NORTHERN ARIZONA UNIVERSITY

PREDICTIVE COMMUNICATIONS

USER MANUAL

Table of Contents

Preface

We are pleased that you have chosen Predictive communications team for your needs. Due to the increasing complexity of current technology, it is necessary to perform the proper research and implementation of methodologies in order to accommodate this increase in complexity. Current Networking methods lack in efficiency when applied to dynamic fast-moving networks. The proposed algorithm tries to fix the problem of efficiency in regard to Power and End to End delay by providing a predictive routing algorithm that will increase efficiency by 10% for moderate sized networks, and more gains for larger networks. It is our job to provide a platform to test and evaluate the proposed algorithm. By creating a network of five car robots operating in linear fashion, the user can test the efficiency gain of the proposed algorithm vs conventional techniques. The system has been designed to the specifications of the client, some of these specifications include ;

1. Decentralized distributed system design
2. Graphical user interface (GUI)
3. Distributed localization method
4. Path prediction Interface

The purpose of this User manual is to inform and instruct the user on how to use, how to maintain, and troubleshooting the predictive communication system for your Business/Scientific use. Attached to this manual in the form of Appendices you will find Schematics, Parts used, and a Journal Paper pertaining to the Predictive Algorithm itself and the research behind it. Our aim is to provide the user with the means to benefit from the product, and to keep benefiting from it with time.

Getting Started

In order to run the code involved with operating the Predictive Communications system you need to be sure to have the required software/hardware/equipment to run the various different types of languages.

Required Software:

- MATLAB: Used to Run GUI & Predictive Routing Algorithm
- PyCharm: Used to Run Lidar & Platform
- Java: Used to Run
- MySQL: Used to Run remote database & connection with GUI.

Required Hardware:

- A Computer that's able to run the above-mentioned software
- Wireless Router to connect everything
- Access to the Internet for the database

Required Equipment:

- Lithium Battery Charger
- 3 Tall reference points (can be any uniform object)
- 5 x 5 Test area

Note: These are the necessary items needed in order to run the system. Some items can be replaced for example PyCharm can be replaced with any other Python software.

Set up procedure:

1. The first step would be to find yourself a 5 x 5-meter area, if you do not have the required amount of space, please check the troubleshooting section for a solution

2. Set up three reference points in order to provide reference for the localization system

3. Place camera at center of area looking down at a high elevation

4. Supply power to Camera Module

5. Connect Camera module to the available router, check troubleshooting section to see how to connect Camera module to a router.

6. Place Carbots in desired initial positions

7. Supply power to Carbots by switching on the battery pack located in the middle level of the Carbots.

8. Wait and Confirm that the Carbots are connected to the available router, check troubleshooting section to see how to connect Carbots to a router.

9. Open MATLAB and Select the GUI Program called “Predictive Comms GUI.m”

10. Click the buttons “Set CAR1” to “Set CAR5” to generate and set the initial condition to cars or you can set it manually and click “RUN” until the lights on graphical user interface turn green.

Maintenance

In this section you will find replaceable parts in case of damages, there are no parts that need constant management and maintenance.

Software and Programming Environment:

Python: python 3.5.3 and PyCharm 2017.3.3 professional edition
Java: java 8 and IntelliJ IDEA 2017.3.5 professional edition
Database: MariaDB 10.2.14
MATLAB: MATLAB R2017a

Replacement parts:

Lidar Mechanism:
Slipring-GeeBat Mini Capsule Electrical Slip Ring 12.5mm 250RPM 250VDC/VAC
Bearing- 608 Bearing
3D printed parts-Dropbox.com EE 2017 Team folder, 3D printed parts

Platform:
Raspberry Pi-Raspberry Pi 3 Model B
Lidar-Benewake TFMINI Micro LIDAR Module (12 m)
Chassis-Adafruit (PID 3244) Mini 3-Layer Round Robot Chassis Kit - 2WD with DC Motors
Battery Pack-Raspberry Pi Lithium Battery Power Pack Expansion Board-Plus Power Module

Troubleshooting FAQ

Problem:
I don't have a 5 x 5-meter space available!

Solution:
Open the Lidar Localization algorithm file called “uartReader.py”. Within the source code locate line 52, Locate the condition function d<?. This is a kind of filter to get the proper distance. This controls the limits of the lidar which means it controls how big the area is. It is 500 by 500 as default, you may adjust to desired size, keep in mind we do not recommend going below a 2 x 2-meter area

Problem:
I can’t connect the Camera Module to the router!

Solution:
Double check if the server can connect the camera robot by using command “ping”. If so, reboot the camera robot. Restart the camera program by using python3 to run script “camera.py” under the project file.

Problem:
I can’t connect the Carbots to the router!

Solution:
Double check if the server can connect the car robot by using command “ping”. If so, reboot the car robot. Restart the camera program by using python3 to run script “main.py” under the project file.

Problem:
Lidar gives me a messy result!

Solution:
Check the filter configuration which can be found in “uartRearder.py” line 52 to see if this filter condition fit your test area. Rerun the script “main.py”.

Problem:
I don’t know why but the robot just doesn’t follow the program!

Solution:
Check if you forgot to update the program after you edit the code. You can upload it through the IDE like PyCharm and Eclipse. Or you can upload the code by yourself. The Raspberry Pi offer a file transfer protocol called sftp. You may need to zip your code and upload it through the sftp and unzip it in Raspberry Pi.

Problem:
The robot gave me an error when I try to run the script!

Solution:
This program is based on python version 3.5 and it should be used in the environment of python 3. You should run it using command “python3” instead of “python”.

Problem:
The platforms on the GUI cannot move!

Solutions:
The GUI is connected to the database to update positions, please check has the database has been installed and opened.

Status of Planned Features

Network: Qiyuan Huang

Team Member: Qiyuan Huang				
ID	**Activity**	**Description**	**Deliverable**	**Other People**
1.1	System Design	We have three subsystems in networking system	Draw a diagram of the design	Prof. Razi
1.1.1	Communication Protocol	The communication is based on TCP-IP protocol	Communication protocol	
1.1.2	Control Unit	Build up a computer base control unit (Contain wireless communication module, path planning module and GUI)	Full function	
1.1.3	Computer client	Write computer control client, send instruction to robot cars	Computer interface for user	
1.2	Network Construct	Construct our final network system	Network works in order	
1.3	Test & Debug	See if the network between computer, robot cars and localization module is working properly	Acceptable error	Predictive communication team

Work Breakdown Structure summary:

In this semester, the goal of network system is design and build up a communication system between each robot cars and our computer-based control unit. Basically, the works of network subsystem were divided into two parts, the hardware(platform), and the software (network/GUI). In addition, our localization module is also contained in the network subsystem. The localization system sends positions and GUI system gets positions back through it, and network system works as communication center. My work is design and build up a communication system between each robot cars and computer-based control unit. Since we have 5 ground robot cars running in a 5 x 5-meter area during the execution of the test, the data transformation should over wireless network. Thus, the most feasible idea is that wireless communication module transforms information with each single robotic car. Then the control unit also communicates with the localization module based on WIFI module.

In design part, the network system consists of three main modules including computer-based control unit, localization unit, and 5 ground robots. Our wireless communication module is based on Raspberry pi 3 Model B (include the control module). In addition, we have two different localization methods, the QR code localization system and Lidar localization system.

To select a suitable communication protocol for the network system, I considered implementation convenience and capability of exchanging high data rates as two main requirements of the communication protocol. Under these requirements, a communication protocol that suitable for our project is TCP/IP over WIFI (IEEE 802.11 family) connectivity, which includes built-in per-link routing, framing and integrity check. However, the challenge is how to simulate the waiting times at intermediate nodes. To solve this problem, I program the nodes to hold data packets for a pre-programmed time before forwarding to the next node. In addition, we also used secure file transfer protocol in our project.

Basically, the computer base control unit is developed in MATLAB environment with a graphic user interface, which includes: path planning module, control module, a bi-way communication module, topology prediction module and a predictive routing algorithm. But MATLAB is not use for communication system, the system is construct in database,

To facilitate information exchange among the robots and CBCU is the principle of our network construct. We assign ID=1,2~5 to the 5 robots, ID=0 to CBCU, and ID=6 to localization module. Start and Stop Flags with constant patterns are also set to mark the beginning and end of the packet. Since the TCP/IP protocol includes built-in per-link routing, framing and integrity check. Then it can be applicated to let source and destination fields define the two ends of a per-link communications, and it is changed per link based on the optimal path that determined by the predictive routing algorithm. Finally, the test and debug work is continued follow-up with the system construct.

Platform: Fahad Al Maraghi

Team Member: Fahad Al Maraghi

ID	Activity	Description	Deliverable	Other People
1.1	Prototype Platform	Put together the final prototype platform with all subsystems included	Functional prototype platform	
1.1.1	PCB Prototype	Currently we are using breadboards We need a unified PCB design for all platforms	PCB Circuit schematic	
1.1.2	Lidar Mount	Develop a mount for our Lidar system	The mount design	
1.1.3	Power the Platform	Find best way to power the platforms via batteries	Lithium ion battery capable of powering platform	Predictive communication team
1.2	Design Lidar system/ Scanner Code	Design how our lidar will scan, code to run it, and system	Method of scanning via servo motor for example, solution, and code	Prof. Razi
1.3	Movement System Code	Develop code to run motors	Source code of actual movement system	
1.4	Aesthetics	Design aesthetically pleasing design for platform	Final product Design	Predictive communication team
1.5	Final Platform	Build final version of platform	Final platform	
1.6	Testing of each system	Make sure platform/Lidar subsystem is fully functional	Product approval	Predictive communication team
1.6.1	Simulation	Perform full system simulations and testing	A complete product	Predictive communication team

Work Breakdown Structure summary:

The main part of the work that was assigned to me was the platform creation with all of its mechanisms. Although i also worked on all the other parts of the project (which was most of my work), I'm not being graded on that so let me talk about what I am going to be graded on.
The prototype platform was completed last semester in addition to the code to run the motors aka movement system code. In addition to this i was tasked to create the Lidar mechanism that would operate the Lidar and develop Lidar Localization code. The Lidar Subsystem mechanism needed to be able to perform a 360-degree spin while reading.

