


Tokyo Institute of Technology

Experimental Study on Coverage Control with Anisotropic Sensors: Voronoi 1D



Fujita Lab, Dept. of Control and System Engineering,
FL07-26-2: January 15, 2008
David Asikin

Tokyo Institute of Technology

Fujita Laboratory

Tokyo Institute of Technology

Outline

- Introduction

- Experiment System
- Voronoi Partition in C++
- E-Nuvo Programming
- Wireless Networking
- Digital Image Processing

- Conclusion & Work Plan

Tokyo Institute of Technology

Fujita Laboratory

Tokyo Institute of Technology

Introduction


1. Coverage Control:
 - Definition
 - Application
 - FAQ
 - Experiment Objective
2. Review:
 - Voronoi
 - Lloyd's algorithm
 - Collective Motion

Tokyo Institute of Technology

Fujita Laboratory

Tokyo Institute of Technology

Coverage Control

- Definition:
 
- Application:

Search and rescue, surveillance robots, planet exploration, environmental monitoring, military and defense, etc.

Tokyo Institute of Technology

Fujita Laboratory

Tokyo Institute of Technology

Coverage Control

- **Why multiple robots?**
Robustness, improve reliability & probability of finding events, able to handle complex task, deepen understanding of nature, etc.
- **What is the goal in coverage control?**
Optimum placement of sensors.
- **Why is that the goal?**
Maximum possible utilization of the sensors: enhance network coverage, extend the system lifetime.

Tokyo Institute of Technology

Fujita Laboratory

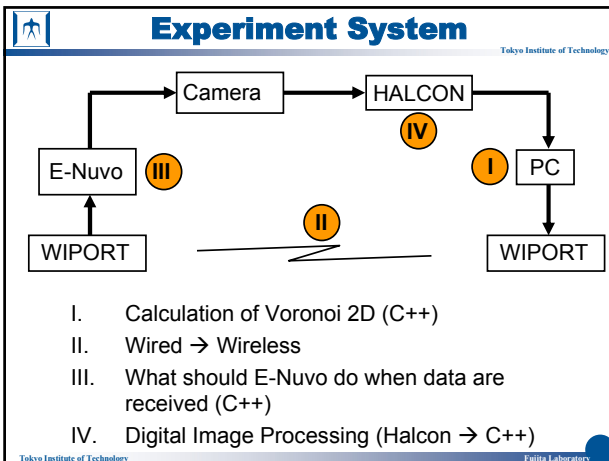
Tokyo Institute of Technology

Coverage Control

- **Objective of the experiment:**
 - **Verification** of Coverage Control Theory based on Lloyd Algorithm.
 - **Novelty** of the field (no such experiment hitherto).
 - Study of Coverage Control through **hands-on experience** of micro-controller, C programming, serial and wireless networking, and digital image processing.

Tokyo Institute of Technology

Fujita Laboratory



Experiment System

Tokyo Institute of Technology

Fujita Laboratory

- ❖ As a first step to Voronoi 2D → Voronoi 1D.

WHY?

- **Easier** calculation of Voronoi partition → easier C++ programming (no integral calculation for Centroid, etc).
- **Easier** control of E-Nuvo (do not have to worry about angle).
- **Expansion** of the preceding Voronoi 1D experiment with 4-wheel vehicles.
- **Familiarizing** myself with E-Nuvo programming (not Simulink).

Voronoi Partition in C++

Tokyo Institute of Technology

Fujita Laboratory

- **Objectives of the program (PC):**
 - Calculate Voronoi partition & centroid for each agent.
 - Move each agent to newly calculated centroids.

<ul style="list-style-type: none"> • Old program: 1. Halcon must receive image data from 3 agents not less, not more (otherwise, error) 2. There must be 3 agents in the field (Simulink → inconvenient, cannot create new variables on an ad hoc basis) 	<ul style="list-style-type: none"> • New program: 1. Halcon can receive data from less/more than 3 agents 2. Number of agents can be dynamic (C → can create new variable when necessary)
--	---

Voronoi Partition in C++

Tokyo Institute of Technology

Fujita Laboratory

- Final convergence point: 1/6, 3/6, 5/6

- Result: Contradicting theory of Voronoi Partition
- **WHY?** Simulink → Must define agents to be compared with using block diagram. Compare #1 with who?

