 7. **for** k **in range** $(1, |\pi_{A0}| - 1)$ **do:**
 8. $\langle q_h, q_m \rangle \leftarrow \langle \pi_{A0}[k], \pi_{A0}[k + 1] \rangle$
 9. found $\leftarrow \perp$
 10. **for** π_D **in** D **do:** // π_D is a sequence of nodes in Q_{TS}
 11. **if** $\pi_D[1] = q_h$ **and** $\pi_D[|\pi_D|] = q_m$ **and** π_D is valid in A **then:**
 12. append π_D to π_{ltl}
 13. found $\leftarrow \top$
 14. **if** \neg found **then:**
 15. find the shortest path π'_D from q_h to q_m in TS
 16. append π'_D to π_{ltl}
 17. append $\pi_{A0}[|\pi_{A0}|]$ to π_{ltl} //this is q_{accept}
 18. **return** π_{ltl}
-

At line 5, α is infinitesimal and $\alpha \in \mathbb{R}_+$. It is much smaller than the smallest weight in W .

At line 11, valid means this path never visit any avoiding states in A .

At line 12, 16, 17, each append operation to π_{ltl} adds the element to the tail of the list.

Algorithm 3 works as following. Assume $\pi_{ij} \in D$, the algorithm first checks if there is transition $(q_i, q_j) \in \delta$ (line 4). If such transition exists, it changes its weight to α . Here, $\alpha \in \mathbb{R}_+$ denotes an infinitesimal value. This can increase the priority of the preferred path set when calculating the shortest path π_{A0} from q_{init} to q_{accept} in line 6. After finding π_{A0} , we need to replace each transition $(q_i, q_j) \in \pi_{A0}$ with a corresponding transition from either the preferred path set D or the transition system TS . As the preferred path set D has higher priority, if π_D exists in the preferred set, we add it to the path π_{ltl} .

Otherwise, we find a shortest alternative π'_D in TS and add it to $\pi_{l tl}$. After q_{accept} is visited, $\pi_{l tl}$ is completed.

Chapter 5

Experiments

In this section, we are going to test our interface and planner on a real robot - turtlebot. Turtlebot can be controlled using a ROS package. It contains two major hardware devices: Kinect and iRobot base. The Turtlebot project also contains many useful packages. For example, `turtlebot_navigation` is one of the most popular packages used to localize the robot by itself. Also, we use `turtlebot_rviz` to visualize the environment. The final goal of this experiment is to use the proposed interface to send an LTL specification and a preferred path set to the planner. The planner should generate a path plan and order the turtlebot to execute the plan.

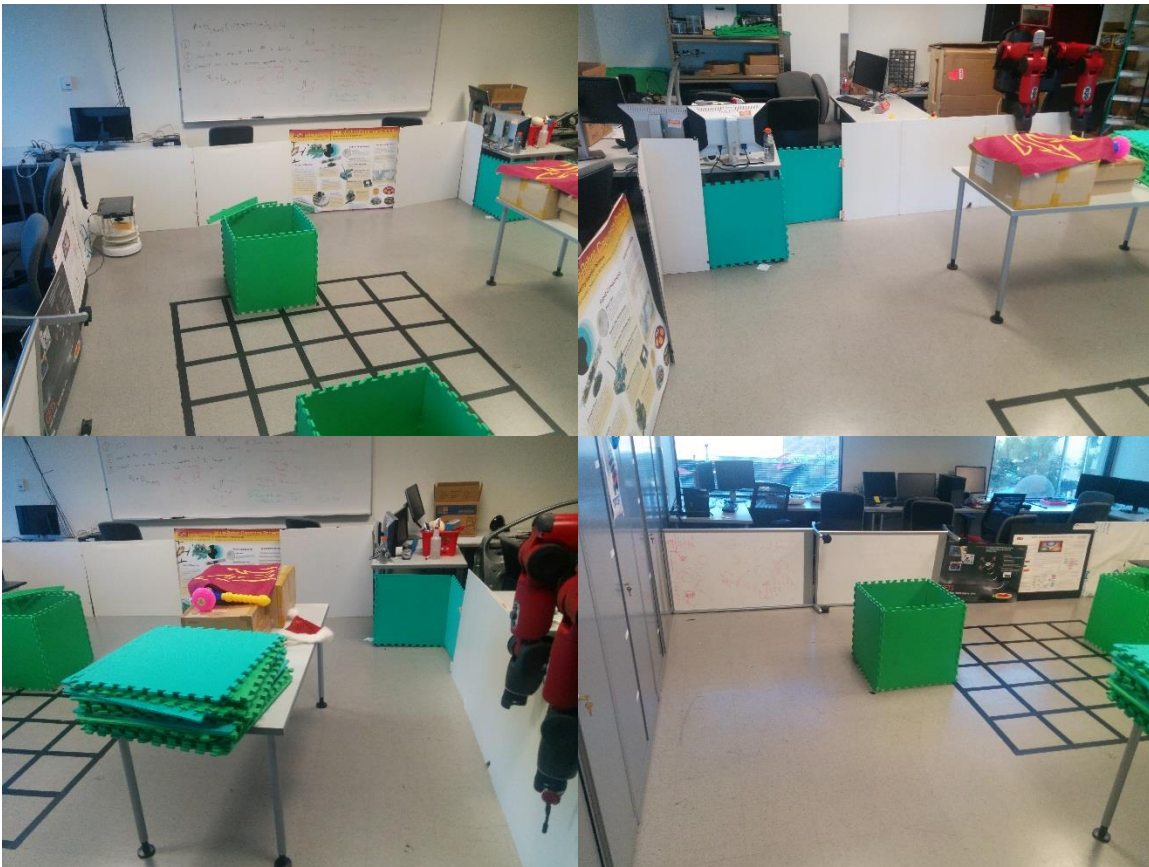


Figure 37: The experiment environment

The testing environment is located in Arizona State University Centerpoint building (Figure 37).

The real layout of the environment is a rough 5 meters by 5 meters area with some objects set up as obstacles. After using *turtleBot_navigation* package to scan the environment, we got a portable graymap format (pgm) file. PGM file uses 8 bits per pixel. For each pixel, the larger the digit is, the higher probability the pixel is occupied by obstacles. Below is the graymap for above environment (Figure 38).

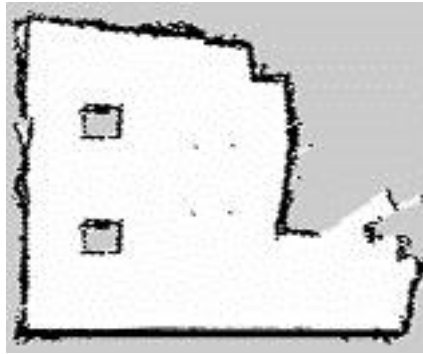


Figure 38: The graymap for above environment

Experiment 1

We have done two experiments. For the first one, we provided an LTL specification:

$$(q0 \rightarrow X q1) \wedge (q0 \wedge Fq2) \wedge (q0 \rightarrow X \neg q2)$$

And two required inputs from E-LTLvis:

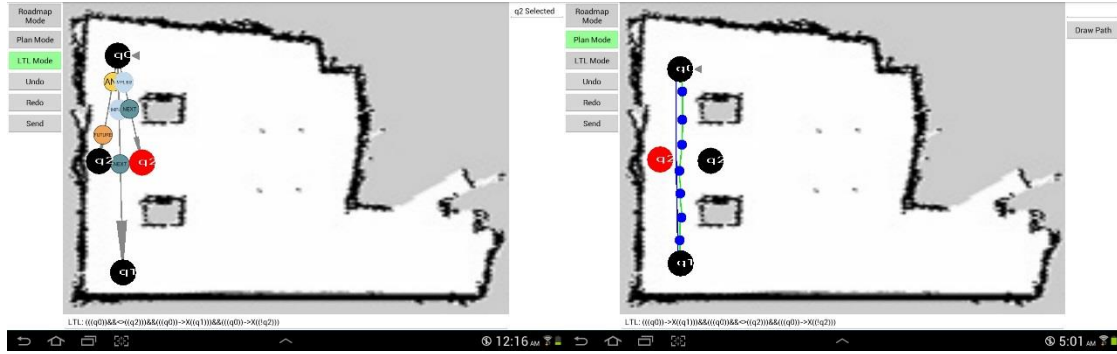


Figure 39: Two required inputs from E-LTLvis

The left figure is the graphic representation of the LTL specification and the right figure is the user sketched path and its corresponding BMP.

The task of the turtlebot is to follow the specification:

$$(q0 \rightarrow X q1) \wedge (q0 \wedge Fq2) \wedge (q0 \rightarrow X \neg q2)$$

In natural language, it means “the turtlebot is required to start from q0 and head for q1 while avoiding q2 before q1 is reached. Then it will reach q2 eventually”.

However, we also provided a preferred path from q0 to q1 via p2. This path violates the LTL specification. Thus, the turtlebot should ignore the preferred path and find an alternative path to reach q1. The process is displayed in below figures (Figure 40 to Figure 45)

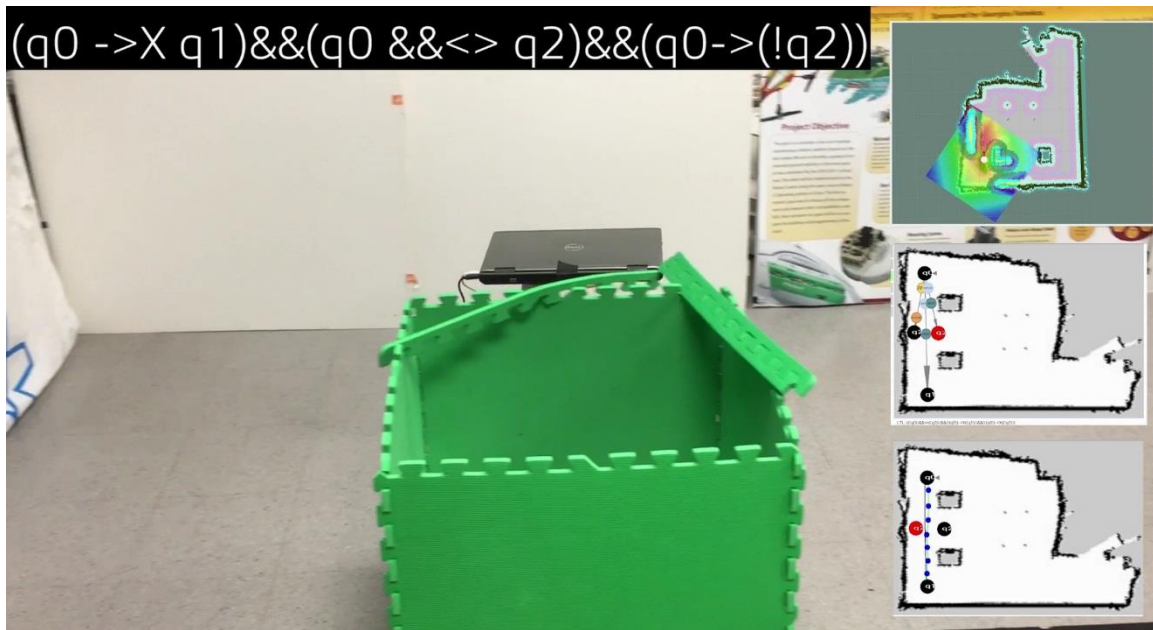


Figure 40: The turtlebot started from q_0 and went north.

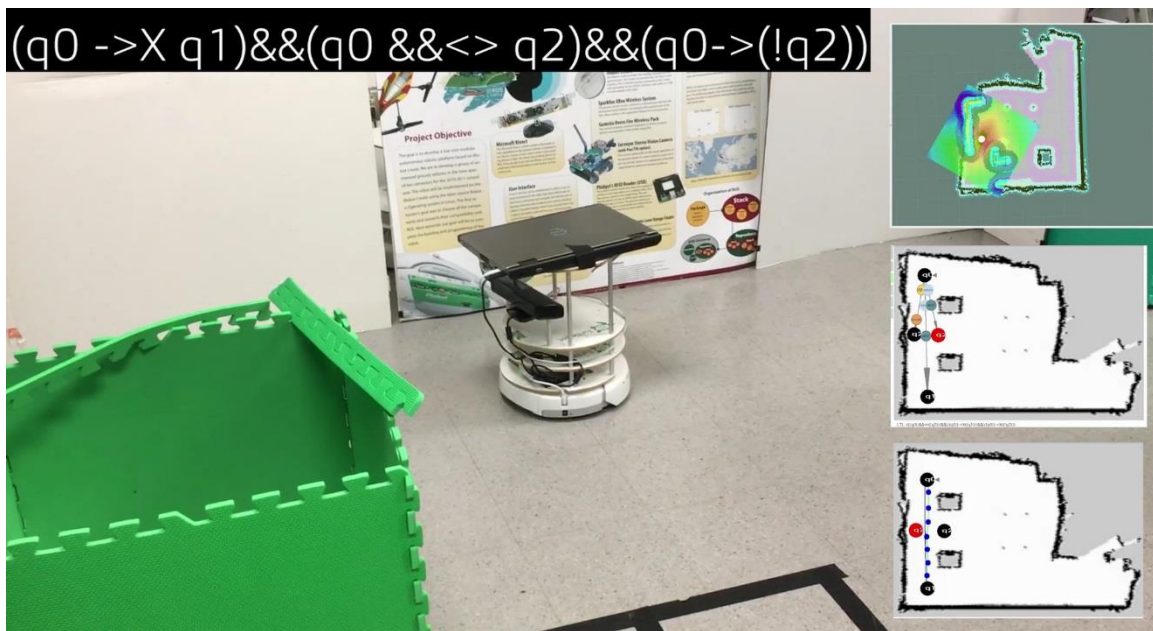


Figure 41: The turtlebot reached the turn point and was ready to go east.