To achieve this, i managed to design a mechanism that the Lidar would fit in that would allow it to spin without pulling any wires using 3D printing Software. The mechanism needed some mechanical parts to make it function smoothly namely a 608 bearing (used for skateboards) a slip ring and a continuous rotation servo with proper screws and nuts to make it stable and fixed onto the body of the mechanism. During the prototype phase i designed a temporary PCB that was later developed into the final version PCB that was used in the final Version of the product (Check Appendix A).

This PCB includes a L293D motor driver, 3 LEDs, 3 1K Resistors and inputs and outputs for the Lidar Mechanism. In addition to this it comes without saying that most of the soldering and wirework done was sent my way since I was already doing that. Onto the parts that were not completed; The aesthetics of the platform and the Simulation of the entire system were not completed due to differing reasons. Our final parts order that included the bulk of the items necessary to complete the project namely Lidars, battery pack, and raspberry Pis did not arrive on time, and the battery packs/Lidars have still not arrived as of 3rd of May. So far, we have two fully functional platforms, the other three are still awaiting their power source and Lidar.

I believe that the parts did not arrive on time due to an ordering mix up. This is the primary reason why the other 3 platforms could not be completed, and final simulations can begin with 5 Carbots. As for the Aesthetics, the final version of the Platform appears to be aesthetically pleasing, therefore a new aesthetic for it was unnecessary.

The idea was to cover up the wiring with something aesthetically pleasing, but since we designed our PCB to fit onto the raspberry Pi with headers there were no wires therefore we did not need to redesign the aesthetic. The Primary reason the Client wished a different aesthetic was to cover up the messy wires. One of the main problems i encountered when completing the final version of the Carbots was an issue of Current. When testing the individual subsystems with the Lithium Battery pack I did not encounter any problems with power.

After completing the final platform and joining these subsystems together, the platforms seems to move at a very slow rate and the Lidar was scanning very slowly. This was due to there being a high amount of Load, to solve this problem i split the subsystems powering method. To solve this i connected the Battery pack to power the motors at a higher rate of current and voltage instead of the limitations of the Raspberry Pi output pins which was 5V maximum. The motors that were used on the platform can handle voltages higher than 5V, the lidar mechanism couldn't so this was the only quick fix.

GUI: Yuting Zhang

Person Primarily Responsible: Yuting Zhang				
ID	Activity/Task	Description	Deliverable	Other People
1	**GUI Design**			
1.1.1	Function Design	Design the functions will be realized in GUI.	• Function list • Related Other subsystems	Professor Razi
1.1.2	Draft graph design	Design how to display platforms in GUI.	• Manuscript	Chaoju Wang
1.1.3	Aesthetics	With consideration of aesthetics.	• basic structure	
1.2	**GUI coding**	realize the design		
1.2.1	Basic Graphic display	Realize the basic graph display.	• Basic graph	
1.2.2	Update positions	Get positions from localization systems through network systems.	• Interaction interface • real-time position update.	Fahad Almaraghi
1.2.3	Update path	Apply the shortest path algorithm	• real-time route update.	Arnau Rovira
1.2.4	Realize interaction with users	Design buttons callback events	• Interaction interface	
1.3	**Testing and Analyze**	Test GUI to see can users interact successfully with platform.		
1.3.1	Test GUI Commands	Send commands to see can GUI response correctly to each command.	• gather test data for mutable experiments • calculated successful rate.	All group members
1.3.2	Test GUI interaction with other subsystems	Test the interact ports between GUI and localization systems/networking system	• Good communication with other subsystems. • interact ports.	All group members
1.3.3	Correct GUI & minimize error	Make adjustment of GUI based on the data and analysis.	• corrected GUI	

Work Breakdown Structure summary:

The “GUI Design” part and the “GUI Coding” part are 100% finished, and the “Testing and Analyze” part is 90% finished. The testing parts need to get positions from localization subsystems. We have not find suitable 5x5 blank area with suitable elevation for camera system. The roofs of lab room and other rooms we can find are too high or too close for camera localization system, so the test for all subsystems mixed as a whole project has no suitable chance to realize. But we did test each subsystem separately, and all parts works well.

The full name of GUI is graphical user interface, the purpose of graphical user interface is to realize visualization and operability of the whole system. The whole system is complex, so we need GUI as a information collector and performance displayer. The first step is to design the GUI. Firstly, I decide what functions this GUI are supposed to realize. I use MATLAB language to develop the graphic user interface. There are three functions of the graphic user interface: connect database, set platforms’ speed and delay, display positions and path in a cartesian coordinate system. The most important function is to connect GUI with the database to use database through MATLAB system. The purposes are as follows: (1) get position information; (2) insert path and command; (3) update delay and speed information. Secondly, I design and draw the outlook of this graphic user interface on paper by hand. In this step, I did a lot changes by drawing several versions to achieve a balance between aesthetics and functions.

The second step is coding this GUI in MATLAB. At first, I realized the outlook of the graphical user interface. The main part of this GUI is the cartesian coordinate system, below that are five set buttons as “SET CAT1”, “SET CAT2”, “SET CAT3”, “SET CAT4”, “SET CAT5”. Once the user clicks these buttons, random initial speed and delay will be generated by MATLAB for each platform respectively. Under these buttons are five status indicators and one show state button. Once the user clicks the state button, GUI will check whether the parameters (speed, delay) have been set successfully to platforms, and the GUI will set the lamps on if corresponded platform has been set successfully. After all the preparation, we can click move to see the five platform’s real time updated positions and predictive path. To realize the click, function of the buttons, I write callback functions respectively for each button. The range of random delay is from 1 to 5, and the range of random speed is from 50 to 100. These generated delays and speeds will be stored in database, and commands will also be stored in database. In my callback function for button “MOVE”, the shortest path algorithm is used. Once the user clicks the button, MATLAB will automatically get positions from database and call the shortest algorithm to calculate the path. Once the graphical user interface gets the path, it will display both the positions and the path in the cartesian coordinate system.

Here I introduce why we need database, and what data structures we use. The system has five vehicles, each of them has their own positions, delay and noise. The two localization systems (QR code system and Lidar system) can update the real-time positions of the five vehicles. We also use the shortest path algorithm to calculate the package routes using positions. There are a lot of information (positions and paths) needed to be stored in the cloud database. Here we use database as our database. In the remote base database, we have three tables. The first table is called “COMMAND”, which includes the stop command and move command with number identification. Thus, once the GUI stores ‘1’ in the database, the control side will know the user wants the platforms run. The second table is called “PATH”, the GUI will store every hop of the path in database, the description of each hop includes source (the platform sending packages) and destination (the platform receiving packages). The third table is called “USER”, all the detailed information is stored for each car in each row of this table. Thus, the table “USER” has five rows for the five platforms. The database is important for this project, and it is not on the plan list because it appears in our mind in the process of doing the project. There are new needs emerge when we solve the problems, that is a normal situation.

In conclusion, the GUI is easy to use and show things clear for users. It connects the localization subsystem and the shortest path algorithm. The outlook and function descriptions are instructed in the appendix.

Localization System, Communication System, & Software: Chaoju Wang

Team Member: Chaoju Wang			
ID	**Activity**	**Description**	**Deliverable**
1.1	Camera System	First localization system for this Project. This is a centralized system which use a camera to get the locations of all cars by taking a photo and recognizing the QRcode of each car.	Full functional camera system.
1.2	Movement System	The code for operating the step motor of the robot.	Full functional movement system.
1.3	Software Kernel	The central control unit in each robot which can unit all subsystems and make them working synchronously.	Full functional software kernel.
2.1	Lidar System	The hardware and the software of the lidar system	Full functional Lidar system
2.1.1	Lidar Scan	Python code of the lidar reading and locating	Code
2.1.2	Lidar Servo	Python code and the available mechanical structure	Code and eligible servo
2.1.3	Test	Test the whole lidar system.	Final Lidar System
2.2	Database System	New communicating method of the project	Full functional database subsystem
2.2.1	Design The Relationship	Design the relationship between the entities of our project in this database	Entity Relationship Diagram (ERD)
2.2.2	Set up the database	Set the database based on ERD	Actual Database in local or internet
2.2.3	Program the interface of user	Develop the interface which allow the user to access the database.	Code

Work Breakdown Structure summary:

In this project we need to design the software control system for all the hardware parts so the end user does not need to modify the hardware part and all the user need to do is click the button in the software. Since we chose Raspberry Pi and Python as our tools, all of our subsystem can be modularized.

Since we cannot use the GPS because its low accuracy in 5*5-meter test area, we chose camera and QR code as our first indoor localization system. Each car has a printed QR code on its top so the camera robot which is above all cars can take a photo which include all 5 cars with their QR code. The program can recognize the QR codes and get the location of each QR code and also, the location of car which is represented by the QR code.