Voronoi Partition in C++

Tokyo Institute of Technology

Fujita Laboratory

- Result: Agents will react autonomously upon the change of situation (= number of agents).

Voronoi Partition in C++

Tokyo Institute of Technology

Fujita Laboratory

HOW?

To be able to react autonomously upon dynamic situation

↓

Must know **WHO** are the neighbors

↓

Compare the agent **with ALL** other agents

↓

Comparison with the **least result**: Neighbors

Voronoi Partition in C++

Tokyo Institute of Technology

- NOTE: Must differ LEFT and RIGHT
- WHY?

the agent in consideration

- Result: p3 and p4 will be the neighbors of p2

WRONG

Tokyo Institute of Technology Fujita Laboratory

Voronoi Partition in C++

Tokyo Institute of Technology

- Then, HOW?

$p3 - p1 = (+) (+)$
 $p3 - p2 = (+)$

Find the min |.....| → p2

$p3 - p4 = (-)$
 $p3 - p5 = (-) (-)$

Find the min |.....| → p4

- Must separate (+) & (-)

Tokyo Institute of Technology Fujita Laboratory

Voronoi Partition in C++

Tokyo Institute of Technology

- HOW in C++?

- Create a matrix comprising of position difference between neighbors:

$$\text{difference_between_neighbors} = \begin{bmatrix} 0 & -1 & -3 & -4 & -5 \\ \dots & 0 & \dots & \dots & \dots \\ \dots & \dots & 0 & \dots & \dots \\ \dots & \dots & \dots & 0 & \dots \\ \dots & \dots & \dots & \dots & 0 \end{bmatrix}$$
- Create variables for (+) & (-) : plus[i] & minus[i]
- Find the minimum of the variables: plus[3]=p1 if (plus[3] > p2), then plus[3]=p2

Tokyo Institute of Technology Fujita Laboratory

Voronoi Partition in C++

Tokyo Institute of Technology

- Result:
- Able to know neighbors
- Able to calculate Voronoi & Centroid using information from **appropriate neighbors**

Problem #3: CLEAR!

Tokyo Institute of Technology Fujita Laboratory

Voronoi Partition in C++

Tokyo Institute of Technology

How to make agents autonomously respond to dynamically changing neighbors in C++?

Initial: 0, 1/6, 3/6, 5/6, 1

Final: 0, 1/4, 3/4, 1

Tokyo Institute of Technology Fujita Laboratory

Voronoi Partition in C++

Tokyo Institute of Technology

- KEY:

- In C++, a variable is needed to save the data of an agent.
 - Dynamically changing agent numbers
 - Needs variable with dynamically flexible size
- To calculate the actual position difference between agents, we need to know which agents are on the field and which are not.

Tokyo Institute of Technology Fujita Laboratory

Voronoi Partition in C++

Example 1: **static variable** (create variable with size greater than the number of agents)

PC (fixed-size)

HALCON AGENTS[5]

1	3.1
2	0
3	2.5
4	0
5	7.8
6	?

position

$difference_between_neighbors =$

0	0.6	4.7
...	0	...
...	...	0
...
...	...	0
...

No need to calculate agent #2 & #4!

Problem: What if there are 6 agents? → need to redefine variable size manually.

Voronoi Partition in C++

Example 2: **dynamic variable** (using malloc)

PC (dynamic variable)

HALCON AGENTS[?]

1	3.1
2	2.5
3	7.8
4	...
5	...
6	...

position

$difference_between_neighbors =$

0	0.6	-4.7
...	0	...
...	...	0
...

Which agents are on the field?

Result:

- PC automatically addresses new agents to new variable.
- Cut unnecessary memory for variables.

Voronoi Partition in C++

Conclusion:

To make agents respond autonomously to dynamically changing agent number, we need the following information:

1. How many & which agents are on the field?
2. Which agents are neighbors?
3. Dynamic variables to save agents' position.
4. Position of the agents.