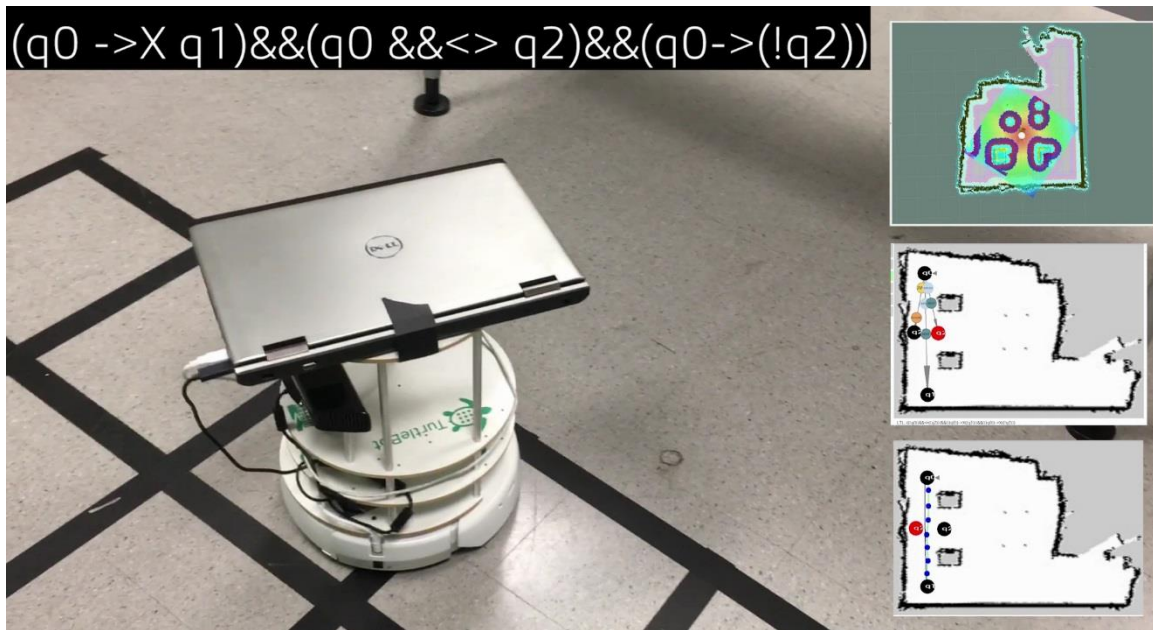


Figure 42: The turtlebot reached the point in the north of q_2 and continued to go east.

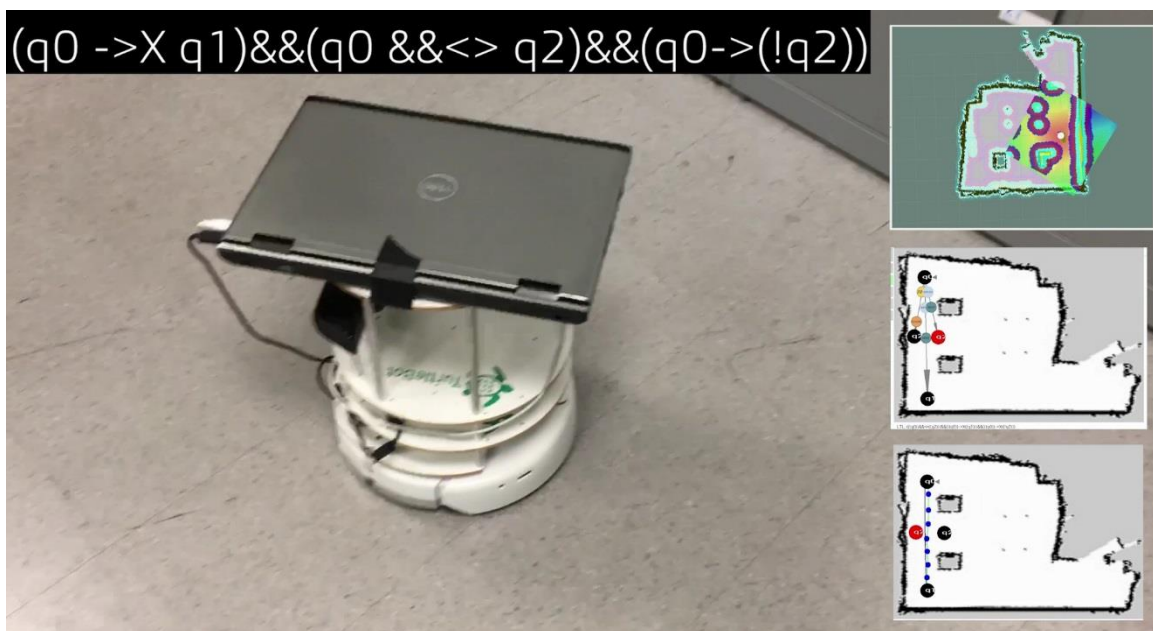


Figure 43: The turtlebot reached the turn point and was ready to go south.

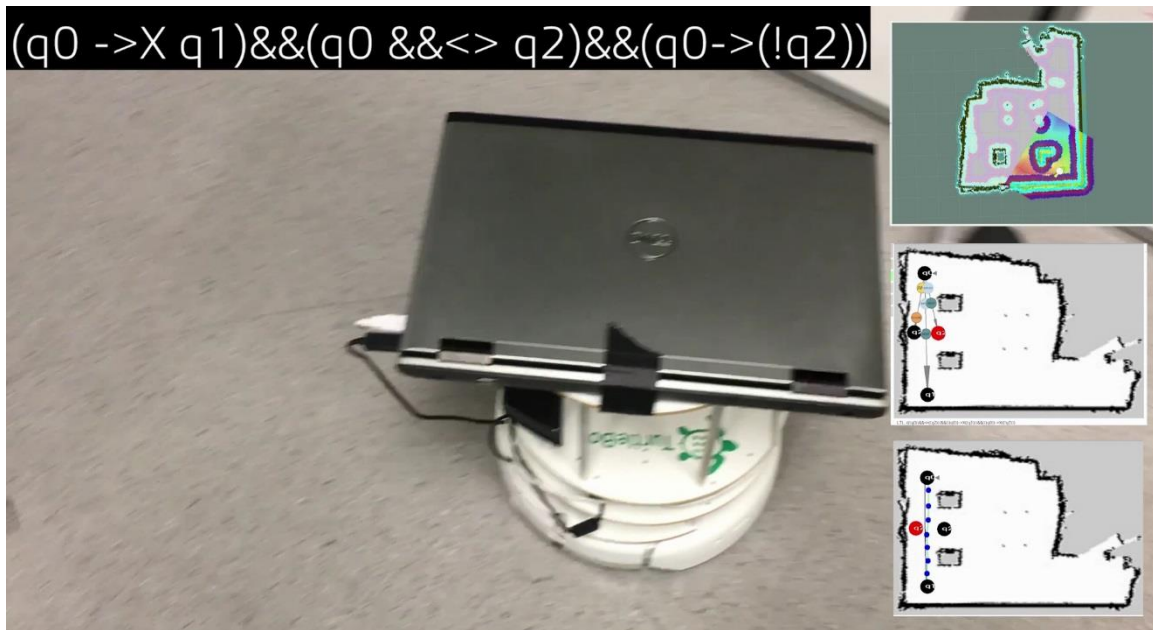


Figure 44: The turtlebot reached q_1 and was ready to go to q_2 .

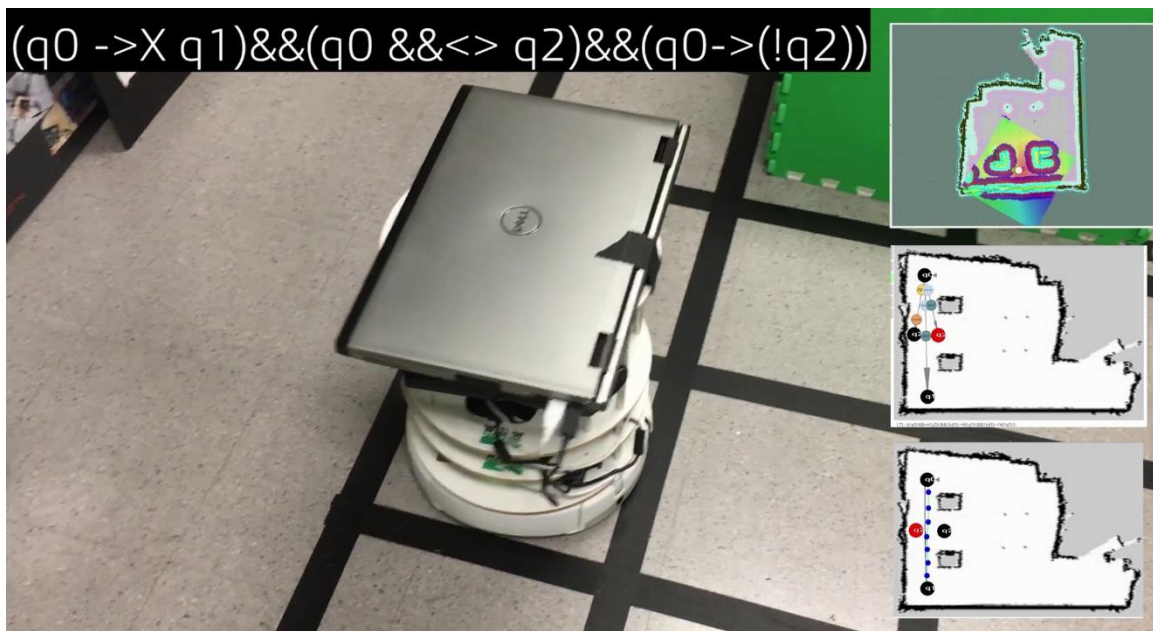


Figure 45: The turtlebot reached q_2 . Task completed.

In this experiment, the turtlebot succeeded in avoiding q2 before visiting q1. The preferred path was not adopted. The LTL specification was satisfied.

Experiment 2

For the second experiment, we provided the LTL specification:

$$(q0 \wedge GF (q1 \wedge F q2))$$

And two required inputs for E-LTLvis:

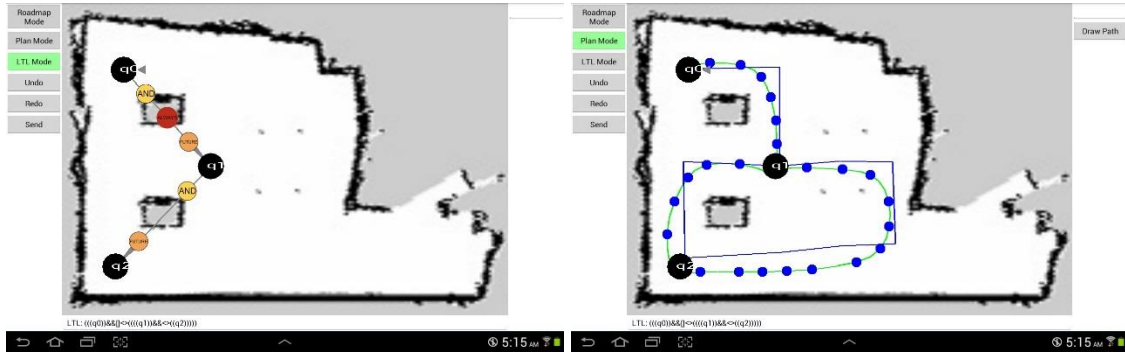


Figure 46: Two required inputs for E-LTLvis

The task of the turtlebot is to follow the specification $(q0 \wedge GF (q1 \wedge F q2))$. In natural language, it means “the Turtlebot is required to start from q0 and head for q1 then to q2 and loop between q1 and q2”. Furthermore, we also provided a preferred path set from q0 to q1, q1 to q2 and q2 to q1. These paths do not violate the LTL specification. Thus, the Turtlebot should obey the preferred path set. The process is displayed in below figures (Figure 47 to Figure 54)

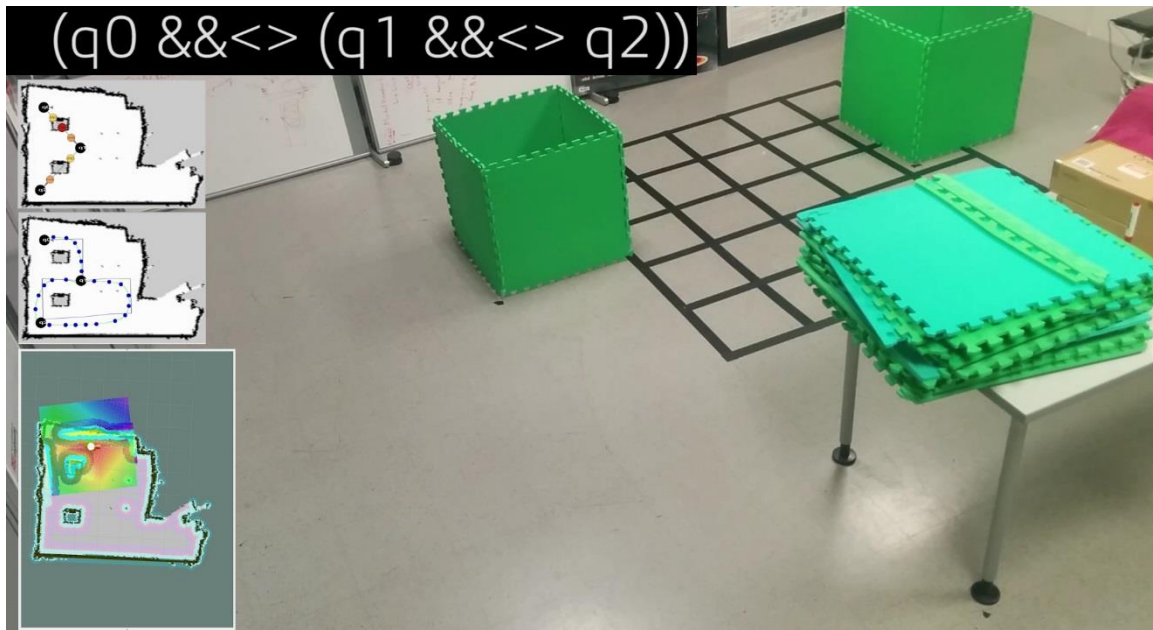


Figure 47: The Turtlebot started from q_0 and headed east.

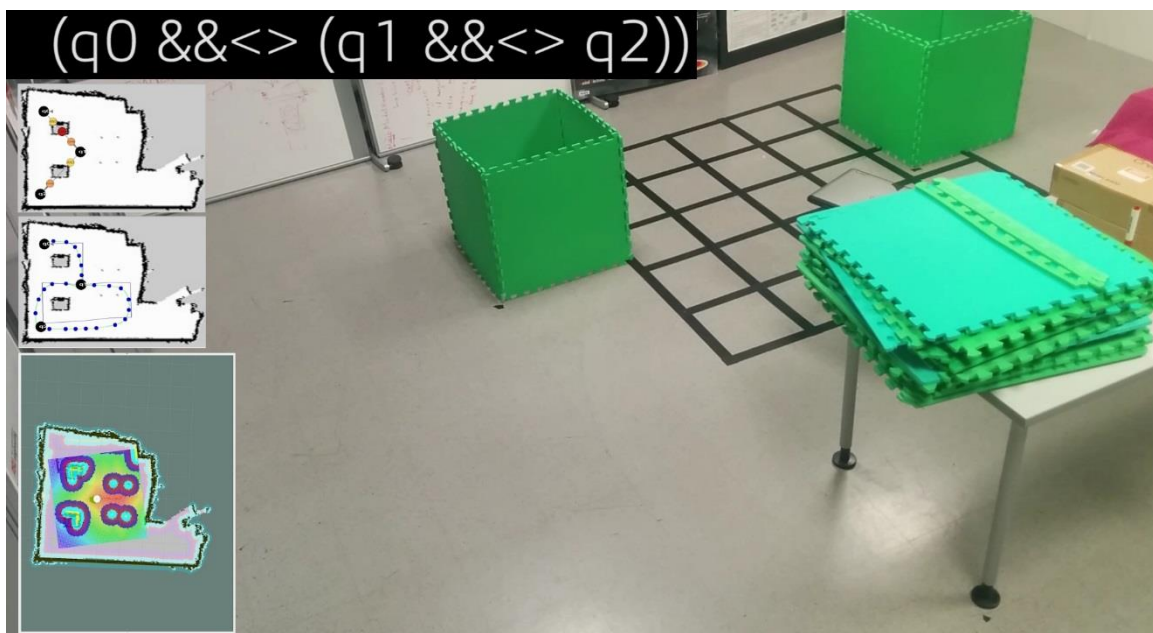


Figure 48: The Turtlebot reached q_1 .