QR code localization is a centralized system which means that we must have a center (camera robot) to get the locations of all cars. But, sometimes, the car need to get their location without communication. So, we design the lidar system as the distributed system that every car can get their own position without other robots. The basic idea is that the robot transmits an invisible light and the light will be reflected by objects which in our case are reference points and go back to the receiver on the robot. The robot can get the distance between the robot and the reference point by calculating using the time of flight (TOF). In our test area, we set up 3 high cylinders as the reference points. After the robot get 3 distances by spanning the lidar for a cycle, it can calculate the position using the triangle-positioning algorithm.

Movement system is the easiest one, all we need to do is set up the pin number for the input and output. We can control the spanning orientation of the step motor by controlling the driver chip. We also can control the speed using the PWD function.

Software kernel is the mind of the whole software. It can unite all subsystems and coordinate them. We use the event driving model as the kernel. The basic idea is that every command in all subsystem are considered as different event. The kernel will collect event from all subsystems and distribute them to the corresponding destination subsystem, so event can be solved by proper subsystem. For example, the communication system receives a command from user which ask the robot to move forward. The communication system will package this command as an event and notify the kernel. After the kernel get the event form communication system, kernel will store it to a queue which is design for events so event will be dispatched in order. When the move event come to the head of the queue, the kernel will send this event to movement system by check the registration table for all kinds of events.

In the beginning of the project, we use WIFI and package based on TCP/IP as our communication system, but we found that MATLAB is hard to be programed in multi-threads. In other words, MATLAB is hard to send and receive the command while calculating the path and showing the graph, so we changed the communication system into database, so MATLAB only need to send the initial constants in the beginning.

PCB: Hanxiao Lu

Team Member: Hanxiao Lu				
ID	**Activity**	**Description**	**Deliverable**	**Other People**
1.1	**PCB Design**			
1.1.1	Design schematic	Complete the connection of the lines to make the platform look better Include: Raspberry pi (40 pins header, 3V and 5V input, lidar, motor chip L293D, resistance and LED)	The schematic is 100% right	
1.1.2	Drawing component packaging	Ensure that components are properly mounted on the PCB	100% done	
1.1.3	Design lines on PCB	Use double-layer traces	100% done	
1.2	**Making PCB board**			
1.2.1	Order PCB board	Contact manufacturer to make PCB board	Ordered 10 PCB boards, five more for back up	Fahad Almaraghi
1.2.2	Assemble the PCB board	Solder all components to PCB board. Ensure that all devices are securely attached together.	The interface of some devices has been modified, and the PCB is welded completely.	Fahad Almaragh, Yuting Zhang, Chaoju Wang
1.3	**Test PCB board**			
1.3.1	Check whether the circuit is properly linked	Use a circuit meter to measure each circuit, as well as LED and resistors.	100% right	
1.3.2	Test integrity	Run radar, Raspberry pi and motor to see if it can works well.	100% right	All group members

Work Breakdown Structure summary:

Printed circuit board(PCB) is an important part of our project. The purpose of designing PCB is to reduce the volume of our circuit and increase the aesthetics of our robot car. The design software I used is Altium Designer 18. Schematic and PCB layer schematic will show at Appendix A part.

The PCB board mainly contains the following devices：

P1: Power input and output. Connect to battery pack (the power system).
P2, P3: The output of the motor.
P4: The 40 pin female headers are used to cover the link to the Raspberry pi 3, greatly simplifying the volume of the electronic device.
P5, P6: Used to test whether the various interfaces of the Raspberry pi 3 are working properly.
P7: The input and output interface of the lidar's power and signal.
P8: The input and output interface of the servo motor’s power and signal.
P9: 3.3V and 5V output of Raspberry pi 3. Used to test whether the output voltage is stable and effective.
P10: L293D, the control chip of the motor.
D1, D2, D3: LED. Used to detect packages transmission.
R1, R2, R3: Resistance. Used to protect the LED from current voltage overload.

Ending Statement

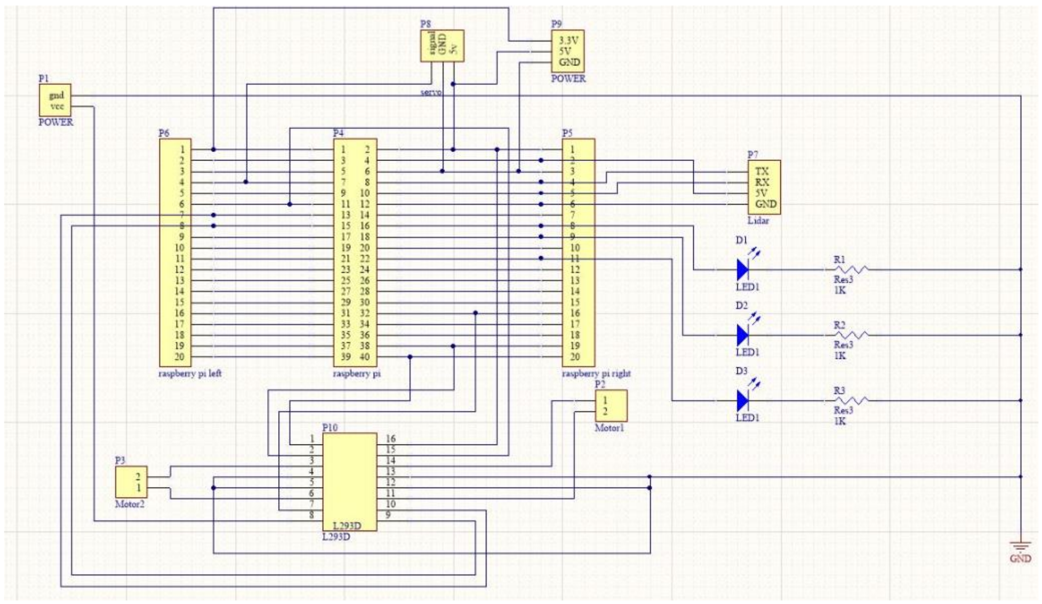
We would like to formally thank the Client, Professor Abolfazl Razi, for giving us this opportunity to work on this exciting topic. We have gained a significant amount of experience during this project that will be utilized inevitably in the near future. Professor Razi's guiding hand steered us in the proper direction during trivial times, which is much appreciated. We hope that the client utilizes and benefits from the system to the highest degree.

For any inquiries or information regarding the system feel free to contact us;

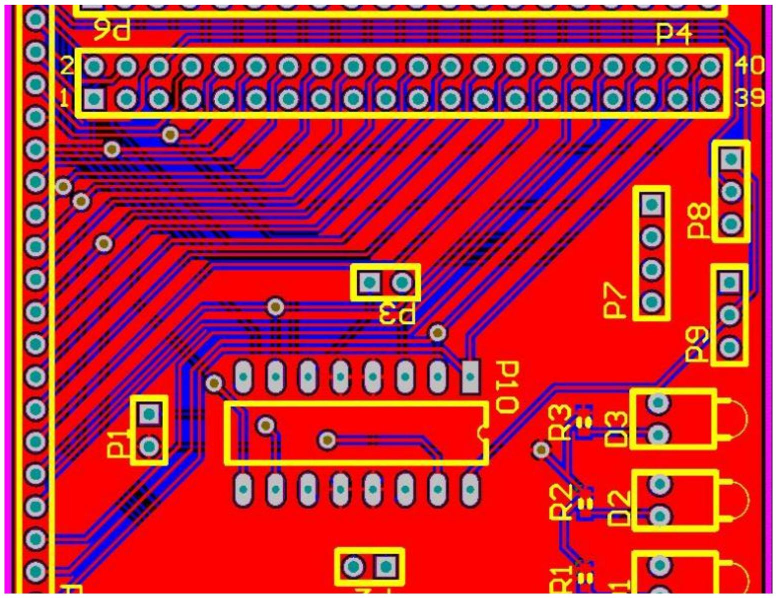
Chaoju Wang	cw892@nau.edu
Fahad Al Maraghi	fahadalmaraghi@gmail.com
Hanxiao Lu	hl443@nau.edu
Qiyuan Huang	qh33@nau.edu
Yuting Zhang	yz292@nau.edu

Appendix A

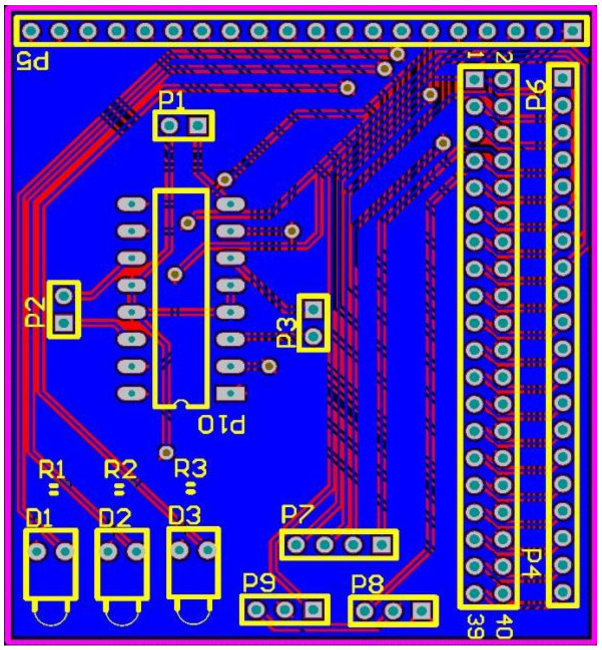
PCB Schematic:



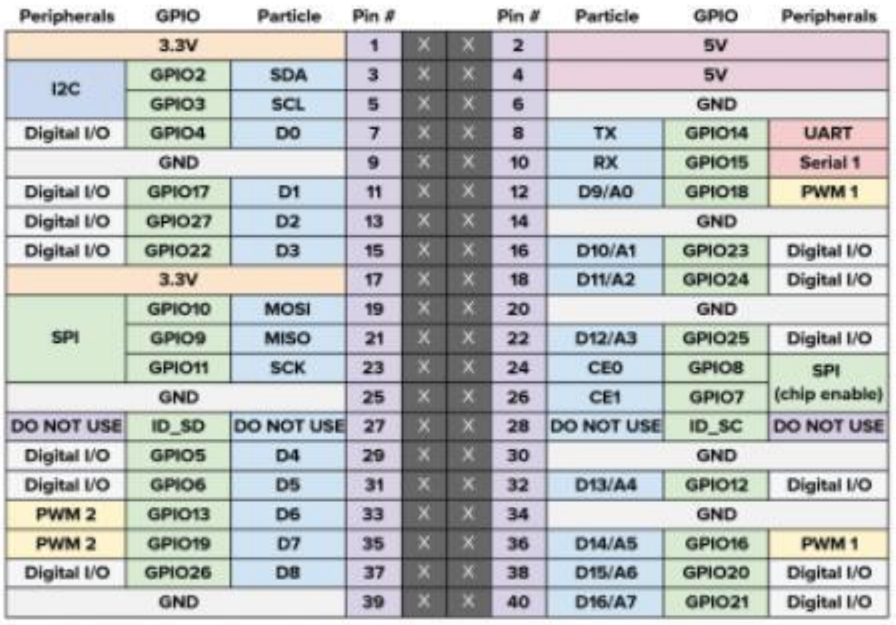
PCB Top Layer:



PCB Bottom Layer:

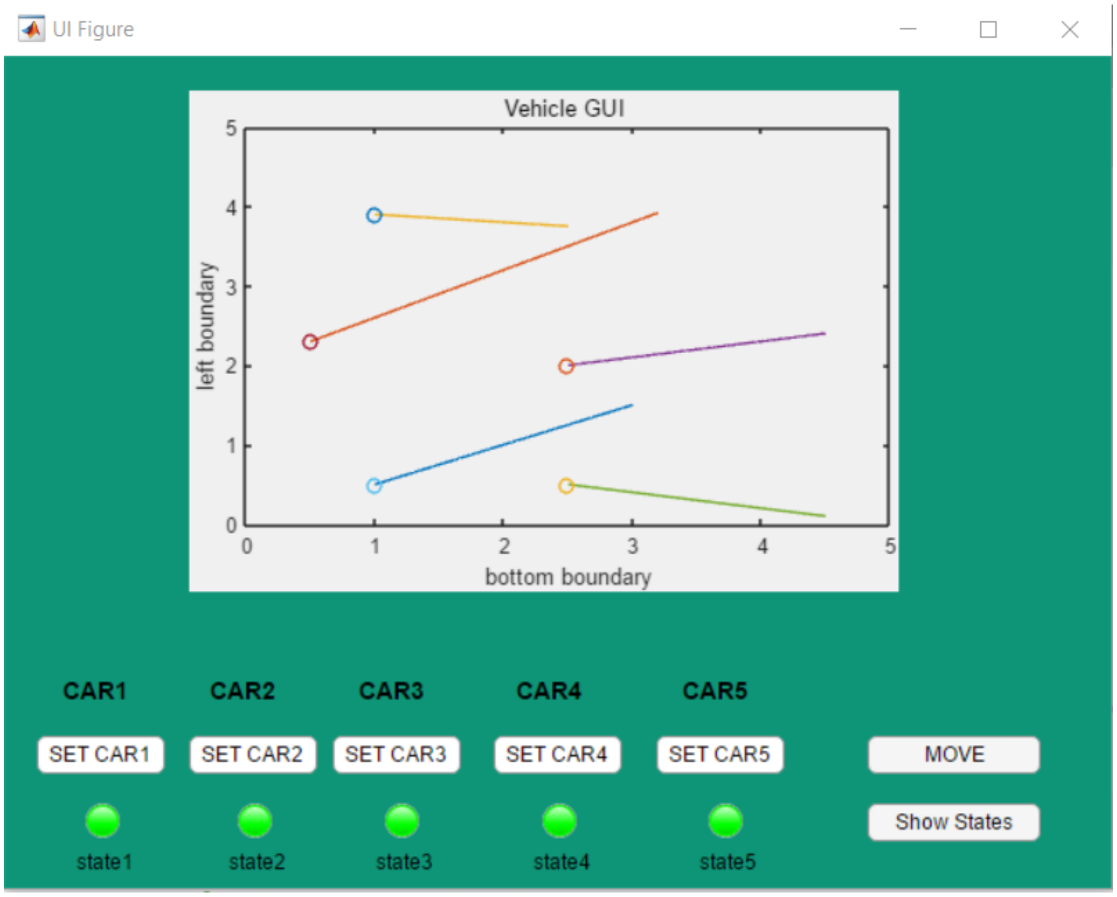


Raspberry pi 3



Peripherals	GPIO	Particle	Pin #			Pin #	Particle	GPIO	Peripherals
	3.3V		1	X	X	2		5V	
I2C	GPIO2	SDA	3	X	X	4		5V	
	GPIO3	SCL	5	X	X	6		GND	
Digital I/O	GPIO4	D0	7	X	X	8	TX	GPIO14	UART
	GND		9	X	X	10	RX	GPIO15	Serial 1
Digital I/O	GPIO17	D1	11	X	X	12	D9/A0	GPIO18	PWM 1
Digital I/O	GPIO27	D2	13	X	X	14		GND	
Digital I/O	GPIO22	D3	15	X	X	16	D10/A1	GPIO23	Digital I/O
	3.3V		17	X	X	18	D11/A2	GPIO24	Digital I/O
SPI	GPIO10	MOSI	19	X	X	20		GND	
	GPIO9	MISO	21	X	X	22	D12/A3	GPIO25	Digital I/O
	GPIO11	SCK	23	X	X	24	CE0	GPIO8	SPI (chip enable)
	GND		25	X	X	26	CE1	GPIO7	
DO NOT USE	ID_SD	DO NOT USE	27	X	X	28	DO NOT USE	ID_SC	DO NOT USE
Digital I/O	GPIO5	D4	29	X	X	30		GND	
Digital I/O	GPIO6	D5	31	X	X	32	D13/A4	GPIO12	Digital I/O
PWM 2	GPIO13	D6	33	X	X	34		GND	
PWM 2	GPIO19	D7	35	X	X	36	D14/A5	GPIO16	PWM 1
Digital I/O	GPIO26	D8	37	X	X	38	D15/A6	GPIO20	Digital I/O
	GND		39	X	X	40	D16/A7	GPIO21	Digital I/O

Graphic User Interface:


UI Figure
Vehicle GUI
left boundary
bottom boundary
0
1
2
3
4
5
CAR1
CAR2
CAR3
CAR4
CAR5
SET CAR1
SET CAR2
SET CAR3
SET CAR4
SET CAR5
MOVE
Show States
state1
state2
state3
state4
state5

Appendix B

Predictive Routing for Wireless Networks: Robotics-Based Test and Evaluation Platform

Chaoju Wang, Fahad Almaraghi, Qiyuan Huang, Yuting Zhang, Hanxiao Lu, Arnau Rovira-Sugranes, Abolfazl Razi
School of Informatics, Computing and Cyber Security (SICCS)
Northern Arizona University, Flagstaff, AZ 86011

***Abstract*—Emerging Internet of Things (IoT) provides connectivity to a wide range of mobile nodes including indoor wireless users, pedestrian, ground robotics, vehicles, and flying objects. Such decentralized network require rethinking user-centric communication protocols which accommodate extremely dynamic environments of autonomous nodes. The authors recently proposed a predictive routing algorithm, which enables a delay-optimal communication through incorporating network topology prediction into the Dijkstra's shortest path algorithm. In this work, we extend the proposed solution to jointly optimize the end-to-end latency and total transmission power. Further, we develop a ground robotics platform in order to study the utility of the proposed algorithm in real-world applications. The simulation results which verified by the test platform, confirm the superiority of the proposed algorithm compared to the conventional shortest path algorithms by improving the delay and power consumption by a factor of 10% to 15%.**

I. Introduction

Internet of Things is an emerging technology to provide connectivity for a wide range of mobile nodes to realize smart cities [1]. The idea is to integrate furnish everyday objects with embedded control units and integrated them into a connected network with ubiquitous access [2], [3]. This large-scale network encircles a wide range of heterogeneous nodes with different levels of autonomy, mobility, storage and data communication requirements, which demands for more efficient Machine to Machine (M2M) communications [4].

In this works, we focus on developing optimized communication protocols appropriate for large-scale networks which include freely-moving nodes with probabilistic but predictable motion trajectories. Adaptive communication is a long-lasting paradigm studies from many different perspectives. The objective of a typical adaptive communication approach is to tune communication parameters or make smarter decisions at different layers of the communication protocol (e.g. packet forwarding in network layer) such that a desired performance metric (e.g. data throughput, transmission delay, age of information, network connectivity, etc.) is optimized [5]–[8].

Majority of the previously reported network optimization frameworks take into account the current network situation in the optimization process. This is a careless ignorance of predictive information that is currently available using advanced machine learning algorithms. For instance, selecting a communication link which maximized the instantaneous performance metric at the current time but is subject to link loss during the transmission session is extremely inefficient [9], [10]. This issue becomes more challenging in flying ad-hoc networks (FANET), where network parameters constantly change due to dynamic network topology [11].

