Collision-Free Coverage Control of Swarm Robotics Based on Gaussian Process Regression to Estimate Sensory Function in non-Convex Environment

Yaqub Aris Prabowo[1] and Bambang Riyanto Trilaksono[2]

Control and Computer Systems Research Group
School of Electrical Engineering and Informatics
Bandung Institute of Technology, Indonesia
[1]yaqub_aris@s.itb.ac.id and [2]briyanto@lskk.ee.itb.ac.id

Abstract: In this paper, Gaussian Process Regression (GPR) is used to estimate the time-invariant sensory function by multi-robots while performing coverage control in an environment with known obstacles. Multi-robots are deployed to explore and exploit the unknown sensory function in a given area based on their centroid of Voronoi partition. The trade-off between exploration and exploitation is weighted based on the maximum posterior variance calculated using GPR. Each robot uses the Hybrid Reciprocal Velocity Obstacle (HRVO) method to navigate without having a collision with either its neighbors or the obstacles. The presence of the obstacles may cause the Voronoi polygon is not convex so that the centroid is probably located inside the obstacle. Consequently, the centroid must be moved to the reachable point. Then, the reachable point is chosen to make the cost of coverage function, which is a function of the reachable distance and its sensory function, to be minimum. The two main contributions in this paper are: (1) Integrating estimation and coverage control based on GPR with HRVO method and (2) An enhanced strategy which adds termination iteration rule to increase the estimation and coverage time performance. Software simulations are conducted to compare the performance results by using different the number of obstacles and their locations, the number of robots, and the strategies.

Keywords: Coverage control, Swarm robotics, Gaussian Process Regression (GPR), Hybrid Reciprocal Velocity Obstacle (HRVO)

1. Introduction

Complexity in the process of prediction and mapping of an environment contaminated with hazardous substances triggers the use of static sensor network installed in the area where dangerous events potentially occurred such as radioactive leaks in a nuclear reactor, oil spills in a refinery, substances arising from volcanic eruptions, etc. However, the static sensor network has disadvantages compared with the mobile sensor network. Mobile sensor network has more flexibility and capability to cope with variable network conditions and environmental situations as it is conducted in [1]. Static sensor network has limited coverage while mobile sensor network can dynamically reorganize correspond to the distribution of sensory function (or usually called density function).

Various concepts and missions have been researched and implemented to measure, detect, and map unknown sensory objects. Reference [2] used three low-cost fixed-wing UAVs for contour mapping of a nuclear radiation field. The UAVs flew with different altitudes and followed given waypoints around the nuclear reactor before mapping the contour of nuclear density. Reference [3] sensed gas leakage by using MOX sensor mounted in Hexacopter UAV. An overview was given by [4] comparing among the type of UAVs to make an assessment of air quality. Reference [5] offered a novel solution to confront oil spills using multi autonomous unmanned oil cleaning. Multiple robots can also be used for forest fire fighting mission in [6] or built chemical concentration map in [7].

Various solutions have been proposed by many researchers starting from Lloyd's invention in [8]. Reference [9] uses the Lloyd algorithm to perform distributed sensing tasks using autonomous vehicles. Reinforcement learning approach for coverage control of multi-robot was

Received: December 26th, 2017. Accepted: March 21st, 2019
DOI: 10.15676/ijeei.2019.11.1.8

proposed in [10] with a given time-varying sensory function. Another approach, by using the backpropagation neural network, successfully proves the stability of coverage control in the sense of Lyapunov in an unknown field [11].

The coverage method in [12,13] using Gaussian Regression to estimate and coverage density function in the convex area without considering collision avoidance. If the environment to be explored contains several obstacles, then the multi-robot need to have obstacle avoidance capabilities. The obstacle avoidance method used for this multi-robot mission is HRVO from [14] because it guarantees multi-robot navigation without any collision. HRVO performs no oscillations or reciprocal dances over the trajectory path for any robot while avoiding collision compared with Reciprocal Velocity Obstacle from [15] and Velocity Obstacle from [16]. There is a special case if the robot faces up one or more obstacles towards its goal position. In [14], explained that if that case has happened, use some nodes which indicate some subgoals delivering to the actual goal. The A* algorithm originally proposed in [18] is used to generate subgoals.

The use of the Lloyd algorithm, also known as Voronoi iteration, requires all of the Voronoi partitions to be convex. Nevertheless, there may be one or more concave polygons. If there is a centroid of Voronoi partition placed inside the obstacle, then the robot at that Voronoi partition will confuse (cannot reach the goal position) although it will not hit the obstacle. Reference [17] called unreachable centroid point as a virtual point which was moved to the reachable point with minimum distance towards the virtual point. Because there is a sensory function in this research case, then the virtual point moved to the point which makes the cost function defined as the total of multiplication of distance and sensory value to be minimum.

There is a probability that the goal of two or more robots are almost stuck together when those robots are performing exploration. As a result, because the robots have their radius and clearance, they cannot exactly move to their goal position. If that happens, then the goal positions have to move so that the robots can be placed exactly to their goal position.

There are two main contributions in this paper. The first contribution is combining estimation and coverage control based on Gaussian Process from [13] with an obstacle avoidance algorithm based on HRVO method from [14] because of the existence of several obstacles in the environment. The second contribution is an enhanced algorithm which adds an iteration termination rule for multi-robot estimation and coverage control with faster estimation and coverage time to reach a certain variance threshold. When each robot does the estimation and coverage mission, there is a difference in trajectory length from the robots and also some robots which probably stuck at the point because the environment has too many robots. By purely using the algorithm from [13], some robots that have reached their goal will wait for other robots which have not yet reached their goal. Moreover, intuitively, with the existence of obstacles, the difference in trajectory length among all of the robots is increasingly striking. The robots waiting for other robots become unproductive. Hence, it is decided to instruct the robots to take a new measurement and carry on the next iteration if at least almost half of the number of robots have reached the goal destination. The robots which have reached the goal position will again decide whether to explore or exploit while the robots which have not yet arrived at the goal position still follow the previous decision (explore or exploit) with new estimated sensory function and Voronoi partitions. In section IV.C, the improvement of an algorithm which we called 2nd strategy can cover the environment faster than the original algorithm which we called 1st strategy while multi-robot achieve the same variance of sensory function.

The rest of this paper is organized as follows. Problem formulation is presented in Section II. The concept of the Lloyd algorithm, the Voronoi partition, the Gaussian Process Regression, and the Hybrid Reciprocal Velocity Obstacle will be explained. In Section III, a detailed algorithm to estimate density function, a trade-off between exploration and exploitation, path planning to avoid a collision, and a modified coverage strategy towards better time performance are described. The scenarios and results of simulations are illustrated in Section IV and eventually, conclusion and future research are presented in Section V. For ease of reference, a list of symbols and notations used in this paper are given in Table 1.