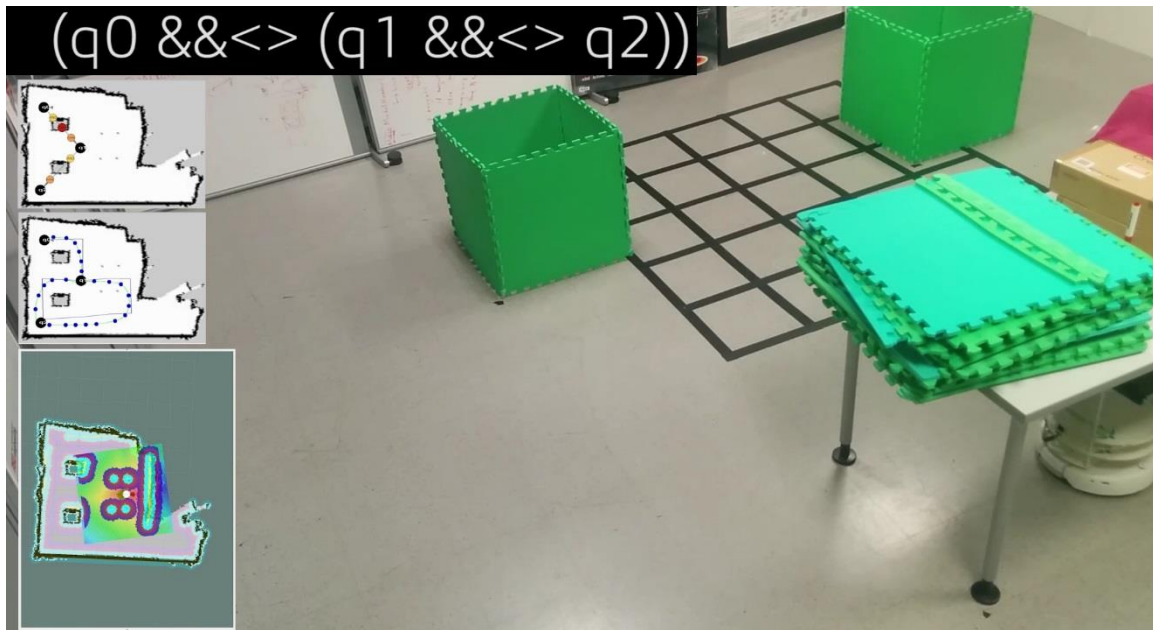


Figure 49: The Turtlebot continued to go east.

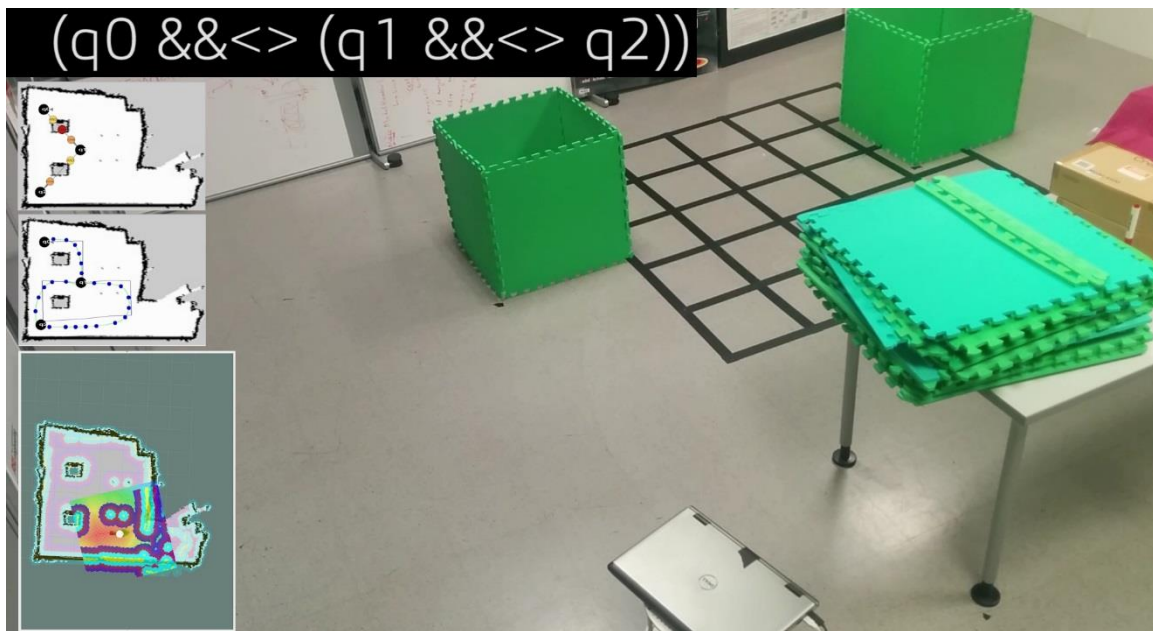


Figure 50: The Turtlebot made a U turn around the legs of the table.

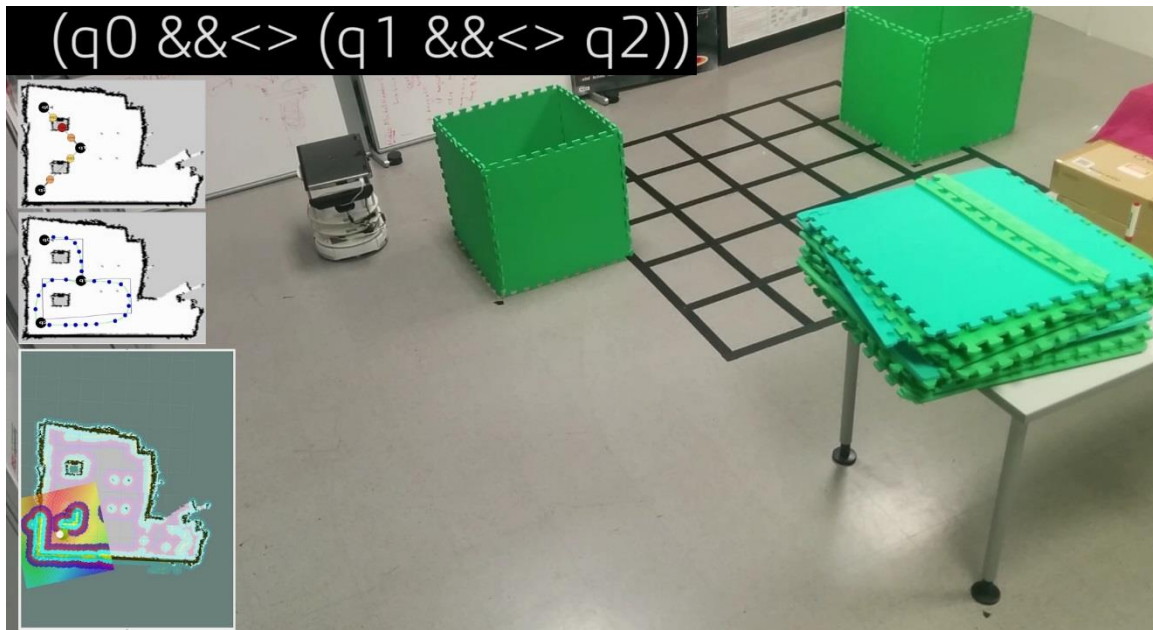


Figure 51: The Turtlebot headed west and reached q2.

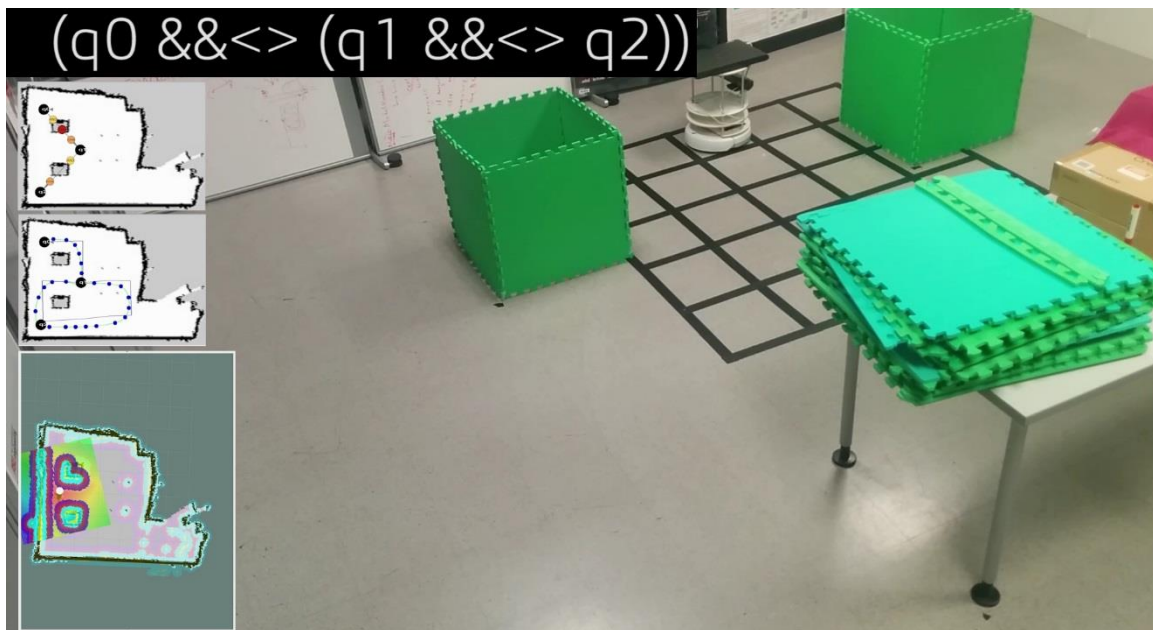


Figure 52: The Turtlebot headed north and reached the turn point.

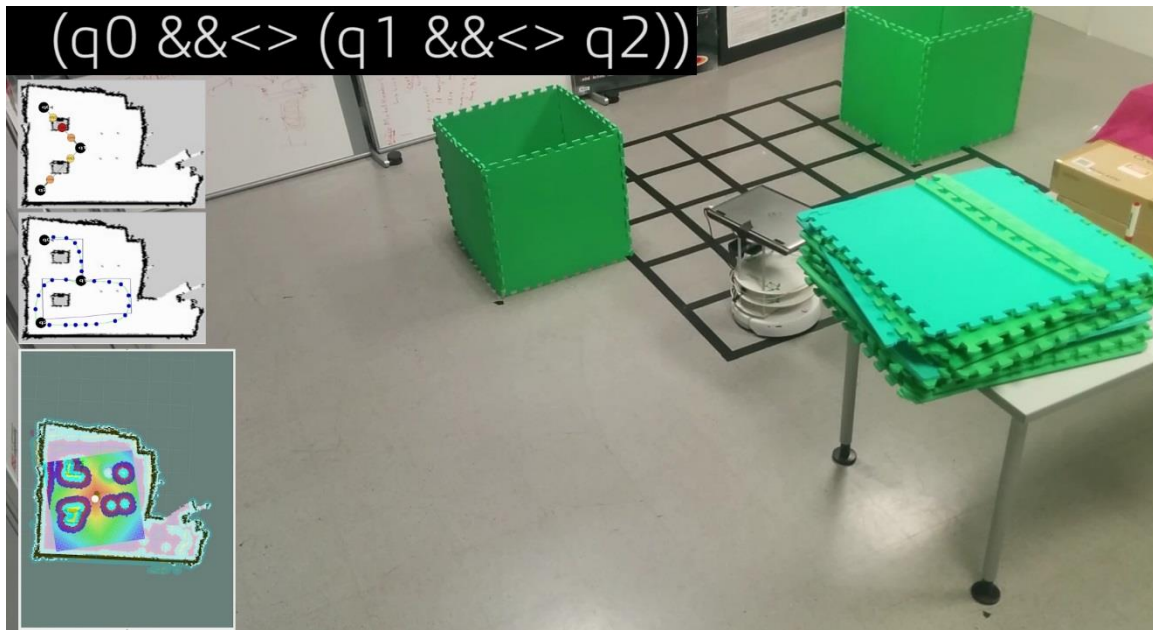
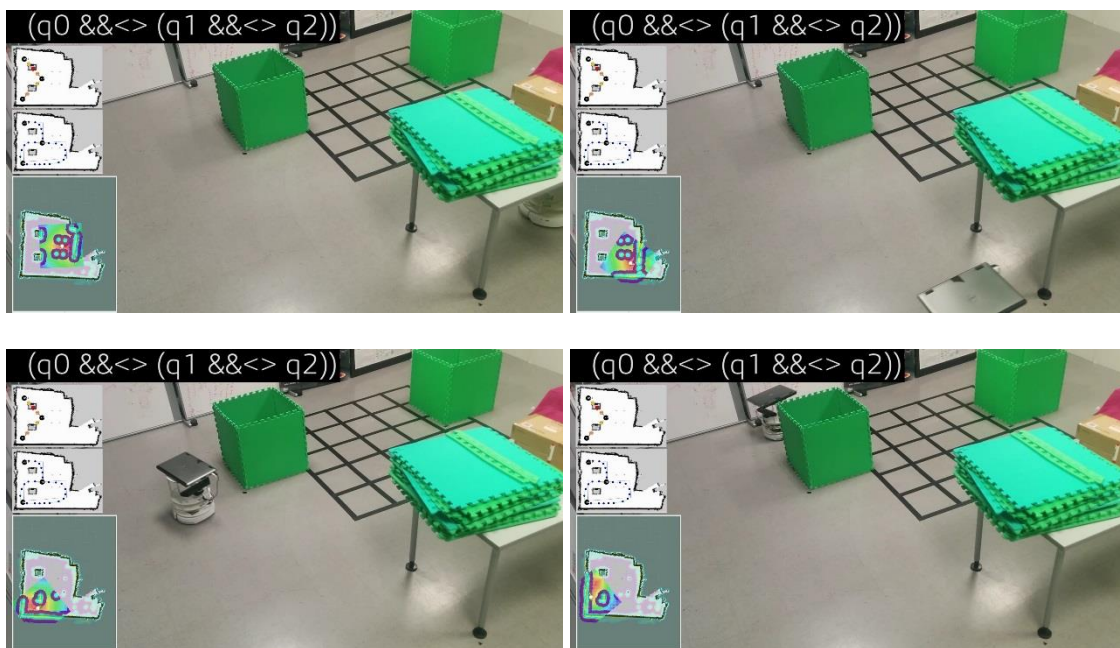


Figure 53: The turtle reached q1 again.