Recently, the idea of incorporating predicted network topology into communication protocols is introduced in order to optimize the end-to-end delay in Unmanned Aerial Vehicles (UAV) networks [12], [13]. However, these works either rely on GPS information, or consider only delay optimization. In this work, we plan to extend this idea to jointly optimize the end-to-end delay and the total transmission power, since both influenced by the network topology. Further, to demonstrate the applicability of the proposed algorithm in real-world applications, we develop a test and evaluation platform based on autonomously controlled ground robotics. We choose ground robots for their motion stability and predictability [14]. However, the idea is general and applicable to other mobile objects such as drones, small satellites and airplanes [15].

The rest of this paper is organized as follows. In section II, the system model and the utilized mobility model are presented. Section III elaborates on the proposed predictive routing algorithm. In section IV, the test and evaluation platform is presented. Finally, simulation results are provided in section V, followed by concluding remarks in section VI.

II. Network Mobility Model

Consider a network of freely moving objects that communicate with high bitrate through a queued communication platform. An example is a network of UAVs that exchange video information for border patrolling. The purpose of a routing algorithm is to find an end-to-end communication path from the source to the destination such that a desired performance metric is optimized. Typical performance metrics including channel rate, transmission power, and transmission delay usually depend on the pairwise distances between the nodes at each communication link. In a queue-less systems

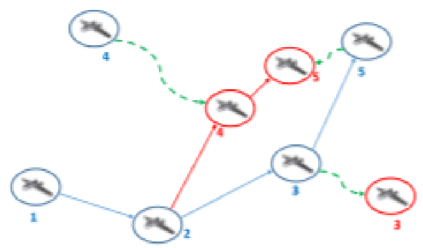


Fig. 1: Utility of predictive routing in finding optimal paths for dynamic networks.

with light traffic regime, the propagation delay is negligible and hence the network topology does not significantly change during a short transmission session. However, in a queued system with heavy traffic regime, network topology can change significantly, while a data packet is waiting in a transmission buffer in the intermediate nodes during his journey from the source to destination. Thereby, the optimal path, if found by the source node based on the initial network topology using a typical shortest path algorithm, may not remain optimal. An illustrative example is shown in Fig. 1, where the blue and red circles, respectively show the original and the updated positions of the nodes (after motions shown by dashed green arrows). A conventional algorithm would determine (1-2-3-5) as the optimal path from source node 1 to destination 5 (represented by blue arrows) based on the original positions (blue circles), whereas the proposed predictive routing algorithm selects the path (1-4-5) (represented by red arrows) taking into account predicted network topology change, while the packet is waiting in the transmission buffer of node 2.

Here, the main idea is to utilize prediction algorithms and incorporate the predicted network topologies into the optimal path selection algorithm. Before elaborating on the details of the proposed algorithm, we present the mobility model used in this work.

The network is composed of N nodes uniformly distributed in a two-dimensional squared grid. Therefore, the initial node positions $\vec{l}_i(0)$=$[x_i(0), y_i(0)] \sim \mathcal{U}(-L, L)$ follow a Uniform distribution within the predefined range $(-L, L)$. Edges between nodes represent bi-way communication links in terms of a contact graph, where existence of each link following an i.i.d. Bernoulli distribution with sparsity level s, i.e. $Pr(e_{ij} = 1) = 1 - Pr(e_{ij} = 0) = s$.

The edge metrics $w_{ij}(t)$ are based on the previously set up contact graph and the distances between the nodes $d_{ij}(t)$, where $w_{ij}(t) = w_{ji}(t)$. Initial velocities $\vec{v}_i(0) = [v_i^{(x)}(0)\ v_i^{(y)}(0)]$ and acceleration vectors $\vec{a}_i = [a_i^{(x)}\ a_i^{(y)}]$ are randomly generated using uniform distributions within a predefined range.

The motion trajectories can be simply obtained using Durbin's curve equations in terms of a discrete state transition model as follows:

$$\begin{cases} \vec{s}_i(k+1) & = A\vec{s}_i(k) + B\vec{a}_i(k) + \vec{w}_i(k), \\ \vec{o}_i(k) & = \gamma_i(k)C\vec{s}_i(k) + \vec{z}_i(k), \end{cases} \tag{1}$$

where we have:

$$\vec{s}_i(k) = \left[x_i(k)\ y_i(k)\ v_i^{(x)}(k)\ v_i^{(y)}(k)\right]^T, \quad \text{(state vector)}$$

$$\vec{a}_i(k) = \left[a_i^{(x)}(k)\ a_i^{(y)}(k)\right]^T, \quad \text{(input vector)}$$

$$\vec{o}_i(k) = \left[o_i^{(x)}(k)\ o_i^{(y)}(k)\right]^T, \quad \text{(observation vector)}$$

$$A = \begin{bmatrix} \vec{I}_{2\times2} & dt\vec{I}_{2\times2} \\ \vec{0}_{2\times2} & \vec{I}_{2\times2} \end{bmatrix}, B = \left[\vec{0}_{2\times2} \quad \vec{I}_{2\times2}\right], C = \left[\vec{I}_{2\times2} \quad \vec{0}_{2\times2}\right]^T$$

$$\vec{w}_i(k) \sim \mathcal{N}(\vec{0}, \vec{R}_i), \quad \vec{z}_i(k) \sim \mathcal{N}(\vec{0}, \vec{Q}_i). \tag{2}$$

Here k represents the discrete time point $k$dt for an arbitrarily chosen time step dt. Also, $\vec{w}_i(k)$ and $\vec{z}_i(k)$ are zero mean Gaussian distributed random vectors of covariances R_i and Q_i that respectively represent the system and observation noise terms. Finally, $\gamma_i(k)$ is a Bernoulli distributed random variable $\Pr(\gamma_i(k) = 1) = \lambda$ to capture object tracking success. The optimal estimation of the node locations when measurement is available ($\gamma_i(k) = 1$) as well as the optimal online prediction of locations when the measurement is absent ($\gamma_i(k) = 0$) for known input vector $\vec{a}_i$ can be obtained using Kalman filtering with intermittent observation [16].

From motion trajectories we can determine actual node locations $\vec{l}_i(t) = [x_i(t)\ y_i(t)]$ and we assume that the location estimates and predictions $\hat{\vec{l}}_i(t) = [\hat{x}_i(t)\ \hat{y}_i(t)]$ for all neighbor nodes are available to each node n_i using a proper tracking system, where we include the prediction error term $\vec{e}_i(t) = \hat{\vec{l}}_i(t) - \vec{l}_i(t)$. Here we take a realistic consideration that the prediction certainty declines over time. In other words, we have $e_i(t) \sim \mathcal{N}(0, \sigma^2(t)\mathbf{I}_{2\times2})$, where the prediction noise variance σ^2 increases over time as

$$\sigma^2(t) = \sigma_0^2 + \alpha t, \tag{3}$$

where α is a predefined discount factor to capture the rise in localization uncertainty. Lastly, to model queuing delays, we assign a random waiting time to each node, $w_i(t) \sim \mathcal{U}(0, W)$. Likewise, each node has an estimate of other node's waiting time as $\hat{w}_i(t) = w_i(t) + e_{wi}(t)$, where $e_{wi}(t) \sim \mathcal{N}(\mu_w, \sigma_w^2)$ is the error measurement for the waiting time, which has variance σ_w^2.

Path from source to destination is determined based on the predicted locations and waiting times, whereas the actual end-to-end objective function calculations are made based on the actual locations and waiting times.

III. Predictive Routing Algorithms

The predictive algorithm is a modification of the well-known conventional Dijskstra's shortest path algorithm [17]. The algorithm is efficient and is the fundamental core of most shortest path algorithms. However, for scenarios where the edge metrics are time-varying, the conventional version is not appropriate, especially for queued communications.

In our network model, we use both conventional and predictive routing , respectively using initial and predicted location information to find the shortest path from the source node to the destination. The optimization goal is find the optimal path $\mathcal{P}_{\text{opt}}$, which maximizes the function in (4), where $d(\mathcal{P})$ and $p(\mathcal{P})$ represent the end-to-end delay and the total transmission power for path $\mathcal{P}$. Then, a desired importance factor γ is used to balance between the two objective functions.

$$f(\mathcal{P}) = \gamma d(\mathcal{P}) + (1 - \gamma)p(\mathcal{P}) \tag{4}$$

The proposed predictive routing algorithm starts from the source node at time 0 and finds the next intermediate node in the path by including the predicted locations and estimating when the packet would reach each of the possible intermediate nodes. On this basis, it select the best intermediate node according to the lowest objective function and updates the topology at the time, the packet is ready for transmission in the intermediate node. This process repeats for the intermediate node, excluding previously visited nodes, until we reach the destination. Consequently, we obtain the path which minimizes the multi-objective function. The following is the edge update procedure.