Table 1. The list of symbols and notations used in this paper

Symbol	Description
Ω	Two-dimensional square environment domain
V_i	Voronoi partition of robot-i
p_i, q	The position of robot-i and arbitrary point respectively
O	Set of circle obstacles; $O = \{o_1, o_2, \dots, o_m\} \in \Omega$
$\phi(x)$	Sensory/density function in point x
$H(p_i, \phi)$	Cost function according to the position of the robot and sensory function
M_{V_i}	Mass of Voronoi partition of robot-i
C_{V_i}	Centroid of Voronoi partition of robot-i
y_i	Measurement of a sensory/density value
ε, σ^2	Zero mean Gaussian noise and noise variance respectively
$\mathcal{N}(0, \sigma^2)$	Gaussian function with zero mean and variance σ^2
$\hat{\phi}(x)$	Estimated density/sensory function at point x
$\overline{K}, K(x_1, x_2)$	Covariance matrix and covariance function/Gaussian kernel respectively
$Var_{\phi}(x)$	Posterior variance at point x
$VO_{A\|B}$, $RVO_{A\|B}$, $HRVO_{A\|B}$	Velocity Obstacle, Reciprocal Velocity Obstacle, and Hybrid Reciprocal Velocity Obstacle region of robot A induced by robot B, respectively
$VO_{A\|O}$	Velocity Obstacle region of robot A induced by obstacle O
v_i	The velocity of robot-i
n, N_{total}	The number of robots and the total number of iterations respectively
$T_{timeout}$	Timeout robot traveled for one iteration
V_i^{new}	Voronoi partition of robot-i subtracts by a set of obstacles ($V_i^{new} = V_i - O$)
$C_{V_i^{new}}$	Voronoi centroid of V_i^{new} (maybe inside O)
$C_{V_i^{new}}^{new}$	New Voronoi centroid of V_i^{new} inside V_i^{new}, $C_{V_i^{new}}^{new} = \underset{p_i \in V_i^{new}}{\operatorname{argmin}} H(p_i, \phi)$
MV_i	Maximum posterior variance in V_i^{new} , $MV_i = \max_{x \in V_i^{new}} Var_{\phi}(x)$
x_i^{goal}	Goal position of robot-i
r_i, c_i	Radius and the clearance of robot-i respectively
$(a, b), r_o$	Center point and radius of obstacle respectively
d	A projected line from (a, b) to the line constructed from start and goal point
v_{des_i}, k_{prop}	The desired velocity of robot-i and a proportional gain respectively
$AV_i(v_i)$, v_i^{suit}	Set of available velocity and suitable velocity of robot-i respectively
v_{max}, v_i^{next}	Maximum velocity and the next chosen velocity of robot-i respectively

$penalty_i(v_i')$	The penalty of an available velocity v_i'
$w_i, tc_i(v_i')$	Weight and collision time for computing the penalty value of velocity-i
x_i^{k+1}	Robot-i position at time k+1 (after navigating with v_i^{next})
$\Delta t, \delta$	Time sampling and small distance threshold respectively
$stat$	Set of robot status;$stat = \{st_1, st_2, \dots, st_n\}$; ($st_i$=0: robot-$i$ reaches the goal, st_i=1: robot i do exploration, st_i=2: robot i do exploitation).
Var_{thres}	Variance threshold for terminating the coverage mission

2. Problem Formulation

A. Coverage Control Optimization

Basically, in a square convex environment $\Omega \subset R^2$ with n number of robots could be defined $\{V_1, V_2, \dots, V_n\}$ as the Voronoi partition of Ω i.e.

$$V_i(p) = \{q \in \Omega \,|\|q - p_i\| \leq \|q - p_j\|, \forall \mathrm{j} \neq \mathrm{i}\} \tag{1}$$

with p_i and q are the position of the robot-i and an arbitrary point in Ω respectively. The set of m circle obstacles $O = \{o_1, o_2, \dots, o_m\} \in \Omega$ may change the environment to be concave. Then, define the new Voronoi partition $V_i^{new} = V_i - O$. Due to the existence of sensory function $\varphi(q)$, defined the cost of coverage function as follows:

$$H(p_i, \phi) = \sum_{i=1}^{n} \int_{V_i} (q - p_i)^2 \phi(q) dq \tag{2}$$

Afterward, the Voronoi mass (M_{V_i}) and centroid (C_{V_i}) can be computed using (3) and (4).

$$M_{V_i} = \int_{V_i} \phi(q) dq, \tag{3}$$

$$C_{V_i} = \frac{1}{M_{V_i}} \int_{V_i} q\phi(q) dq \tag{4}$$

B. Gaussian Process Regression

Let $y_i = \varphi(x_i) + \varepsilon$ where y is the measurement value, $\varphi(x)$ is the density function at x, and ε is the zero-mean Gaussian noise $\varepsilon \sim \mathcal{N}(0, \sigma^2)$. Then, the estimated density function is defined as follows:

$$\hat{\phi}(x) = E[\phi(x)|x, y] = \boldsymbol{c}K(x_i, x), \forall x \in \Omega \tag{5}$$

Coefficient c and the covariance matrix are defined in (6) and (7) respectively.

$$\boldsymbol{c} = \begin{bmatrix} \boldsymbol{c}_{1_1} \\ \vdots \\ \boldsymbol{c}_{N_n} \end{bmatrix} = (\overline{K} + \sigma^2 I)^{-1} \begin{bmatrix} y_{1_1} \\ \vdots \\ y_{N_n} \end{bmatrix} \tag{6}$$

$$\overline{K} = \begin{bmatrix} K(x_{1_1}, x_{1_1}) & \dots & K(x_{1_1}, x_{N_n}) \\ \vdots & \ddots & \vdots \\ K(x_{N_n}, x_{1_1}) & \dots & K(x_{N_n}, x_{N_n}) \end{bmatrix} \tag{7}$$

where $c \in \mathrm{R}^{Nn \times 1}$, $y \in \mathrm{R}^{Nn \times 1}$, $\overline{K} \in \mathrm{R}^{Nn \times Nn}$, and $I \in \mathrm{R}^{Nn \times Nn}$ is identity matrix with N is the current number of iteration and n is the number of robots. Afterward, it is needed to compute the posterior variance of the estimate given in (8) and (9)

$$Var_{\phi}(x) = Var[\phi(x)|x,y] = K(x,x) - (K^*)^T(\overline{K} + \sigma^2 I)^{-1}K^* \quad (8)$$

$$K^* = [K(x_{1_1},x) \quad \dots \quad K(x_{N_n},x)] \quad (9)$$

where $K^* \in R^{Nn\times 1}$. It can be seen from the above formula that the process is non-Markovian so that the time computation will be longer along with the increase of the number of iteration.