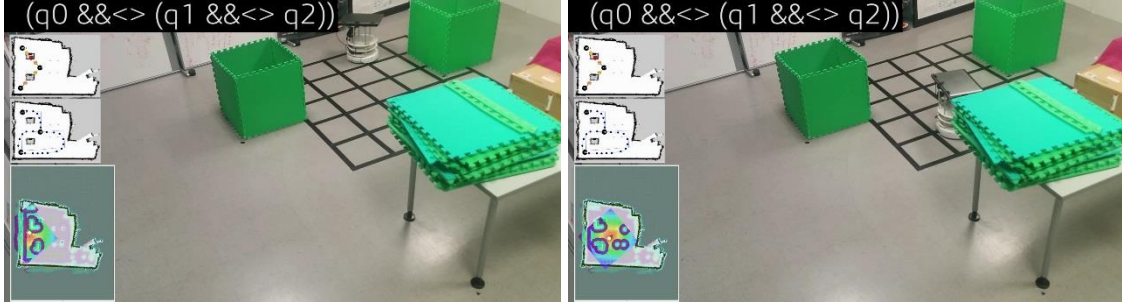


Figure 54: The turtle continued above actions until we manually stopped the planner.

During the second experiment, the Turtlebot always followed the preferred path. The task was completed.

Extra Experiments

We also repeated the above experiments in another environment and recorded the trace of the robot.

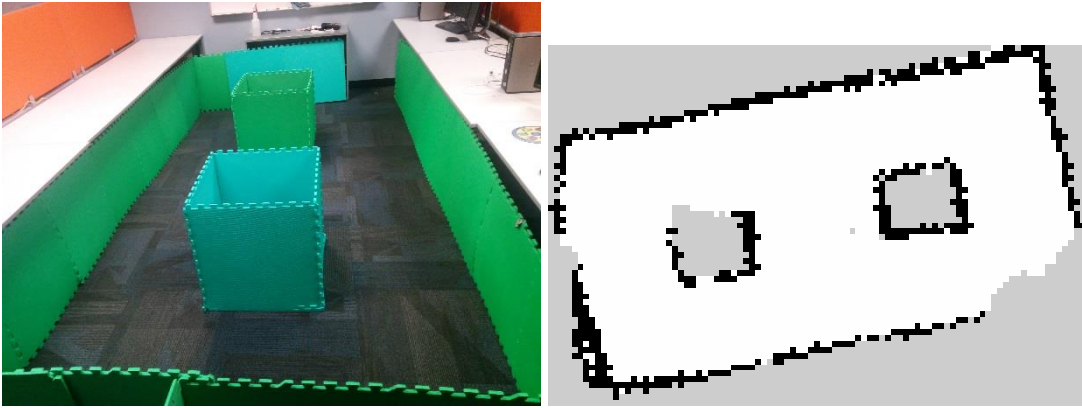


Figure 55: Experiment environment and scanned greymap.

Experiment 1

For the first one, we provided an LTL specification:

$$(q0 \rightarrow X q1) \wedge (q0 \wedge Fq2) \wedge (q0 \rightarrow X \neg q2)$$

And two required inputs from E-LTLvis:

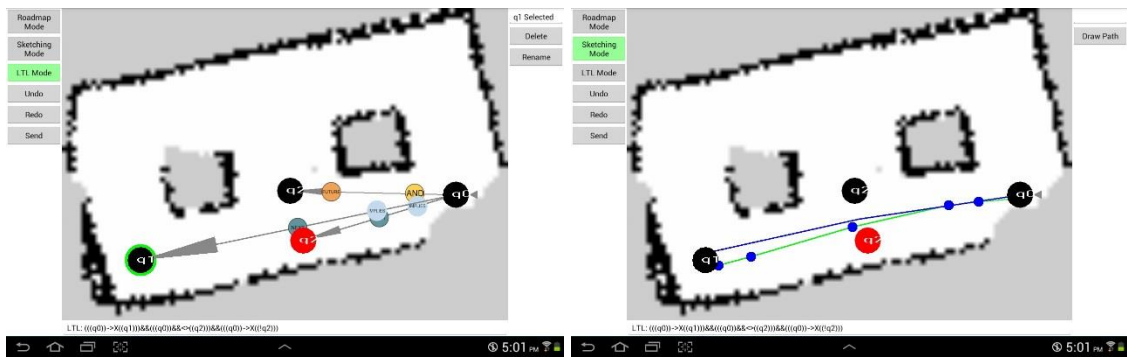


Figure 56: Two required inputs from E-LTLvis

In this experiment, the Turtlebot succeeded in avoiding q2 before visiting q1. The preferred path was not adopted. The LTL specification was satisfied. Figure 57 shows the robot trajectory.

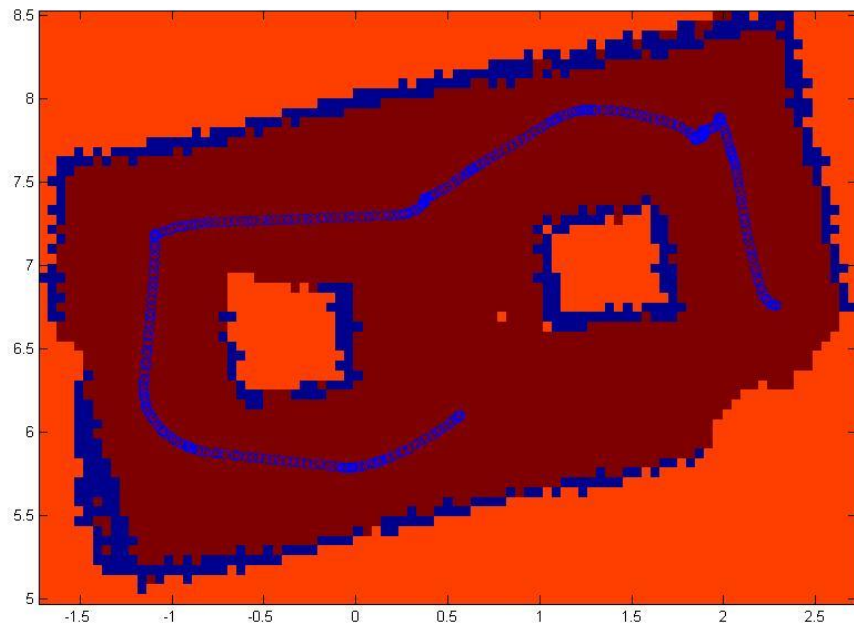


Figure 57: Robot trajectory for experiment 1

Experiment 2

For the second experiment, we provided the LTL specification:

$$(q0 \wedge GF (q1 \wedge F q2))$$

And two required inputs for E-LTLvis:

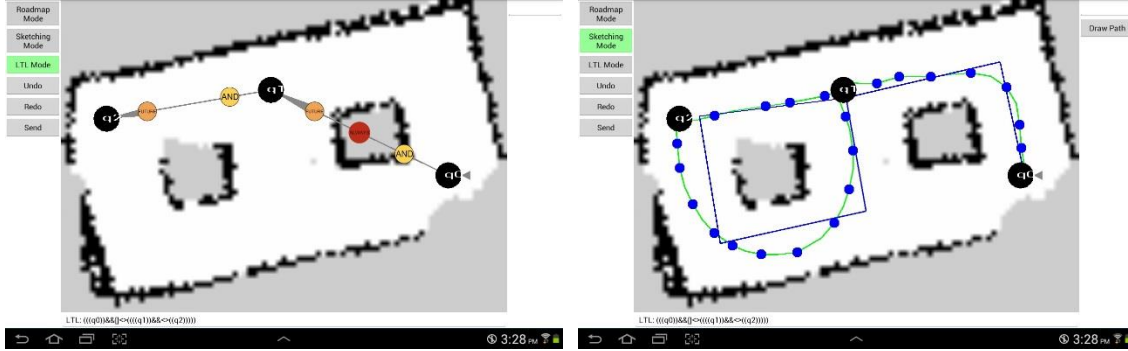


Figure 58: Two required inputs for E-LTLvis

During the second experiment, the Turtlebot always followed the preferred path. Figure

59 shows the robot trajectory.

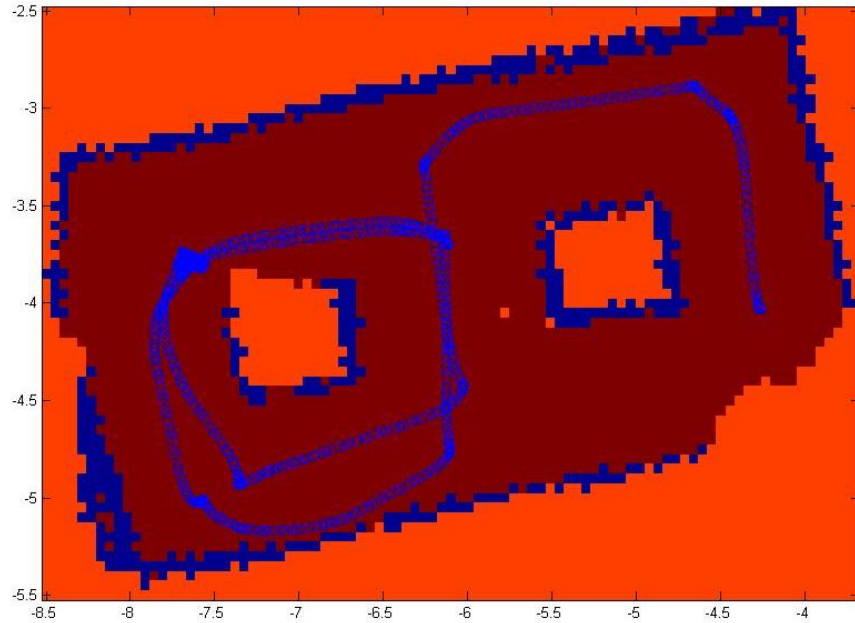


Figure 59: Robot trajectory for experiment 2

Chapter 6

RELATED WORK

Control Interface

In [13], the authors had shown that planning using sketch based interfaces can be improved using path correction. The interface is designed to be friendly to all users including the ones with Parkinson disease. Once users draw a path bypassing an invalid region (collisions), this interface will auto-correct the invalid sub-path to a valid Bezier curve. It can also correct multiple collisions in the same path.

Previous attempts to combine planning graphical user interfaces with Linear Temporal Logic are presented in [5]. The authors there define a graphical language for LTL. With the proposed interface, the user can ‘draw’ the formula with edges and nodes. This interface supports LTL operators such as ‘AND’, ‘OR’, ‘ALWAYS’, ‘FUTURE’, ‘UNTIL’, ‘IMPLIES’, ‘NEXT’ and ‘UNTIL’. Users can simply drag and drop the nodes and edges to generate the LTL formula.

Most of home robot users are non-expert programmers. Sakamoto et al. proposed a robot control interface especially for home cleaning robots [1]. In their work, a vacuum machine was controlled by the proposed interface to complete a set of tasks. The authors define a set of gesture commands for a set of actions. They include *move* with an open curve, *vacuum* with a closed curve, *stop* with a cross mark, etc.

The proposed work from [3] is similar to our research. But instead of commanding a robot to follow a path, they assign a start and an end position for the robot and the robot will explore its path by searching the RRT of the given map. Their interface and planner system allow users to assign destinations outside of the given map by assuming these

unknown regions are spatially free. They also conducted tests to prove that the autonomous control is more efficient than manual control, and to prove that two copies of this system can run simultaneously.

The work presented in [12] proposed an interface solution controlling multiple robots. In the interface design, users are able to give commands to each individual robot in each individual interface layer. Only one interface layer can be active at a time. Users can also set the active time for each layer. When there is time conflict between layers, the system will automatically delay the later one, so that the robots will not conflict during the tasks.

In our interface, when a user sketch crosses over an undefined region, the interface can still find its BMP. But if this BMP conflicts with the LTL specification, it will be ignored without notifying the user. The interface also inherits all the gesture language from LTLvis [5]. For example, long pressing the screen can create a new node and single tabbing two nodes can link them with “AND” relation. This interface also has multiple layers (modes). Instead of configuring different robots in different layers, we configure one robot in different layers for different requirements. For example, the sketch mode is used to configure user sketch path and LTL mode is used to edit the LTL specification.

LTL Planner

In terms of LTL planning, in [18], the authors had proposed a solution to generate the optimal plan under a temporal logic specification. The system accepts the LTL specification and the transition system as input and creates a product automaton. Then, it uses a predefined weight heuristic to find the optimal cycle path including at least one accepting node with minimum cost in this new automaton.

To let the robot complete the mission in a dynamic environment, Ulusoy et al. proposed a solution in [14]. As the robot sensors have limited ability to scan the whole environment, they define a limited region as the local environment. The local environment can be changed as time proceeds and the robot needs to be aware of these changes. In [14], the environment is a $n \times m$ grid map. Every time, the robot enters into the next grid cell, the solution will regenerate the local environment and re-calculate the optimal path under this new local environment.

Dynamic Time Warping

Dynamic Time Warping (DTW) is a similar approach to ours calculating the similarity of two trajectories [25, 26]. For two trajectories $X_N = (x_1, x_2 \dots, x_N)$ and $Y_M = (y_1, y_2 \dots, y_M)$,

$$DTW(X_N, Y_M) = cost(x_N, y_M) + \min(DTW(X_{N-1}, Y_M), DTW(X_N, Y_{M-1}), DTW(X_{N-1}, Y_{M-1}))$$

Where $cost()$ is distance between two symbols x_N, y_M . In my work, $cost()$ is defined as the distance from a point in path A to a corresponding line segment in path B and DTW is a similar concept as CWPD. BMP is an approach to find the best matching among a large number of sequences based on the CWPD.