In a general formulation, we can consider edge weights as $w_{ij}(t) = f(d_{ij}(t)) + g_{ij}(t)$, where $f(d_{ij}(t))$ mimics the distance-related terms as can be the propagation delay and $g_{ij}(t)$ represents other terms, for example the waiting delay. In this paper, the goal is to minimize the transmission delay and power, represented by $f()$ in (4). To optimize the power, noting that transmission power is proportional to distance squared, we use $d_{ij}^2(t)$ as the surrogate of power in selecting the optimal path. To optimize the end-to-end delay, we consider an edge metrics $w_{ij}(t)$ that represents the delays associated with the edge e_{ij} accounting for time delays for a packet when it reaches node i denoted by t_i until the epoch it is delivered to node j, denoted by t_j. This time is composed by the waiting time in transit buffer of node i at time t_i, denoted by $w_i(t_i)$ and the actual propagation time. The propagation time for a packet that leaves node n_i at time $t_i + w_i$ and reaches the mobile node j, denoted by $p_{ij}(t_i+w_i)$ is calculated by solving the following equations:

$$\begin{cases} d_{ij} = |\hat{l}_j(t_i + \hat{w}_i + p_{ij}(t_i + \hat{w}_i)) - l_i(t_i + \hat{w}_i)|_2, \\ p_{ij}(t_i + \hat{w}_i) = d_{ij}/c, \end{cases} \tag{5}$$

where the first equations characterizes the predicted distance between node n_i (when the packet leaves this node at time $t_i + w_i$) and node n_j (when the packet reaches this node at time $p_{ij}(t_i + w_i)$). The second equations relates this distance to the propagation time $p_{ij}(t_i + w_i)$ using the wave propagation phenomenon with the speed of light c. The operator $|\vec{x}|_2 = \sqrt{x_1^2 + x_2^2}$ is the second norm of vector $\vec{x} = [x_1 x_2]$. Note that (5) may require numerical methods to solve the nonlinear equation, since $\hat{l}_j(t)$ is the solution of (1) and in general is not linear. However, noting the fact that propagation time is negligible compared to waiting times, we can use the convenience of $p_{ij} = 0$ in (5), thereby the end-to-end delay is the accumulated waiting times through the path.

Once, we solve (5) with or without the simplifying assumption, the edge metrics are determined as

$$w_{ij}(t_i) = w_i + p_{ij}(t_i + w_i), \quad t_j = t_i + w_{ij}. \tag{6}$$

The total transmission delay of a K-hop path $\mathcal{P} = (n_{i_1}, n_{i_2}, \dots n_{i_K})$ is calculated as:

$$d(\mathcal{P}) = \sum_{k=1}^{K-1} w_{i_k i_{k+1}}(t_{i_k}) = \sum_{k=1}^{K-1} w_{i_k} + p_{i_k i_{k+1}}(t_{i_k} + w_{i_k}) \tag{7}$$

Likewise, the power can be represented by

$$p(\mathcal{P}) = \sum_{k=1}^{K-1} d^2_{i_k i_{k+1}} \tag{8}$$

The following is the summary of the proposed algorithm.

Algorithm 1: Optimal shortest path algorithm

```
Data: N: number of nodes, $n_s$: source, $n_d$: destination,
      $\mathcal{G} = (V, E_{ij}(t))$, $\vec{l_i}(t)$, $\hat{l}_i(t)$, $w_i(t)$, $\hat{w}_i(t)$ $c$
Result: $\mathcal{P}$: optimal path, $obj(\mathcal{P})$: path objective function,
        $d(\mathcal{P})$: path delay, $p(\mathcal{P})$: path power
(initialization):
$\{Visited\} \leftarrow \{n_s\}$; $t_{n_s} = 0$;
$\{Unvisited\} \leftarrow V \setminus \{n_s\}$; $t_n = \infty \forall n \in \{Unvisited\}$;
$n_i \leftarrow n_s$;
(finding optimal path:)
while $n_d \notin \{Visited\}$ do
    for $n_j \in \{Unvisited\}$ do
        if $e_{ij} = 1$ then
            update $w_{ij}(t_i)$ using (6);
            update $d_{ij}$ using (5);
            update $obj_{ij}$ using $w_{ij}(t_i)$ and $d_{ij}$ in (4);
            $t_j = t_i + w_{ij}(t_i)$;
            $d_{sj} = d_{si} + d_{ij}$;
            $obj_{sj} = obj_{si} + obj_{ij}$;
        $\{Visited\} \leftarrow \{Visited\} \cup \{n_i\}$;
        $\{Unvisited\} \leftarrow V \setminus \{n_i\}$;
        update next node: $n_i \leftarrow \underset{j \in \{Unvisited\}}{\text{argmin}} \{obj_{sj}\}$ ;
        Previous($n_j$)=$n_i$;
(form the optimal path:)
$\mathcal{P} = \{n_d\}$; $n = n_d$;
while $n \neq n_s$ do
    $n \leftarrow$ Previous($n$);
    $\mathcal{P} \leftarrow n \cup \mathcal{P}$
(calculate delay:)
t=0; dist=0; obj=0;
for $n \in \mathcal{P}$ do
    $i = n; j =$ next($n$);
    calculate $d_{ij}$ using (5) by substituting $\hat{l}_j(t)$ with $l_j(t)$
      and $\hat{w}_i(t)$ with $w_i(t)$;
    update $w_{ij}(t)$ using (5) and (6);
    update $obj_{ij}$ using $w_{ij}(t_i)$ and $d_{ij}$ in (4);
    $t \leftarrow t + w_{ij}(t)$;
    $dist \leftarrow d + d_{ij}$;
    $obj \leftarrow obj + obj_{ij}$;
$d(\mathcal{P}) = t$;
$p(\mathcal{P}) = p(\mathcal{P}) + d_{ij}^2$;
$f(\mathcal{P}) = \gamma d(\mathcal{P}) + (1 - \gamma) p(\mathcal{P})$;
```

We observe that the objective function in Algorithm 1 is the combination of the end-to-end delay and power along.Also, note that the predicted node locations $\hat{l}_j(t)$ and estimated waiting times $\hat{w}_i(t)$ are used to find the optimal intermediate nodes in the path, while actual node locations $l_j(t)$ and node delays $w_i(t)$ are used to calculate the actual delay and power metrics. In the case of linear motions, we have $B = 0$ in (1) and the state transition equations are reduced to $\vec{l_i}(t+1) = \vec{l_i}(t) + dt\vec{v_i}$+noise, which further simplifies the calculations by linearizing (5).

IV. Test and Evaluation Platform Design

In order to test the practicality of the developed Predictive Routing algorithm in practical Dynamic Networks, we develop a test and evaluation platform using ground robots. The main goal is to assess the superiority of the proposed algorithm

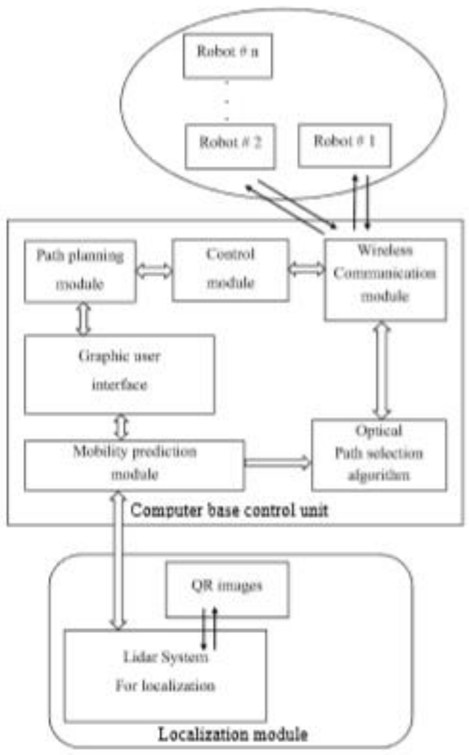

Robot # n
Robot # 2
Robot # 1
Path planning module
Control module
Wireless Communication module
Graphic user interface
Mobility prediction module
Optical Path selection algorithm
Computer base control unit
QR images
Lidar System For localization
Localization module

Fig. 2: Block diagram of the test and evaluation platform. Green color represents the external modules.

in finding optimal paths in real scenarios compared to conventional shortest path algorithms. In order to validate the credibility of the obtained simulation results, we compare the actual end-to-end distances for an end-to-end communication extracted from the captured video against the numerical values obtained by processing the planned motion trajectories. Fig. 2 illustrates the block diagram of the proposed system.

The system consists of three main modules including computer-based control unit (CBCU), localization unit (LU), and 5 ground robots. The ground robots are three-wheel car platform equipped with two electric motors, a LiDAR based localization module, and a control and wireless communication module based on *Raspberry pi 3 Model B*.

The control unit is developed in MATLAB environment with a graphic user interface (GUI) which includes the following modules: i) path planing module (PPU) in order to program the robots motion trajectories according to the probability distributions presented in section II, ii) command and control unit (CCU) in order to convert the planned motion trajectories as well as the communication parameters into a sequence of control commands, iii) a bi-way communication module in order to send the control commands to the ground robots and receive the relevant measurements from the robots, iv) topology prediction module in order to predict network topology based on the localization information, and v) a predictive routing algorithm in order to find the optimal path from the source to the destination, as detailed in section III. Finally, we note that the localization module includes two different approaches based on QR imaging and LiDAR system, as detailed in sections IV-B and IV-C.

A. Communication Protocol

The information exchange occurs between the CBCU and the robots through WiFi connectivity and using TCP/IP protocol. We select TCP/IP over WiFi (IEE 802.11 family) connectivity as opposed to other candidates such as ZigBee protocol (based on IEEE 802.15) for the implementation convenience and also its capability of exchanging high data rates. In order to simulate the waiting times at intermediate nodes, we program the nodes to hold data-packets for a pre-programmed time before forwarding to the next node. In order to facilitate information exchange among the robots and CBCU, we propose to use the following template for data packets. Note that TCP/IP includes built-in per-link routing, framing and integrity check, but include relevant fields in our packet format to make it independent from the underlying communication protocol and make it appropriate for Ad-hoc routing as well.