C. Hybrid Reciprocal Velocity Obstacle

Velocity Obstacle is used to avoid collision with obstacles and other robots. In this subsection, it is briefly reviewed the concepts of velocity obstacle [16], reciprocal velocity obstacle [15], and hybrid reciprocal velocity obstacle [14]. Define that $VO_{A|B}$ in (10) as a velocity obstacle region of robot A (v_A) induced by dynamic robot B assumed the velocity of robot B (v_B) is constant.

$$VO_{A|B} = \{v|\lambda(p_A, v - v_B) \cap B \oplus -A \neq 0\} \quad (10)$$

p_A and p_B are the positions of the robot A and B respectively. $\lambda(p,v) = \{p+tv|t > 0\}$ is a ray with initial position p, direction v for time t. $A \oplus B = \{a+b|a \in A, b \in B\}$ is the Minkowski sum of robot A and dynamic obstacle B with $-A = \{-a|a \in A\}$.

Assume that the robot and the obstacle are disc-shaped. $D(p,r)$ is the disc shape with the center point in p, radius r, and clearance c. Then, equation (10) will be easier to understand with the new definition given in (11).

$$VO_{A|B} = \{v|\exists t > 0 :: t(v - v_B) \in D(p_B - p_A, r_A - r_B + 2c)\} \quad (11)$$

With the definition in (11), robot A will obstruct with dynamic obstacle B if $v_A \in VO_{A|B}$ is chosen.

However, obstacle velocity generates an oscillation trajectory. So, instead of only considering the current velocity of the neighbor (v_B), it will be more natural if it is also considering the current velocity of the robot itself (v_A). Then, the idea of reciprocal velocity obstacle arises by averaging its robot current velocity and the current velocity of the neighbor. More formally, the definition of reciprocal velocity obstacle is defined in (12).

$$RVO_{A|B} = \{v|2v - v_A\} \in VO_{A|B} \quad (12)$$

That definition, geometrically, can be translated so that the apex of $RVO_{A|B}$ is located in $(v_A - v_B)/2$ yet it still produces a reciprocal dance because there is no deal between each robot where they should pass.

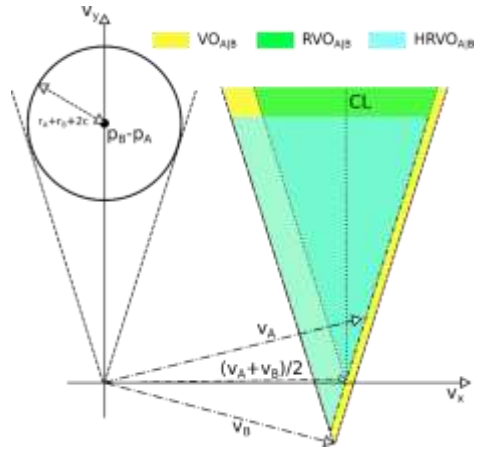


Figure 1. Illustration of $VO_{A|B}$, $RVO_{A|B}$, and $HRVO_{A|B}$ region

To resolve its problem, The Hybrid Reciprocal Velocity Obstacle concept was invented. The centerline (*CL*) of $RVO_{A|B}$ is used to determine the region of $HRVO_{A|B}$. If v_A lies on the right side of *CL*, then the apex of $HRVO_{A|B}$ will be placed at the intersection between left boundary of $VO_{A|B}$ and right boundary of $RVO_{A|B}$. Mirror the procedure if v_A located on the left side of *CL*. It also applies symmetrically to $HRVO_{B|A}$. Hence, by using hybrid reciprocal velocity obstacle, each robot has an agreement where they should pass. Hence, as seen in Figure 1 the difference of *VO*, *RVO*, and *HRVO* lies in the position of the apex. $VO_{A|B}$ has an apex at v_B, $RVO_{A|B}$ has an apex at $(v_A+v_B)/2$, and $HRVO_{A|B}$ apex is located with considering the centerline (*CL*) of $RVO_{A|B}$. If vector v_A is at the right of *CL* of $RVO_{A|B}$, then the apex of $HRVO_{A|B}$ places at the intersection between left boundary of $VO_{A|B}$ and right boundary of $RVO_{A|B}$.

3. The Algorithm

There are four outlines of algorithms i.e. estimating the density function, performing distributed coverage control, planning the trajectory to avoid a collision, and terminating the path planning. First, initialize the domain of environment Ω which want to observe with certain grid sampling, the domain of obstacles O, n number of robots, N_{total} number of iteration.

Algorithm 1 Pseudo Code

Requires: $\Omega, n, O, N_{total}, p_i, \sigma^2, strategy, stat, r_i, c_i$

for $j = 1 \rightarrow N_{total}$ **do**
- **Gaussian Process:**
- Collect y_i, $\hat{\phi}(x) = E[\phi(x) \mid x, y]$, $Var_\phi(x) = Var[\phi(x) \mid x, y]$
- **Compute Voronoi and Centroid:**
- **for** $i = 1 \rightarrow n$ **do**
  - **if** $j = 1$ **then**
    - Build $V_i(p)$ then $V_i^{new}(p)$ ▷ calc. Voronoi block accord. to position at the 1_{st} iter.
    - $C_{V_i^{new}} = \frac{1}{M_{V_i^{new}}} \int_{V_i^{new}} q\phi(q)dq$
  - **end**
  - Build $V_i(C_{V_i})$ then $V_i^{new}(C_{V_i})$
  - $C_{V_i^{new}} = \frac{1}{M_{V_i^{new}}} \int_{V_i^{new}} q\phi(q)dq$
  - **if** $C_{V_i^{new}} \in O$ **then**
    - $C_{V_i^{new}}^{new} = \text{argmin}_{p_i \in V_i^{new}} H(p_i, \phi)$ ▷ calculate new centroid outside obstacle
    - $C_{V_i^{new}} = C_{V_i^{new}}^{new}$
  - **end**
  - **Compute Goal Position:** ▷ st_i come from list $stat$ for termination
  - **if** $st_i = 0$ **then**
    - $MV_i = \max_{x \in V_i^{new}} Var_\phi(x)$
  - **end**
  - **if** $(st_i = 0$ ***and*** $B(1, MV_i) = 1)$ ***or*** $st_i = 1$ **then**
    - $x_i^{goal} = \text{argmax}_{x \in V_i^{new}} Var_\phi(x)$
    - x' =circle with center point is x_i^{goal} and radius $2r_i + c_i$
    - $Var_\phi(x') = -1$
  - **else**
    - $x_i^{goal} = C_{V_i^{new}}$ ▷ $(st_i = 0$ **and** $B(1, MV_i) = 0)$ **or** $st_i = 2$
  - **end**
- **end**
- x_i^{goal} =**Update Coinciding Goal Position**
- **Run Path Planning Algorithm**

end

The algorithms are summarized in the form of pseudo code in Algorithm 1, the line of sight algorithm in Algorithm 2, path planning algorithm in Algorithm 3 and termination algorithm in Algorithm 4.