☀ = NEW updates from the old program

Voronoi Partition in C++

3 ways of programming:

```

    graph TD
      Camera --> HALCON
      HALCON --> Simulink
      HALCON --> Matlab
      HALCON --> Cplusplus[C++]
      Simulink --> Dspace[D-space]
      Matlab --> E-nuvo[E-Nuvo]
      Cplusplus --> E-nuvo
      Dspace --> E-nuvo
      E-nuvo --> Camera
  
```

- ☒ Simulink → easy coding, but inconvenient in writing Lloyd algorithm for dynamic situation.
- ☒ Matlab → easy coding, but same problem as Simulink & takes too much time.
- 3. C++ → fast & easy adaptation for serial comm.

E-Nuvo Programming

Wireless:

Nuvo → RS 232 Serial Comm. → PC

Nuvo → Wiport → Wiport → PC

As a first step to wireless networking, we need to program E-Nuvo to make it able to receive data online (wired)

E-Nuvo Programming

How serial communication works:

C++ Code:

Sec	Velocity
0	0 m/s
10	10 m/s
20	10 m/s
30	15 m/s
40	0 m/s

OUTPUT BUFFER

INPUT BUFFER

atof

PC

E-Nuvo Programming

Tokyo Institute of Technology

- Problem:
E-Nuvo is **NOT designed** to receive data online.

↓

- The original program was:
IF (data_received = TRUE) → return(1)
- The original program can only receive 1 character at a time. Example: Input "100" to target_velocity → E-Nuvo can only receive "1", "0", "0".

Tokyo Institute of Technology Fujita Laboratory

E-Nuvo Programming

Tokyo Institute of Technology

- Solution:

- Using pointer, link the variable *data_received* in input buffer with a new variable.
- Program E-Nuvo to make it accumulate *data_received* until "Enter" button is pressed before executing command.

①

```
data_received → save → atof → target_velocity
```

②

C++ Code:

```
velocity
1
0
0
0
```

Tokyo Institute of Technology Fujita Laboratory

E-Nuvo Programming

Tokyo Institute of Technology

- Difficulties:

- Complicated coding using pointer and array.
- Adjusting variable types:

unsigned char

```
data_received
```

char

```
save
```

atof

float

```
target_velocity
```

- How to save different characters into a single array (strcpy? sprintf?)

Tokyo Institute of Technology Fujita Laboratory

E-Nuvo Programming

Tokyo Institute of Technology

- Result:

C++ Code:

Sec	Velocity
0	0 m/s
10	10 m/s
20	10 m/s
30	15 m/s
40	0 m/s

*PC program: teraterm (to monitor char sent & rcvd via serial port)

Tokyo Institute of Technology Fujita Laboratory

E-Nuvo Programming

Tokyo Institute of Technology

- Exercise program:

- "Hello world!" program.
- Menu and angle controlling program.
- Menu and velocity controlling program.
- Online controlling of E-Nuvo using arrow keys program.

Program #4 is required for smooth controlling of Nuvo when expanding the problem to 2D-space!

Tokyo Institute of Technology Fujita Laboratory

E-Nuvo Programming

Tokyo Institute of Technology

- Program #4: E-Nuvo motion controlling program.

- Velocity:

sec [s]	velocity [m/s]
0	0
1	10
2	10
3	0
- Angle:

sec [s]	omega [rad/s]
0	0
1	pi/4
2	0

Tokyo Institute of Technology Fujita Laboratory

E-Nuvo Programming

- How to turn RIGHT or LEFT?
Combine *target_velocity* and *target_angle*
- Example:

E-Nuvo Programming

- Problem #1:
To stop or turn direction, Nuvo must tilt to opposite direction → used in Voronoi 1D convergence.

- Point:
Need to find appropriate velocity so that Nuvo can still maintain its balance while moving at a speed fast enough to satisfy viewers.

E-Nuvo Programming

- Problem #2:
Need smooth transition (speed deceleration) when changing direction.

- If $0.075 \text{ m/s} \rightarrow -0.075 \text{ m/s}$, E-Nuvo will fall.
- If the difference between a and b is too big, E-Nuvo will fall.