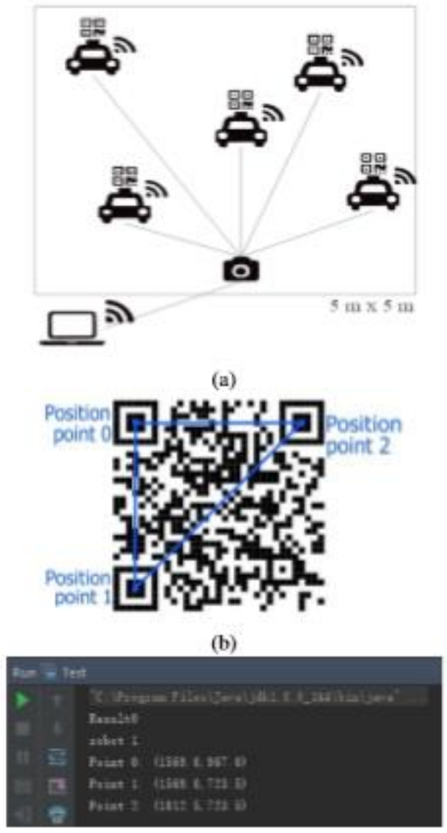
TABLE I: Communication Protocol: Unified Packet Format

Field	Values: Options
Start Flag	Fixed value: 01111110
Source Id	Unique source node ID
Destination Id	Unique destination node ID
Command	Options: Localization Info, Control command, Data Packet, ...
Length	Length of the payload data
Payload Data	Measurement information, motion trajectory information, ...
CheckSum	Module-256 addition
End Flag	Fixed value: 01111110

The *Source* and *Destination* fields include uniques IDs of the source and destination nodes. We assign ID=1,2,. . . ,5 to the 5 robots, ID=0 to CBCU, and ID=6 to localization module. *Start* and *Stop Flags* with constant patterns are used to mark the beginning and end of the packet. The fields *Length* and *CheckSum* are used for additional integrity check. The *Source* and *Destination* fields define the two ends of a per-link communications, and it is changed per link based on the optimal path determined by the predictive routing algorithm. This information is programmed in terms of routing tables in robots at the beginning of a transmission session. The *Command* filed defines the type of the packet including i) path planning command, ii) delay programming, ii) routing update command, iii) datapacket, and iv) localization measurement. The Payload data has a variable length and its content depends on the command. For instance, the payload data includes the velocity and initial directions of robots for a *path planning* command. Details of commands and payload data are omitted here for the sake of brevity.

B. QR-Code Based Identification and Localization

In this project, we use Quick Response (QR) code for both identification and localization purposes. QR codes are extensions of bar-codes into matrix format and can be used for image-based identification. Here, we attach a unique QR code printout on the upper side of each robot, representing its



```
"C:\Program Files\Java\jdk1.8.0_[illegible]\bin\java" ...
Result0
robot 1
Point 0: ([illegible])
Point 1: ([illegible])
Point 2: ([illegible])
```

Fig. 3: Image based identification/localization based on QR codes. (a): the localization module, (b): sample QR code, (c): sample localization information.

unique ID. A central camera pointing downwards is used 3 meters above the field level to locate the robots as shown in Fig. 3a.

We use the *Zxing: zebra crossing* package developed in Python programming language to identify and locate the objects labeled with QR codes. This package provides accurate readings for three corner points of QR codes (Fig. 3c). The dimensions of the coverage area is fixed ($5m \times 5m$), and the exact position of the camera are known. Even without these information, relative distances between labeled objects can be easily found with high accuracy by scaling the distances in the captured images (video frames). If the actual distance between the QR points 0 and 1 in a QR label is d_{01} and the program output is Point 0: (x_0, y_0) and Point 1:(x_1, y_1), the distance between any arbitrary readings (x_i, y_i) and (x_j, y_j) is:

$$d_{ij} = d_{01}\sqrt{\frac{(x_i - x_j)^2 + (y_i - y_j)^2}{(x_0 - x_1)^2 + (y_0 - y_1)^2}} \tag{9}$$

The QR readings can also be used to determine the orientation of each labeled object with respect to a reference direction. For instance if Points 0 and 2 are aligned with the reference direction (horizontal direction in Fig .3b), the relative angle θ is calculated as follows:

$$\theta = \tan^{-1}[(y_2 - y_0)/(x_2 - x_0)] \tag{10}$$

The obtained information is shared with other robots and the CBCU for predicting future locations and executing predictive routing algorithm. Note that only one central camera is used to reduce the implementation cost, but a distributed version can be implemented if one camera is deployed by each robot.

C. LiDAR-based Localization Method

The QR-based imaging provides accurate localization for both centralized and distributed methods. However, it requires high-resolution camera and image processing which increases the implementation cost. To implement low-cost solution, we plan to use laser based localization, as depicted in Fig. 4. In this approach, a reference rod is located in the center of the field. Also, a controllable turret with LiDAR transmitter and detector of type *Benewake TFMINI Micro LiDAR Module* mounted on the robot's body. The LiDAR module rotates a full circle using an embedded servo motor in order to detect and locate the reference rod. The strength of the reflected light as well as the angular phase, which maximizes the signal strength provides an accurate estimate of the robots location. Each robot share its location information with other robots as well as the CBCU in order to realize predictive communications. Note that similar to the QR-based localization, this approach is centralized, but it is easily extend-able to centralized method if the LiDAR transceiver keep track of all surrounding objects. This approach can be integrated with QR-based imaging to realize a fully distributed joint identification and localization method. The accuracy of utilized LiDAR transceiver is the range of few millimeters within the localization 30cm to 12m.

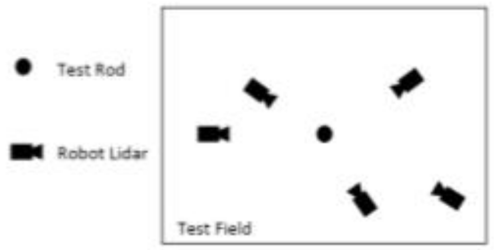


Fig. 4: LiDAR localization based on wireless networks.

Finally, Fig. 5 shows different building blocks of each robot.

V. Experimental Results

In this section, we provide simulations results as well as the practical tests in order to verify the optimality of the proposed predictive routing algorithm as well as the utility of the developed test and evaluation platform.

A. Experimental Setup

The objective of the practical test is to program robots such that they follow a pre-planned motion paths. The motion trajectory of each node is determined by its initial position, orientation, and velocity. These parameters are randomly

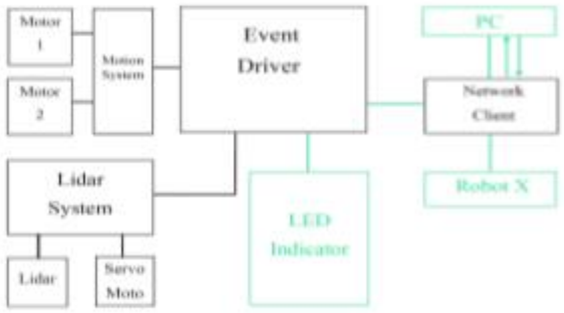


Fig. 5: Block diagram of the designed Robotic vehicle.

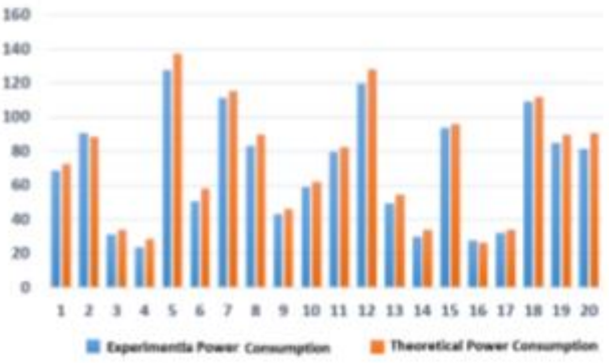


Fig. 6: Comparison of expected transmission power with the experimental values.

generated by the CBCU in the initialization phase using probability distributions discussed in section II and sent as a set of control commands to robots using the wireless communication module. Also, waiting time for each robot is randomly generated and programmed. The robots are programmed to send packets based on the determined optimal path all the way to the final destination. Each node once receives a packet, holds it according to its programmed waiting time and relays it to the next node. Transmit, receive and waiting modes are indicated by three LEDs for easy visualization. This framework can be used to verify the optimality of the developed predictive routing algorithm based on the collected localization information in real-world applications. In order to confirm the accurate operation of different steps (path planning, robot programming, routing, and waiting time enforcement), we develop a simple scenario where two robots exchange a data packet 5 times back and forth under different scenarios, and we compare the experimental values of the link lengths during each transmission phase (obtained from the captured video) with mathematically calculated values (based on the programmed parameters). The results are shown in Fig. 6. The transmission power in this figure is calculated as $\sum_{j=1}^{|\mathcal{C}|} d_{i_j i_{j+1}}$, where $\mathcal{C} = \{i_1, i_2, \ldots, i_{|\mathcal{C}|}\}$ is the selected path which includes the ordered set of $|\mathcal{C}|$ nodes. The error between the analytically derived and practically obtained values for total power consumption across 20 scenarios is less than 1%, which ensures the accuracy of the subsequent simulation results.

B. Simulation Results

For testing the performance of the predictive routing protocol, the path planning module first generates random network topologies by defining initial positions $\vec{l}_i(t)$, initial velocities, and the acceleration profile for each nodes based on distributions presented in section II. The robots are programmed with path-planning parameters along with the randomly generated waiting time.