A. Estimating the Density Function

To compute the estimation of the sensory function $\hat{\varphi}(x)$ and the posterior variance $Var_{\varphi}(x)$ at $x \in \Omega$, it p_i the position of robot-i, and the measurement values y_i of robot-i are necessary. For simulation purpose, the measurement noise is zero-mean Gaussian declared as $\mathcal{N} \sim (0,\sigma^2)$.

B. Performing Coverage Control

Compute the Voronoi partition according to the position of each robot $V_i(p)$ then $V_i^{new}(p)$ merely at the first of iteration because the information of the centroid C_{Vi} and $C_{V_i^{new}}$ of each Voronoi partition cannot be computed yet. Then, in accordance with the position of each robot and the estimated density function, the centroid $C_{V_i^{new}}$ can be calculated. Afterward, the Voronoi partition must be altered depending on the centroid $V_i^{new}(C_{V_i^{new}})$. If $C_{V_i^{new}}$ lies inside the known obstacle, move it to $C_{V_i^{new}}^{new}$ explained in the next paragraph.

The presence of obstacle doesn't alter the Voronoi partition however the centroid probably placed inside the obstacle. Therefore, discretize Ω become an occupancy grid with a certain resolution. Then, instead of using $C_{V_i^{new}}$, use $C_{V_i^{new}}^{new}$ (the new centroid location inside V_i^{new}) defined in (13) which is calculated using the greedy best search if $C_{V_i^{new}}$ is placed inside O as follows

$$C_{V_i^{new}}^{new} = \underset{p_i \in V_i^{new}}{\operatorname{argmin}} H(p_i, \phi) \tag{13}$$

The maximum of the posterior variance in each Voronoi region outside the obstacles is defined in (14).

$$MV_i = \max_{x \in V_i^{new}} Var_{\phi}(x) \tag{14}$$

MV_i is used as the mean of a random binomial distribution with a trial $B(1,MV_i)$ to determine the next decision towards the coverage. If the random value result is one, then the robot will explore the area. Otherwise, the robot will carry on exploitation. If the robot decides to explore, then the goal position of the robot is the point where the maximum posterior variance inside the new Voronoi domain of its robot. In another case, if the robot chooses to exploit, then the goal position of the robot (x_i^{goal}) is placed at the centroid of its Voronoi region.

Considering that the position at the maximum variance of one robot with another probably almost coincide, then, modify [13] by changing the posterior variance after one robot has its goal position at its maximum variance in its Voronoi region. The idea is to assign some constants less than zero around the goal point selected at the maximum variance location so that the next robot will not identify that region if it actually contains the maximum variance. The next robot will search for another maximum variance point. Then, the value of variance at the location circling around the robot's goal position with radius is the sum of twice of the robot's radius and clearance has to be changed. As a result, there will be no goal position that is located very close to each other and the robots can exactly track their goal positions. Figure 2 illustrates the difference when considering the almost coinciding goal locations and not. Without considering this issue, the robots will oscillate because they cannot track the goal positions while conversely the robots can exactly be placed into their goal positions. In the algorithm, change some values of posterior variance matrix in coordinate that intersect a circle defined with x_i^{goal} as the center of the circle with radius $(2r_i + c_i)$

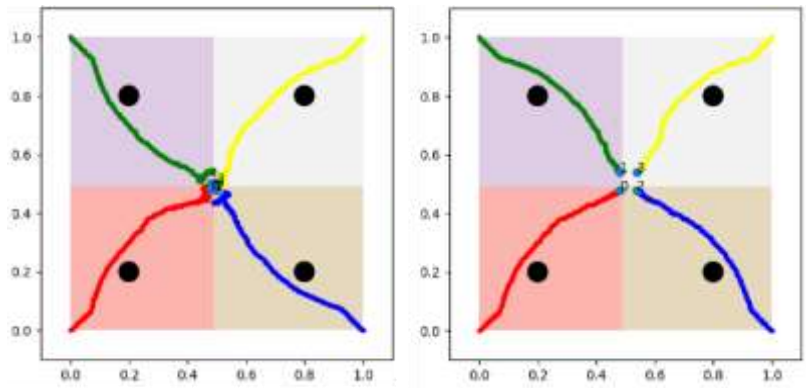


Figure 2. Comparison of HRVO path trajectories using 4 robots initially started from [0,0],[0,1],[1,0],[1,1] to each goal positions around the center of Ω where the maximum variance located, without considering almost coincided goal location (left) and with considering almost coincided goal location (right).

The line of sight condition of the robot and its goal position is necessary to compute. Because it is assumed that the shape of the obstacle in this work is a circle. Then the calculation is simply done by comparing the radius of the circle (r_o) and the line (d) which project the midpoint (a,b) of the circle obstacle perpendicular to the line construct from start and goal position of the robot. It is formally defined in (15).

$$d = \frac{\left\|(p_i - x_i^{goal}) \times (x_i^{goal} - (a,b))\right\|}{\left\|p_i - x_i^{goal}\right\|} \tag{15}$$

That calculation is only done if geometrically the obstacle does not intersect the rectangle built with the starting point is the upper left corner and the goal point is the lower right corner or vice versa. The calculation of the line of sight condition is shown in Algorithm 2.