Others

In [27], the authors propose an approach to extract navigation states in the form of relative, robot-centered spatial descriptions from a hand-drawn map. First, the user needs to draw objects by sketching a polygon. During the sketching process, a delimiter is included to separate the string of coordinates for each object. After all objects have been

drawn, another delimiter is included to indicate the start of the robot trajectory. The post processing will start once the robot trajectory is drawn. For each point in the trajectory. System will build a view within the radius of the sensor range and a spatial description is generated relative to the objects within the range. If there is an object detected at the current point in the robot trajectory, a formal description “Object is to the left of the Robot” will be generated. The approach represents a first step in studying the use of spatial relations as a symbolic language between a human user and a robot for navigation tasks.

The work in [28, 29, 30] is based on [27]. The authors present a strategy for extracting qualitative route information from a sketched route map and then they show how this information can be used for robot navigation along the sketched path. Landmark states are the labeled objects on the map. Instead of extracting state information from all objects, this strategy only extracts landmark states at critical path nodes. When there is an action change at a node in the path, like “stop”, “turn left” etc., this node can be treated as a critical path node. A sequential compilation of steps is being generated as they are encountered along the sketched path. They also did experiments on a robot simulator in [28]. First, a path is drawn on a hand-drawn map. Then a sequential compilation of steps is generated for the critical path nodes along the sketch path. Then, the robot goes forward at the starting position. When the robot approaches a critical path node, after checking all state information matched at the current state, the robot will perform the corresponding actions. Then, the robot will keep executing the last action until it reaches next critical path node. When all the critical path nodes are visited, the task is completed. In [30], they solve the problem when objects are too far from the robot by incorporating an adaptive sensory radius algorithm.

In [2], the authors have updated the interface proposed above with some useful features. The interface now can distinguish the objects by recognizing any closed polygon. The objects can be deleted, moved and labeled. The user can undo the most recent operation. During the sketching, duplicate points are pruned to make the algorithm more efficient. They conducted experiments to test the improvements. The results showed that the interface is as easy as using pencil and paper.

Based on [28], the authors increased accuracy when searching the match of the compilation of steps in [31]. They added an adjustable threshold, landmark distance to the qualitative landmark states (QLS) conditions to provide better confidence. They also used a reactive obstacle avoidance algorithm to ensure that the robot does not collide with any objects.

In [32, 33], the authors propose an interface allowing one operator to manage a team of robots. In the interface, users can originate multiple destinations and the interface will measure the distance between destinations to the starting position of the robots, and then assign the closest target to each robot. To avoid congestion in navigation, robots need to wait to begin moving before the last robot finishes its task. The interface also supports path commands. The path command will be segmented into a series of sub-destinations on a fixed interval length. The robot position will be updated in the interface in real time as a relative position to the landmarks. They also did a usability study to prove its ease of use.

There are major constraints for [32, 33]. Before navigation, the sketched map and the map detected by the robot should be matched and their objects should be labeled correctly. Fail to match or inaccuracy would cause the robot to not be able to navigate

through the scene. In [34], authors present a technique to perform scene matching between a sketched map and a map of environment. The approach neutralizes the difference between the two maps by scaling, orientating, translating the objects. In the experiment, the approach successfully matched 7 out of 8 cases.

CONCLUSIONS AND FUTURE WORK

Our current research aims to solve the path planning problem with a path requirement under an LTL specification for a single robot. We combined the ease of use of a sketch interface and LTLvis [14] into a hybrid interface to allow users input customized paths. We conducted two sets of experiments. The interface can express user demands and the planner can realize these demands correctly in the experiments.

In terms of future research, the interface can be extended to multiple robots by adding a cooperation module. Second, we can add a real-time feedback module to the planner so that the users will know how the robots are running. Third, we plan to perform a usability study to test its ease of use.

REFERENCES

- [1] D. Sakamoto, K. Honda, M. Inami and T. Igarashi, "Sketch and run: a stroke-based interface for home robots," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2009.
- [2] M. Skubic, C. Bailey and G. Chronis, "A Sketch Interface for Mobile Robots," in *in Proc. of the IEEE 2003 Conf. on SMC*, Washington, D.C., 2003.
- [3] Y. Ochiai, K. Takemura, A. Ikeda, J. Takamatsu and T. Ogasawara, "Remote control system for multiple mobile robots using touch panel interface and autonomous mobility," in *Intelligent Robots and Systems (IROS 2014)*, 2014.
- [4] G. E. Fainekos, A. Girard, h. Kress-Gazit and G. J. Pappas, "Temporal logic motion planning for dynamic robots," *Automatica*, vol. 45, pp. 343-352, 2009.
- [5] S. Srinivas, R. Kermani, K. Kim, Y. Kobayashi and G. Fainekos, "A graphical language for LTL motion and mission planning," in *Robotics and Biomimetics (ROBIO)*, Shenzhen, 2013.
- [6] A. Bhatia, L. E. Kavraki and M. Y. Vardi, "Sampling-based motion planning with temporal goals," in *International Conference on Robotics and Automation*, 2010.
- [7] T. Wongpiromsarn, U. Topcu and R. M. Murray, "Receding horizon control for temporal," in *Proceedings of the 13th ACM international conference*, 2010.
- [8] B. Hoxha, N. Mavridis and G. Fainekos, "VISPEC: A graphical tool for elicitation of MTL requirements," in *In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Hamburg, Germany, September 2015.
- [9] C. Finucane, G. Jing and H. Kress-Gazit, "LTLMoP: Experimenting with language, Temporal Logic and robot control," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010.
- [10] H. Kress-Gazit, G. E. Fainekos and G. J. Pappas, "Translating Structured English to Robot Controllers," in *Advanced Robotics*, 2008.
- [11] B. Lumpkin, "A High Level Language for Human Robot Interaction," MS Thesis, Arizona State University, 2012.

- [12] K. Liu, D. Sakamoto, M. Inami and T. Igarashi, "Roboshop: multi-layered sketching interface for robot housework assignment and management," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2011.
- [13] J. A. Frank and V. Kapila, "Path Bending: Interface Human-Robot Interfaces With Collision-Free Correction of User-Drawn Paths," in *Proceedings of the 20th International Conference on Intelligent User Interfaces*, 2015.
- [14] A. Ulusoy, M. Marrazzo and C. Belta, "Receding Horizon Control in Dynamic Environments from Temporal Logic Specifications," *Robotics: Science and Systems*, 2013.
- [15] W. Wei, K. Kim and G. Fainekos, "Extended LTLvis Motion Planning interface," in *Conference on Systems, Man, and Cybernetics (SMC 2016)*, 2016.
- [16] A. Pnueli, "The temporal logic of programs," in *Proceedings of the 18th Annual Symposium on Foundations of Computer Science (FOCS)*, 1977.
- [17] G. E. Fainekos, H. Kress-Gazit and G. J. Pappas, "Temporal Logic Motion Planning for Mobile Robots," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference*, 2005.
- [18] S. L. Smith, J. Tumova, C. Belta and D. Rus, "Optimal Path Planning under Temporal Logic Constraints," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on.*, 2010.
- [19] P. Gastin and D. Oddoux, "Fast LTL to B uchi Automata Translation," in *Proceedings of the 13th CAV, volume 2102 of LNCS*, Paris, France, 2001.
- [20] I. Lee and O. Sokolsky, "A Graphical Property Specification Language," In *Proceedings of High-Assurance Systems Engineering Workshop*, 1997.
- [21] M. H. Smith, G. J. Holzmann and K. Etessami, "Events and Constraints: A Graphical Editor for Capturing Logic Requirements of Programs," *Requirements Engineering*, 2001. *Proceedings. Fifth IEEE International Symposium on*, 2001.
- [22] K. Kim and G. Fainekos, "Revision of Specification Automata under Quantitative Preferences," in *IEEE International Conference on Robotics and Automation*, Hong-Kong, 2014.

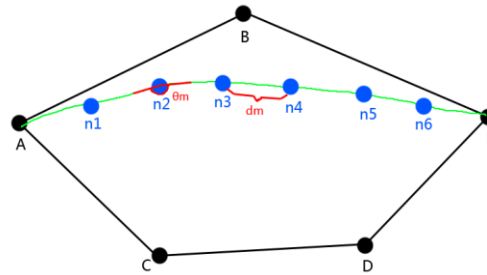
- [23] S. M. LaValle, *Planning Algorithms*, Cambridge University Press, 2006.
- [24] T. H. Cormen, C. E. Leiserson, R. I. Rivest and C. Stein, *Introduction to Algorithms*, MIT Press, 2009.
- [25] M. Müller, "Dynamic Time Warping," in *Information Retrieval for Music and Motion*, Springer Berlin Heidelberg, 2007, pp. pp 69-84.
- [26] E. J. Keogh and M. J. Pazzani, "Scaling up dynamic time warping for datamining applications," in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, New York, NY, 2000.
- [27] M. Skubic, S. Blisard, A. Carle and P. Matsakis, "Extracting Navigation States from a Hand-Drawn Map," in *Proceedings of the 2001 IEEE International Conference on Robotics and Automation*, Seoul, Korea, 2001.
- [28] G. Chronis and M. Skubic, "Sketch-Based Navigation for Mobile Robots," in *Proc. of the IEEE 2003 International Conference on Fuzzy Systems*, Louis, MO, 2003.
- [29] M. Skubic, S. Blisard, A. Carle and P. Matsakis, "Hand-Drawn Maps for Robot Navigation," AAAI 2002 Spring Symposium, Sketch Understanding Workshop, Stanford University, 2002.
- [30] M. Skubic, S. Blisard, C. Bailey, J. Adams and P. Matsakis, "Qualitative Analysis of Sketched Route Maps: Translating a Sketch into Linguistic Descriptions," in *IEEE Transactions on SMC*, 2004.
- [31] G. Chronis and M. Skubic, "Robot Navigation Using Qualitative Landmark States from Sketched Route Maps," in *in Proc. of the IEEE 2004 Intl. Conf. on Robotics and Automation*, New Orleans, LA, 2004.
- [32] M. Skubic, D. Anderson, S. Blisard, D. Perzanowski and A. Schultz, "Using a Qualitative Sketch to Control a Team of Robots," in *in Proceedings of the IEEE 2006 Intl. Conf. on Robotics and Automation*, Orlando, FL, May, 2006.
- [33] M. Skubic, D. Anderson, S. Blisard, D. Perzanowski and A. Schultz, "Using a Hand-Drawn Sketch to Control a Team of Robots," in *Autonomous Robots*, May, 2007.

- [34] G. Parekh, M. Skubic, O. Sjahputera and J. Keller, "Scene Matching Between a Map and a Hand Drawn Sketch Using Spatial Relations," in *Proc., IEEE Intl. Conf. on Robotics and Automation*, Rome, Italy, April, 2007.
- [35] G.-H. Kuo, C.-Y. Cheng and C.-J. Wu, "Design and implementation of a remote monitoring cleaning robot," in *Automatic Control Conference (CACCS), 2014 CACS International*, Kaohsiung, 2014.
- [36] B. Korte and J. Vygen, *combinatorial optimization: theory and algorithm*, Springer Berlin Heidelberg, 2012.
- [37] G. E. Fainekos, H. Kress-Gazit and G. J. Pappas, "Temporal Logic Motion Planning for Mobile Robots," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference*, 2005.
- [38] M. Skubic, D. Anderson, S. Blisard, D. Perzanowski, W. Adams, J. G. Trafton and A. C. Schultz, "Using a Sketch Pad Interface for Interacting with a Robot Team," in *in Proceedings of the 20th national conference on Artificial intelligence, AAAI*, 2005.
- [39] M. W. Kadous, R. K.-M. Sheh and C. Sammut, "Effective User Interface Design for Rescue Robotics," in *In Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, 2006.

APPENDIX A

PROCEDURE OF ALG. 1

Step0: initialize the tables



Algorithm1 FIND_BMP

Input: p^0, TS

Output: p_{bmp}

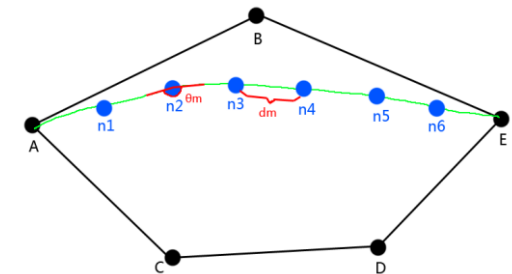
```

1.  $M \leftarrow |Q_{TS}|$ 
2.  $N \leftarrow |p^0|$ 
3.  $cwpd[:, :] \leftarrow \infty$  //for  $N \times M$  matrix
4.  $bmp[:, :] \leftarrow \emptyset$  //for  $N \times M$  matrix
5.  $\langle start, end \rangle \leftarrow \langle index(p^0[1]), index(p^0[-1]) \rangle$  //index of
   nodes in  $Q_{TS}$ 
6.  $\langle cwpd[1][start], bmp[1][start] \rangle \leftarrow \langle 0, \{p^0[1]\} \rangle$ 
7. For  $i$  in range (2, N) do:
8.   For  $j$  in range (1, M) do:
9.     UPDATE( $cwpd, bmp, i, j, p^0, TS$ )
10.  $p_{bmp} \leftarrow bmp[N][end]$ 
11. Return  $p_{bmp}$ 
   where  $Q_{TS} = \{q_0, q_1 \dots q_{M-1}\}$  is the set of nodes in the TS
  
```

BMP	A	B	C	D	E
A	\emptyset	\emptyset	\emptyset	\emptyset	\emptyset
n1	\emptyset	\emptyset	\emptyset	\emptyset	\emptyset
n2	\emptyset	\emptyset	\emptyset	\emptyset	\emptyset
n3	\emptyset	\emptyset	\emptyset	\emptyset	\emptyset
n4	\emptyset	\emptyset	\emptyset	\emptyset	\emptyset
n5	\emptyset	\emptyset	\emptyset	\emptyset	\emptyset
n6	\emptyset	\emptyset	\emptyset	\emptyset	\emptyset
E	\emptyset	\emptyset	\emptyset	\emptyset	\emptyset

CWPD	A	B	C	D	E
A	∞	∞	∞	∞	∞
n1	∞	∞	∞	∞	∞
n2	∞	∞	∞	∞	∞
n3	∞	∞	∞	∞	∞
n4	∞	∞	∞	∞	∞
n5	∞	∞	∞	∞	∞
n6	∞	∞	∞	∞	∞
E	∞	∞	∞	∞	∞

Step0: start point.
p0: [A]