Different simulation scenarios include investigating the impact of different network parameters including i) the number of nodes, ii) the average node velocity and iii) the average waiting time. In particular, we are interested to see the improvement of the delay and power consumption performance for the proposed method compared to the Dijkstra's shortest path algorithm. The idea is to identify the optimal path using the predicted locations and estimated waiting times and quantify the objective function with the actual node motions and waiting times.

Fig. 7 shows the effect of the number of nodes in the efficiency of the method proposed. It is observable that as number of nodes in the network increases, the optimal algorithm exhibits a higher performance improvement in terms of delay and power utilization compared to the conventional algorithm. The reason is that, more decision making are taken place for a larger network, and thereby there is larger margin between the two algorithms. Further, the network topology change is more extreme for larger networks, simply because it takes a longer time for a packet to reach the destination. For a network of 50 nodes, the performance of the optimal method shows about 10% improvement. For a larger network, this improvement is expected to rise.

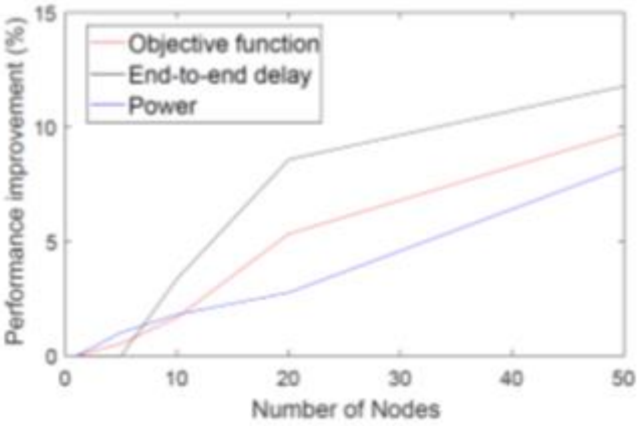


Fig. 7: The performance improvement (in percentage) by using predictive routing compared to conventional shortest path method. Objective function, the end-to-end delay and the power consumption show improvement consistently.

Next, the objective function for both methods regarding the average waiting time has been evaluated. Simulation results, as shown in Fig. 8, suggest higher performance gain for the optimal algorithm when the average waiting times are longer.

As we increase the waiting time for each node, topology changes are more severe and we obtain higher gains by predicting network topology. Therefore, substantial benefits can be obtained for queued networks with heavy traffic mode.

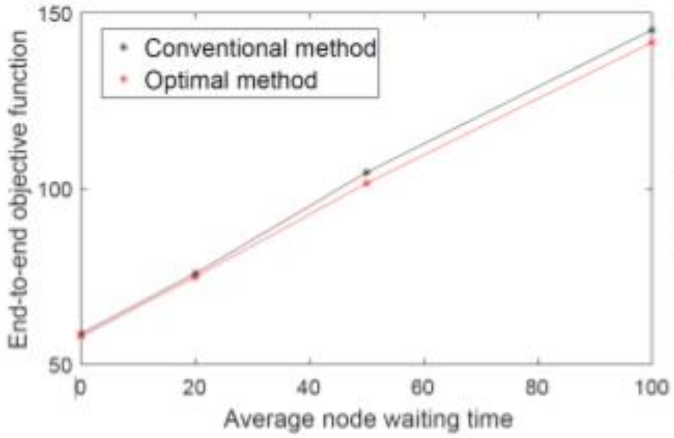


Fig. 8: Objective function for the proposed and the conventional Dijkstras shortest path algorithms based on the average node waiting time.

Lastly, we test the performance gain with respect to the average node velocity. For more dynamic networks, where the node velocities are higher, the optimal method improves compared to the conventional, as shown in Fig. 9. The reason is that for higher node velocities the network topology is evolving rapidly and consequently, the proposed algorithm shows a better performance. For this reason, the proposed algorithm is well suited to FANET with fast-flying objects.

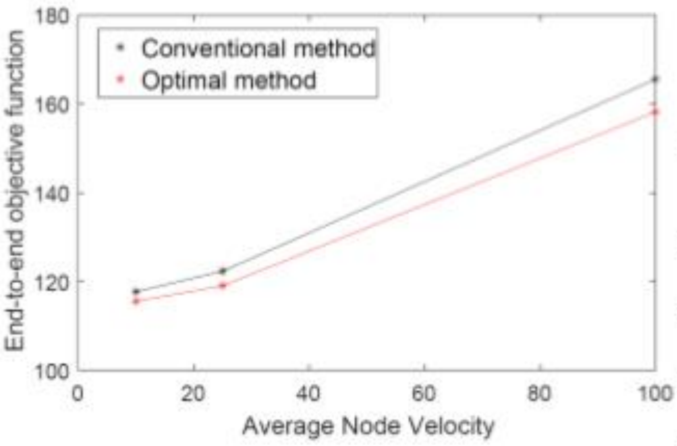


Fig. 9: Objective function for the proposed and the conventional Dijkstras shortest path algorithms based on the average node velocity.

VI. Conclusions

In this paper, an optimal routing method based on Dijkstra's shortest path algorithm is proposed for UAV networks by incorporating predicted network topology into the path selection criterion. Using a multi-objective function, we were able to jointly optimize the end-to-end delay and the transmission power. In order to mimic the real-world prediction uncertainty, we let the prediction error rise over time. We condiucted preliminary tests on ground robots to showcase the optimality and applicability of the proposed method in real-world application. Simulation results conform an improvement of about 10% for moderate network sizes. The performance gain increases for larger networks, larger average waiting time and higher node velocities.

This method can be viewed as a primary step towards developing predictive communications and we envision that much larger gains can be obtained by incorporating predictive network topology into different layers of communication protocols.

References

[1] E. Theodoridis, G. Mylonas, and I. Chatzigiannakis, "Developing an iot smart city framework," in *Information, intelligence, systems and applications (iisa), 2013 fourth international conference on*. IEEE, 2013, pp. 1–6.

[2] F. Xia, L. T. Yang, L. Wang, and A. Vinel, "Internet of things," *International Journal of Communication Systems*, vol. 25, no. 9, p. 1101, 2012.

[3] S.-H. Yang, "Internet of things," in *Wireless Sensor Networks*. Springer, 2014, pp. 247–261.

[4] S. Andreev, O. Galinina, A. Pyattaev, M. Gerasimenko, T. Tirronen, J. Torsner, J. Sachs, M. Dohler, and Y. Koucheryavy, "Understanding the iot connectivity landscape: a contemporary m2m radio technology roadmap," *IEEE Communications Magazine*, vol. 53, no. 9, pp. 32–40, 2015.

[5] V. Asghari and S. Aissa, "Adaptive rate and power transmission in spectrum-sharing systems," *IEEE Transactions on Wireless Communications*, vol. 9, no. 10, pp. 3272–3280, 2010.

[6] A. Razi, A. Valehi, and E. Bentley, "Delay minimization by adaptive framing policy in cognitive sensor networks," in *Wireless Communications and Networking Conference (WCNC), 2017 IEEE*. IEEE, 2017, pp. 1–6.

[7] W. Jiang, X. He, and T. Matsumoto, "Power allocation in an asymmetric wireless sensor network," *IEEE Communications Letters*, vol. 21, no. 2, pp. 378–381, 2017.

[8] M. Hammoudeh and R. Newman, "Adaptive routing in wireless sensor networks: Qos optimization for enhanced application performance," *Information Fusion*, vol. 22, pp. 3–15, 2015.

[9] Y. He, W. Xu, and X. Lin, "A stable routing protocol for highway mobility over vehicular ad-hoc networks," in *Vehicular Technology Conference (VTC Spring), 2015 IEEE 81st*. IEEE, 2015, pp. 1–5.

[10] T. Sivakumar, R. Manoharan, and K. Kuppusamy, "A stable routing protocol for vehicular ad hoc networks," in *Networks & Soft Computing (ICNSC), 2014 First International Conference on*. IEEE, 2014, pp. 46–49.

[11] O. K. Sahingoz, "Networking models in flying ad-hoc networks (fanets): Concepts and challenges," *Journal of Intelligent & Robotic Systems*, vol. 74, no. 1-2, p. 513, 2014.

[12] A. Rovira-Sugranes and A. Razi, "Predictive routing for dynamic uav networks," in *IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE)*, Oct 2017.

[13] S. Rosati, K. Kruzelecki, L. Traynard, and B. Rimoldi, "Speed-aware routing for uav ad-hoc networks," in *Globecom Workshops (GC Wkshps), 2013 IEEE*. IEEE, 2013, pp. 1367–1373.

[14] E. F. Flushing, L. M. Gambardella, and G. A. Di Caro, "On using mobile robotic relays for adaptive communication in search and rescue missions," in *Safety, Security, and Rescue Robotics (SSRR), 2016 IEEE International Symposium on*. IEEE, 2016, pp. 370–377.

[15] A. Razi, F. Afghah, and J. Chakaresk, "Optimal measurement policy for predicting uav network topology," in *Asilomar Conference on Signals, Systems, and Computers*, Oct 2017.

[16] J. Rohde, J. E. Stellet, H. Mielenz, and J. M. Zllner, "Localization accuracy estimation with application to perception design," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 4777–4783.

[17] H. Ortega-Arranz, D. R. Llanos, and A. Gonzalez-Escribano, *The Shortest-Path Problem:Analysis and Comparison of Methods*. Morgan & Claypool, 2014.