Algorithm 2 line of sight

Requires: O, p_i, x_i^{goal}

for $o \in O$ **do**
- $(a, b, r_o) = o$
- $(x_1, y_1) = p_i$
- $(x_2, y_2) = x_i^{goal}$
- $\mathbf{x} = (x_1, x_2)$
- $\mathbf{y} = (y_1, y_2)$
- **if** $((b - r_o) > max(\mathbf{y}))$ **or** $((a - r_o) > max(\mathbf{x}))$ **or** $((b + r_o) < min(\mathbf{y}))$ **or** $((a + r_o) < min(\mathbf{x}))$ **then**
  - **return** *False*
- **else**
  - $d = \frac{\|(p_i - x_i^{goal}) \times (x_i^{goal} - (a,b))\|}{\|p_i - x_i^{goal}\|}$
  - **if** $d > r_o$ **then**
    - **return** *False*
  - **else**
    - **return** *True*
  - **end**
- **end**

end

C. Path Planning

After all of the robots have the subgoal and goal position, HRVO method is used to get the path trajectory. The robots gather the position and velocity information from the neighbors i.e. p_{B_j} and v_{B_j} respectively then use its position p_i and calculate its desired velocity towards the goal position v_{des_i} to build the $HRVO_{A|B}$ region. As stated in [9], v_{des_i} is defined as follows

$$v_{des_i} = -k_{prop}(p_i - x_i^{goal}) \tag{16}$$

with k_{prop} is a positive scalar gain. Assume that the radius r_i and the clearance c_i of the robots are the same. All of the robots also have to know the obstacles position to make $VO_{A|O}$ region of the robot according to the obstacles.

Then, choose the next velocity (v_i^{next}) of each robot from the list of available velocity vector AV_i which is formally defined in (17).

$$AV_i(v_i) = \{v_i' | \|v_i'\| < v_i^{max}\}. \tag{17}$$

v_{des_i} is constrained to the given maximum value of its robot velocity v_i^{max}. If there is any suitable velocity v_i^{suit} i.e. the velocity that does not intersect the $HRVO_{A|B}$ and $VO_{A|O}$ regions:

$$v_i^{suit} = \{v | v \in AV_i(v_i) - (HRVO_{A|B} \cup VO_{A|O})\}, \tag{18}$$

then choose the nearest suitable velocity to the desired velocity towards the goal position using (19).

$$v_i^{next} = \underset{v \in v_i^{suit}}{\operatorname{argmin}} ||v - v_{des_i}|| \tag{19}$$

Nevertheless, if there is no suitable velocity, as explained in [15], the penalty of a candidate available velocity v_i' which contains the expected time to collision $tc_i(v_i')$ and the distance from suitable velocity as follows:

$$penalty_i(v_i') = w_i \frac{1}{tc_i(v_i')} + \|v_i^{suit} - v_i'\| \tag{20}$$

The faster the robot to collide and the longer the velocity distance, the larger the penalty value. Symbol w_i is a scalar weight which makes the robot tend to more or less aggressive. Subsequently, get the velocity that makes the penalty value minimum which is defined in (21).

$$v_i^{next} = \underset{v_i' \in AV_i}{\operatorname{argmin}} penalty_i(v_i') \tag{21}$$

Algorithm 3 Path Planning

Requires: $\Omega, n, O, p_i, x_i^{goal}, v_i^{max}, k_{prop}, r_i, c_i, \delta, T_{timeout}, k = 1$
if *line of sight*(O, p_i, x_i^{goal}) **then**
  $x_i^{subgoal}$=**Construct with A* Algorithm**
  $x_i^{endgoal} = x_i^{goal}$
  $x_i^{goal} = x_i^{subgoal}\{0\}$
end
while $||p_i - x_i^{endgoal}|| > \delta$ ***and*** $k < \frac{T_{timeout}}{\Delta t}$ **do**
  Desired Velocity:
  $v_{A_i} = -k_{prop}(p_i - x_i^{goal})$
  Build $HRVO_{A|B}$ and $VO_{A|O}$
  Build $AV_i(v_i)$ and v_i^{ait}
  if $v_i^{ait} \neq \emptyset$ **then**
    $v_i^{next} = \text{argmin}_{v \in v_i^{ait}} ||v - v_{des_i}||$
  else
    $penalty_i(v_i') = w_i \frac{1}{ttc_i(v_i')} + ||v_{des_i} - v_i'||$
    $v_i^{next} = \text{argmin}_{v_i' \in AV_i} penalty_i(v_i')$
  end
  $x_i^{k+1} = x_i^k + v_i^{next}\Delta t$
  $p_i = x_i^{k+1}; k = k + 1$
  if $||p_i - x_i^{goal}|| < \delta$ **then**
    $x_i^{goal} = x_i^{subgoal}\{next\}$
  end
end

Define x_i^{k+1} as the position at time k+1 or the next position after x_i^k of robot-i with initial position $x_i^0 = p_i$ and Δt as the time sampling. Assume that there is no uncertainty factor, then the next position can be calculated by (22).

$$x_i^{k+1} = x_i^k + v_i^{next}\Delta t \tag{22}$$

A timeout limit ($T_{timeout}$) for every robot to travel from its initial position is needed because there might be a goal position that is unreachable or too far. The position of the robot is changed after (22) is calculated, i.e $p_i = x_i^{k+1}$. However, if the distance between robot and goal position is less than δ (the small distance threshold) for all robots s.t. $\left\|p_i - x_i^{goal}\right\| < \delta, \forall i = 1, .., n$, or timeout then it is decided that the trajectory planning has been finished.

D. Terminating The Path Planning

In [13], the algorithm is presented without trajectory planning where the termination happens after all of the robots reach their goal position. In subsequent discussions, such algorithm is called the 1st strategy while our proposed algorithm is called the 2nd strategy which terminates the path planning at least only half of the number of robots arriving at the goal position. The robots reaching the goal position will decide again whether it will explore or exploit while the robot which has not yet reached the goal is still at the previous decision but with the new estimation of the density function and Voronoi regions.

Algorithm 4 Terminating Iteration

Requires: $strategy = (1^{st} \text{ or } 2^{nd}), stat = \{st_1, st_2, \ldots, st_n\}, n, p_{A_i}, x_i^{goal}$
$terminator\_flag = False$
if *using* 1^{st} *strategy* **then**
  if *all* $(||x_i^k - x_i^{goal}|| < \delta)$ **then**
    $terminator\_flag = True$
  end
else
  ▷ *using* 2^{nd} *strategy*
  for $i = 1 \rightarrow n$ **do**
    if $||p_{A_i} - x_i^{goal}|| < \delta$ **then**
      $st_i = 0$
    end
    if $length(stat = 0) > (n \; \textbf{div} \; 2)$ **then**
      $terminator\_flag = True$
    end
  end
end
if $terminator\_flag = True$ **then**
  break
end

If all of the robots which have not yet reached their goal position are forced to decide again to make a new decision in the trade-off of exploration and exploitation, then "back and forth" movement is very likely to happen. The "back and forth" movement term in this research means that the robot goes to the centroid to exploit, then goes to maximum variance point which usually places far from the centroid to explore, then goes back to the centroid to exploit again although the robot has not yet arrived exactly at those points before). The "back and forth" movement probability is big when the maximum of posterior variance values is around 0.5 because of the use of binomial distribution. To implement this strategy, define the status of the robots as list named $stat = \{st_1, st_2, \ldots, st_n\}$ with three kinds of states: (0: robot reaches the goal, 1: robot do exploration, 2: robot do exploitation). Details of the algorithm are presented in Algorithm 4.