*dab1: the distance from n_1 to edge e_{ab}

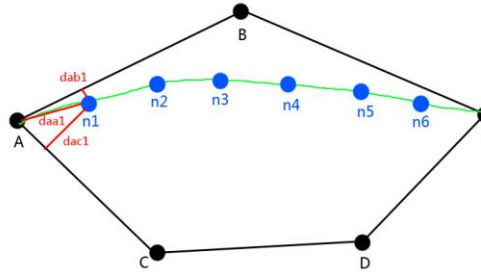
Algorithm1 FIND_BMP
Input: p^0, TS
Output: p_{bmp}
1. $M \leftarrow |Q_{TS}|$
2. $N \leftarrow |p^0|$
3. $cwpd[:, :] \leftarrow \infty$ //for $N \times M$ matrix
4. $bmp[:, :] \leftarrow \emptyset$ //for $N \times M$ matrix
5. $\langle start, end \rangle \leftarrow \langle index(p^0[1]), index(p^0[-1]) \rangle$ //index of nodes in Q_{TS}
6. $\langle cwpd[1][start], bmp[1][start] \rangle \leftarrow \langle 0, \{p^0[1]\} \rangle$
7. For i in range (2, N) do:
8. For j in range (1, M) do:
9. UPDATE($cwpd, bmp, i, j, p^0, TS$)
10. $p_{bmp} \leftarrow bmp[N][end]$
11. Return p_{bmp}
where $Q_{TS} = \{q_0, q_1 \dots q_{M-1}\}$ is the set of nodes in the TS

82

BMP	A	B	C	D	E
A	A				
n1					
n2					
n3...E					

CWPD	A	B	C	D	E
A	0				
n1					
n2					
n3...E					

Step1: find the possible BMPs
and calculate their CWPDs in
row n1
p0: [A,n1]



*dab1: the distance from n_1 to edge e_{ab}

Algorithm1 FIND_BMP

Input: p^0, TS

Output: p_{bmp}

```

1.  $M \leftarrow |Q_{TS}|$ 
2.  $N \leftarrow |p^0|$ 
3.  $cwpd[:, :] \leftarrow \infty$  //for  $N \times M$  matrix
4.  $bmp[:, :] \leftarrow \emptyset$  //for  $N \times M$  matrix
5.  $(start, end) \leftarrow (index(p^0[1]), index(p^0[-1]))$  //index of nodes in  $Q_{TS}$ 
6.  $cwpd[1][start], bmp[1][start] \leftarrow (0, \{p^0[1]\})$ 
7. For  $i$  in range  $(2, N)$  do:
8.   For  $j$  in range  $(1, M)$  do:
9.     UPDATE( $cwpd, bmp, i, j, p^0, TS$ )
10.  $p_{bmp} \leftarrow bmp[N][end]$ 
11. Return  $p_{bmp}$ 
where  $Q_{TS} = \{q_0, q_1 \dots q_{M-1}\}$  is the set of nodes in the TS

```

BMP	A	B	C	D	E
A	A				
n1	AA	AB	AC		
n2					
n3...E					

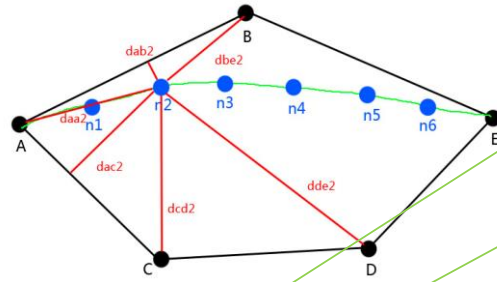
CWPD	A	B	C	D	E
A	0				
n1	daa1	dab1	dac1		
n2					
n3...E					

*Possible BMPs for cell $BMP[n1][A] = (\text{all } BMP[A] \text{ that can access A within one step}^{**}) + A$

**These BMPs can be ended at A or A's neighbors.

For above row n1, previous BMP is A. By adding a node to A, we can get AA, AB, AC.

Step2: find all possible BMPs
and calculate CWPDs in n2
p0: [A, n1, n2]



*dab2: the distance from n_2 to edge e_{ab}

Algorithm2 UPDATE

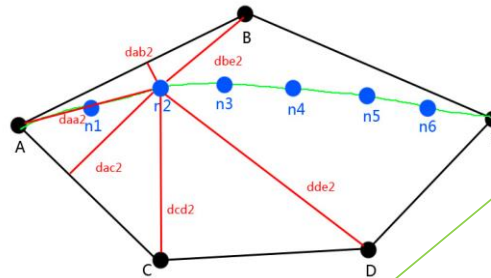
Input: $cwpd, bmp, i, j, p^0, TS$

1. If $bmp[i-1][j] \neq \emptyset$ then:
2. $q_j \leftarrow index^{-1}(j, Q_{TS})$
3. $n_i^0 \leftarrow index^{-1}(i, p^0)$
4. $edge_{prev} \leftarrow GetLastEdge(bmp[i-1][j])$
5. $cwpd_{candi} \leftarrow cwpd[i-1][j] + distance(n_i^0, edge_{self}) // EQ2$
6. If $cwpd_{candi} < cwpd[i][j]$ then:
7. $cwpd[i][j] \leftarrow cwpd_{candi}$
8. $bmp[i][j] \leftarrow bmp[i-1][j] + q_j$ //concatenate q_j to the end
9. For q_k in $neighbors(q_j)$ do:
10. $edge_{curr} \leftarrow (q_j, q_k)$
11. $k \leftarrow index(q_k)$ //index of nodes in Q_{TS}
12. If $edge_{curr} \neq edge_{prev}$ then:
13. $cwpd_{candi} \leftarrow cwpd[i-1][k] + distance(n_i^0, edge_{curr})$
14. If $cwpd_{candi} < cwpd[i][k]$ then:
15. $cwpd[i][k] \leftarrow cwpd_{candi}$
16. $bmp[i][k] \leftarrow bmp[i-1][k] + q_k$

GetLastEdge() returns the last edge of a given path, e.g., GetLastEdge([ABCDE]) returns (DE).

BMP	A	B	C	D	E
n1	AA	AB	AC		
n2	<div> <div> <div>bmp[n1][A]+A -> AAA</div> <div>bmp[n1][B]+A -> ABA</div> <div>bmp[n1][C]+A -> ACA</div> </div> </div>	<div> <div>bmp[n1][B]+B -> ABB</div> <div>bmp[n1][A]+B -> AAB</div> </div>	<div> <div>bmp[n1][C]+C -> ACC</div> <div>bmp[n1][A]+C -> AAC</div> </div>	bmp[n1][C]+D-> ACD	bmp[n1][B]+D -> ABE
n3...E	*Possible BMPs for cell BMP[n2][A]= (all BMP[n1] that can access A within one step**) + A **These BMPs can be ended at A or A's neighbors.				
CWPD	A	B	C	D	E
n1	daa1	dab1	dac1		
n2	<div> <div>cwpd[n1][A]+daa2</div> <div>cwpd[n1][B]+dab2</div> <div>cwpd[n1][C]+dac2</div> </div>	<div> <div>cwpd[n1][B]+dab2</div> <div>cwpd[n1][A]+dab2</div> </div>	<div> <div>cwpd[n1][C]+dac2</div> <div>cwpd[n1][A]+dac2</div> </div>	cwpd[n1][C]+dcd2	cwpd[n1][B]+dbe2
n3...E	*According to EQ2, $cwpd([A, n1, n2], ABA) = cwpd([A, n2], AB) + dab2 = cwpd[n1][B] + dab2$				

Step2.5: find the minimum
cwpd. Its corresponding cell in
BMP table is the correct BMP
for this cell
p0: [A, n1, n2]



*dab2: the distance from n_2 to edge e_{ab}

Algorithm2 UPDATE

Input: cwpd, bmp, i, j, p^0 , T_S
1. If $\text{bmp}[i-1][j] \neq \emptyset$ then:
2. $q_j \leftarrow \text{index}^{-1}(j, Q_{TS})$
3. $n_i^0 \leftarrow \text{index}^{-1}(i, p^0)$
4. $\text{edge}_{\text{prev}} \leftarrow \text{GetLastEdge}(\text{bmp}[i-1][j])$
5. $\text{cwpd}_{\text{candi}} \leftarrow \text{cwpd}[i-1][j] + \text{distance}(n_i^0, \text{edge}_{\text{set}}) // \text{EQ2}$
6. If $\text{cwpd}_{\text{candi}} < \text{cwpd}[i][j]$
7. $\text{cwpd}[i][j] \leftarrow \text{cwpd}_{\text{candi}}$
8. $\text{bmp}[i][j] \leftarrow \text{bmp}[i-1][j] + q_j // \text{concatenate } q_j \text{ to the end}$
9. For q_k in $\text{neighbors}(q_j)$ do:
10. $\text{edge}_{\text{curr}} \leftarrow (q_j, q_k)$
11. $k \leftarrow \text{index}(q_k) // \text{index of nodes in } Q_{TS}$
12. If $\text{edge}_{\text{curr}} \neq \text{edge}_{\text{prev}}$
13. $\text{cwpd}_{\text{candi}} \leftarrow \text{cwpd}[i-1][k] + \text{distance}(n_i^0, \text{edge}_{\text{curr}})$
14. If $\text{cwpd}_{\text{candi}} < \text{cwpd}[i][k]$
15. $\text{cwpd}[i][k] \leftarrow \text{cwpd}_{\text{candi}}$
16. $\text{bmp}[i][k] \leftarrow \text{bmp}[i-1][k] + q_k$

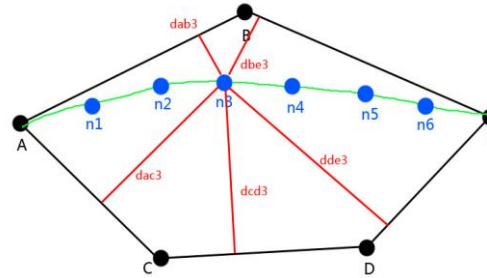
GetLastEdge() returns the last edge of a given path, e.g., GetLastEdge([ABCDE]) returns (DE).

85

BMP	A	B	C	D	E
n1	AA	AB	AC		
n2	bmp[n1][A]+A -> AAA bmp[n1][B]+A -> ABA -> bmp[n2][A] bmp[n1][C]+A -> ACA	bmp[n1][B]+B -> ABB -> bmp[n2][B] bmp[n1][A]+B -> AAB	bmp[n1][C]+C -> ACC -> bmp[n2][C] bmp[n1][A]+C -> AAC	bmp[n1][C]+D -> ACD -> bmp[n2][D]	bmp[n1][B]+D -> ABE -> bmp[n2][E]
n3...E					

CWPD	A	B	C	D	E
n1	daa1	dab1	dac1		
n2	cwpd[n1][A]+daa2 cwpd[n1][B]+dab2 -> cwpd[n2][A] cwpd[n1][C]+dac2	cwpd[n1][B]+dab2 -> cwpd[n2][B] cwpd[n1][A]+dab2	cwpd[n1][C]+dac2 -> cwpd[n2][C] cwpd[n1][A]+dac2	cwpd[n1][C]+dcd2 -> cwpd[n2][D]	cwpd[n1][B]+dbe2 -> cwpd[n2][E]
n3...E					

Step3: find all possible BMPs
and calculate CWPDs in n3
p0: [A, n1, n2, n3]

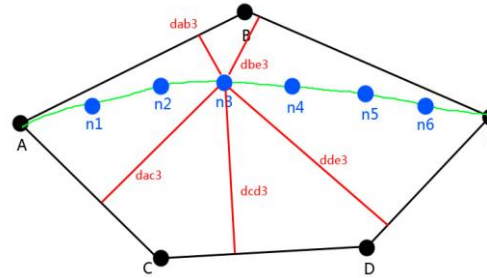


*dab2: the distance from n_2 to edge e_{ab}

98

BMP	A	B	C	D	E
n2	ABA -> bmp[n2][A]	ABB -> bmp[n2][B]	ACC -> bmp[n2][C]	ACD -> bmp[n2][D]	ABE -> bmp[n2][E]
n3	bmp[n2][A]+A -> ABAA bmp[n2][B]+A -> ABBA bmp[n2][C]+A -> ACCA	bmp[n2][B]+B -> ABBB bmp[n2][A]+B -> ABAB bmp[n2][E]+B -> ABEB	bmp[n2][C]+C -> ACCC bmp[n2][A]+C -> ABAC bmp[n2][D]+C -> ACDC	bmp[n2][D]+D -> ACDD bmp[n2][C]+D -> ACCD bmp[n2][E]+D -> ABED	bmp[n2][E]+E -> ABEE bmp[n2][B]+E -> ABBE bmp[n2][D]+E -> ACDE
n4...E					
CWPD	A	B	C	D	E
n2	cwpd[n2][A]	cwpd[n2][B]	cwpd[n2][C]	cwpd[n2][D]	cwpd[n2][E]
n3	cwpd[n2][A]+dab3 cwpd[n2][B]+dab3 cwpd[n2][C]+dac3	cwpd[n2][B]+dab3 cwpd[n2][A]+dab3 cwpd[n2][E]+dbe3	cwpd[n2][C]+dac3 cwpd[n2][A]+dac3 cwpd[n2][D]+dcd3	cwpd[n2][D]+dcd3 cwpd[n2][C]+dcd3 cwpd[n2][E]+dde3	cwpd[n2][E]+dbe3 cwpd[n2][B]+dbe3 cwpd[n2][D]+dde3
n4...E					

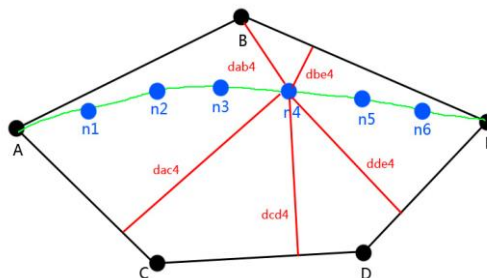
Step3.5: find the minimum
cwpd. Its corresponding cell in
BMP table is the correct BMP
for this cell
p0: [A, n1, n2, n3]



*dab2: the distance from n_2 to edge e_{ab}

BMP	A	B	C	D	E
n2	ABA -> bmp[n2][A]	ABB -> bmp[n2][B]	ACC -> bmp[n2][C]	ACD -> bmp[n2][D]	ABE -> bmp[n2][E]
n3	bmp[n2][A]+A -> ABAA -> bmp[n3][A] bmp[n2][B]+A -> ABBA bmp[n2][C]+A -> ACCA	bmp[n2][B]+B -> ABBB -> bmp[n3][B] bmp[n2][A]+B -> ABAB bmp[n2][E]+B -> ABEB	bmp[n2][C]+C -> ACCC bmp[n2][A]+C -> ABAC -> bmp[n3][C] bmp[n2][D]+C -> ACDC	bmp[n2][D]+D -> ACDD bmp[n2][C]+D -> ACCD bmp[n2][E]+D -> ABED -> bmp[n3][D]	bmp[n2][E]+E -> ABEE -> bmp[n3][E] bmp[n2][B]+E -> ABBE bmp[n2][D]+E -> ACDE
n4...E					
CWPD	A	B	C	D	E
n2	cwpd[n2][A]	cwpd[n2][B]	cwpd[n2][C]	cwpd[n2][D]	cwpd[n2][E]
n3	cwpd[n2][A]+dab3 -> cwpd[n3][A] cwpd[n2][B]+dab3 cwpd[n2][C]+dac3	cwpd[n2][B]+dab3 -> cwpd[n3][B] cwpd[n2][A]+dab3 cwpd[n2][E]+dbe3	cwpd[n2][C]+dac3 cwpd[n2][A]+dac3 -> cwpd[n3][C] cwpd[n2][D]+dcd3	cwpd[n2][D]+dcd3 cwpd[n2][C]+dcd3 cwpd[n2][E]+dde3 -> cwpd[n3][D]	cwpd[n2][E]+dbe3 -> cwpd[n3][E] cwpd[n2][B]+dbe3 cwpd[n2][D]+dde3
n4...E					