E-Nuvo Programming

- Best solution:
Fading out the speed (e.g. $0.075 \text{ m/s} \rightarrow 0.070 \text{ m/s} \rightarrow 0.065 \text{ m/s} \rightarrow \dots \rightarrow 0 \text{ m/s} \rightarrow -0.010 \text{ m/s} \rightarrow \dots \rightarrow -0.075 \text{ m/s}$)
- However, the original program to input speed only provides **ONE INPUT PER SECOND**

Cannot fade out the speed; Too slow for application in Voronoi 1D convergence

Need to find one appropriate midway speed for transition

E-Nuvo Programming

- Velocity changing in 1D: **OK**
- What about angle changing and combination of angle & velocity changing in the case of 2D?

E-Nuvo Programming

- More complicated problem:

- What if order ② comes when executing order ①? Override/ ignore?
- How to maintain stability when changing to complicated motion? (① → ③ or ① → ②)
- When to put "Fade out speed" and when not to? (① → ④ and ① → ③)

E-Nuvo Programming

- Program #4:
Online controlling of E-Nuvo using arrow keys
program: **OK**

Program #4 resembles the situation of giving order from PC to E-Nuvo in actual Voronoi 2D

```

    graph TD
      A[Transmitting target_velocity to E-Nuvo in a great speed] --> B[Holding on arrow keys to order E-Nuvo to move]
  
```

Tokyo Institute of Technology
Fuji Laboratory

Menu

```

    graph TD
      Camera --> HALCON
      HALCON --> E_Nuvo[E-Nuvo]
      HALCON --> PC
      E_Nuvo --> WIPORT_E[E-Nuvo WIPORT]
      PC --> WIPORT_PC[PC WIPORT]
      WIPORT_PC -.-> WIPORT_E
  
```

- I. Calculation of Voronoi 2D (C++)
- II. Wired → Wireless
- III. What should E-Nuvo do when data are received (C++)
- IV. Digital Image Processing (Halcon → C++)

Tokyo Institute of Technology
Fuji Laboratory

Wireless Networking

- ❖ Wireless:
- Hardware: Wiport

```

    graph LR
      Nuvo --- Wiport1[Wiport]
      Wiport1 -.- Wiport2[Wiport]
      Wiport2 --- PC
  
```

- Software to setup Wiport:
 1. Device Installer (GUI)
 2. Web configuration (GUI)
 3. Telnet (e.g. Teraterm) (Text only)

Tokyo Institute of Technology
Fuji Laboratory

Wireless Networking

- Difficulties:
 1. Too little information on Wiport.
 2. Telnet → Device Installer → Web configuration
3 setup processes needed even to change a single mode.
 3. Baud-rate changing in Telnet:
Nuvo Baud-rate: 57600
Wiport setup Baud-rate: 9600

Tokyo Institute of Technology
Fuji Laboratory

Wireless Networking

- Exercise program:
 1. Message sending between PC and PC:

```

    graph LR
      PC1[PC] --- Wiport1[Wiport]
      Wiport1 -.- Wiport2[Wiport]
      Wiport2 --- PC2[PC]
  
```

2. Message sending between PC and E-Nuvo:

```

    graph LR
      Nuvo --- Wiport1[Wiport]
      Wiport1 -.- Wiport2[Wiport]
      Wiport2 --- PC
  
```

Tokyo Institute of Technology
Fuji Laboratory

Wireless Networking

- Message sending between PC and PC:
 - Connection method:
 1. TCP (1 PC to 1 PC):
 - Feedback from remote PC whether data is transmitted
 - Slow but reliable
 - Best for applications that need guaranteed delivery
 2. UDP (1 PC to Multiple PCs):
 - No packet checking
 - Packet loss, not reliable
 - Fast
 - Best for time-sensitive applications

Tokyo Institute of Technology
Fuji Laboratory

Wireless Networking

Tokyo Institute of Technology

❖ TCP Experiment:

1. 1 Host PC to 1 Remote PC

HOST:

- 192.168.11.120/5000
- 57600
- Active startup (any char)

REMOTE:

- 192.168.11.130/10001
- 57600
- Active startup (any char)

- Result: data is successfully transmitted!