4. Simulation Result and Discussion

The simulation is conducted with Python script in Ubuntu 14.04 using Intel Core i5-7200U 2.5 GHz and 12 GB DDR4 RAM. We simulate the algorithm within some scenarios whereas the different position of obstacles, sensory function, number of robots, and strategy used. The environment domain is $\Omega = \{(0,1)\times(0,1)\}$ with sampling grid 0.02. Parameters for HRVO are selected as $v_{max} = 0.2$, $k_{prop} = 3$, $w_i = 1$, $r_i = 0.02$, $c_i = 0.01$, $\Delta t = 0.05$ sec, and $T_{timeout} = 10$ sec. For termination purpose, $\delta = 0.01$ and initialize value of *stat* list all zero are chosen. Gaussian kernel is used as the covariance function s.t. $K(x_1, x_2) = e^{-\frac{\|x_1 + x_2\|}{0.04}}$.

A. Scenario 1

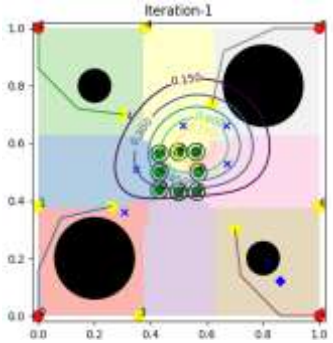
In Scenario 1, the simulation result of the 1^{st} strategy is presented. The sensory function reference is defined in (23) and (24).

$$\phi(x) = 20\left(e^{-\frac{\|x-\phi_0\|}{0.04}} + e^{-\frac{\|x-\phi_1\|}{0.04}}\right) + 7\left(e^{-\frac{\|x-\phi_2\|}{0.04}} + e^{-\frac{\|x-\phi_3\|}{0.04}}\right) \quad (23)$$

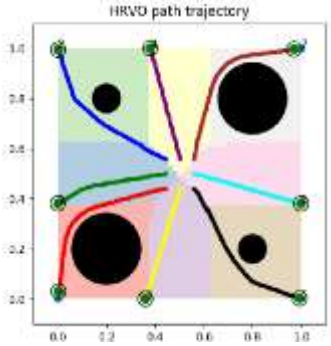
$$\phi_0 = (0.2,0.2), \phi_1 = (0.8,0.8), \phi_2 = (0.8,0.2), \phi_3 = (0.2,0.8) \quad (24)$$

Then, four circle obstacles $O = \{o_1,o_2,o_3,o_4\}$ = {(0.2,0.2,0.12), (0.8,0.8,0.12), (0.8,0.2,0.05), (0.2,0.8,0.05)} are placed in the four center of Gaussian distribution. The positions of the robots

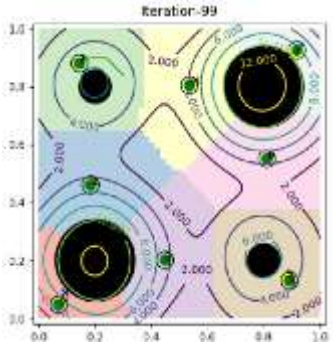
are initially placed at the center of the environment. By using the number of iteration N_{total} = 100 and the first strategy, the result is shown in Figure 3.



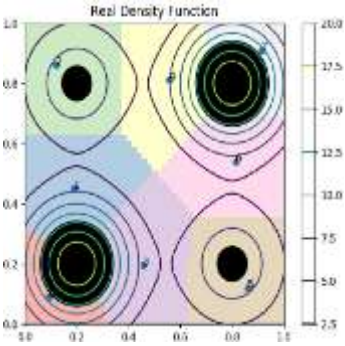
(a)



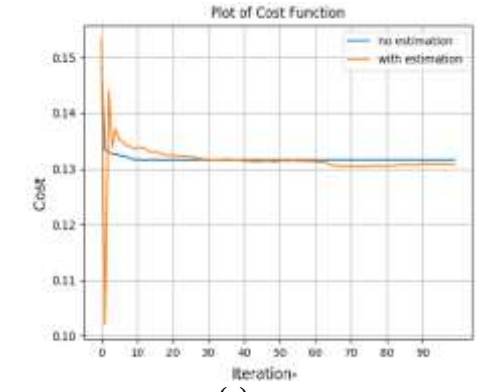
(b)



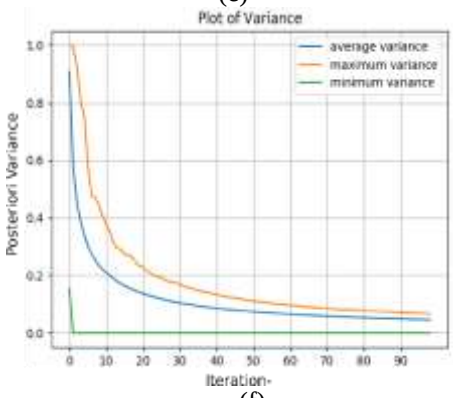
(c)



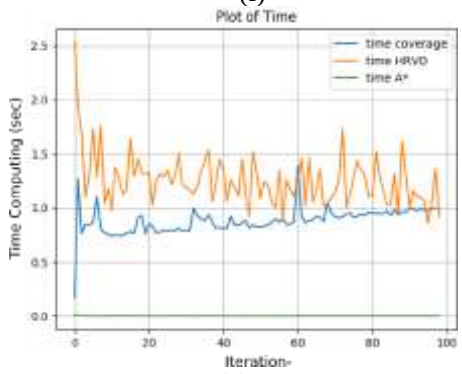
(d)



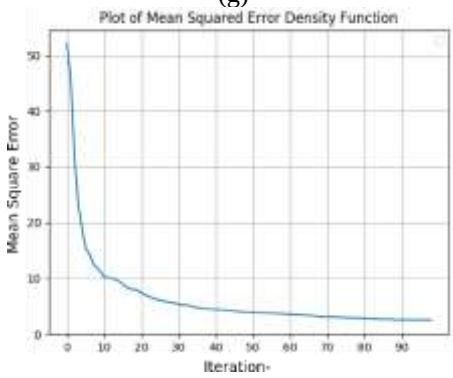
(e)