Step4: find all possible BMPs
and calculate CWPDs in n_4
 $p_0: [A, n_1, n_2, n_3, n_4]$

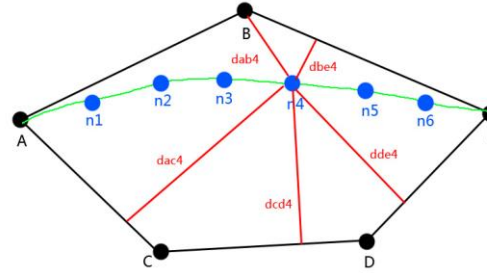


*dab2: the distance from n_2 to edge e_{ab}

BMP	A	B	C	D	E
n3	ABAA -> bmp[n3][A]	ABBB -> bmp[n3][B]	ABAC -> bmp[n3][C]	ABED -> bmp[n3][D]	ABEE -> bmp[n3][E]
n4	bmp[n3][A]+A -> ABAAA bmp[n3][B]+A -> ABBBA bmp[n3][C]+A -> ABCAA	bmp[n3][B]+B -> AB BBB bmp[n3][A]+B -> ABAAB bmp[n3][E]+B -> ABEEB	bmp[n3][C]+C -> ABACC bmp[n3][A]+C -> ABAAC bmp[n3][D]+C -> ABEDC	bmp[n3][D]+D -> ABEDD bmp[n3][C]+D -> ABACD bmp[n3][E]+D -> ABEEED	bmp[n3][E]+E -> ABEEE bmp[n3][B]+E -> ABBBE bmp[n3][D]+E -> ABEDE
n5...E					

CWPD	A	B	C	D	E
n3	cwpd[n3][A]	cwpd[n3][B]	cwpd[n3][C]	cwpd[n3][D]	cwpd[n3][E]
n4	cwpd[n3][A]+dab4 cwpd[n3][B]+dab4 cwpd[n3][C]+dac4	cwpd[n3][B]+dab4 cwpd[n3][A]+dab4 cwpd[n3][E]+dbe4	cwpd[n3][C]+dac4 cwpd[n3][A]+dac4 cwpd[n3][D]+dcd4	cwpd[n3][D]+dde4 cwpd[n3][C]+dcd4 cwpd[n3][E]+dde4	cwpd[n3][E]+dbe4 cwpd[n3][B]+dbe4 cwpd[n3][D]+dde4
n5...E					

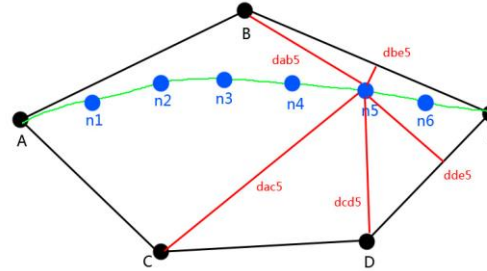
Step4.5: find the minimum
cwpd. Its corresponding cell in
BMP table is the correct BMP
for this cell
p0: [A, n1, n2, n3, n4]



*dab2: the distance from n_2 to edge e_{ab}

BMP	A	B	C	D	E
n3	ABAA -> bmp[n3][A]	ABBB -> bmp[n3][B]	ABAC -> bmp[n3][C]	ABED -> bmp[n3][D]	ABEE -> bmp[n3][E]
n4	bmp[n3][A]+A -> ABAAA -> bmp[n4][A] bmp[n3][B]+A -> ABBBA bmp[n3][C]+A -> ABCAA	bmp[n3][B]+B -> ABBBB bmp[n3][A]+B -> ABAAB bmp[n3][E]+B -> ABEEB -> bmp[n4][B]	bmp[n3][C]+C -> ABACC bmp[n3][A]+C -> ABAAC -> bmp[n4][C] bmp[n3][D]+C -> ABEDC	bmp[n3][D]+D -> ABEDD bmp[n3][C]+D -> ABACD bmp[n3][E]+D -> ABEED -> bmp[n4][D]	bmp[n3][E]+E -> ABEEE bmp[n3][B]+E -> ABBBE -> bmp[n4][E] bmp[n3][D]+E -> ABEDE
n5...E					
CWPD	A	B	C	D	E
n3	cwpd[n3][A]	cwpd[n3][B]	cwpd[n3][C]	cwpd[n3][D]	cwpd[n3][E]
n4	cwpd[n3][A]+dab4 -> cwpd[n4][A] cwpd[n3][B]+dab4 cwpd[n3][C]+dac4	cwpd[n3][B]+dab4 cwpd[n3][A]+dab4 cwpd[n3][E]+dbe4 -> cwpd[n4][B]	cwpd[n3][C]+dac4 cwpd[n3][A]+dac4 -> cwpd[n4][C] cwpd[n3][D]+dcd4	cwpd[n3][D]+dde4 cwpd[n3][C]+dcd4 cwpd[n3][E]+dde4 -> cwpd[n4][D]	cwpd[n3][E]+dbe4 cwpd[n3][B]+dbe4 -> cwpd[n4][E] cwpd[n3][D]+dde4
n5...E					

Step5: find all possible
BMPs and calculate
CWPDs in n5
p0: [A, n1, n2, n3, n4, n5]

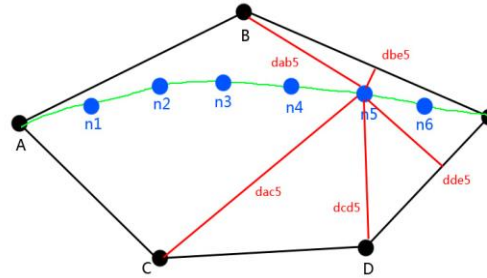


*dab2: the distance from n_2 to edge e_{ab}

BMP	A	B	C	D	E
n4	ABAAA -> bmp[n4][A]	ABEEB -> bmp[n4][B]	ABAAC -> bmp[n4][C]	ABEED -> bmp[n4][D]	ABBBE-> bmp[n4][E]
n5	bmp[n4][A]+A -> ABAAAA bmp[n4][B]+A -> ABEEBA bmp[n4][C]+A -> ABAACA	bmp[n4][B]+B -> ABEEBB bmp[n4][A]+B -> ABAAAB bmp[n4][E]+B -> ABBBEB	bmp[n4][C]+C -> ABAACC bmp[n4][A]+C -> ABAAAC bmp[n4][D]+C -> ABEEDC	bmp[n4][D]+D -> ABEEDD bmp[n4][C]+D -> ABAACD bmp[n4][E]+D -> ABBBED	bmp[n4][E]+E -> ABBBEE bmp[n4][B]+E -> ABEEBE bmp[n4][D]+E -> ABEEDE
n6...E					

CWPD	A	B	C	D	E
n4	cwpd[n4][A]	cwpd[n4][B]	cwpd[n4][C]	cwpd[n4][D]	cwpd[n4][E]
n5	cwpd[n4][A]+dab5 cwpd[n4][B]+dab5 cwpd[n4][C]+dac5	cwpd[n4][B]+dbe5 cwpd[n4][A]+dab5 cwpd[n4][E]+dbe5	cwpd[n4][C]+dac5 cwpd[n4][A]+dac5 cwpd[n4][D]+dcd5	cwpd[n4][D]+dde5 cwpd[n4][C]+dcd5 cwpd[n4][E]+dde5	cwpd[n4][E]+dbe5 cwpd[n4][B]+dbe5 cwpd[n4][D]+dde5
n6...E					

Step5.5: find the minimum
cwpd. Its corresponding cell in
BMP table is the correct BMP
for this cell
p0: [A, n1, n2, n3, n4, n5]

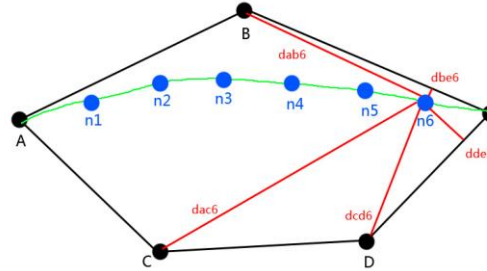


*dab2: the distance from n_2 to edge e_{ab}

BMP	A	B	C	D	E
n4	ABAAA -> bmp[n4][A]	ABEEB -> bmp[n4][B]	ABAAC -> bmp[n4][C]	ABEED -> bmp[n4][D]	ABBBE -> bmp[n4][E]
n5	bmp[n4][A]+A -> ABAAAA bmp[n4][B]+A -> ABEEBA -> bmp[n5][A] bmp[n4][C]+A -> ABAACA	bmp[n4][B]+B -> ABEEBB bmp[n4][A]+B -> ABAAAB bmp[n4][E]+B -> ABBBEB -> bmp[n5][B]	bmp[n4][C]+C -> ABAACC bmp[n4][A]+C -> ABAAAC -> bmp[n5][C] bmp[n4][D]+C -> ABEEDC	bmp[n4][D]+D -> ABEEDD bmp[n4][C]+D -> ABAACD bmp[n4][E]+D -> ABBBED -> bmp[n5][D]	bmp[n4][E]+E -> ABBBEE -> bmp[n5][E] bmp[n4][B]+E -> ABEEBE bmp[n4][D]+E -> ABEEDE
n6...E					

CWPD	A	B	C	D	E
n4	cwpd[n4][A]	cwpd[n4][B]	cwpd[n4][C]	cwpd[n4][D]	cwpd[n4][E]
n5	cwpd[n4][A]+dab5 cwpd[n4][B]+dab5 -> cwpd[n5][A] cwpd[n4][C]+dac5	cwpd[n4][B]+dbe5 cwpd[n4][A]+dab5 cwpd[n4][E]+dbe5 -> cwpd[n5][B]	cwpd[n4][C]+dac5 cwpd[n4][A]+dac5 -> cwpd[n5][C] cwpd[n4][D]+dcd5	cwpd[n4][D]+dde5 cwpd[n4][C]+dcd5 cwpd[n4][E]+dde5 -> cwpd[n5][D]	cwpd[n4][E]+dbe5 -> cwpd[n5][E] cwpd[n4][B]+dbe5 cwpd[n4][D]+dde5
n6...E					

Step6: find all possible BMPs
and calculate CWPDs in n6
p0: [A, n1, n2, n3, n4, n5, n6]

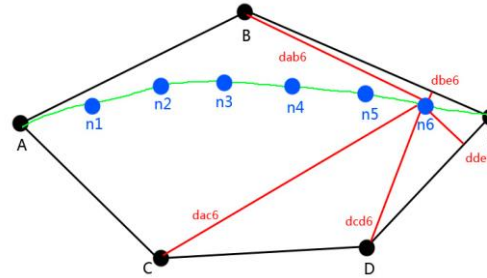


*dab2: the distance from n_2 to edge e_{ab}

BMP	A	B	C	D	E
n5	ABEEBA-> bmp[n5][A]	ABBBEB -> bmp[n5][B]	ABAAAC -> bmp[n5][C]	ABBBED -> bmp[n5][D]	ABBBEE -> bmp[n5][E]
n6	bmp[n5][A]+A -> ABEEBAA bmp[n5][B]+A -> ABBBEBA bmp[n5][C]+A -> ABAAACA	bmp[n5][B]+B -> ABBBEBB bmp[n5][A]+B -> ABEEBAB bmp[n5][E]+B -> ABBBEEB	bmp[n5][C]+C -> ABAAACC bmp[n5][A]+C -> ABEEBAC bmp[n5][D]+C -> ABBBEDC	bmp[n5][D]+D -> ABBBEDD bmp[n5][C]+D -> ABAAACD bmp[n5][E]+D -> ABBBEED	bmp[n5][E]+E -> ABBBEEE bmp[n5][B]+E -> ABBBEBE bmp[n5][D]+E -> ABBBEDE
E					

CWPD	A	B	C	D	E
n5	cwpd[n5][A]	cwpd[n5][B]	cwpd[n5][C]	cwpd[n5][D]	cwpd[n5][E]
n6	cwpd[n5][A]+dab6 cwpd[n5][B]+dab6 cwpd[n5][C]+dac6	cwpd[n5][B]+dbe6 cwpd[n5][A]+dac6 cwpd[n5][E]+dbe6	cwpd[n5][C]+dac6 cwpd[n5][A]+dac6 cwpd[n5][D]+dcd6	cwpd[n5][D]+dde6 cwpd[n5][C]+dcd6 cwpd[n5][E]+dde6	cwpd[n5][E]+dbe6 cwpd[n5][B]+dbe6 cwpd[n5][D]+dde6
E					

Step6.5: find the minimum
cwpd. Its corresponding cell in
BMP table is the correct BMP
for this cell
p0: [A, n1, n2, n3, n4, n5, n6]

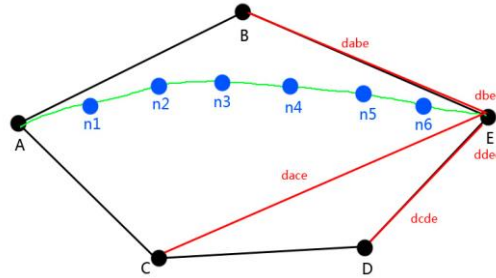


*dab2: the distance from n_2 to edge e_{ab}

BMP	A	B	C	D	E
n5	ABEEBA → bmp[n5][A]	ABBBEB → bmp[n5][B]	ABAAAC → bmp[n5][C]	ABBBED → bmp[n5][D]	ABBBEE → bmp[n5][E]
n6	bmp[n5][A]+A → ABEEBAA bmp[n5][B]+A → ABBBEBA → bmp[n6][A] bmp[n5][C]+A → ABAAACA	bmp[n5][B]+B → ABBBEBB bmp[n5][A]+B → ABEEBAB bmp[n5][E]+B → ABBBEEB → bmp[n6][B]	bmp[n5][C]+C → ABAAACC bmp[n5][A]+C → ABEEBAC bmp[n5][D]+C → ABBBEDC → bmp[n6][C]	bmp[n5][D]+D → ABBBEDD bmp[n5][C]+D → ABAAACD bmp[n5][E]+D → ABBBEED → bmp[n6][D]	bmp[n5][E]+E → ABBBEEE → bmp[n6][E] bmp[n5][B]+E → ABBBEBE bmp[n5][D]+E → ABBBEDE
E					