Tokyo Institute of Technology Fujita Laboratory

Wireless Networking

Tokyo Institute of Technology

2. 1 Host PC to Channel 1: Remote PC 1
Channel 2: Remote PC 2

HOST: 192.168.11.120

Remote1: 192.168.11.130

Remote2: 192.168.11.131

- Result: Choose either Channel 1 or Channel 2

Tokyo Institute of Technology Fujita Laboratory

Wireless Networking

Tokyo Institute of Technology

3. 1 Host PC to Remote PCs with same IP address

HOST: 192.168.11.120

Remote1: 192.168.11.130

Remote2: 192.168.11.130

- Result: No data received on any PC. **WHY?**

Tokyo Institute of Technology Fujita Laboratory

Wireless Networking

Tokyo Institute of Technology

4. 1 Host PC to Multiple PCs: Connect & disconnect using a user-specified character.

HOST: 192.168.11.120

Remote1: 192.168.11.130

Remote2: 192.168.11.131

Result:

- Data is successfully transmitted!
- Use this for Voronoi 1D!** (Fast connect & disconnect in C++)

Tokyo Institute of Technology Fujita Laboratory

Wireless Networking

Tokyo Institute of Technology

5. Informing connection status to Remote PC:
WHY?

E-Nuvo can only receive 1 character at a time and saves the character to an array. When E-Nuvo get disconnected, it needs to reset the array (otherwise, error).

① 1010 + ② 100
③ 10100

Tokyo Institute of Technology Fujita Laboratory

Wireless Networking

Tokyo Institute of Technology

- E-Nuvo dynamics: $\dot{p}_i = -k(p_i - C_{V_i})$

BIG DIFFERENCE between current E-Nuvo position and centroid results in **BIG VELOCITY**

↓

E-Nuvo cannot maintain stability

↓

Fixed velocity is needed (above program → NG, program #4 → OK)

Tokyo Institute of Technology Fujita Laboratory

Wireless Networking

Tokyo Institute of Technology

- Conclusion:
 - By writing C++ code that can swiftly connect & disconnect host PC with multiple remote IPs, data transmitting to multiple E-Nuvos can be realized even with TCP connection.
 - In the case of UDP (simultaneous transmission):

Tokyo Institute of Technology Fujitsu Laboratory

Menu

Tokyo Institute of Technology

- I. Calculation of Voronoi 2D (C++)
- II. Wired → Wireless
- III. What should E-Nuvo do when data are received (C++)
- ➔ IV. Digital Image Processing (Halcon → C++)

Tokyo Institute of Technology Fujitsu Laboratory

Digital Image Processing

Tokyo Institute of Technology

- Software: HALCON (programming by Igarashi)

Tokyo Institute of Technology Fujitsu Laboratory

Digital Image Processing

Tokyo Institute of Technology

Old HALCON program:

- Used size (Large/Medium/Small) to differ agents:
 - Influence balancing in E-Nuvo
 - Incompatible to dynamic variable (**WHY?**)
- Can only detect 3 agents.

New HALCON program:

- Uses color to differ agents.
- Can detect multiple agents

Tokyo Institute of Technology Fujitsu Laboratory

Digital Image Processing

Tokyo Institute of Technology

- Incompatibility of the old program to dynamic variable:

(in size order)

Tokyo Institute of Technology Fujitsu Laboratory

Digital Image Processing

Tokyo Institute of Technology


- Compatibility of the new program to dynamic variable:

	HALCON	PC (dynamic variable)	
	AGENTS[?]	AGENTS[?]	
Red	1	3.1	Red
Green	2	2.5	Blue
Blue	3	7.8	White
Yellow	4		
White	5		

position

- Point: need to define link agents color & IP address!


Tokyo Institute of Technology Fujitsu Laboratory

 **Conclusion** Tokyo Institute of Technology

- Conclusion:
 - I. Calculation of Voronoi 2D (C++)
 - II. Wired → Wireless
 - III. What should E-Nuvo do when data are received (C++)
 - IV. Digital Image Processing (Halcon → C++)


Need to link these 4 elements into a connected system!

Tokyo Institute of Technology Fujitsu Laboratory

 **Future Work** Tokyo Institute of Technology

1. Voronoi 1D experiment
2. Voronoi 2D calculation program (Steve Fortune of Bell's lab)
3. Voronoi 2D experiment

Tokyo Institute of Technology Fujitsu Laboratory

 Tokyo Institute of Technology

Any question?

Tokyo Institute of Technology Fujitsu Laboratory