(f)



(g)



(h)

Figure 3. Simulation results for Scenario 1

Figure 3a shows four circle black obstacles and eight robot positions at the first iteration marked with the green circle with its clearance around theirs. The density function estimated is shown by the contour. There are eight Voronoi regions with the centroid marked with the blue cross mark. In the Voronoi region number 5, originally the centroid is placed inside the obstacle approximately in point (0.8,0.2). The new centroid is calculated so that its position is changed and marked with a blue diamond mark. Robot numbers 0, 2, 5, and 7 run the A* pathfinding algorithm because they cannot directly run into its goal position straightforward because they face up to the obstacles. For the robots which do not do the pathfinding, yellow dot marks illustrate its robot's goal positions. For the robots that compute the pathfinding algorithm, there appeared to be some lines that connected some points with the subgoal position marked by the yellow dot, and the goal position marked by the red dot.

The HRVO trajectory of the robots described in Figure 3b which all of the robots move smoothly avoid the obstacles to their goal locations. At the last iteration in Figure 3c, all robots successfully cover the environment with its sensory function. Compared with the true sensory function in Figure 3d, Figure 3c shows almost similar Gaussian distribution characteristic although the sensory value in the densest part is different from the true function because they cannot take the measurement inside the obstacles.

The coverage cost function with estimation value successfully tracks the cost function without estimating the sensory function illustrated in Figure 3e. Displayed in Figure 3f, the posterior variance value is always falling down. Figure 3g shows the comparison of time computation of coverage and estimation algorithm, A* pathfinding, and HRVO path planning. A* pathfinding computation is very light compared with others. Coverage and estimation algorithm computation size is always slightly growing as the iteration increased nevertheless it almost does not exceed the HRVO time computation. As can be seen in Figure 3h, it's difficult to reach zero mean squared error of density function because there are some points where the robots cannot measure.

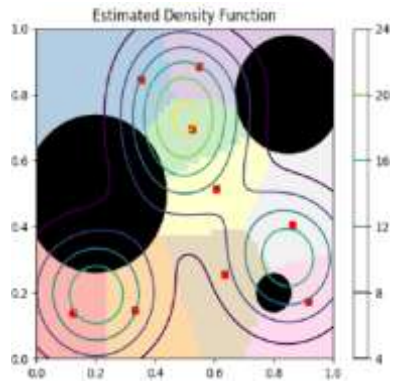
B. Scenario 2

In Scenario 2, the true sensory function is considered as

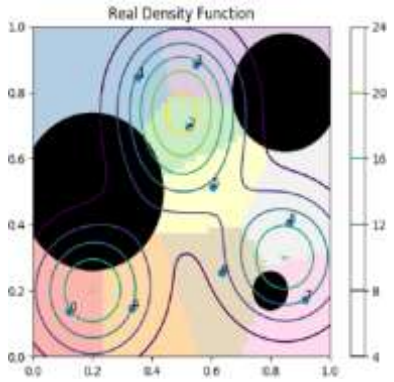
$$\phi(x) = 20\left(e^{-\frac{\|x-\phi_0\|}{0.04}} + e^{-\frac{\|x-\phi_1\|}{0.04}} + e^{-\frac{\|x-\phi_2\|}{0.04}}\right) + 12\left(+e^{-\frac{\|x-\phi_3\|}{0.04}}\right) \quad (25)$$

$$\phi_0 = (0.5,0.8), \phi_1 = (0.2,0.2), \phi_2 = (0.85,0.3), \phi_3 = (0.5,0.6) \quad (26)$$

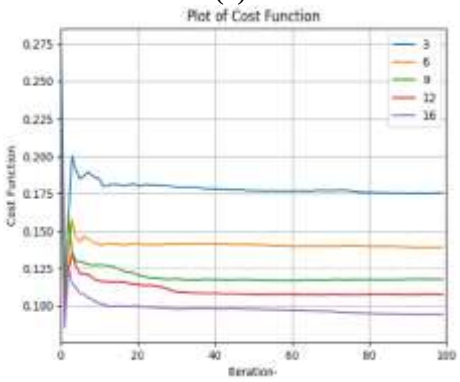
The location of the obstacles is defined s.t. $O=\{o_1,o_2,o_3\}=\{(0.2,0.5,0.2), (0.8,0.2,0.05), (0.85,0.8,0.15)\}$. In this scenario, the comparison of the use of different numbers of robots by using the first strategy with N_{total} = 100 is demonstrated. The result is illustrated in Figure 4.



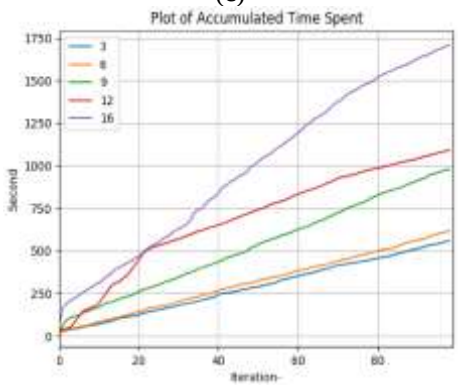
(a)


Real Density Function

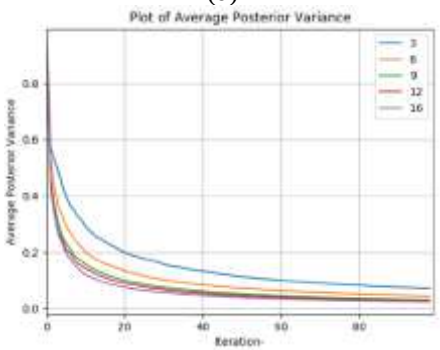
(b)


Plot of Cost Function
Cost Function
Iteration
3
6
9
12
16

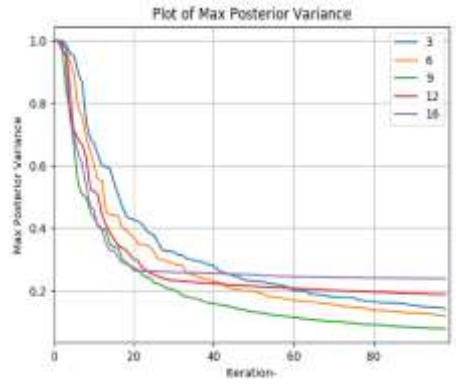
(c)


Plot of Accumulated Time Spent
Second
Iteration
3
6
9
12
16

(d)


Plot of Average Posterior Variance
Average Posterior Variance
Iteration
3
6
9
12
16

(e)



(f.)