CWPD	A	B	C	D	E
n5	cwpd[n5][A]	cwpd[n5][B]	cwpd[n5][C]	cwpd[n5][D]	cwpd[n5][E]
n6	cwpd[n5][A]+dab6 cwpd[n5][B]+dab6 → cwpd[n6][A] cwpd[n5][C]+dac6	cwpd[n5][B]+dbe6 cwpd[n5][A]+dab6 cwpd[n5][E]+dbe6 → cwpd[n6][B]	cwpd[n5][C]+dac6 cwpd[n5][A]+dac6 cwpd[n5][D]+dcd6 → cwpd[n6][C]	cwpd[n5][D]+dde6 cwpd[n5][C]+dcd6 cwpd[n5][E]+dde6 → cwpd[n6][D]	cwpd[n5][E]+dbe6 → cwpd[n6][E] cwpd[n5][B]+dbe6 cwpd[n5][D]+dde6
E					

Step7:find all possible BMPs and
calculate CWPDs in E
p0: [A, n1, n2, n3, n4, n5, n6, E]
The result in green cell will
be the final BMP

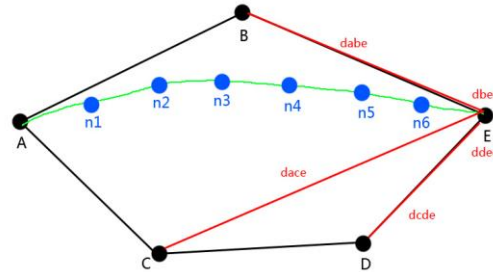


*dabe: the distance from E to edge e_{ab}
*dbee = ddee = 0

BMP	A	B	C	D	E
n6	ABBBEBA-> bmp[n6][A]	ABBBEEB -> bmp[n6][B]	ABBBEDC-> bmp[n6][C]	ABBBEED-> bmp[n6][D]	ABBBEEE -> bmp[n6][E]
E	bmp[n6][A]+A -> ABBBEBA bmp[n6][B]+A -> ABBBEBA bmp[n6][C]+A -> ABBBEDCA	bmp[n6][B]+B -> ABBBEEBB bmp[n6][A]+B -> ABBBEBAB bmp[n6][E]+B -> ABBBEBAB	bmp[n6][C]+C -> ABBBEDCC bmp[n6][A]+C -> ABBBEBA bmp[n6][D]+C -> ABBBEEDC	bmp[n6][D]+D -> ABBBEEDD bmp[n6][C]+D -> ABBBEDCD bmp[n6][E]+D -> ABBBEEDD	bmp[n6][E]+E -> ABBBEEDD bmp[n6][B]+E -> ABBBEEDD bmp[n6][D]+E -> ABBBEEDD

CWPD	A	B	C	D	E
n6	cwpd[n6][A]	cwpd[n6][B]	cwpd[n6][C]	cwpd[n6][D]	cwpd[n6][E]
E	cwpd[n6][A]+dabe cwpd[n6][B]+dabe cwpd[n6][C]+dace	cwpd[n6][B]+dbee cwpd[n6][A]+dabe cwpd[n6][E]+dabe	cwpd[n6][C]+dcde cwpd[n6][A]+dace cwpd[n6][D]+dcde	cwpd[n6][D]+ddee cwpd[n6][C]+dcde cwpd[n6][E]+ddee	cwpd[n6][E]+dbee cwpd[n6][B]+dbee cwpd[n6][D]+ddee

Step7.5: find the minimum cwpd.
 Its corresponding cell in BMP table
 is the correct BMP for this cell
 p0: [A, n1, n2, n3, n4, n5, n6, E]
 The result in green is
 [ABBBEEEE] which is equivalent
 to [ABE]



*dabe: the distance from E to edge e_{ab}

*dbee = ddee = 0

BMP	A	B	C	D	E
n6	ABBBEBA → bmp[n6][A]	ABBBEEB → bmp[n6][B]	ABBBEDC → bmp[n6][C]	ABBBEED → bmp[n6][D]	ABBBEEE → bmp[n6][E]
E	bmp[n6][A]+A → ABBBEBA bmp[n6][B]+A → ABBBEBA → bmp[E][A] bmp[n6][C]+A → ABBBEDCA	bmp[n6][B]+B → ABBBEEBB → bmp[E][B] bmp[n6][A]+B → ABBBEBAB bmp[n6][E]+B → ABBBEBAB	bmp[n6][C]+C → ABBBEDCC bmp[n6][A]+C → ABBBEBAAC bmp[n6][D]+C → ABBBEEDC → bmp[E][C]	bmp[n6][D]+D → ABBBEEDD bmp[n6][C]+D → ABBBEDCD bmp[n6][E]+D → ABBBEEDD → bmp[E][D]	bmp[n6][E]+E → ABBBEEDD → bmp[E][E] bmp[n6][B]+E → ABBBEEBE bmp[n6][D]+E → ABBBEEDD

CWPD	A	B	C	D	E
n6	cwpd[n6][A]	cwpd[n6][B]	cwpd[n6][C]	cwpd[n6][D]	cwpd[n6][E]
E	cwpd[n6][A]+dabe cwpd[n6][B]+dabe → cwpd[E][A] cwpd[n6][C]+dace	cwpd[n6][B]+dbee → cwpd[E][B] cwpd[n6][A]+dabe cwpd[n6][E]+dabe	cwpd[n6][C]+dcde cwpd[n6][A]+dace cwpd[n6][D]+dcde → cwpd[E][C]	cwpd[n6][D]+ddee cwpd[n6][C]+dcde cwpd[n6][E]+ddee → cwpd[E][D]	cwpd[n6][E]+dbee → cwpd[E][E] cwpd[n6][B]+dbee cwpd[n6][D]+ddee

Final BMP Table

BMP	A	B	C	D	E
A	A				
n1	AA	AB	AC		
n2	bmp[n1][A]+A -> AAA bmp[n1][B]+A -> ABA -> bmp[n2][A] bmp[n1][C]+A -> ACA	bmp[n1][B]+B -> ABB -> bmp[n2][B] bmp[n1][A]+B -> AAB	bmp[n1][C]+C -> ACC -> bmp[n2][C] bmp[n1][A]+C -> AAC	bmp[n1][C]+D -> ACD -> bmp[n2][D]	bmp[n1][B]+D -> ABE -> bmp[n2][E]
n3	bmp[n2][A]+A -> ABAA -> bmp[n3][A] bmp[n2][B]+A -> ABBA bmp[n2][C]+A -> ACCA	bmp[n2][B]+B -> AB BB -> bmp[n3][B] bmp[n2][A]+B -> ABAB bmp[n2][E]+B -> ABEB	bmp[n2][C]+C -> ACCC bmp[n2][A]+C -> ABAC -> bmp[n3][C] bmp[n2][D]+C -> ACDC	bmp[n2][D]+D -> ACDD bmp[n2][C]+D -> ACCD bmp[n2][E]+D -> ABED -> bmp[n3][D]	bmp[n2][E]+E -> ABEE -> bmp[n3][E] bmp[n2][B]+E -> ABBE bmp[n2][D]+E -> ACDE
n4	bmp[n3][A]+A -> ABAAA -> bmp[n4][A] bmp[n3][B]+A -> AB BBA bmp[n3][C]+A -> AB CAA	bmp[n3][B]+B -> AB BBB bmp[n3][A]+B -> AB AAB bmp[n3][E]+B -> AB EEB -> bmp[n4][B]	bmp[n3][C]+C -> ABACC bmp[n3][A]+C -> AB AAC -> bmp[n4][C] bmp[n3][D]+C -> ABEDC	bmp[n3][D]+D -> AB EDD bmp[n3][C]+D -> AB ACD bmp[n3][E]+D -> AB EED -> bmp[n4][D]	bmp[n3][E]+E -> AB EEE bmp[n3][B]+E -> AB BBE -> bmp[n4][E] bmp[n3][D]+E -> AB EDE
n5	bmp[n4][A]+A -> ABAAAA bmp[n4][B]+A -> AB EEB A -> bmp[n5][A] bmp[n4][C]+A -> AB AACA	bmp[n4][B]+B -> AB EEBB bmp[n4][A]+B -> AB AAA B bmp[n4][E]+B -> AB BBEB -> bmp[n5][B]	bmp[n4][C]+C -> AB AAAC bmp[n4][A]+C -> AB AAAC -> bmp[n5][C] bmp[n4][D]+C -> AB EEDC	bmp[n4][D]+D -> AB EEDD bmp[n4][C]+D -> AB AAACD bmp[n4][E]+D -> AB BBED -> bmp[n5][D]	bmp[n4][E]+E -> AB BBEE -> bmp[n5][E] bmp[n4][B]+E -> AB EEBE bmp[n4][D]+E -> AB EEDE
n6	bmp[n5][A]+A -> AB EEBAA bmp[n5][B]+A -> AB BBEB A -> bmp[n6][A] bmp[n5][C]+A -> AB AAACA	bmp[n5][B]+B -> AB BBEBB bmp[n5][A]+B -> AB EEBAB bmp[n5][E]+B -> AB BBEEB -> bmp[n6][B]	bmp[n5][C]+C -> AB AAACC bmp[n5][A]+C -> AB EEBAC bmp[n5][D]+C -> AB BBEDC -> bmp[n6][C]	bmp[n5][D]+D -> AB BBEDD bmp[n5][C]+D -> AB AAACD bmp[n5][E]+D -> AB BBEE D -> bmp[n6][D]	bmp[n5][E]+E -> AB BBEEE -> bmp[n6][E] bmp[n5][B]+E -> AB BBEBE bmp[n5][D]+E -> AB BBEEDE
E	bmp[n6][A]+A -> AB BBEBAA bmp[n6][B]+A -> AB BBEEBA -> bmp[E][A] bmp[E][A]	bmp[n6][B]+B -> AB BBEEBB -> bmp[E][B] bmp[n6][A]+B -> AB BBEBAB bmp[n6][E]+B -> AB BBEBAB	bmp[n6][C]+C -> AB BBEDCC bmp[n6][A]+C -> AB BBEBAC bmp[n6][D]+C -> AB BBEE DC -> bmp[E][C]	bmp[n6][D]+D -> AB BBEE DD bmp[n6][C]+D -> AB BBEDCD bmp[n6][E]+D -> AB BBEE ED -> bmp[E][D]	bmp[n6][E]+E -> AB BBEEEE -> bmp[E][E] bmp[n6][B]+E -> AB BBEEBE bmp[n6][D]+E -> AB BBEEDE

Final CWPD Table

CWPD	A	B	C	D	E
A	0				
n1	daa1	dab1	dac1		
n2	cwpd[n1][A]+daa2 cwpd[n1][B]+dab2 -> cwpd[n2][A] cwpd[n1][C]+dac2	cwpd[n1][B]+dab2-> cwpd[n2][B] cwpd[n1][A]+dab2	cwpd[n1][C]+dac2-> cwpd[n2][C] cwpd[n1][A]+dac2	cwpd[n1][C]+dcd2-> cwpd[n2][D]	cwpd[n1][B]+dbe2->cwpd[n2][E]
n3	cwpd[n2][A]+dab3 -> cwpd[n3][A] cwpd[n2][B]+dab3 cwpd[n2][C]+dac3	cwpd[n2][B]+dab3 -> cwpd[n3][B] cwpd[n2][A]+dab3 cwpd[n2][E]+dbe3	cwpd[n2][C]+dac3 cwpd[n2][A]+dac3 -> cwpd[n3][C] cwpd[n2][D]+dcd3	cwpd[n2][D]+dcd3 cwpd[n2][C]+dcd3 cwpd[n2][E]+dde3 -> cwpd[n3][D]	cwpd[n2][E]+dbe3 -> cwpd[n3][E] cwpd[n2][B]+dbe3 cwpd[n2][D]+dde3
n4	cwpd[n3][A]+dab4 -> cwpd[n4][A] cwpd[n3][B]+dab4 cwpd[n3][C]+dac4	cwpd[n3][B]+dab4 cwpd[n3][A]+dab4 cwpd[n3][E]+dbe4 -> cwpd[n4][B]	cwpd[n3][C]+dac4 cwpd[n3][A]+dac4 -> cwpd[n4][C] cwpd[n3][D]+dcd4	cwpd[n3][D]+dde4 cwpd[n3][C]+dcd4 cwpd[n3][E]+dde4 -> cwpd[n4][D]	cwpd[n3][E]+dbe4 cwpd[n3][B]+dbe4-> cwpd[n4][E] cwpd[n3][D]+dde4
n5	cwpd[n4][A]+dab5 cwpd[n4][B]+dab5-> cwpd[n5][A] cwpd[n4][C]+dac5	cwpd[n4][B]+dbe5 cwpd[n4][A]+dab5 cwpd[n4][E]+dbe5 -> cwpd[n5][B]	cwpd[n4][C]+dac5 cwpd[n4][A]+dac5 -> cwpd[n5][C] cwpd[n4][D]+dcd5	cwpd[n4][D]+dde5 cwpd[n4][C]+dcd5 cwpd[n4][E]+dde5 -> cwpd[n5][D]	cwpd[n4][E]+dbe5 -> cwpd[n5][E] cwpd[n4][B]+dbe5 cwpd[n4][D]+dde5
n6	cwpd[n5][A]+dab6 cwpd[n5][B]+dab6-> cwpd[n6][A] cwpd[n5][C]+dac6	cwpd[n5][B]+dbe6 cwpd[n5][A]+dab6 cwpd[n5][E]+dbe6 -> cwpd[n6][B]	cwpd[n5][C]+dac6 cwpd[n5][A]+dac6 cwpd[n5][D]+dcd6 -> cwpd[n6][C]	cwpd[n5][D]+dde6 cwpd[n5][C]+dcd6 cwpd[n5][E]+dde6 -> cwpd[n6][D]	cwpd[n5][E]+dbe6 -> cwpd[n6][E] cwpd[n5][B]+dbe6 cwpd[n5][D]+dde6
E	cwpd[n6][A]+dabe cwpd[n6][B]+dabe-> cwpd[E][A] cwpd[n6][C]+dace	cwpd[n6][B]+dbe6 -> cwpd[E][B] cwpd[n6][A]+dabe cwpd[n6][E]+dabe	cwpd[n6][C]+dcde cwpd[n6][A]+dace cwpd[n6][D]+dcde -> cwpd[E][C]	cwpd[n6][D]+dde6 cwpd[n6][C]+dcde cwpd[n6][E]+dde6 -> cwpd[E][D]	cwpd[n6][E]+dbe6 -> cwpd[E][E] cwpd[n6][B]+dbe6 cwpd[n6][D]+dde6