Figure 4. Simulation results for Scenario 2

It is shown in Figure 4a the estimated density function which approximately has the same sensory function as the real sensory function in Figure 4b using 9 robots. Figure 4c describes that the more the robots used, the less the cost function value. Reciprocally, the more the robots used, the less the average of posterior variance value as shown in Figure 4e but the longer the time consumption as illustrated in Figure 4d. The maximum variance has a moderately different characteristic whereas the maximum of posterior variance by using the number of robots from 3 to 9 always decreased but, by using 12 and 16 robots, the maximum of posterior variance value seems constant from specific iteration steps as seen in Figure 4f.

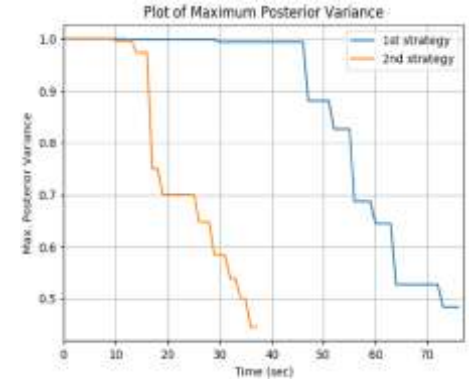
By using too many robots, the environment becomes too dense so that sometimes cause the robot cannot pass. HRVO seems unable to handle a very dense environment because of the repulsive force generated from many velocity obstacles. As a result, some robots may not track the goal position very well so that the maximum of posterior variance will not decrease especially for the goal position which contains the maximum of posterior variance.

C. Scenario 3

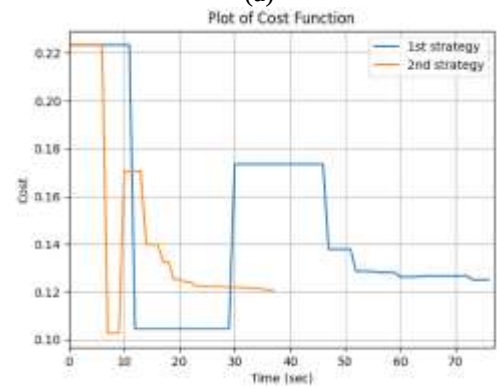
By using the parameter in Scenario 2, the 1st strategy is compared to the 2nd strategy in Scenario 3. It is necessary to know that the number of iterations does not represent the time estimation and coverage time. Therefore, the simulation will not stop because of the number of iteration, but the simulation stops if the maximum of a posterior variance of sensory function is already below the specified threshold Var_{thres}. Hence, some variables are compared according to the estimation and coverage time instead of the number of iterations. This simulation uses Var_{thres} = 0.5 and the comparison between the strategies is presented in Table 2.

Table 2. The comparison of time consumption with the different strategies

Number of robots	Time spent on the 1st strategy	Time spent on the 2nd strategy
3	96.6 sec	65.1 sec
6	81.9 sec	62.2 sec
9	77.5 sec	40.1 sec
12	99.6 sec	65.7 sec
16	100.9 sec	88.89 sec



(a)



(b)

Figure 5. Simulation results for Scenario 3

The time spent consists of the computation of all algorithms and time of HRVO path planning. Overall, the 2nd strategy gives faster time than the 1st strategy to accomplish the mission so that the posterior variance is below the Var_{thres} for the first time. The estimation and coverage mission using 9 robots is faster than using 3 or 6 robots but when using 12 robots or greater, the time spent becomes slower. It happens because the field is so dense that the computation of HRVO is going to be more complex. The comparison of maximum posterior variance and the cost function characteristic for the estimation and coverage mission using 9 robots are depicted in Figure 5.

5. Conclusion and Future Research

The coverage and estimation algorithm based on GPR from [13] have been successfully modified with the existence of obstacles so that the HRVO algorithm from [14] is involved. The simulation results show good performance for coverage, estimation, and path planning using multi-robot without collision. All of the robots can finally converge to the centroid of each Voronoi region where intuitively they tend to have high-density value.

According to the result from Scenario 2 which shows the comparison by using a different number of robots, it can be concluded that there is no guarantee that the more the number of robots used, the better the coverage performance. If the number of robots used is too much, then the environment will be too dense. With HRVO method for obstacle avoidance, if the environment is too dense then each robot will be difficult to navigate because the environment domain is almost filled with restricted (HRVO) region. Based on the result in Scenario 3 which shows the comparison by using different strategies, the estimation and coverage time with the 2nd strategy is faster than the 1st strategy to reach a certain variance threshold.

This work assumes that the obstacles are given a priori. Research on estimation and coverage control based on GPR using unknown and dynamic obstacles still becomes an open problem. If the robots know exactly where the obstacles are then the robots know where the reachable maximum posterior variance and centroid location are. If the environment is unknown, whether the robots must predict the location of the obstacles for simultaneous estimation, coverage, path planning, and mapping or carry on estimation, coverage, and path planning after mapping the environment.

In the future, the simulation using a hardware-in-the-loop simulation scheme will be conducted so that the performance result of the algorithm can be seen by considering the communication time which is very essential for the multi-robot mission. Then, if real robots and sensors are used, the dynamic constraint and uncertainty of the robots must be applied. Moreover, coverage control to estimate time variant sensory function which may be affected by wind velocity or another noise is an interesting issue to be the next research topic.

6. Acknowledgment

Thanks to Marco Todescato (todescat@dei.unipd.it) from the University of Padova, Italy, for the discussion about his prior work.

7. References

[1]. Y. C. Wang, "Mobile Sensor Networks: System Hardware and Dispatch Software," ACM Computing Surveys (CSUR) Surveys Homepage archive, Volume 47 Issue 1, July 2014, Article No. 12. *ACM New York*, NY, USA.

[2]. J. Han, Y. Xu, L. Di, and Y. Chen, "Low-cost Multi-UAV Technologies for Contour Mapping of Nuclear Radiation Field," *Journal of Intelligent and Robotic Systems* April 2013, Volume 70, Issue 14, pp 401410.

[3]. M. Rossi, D. Brunelli, A. Adami, L. Lorenzelli, F. Menna, and F. Remondino, "Gas-drone: Portable Gas Sensing System on UAVs for Gas Leakage Localization," *In Proceedings of the 2014 IEEE SENSORS*, Valencia, Spain, 25 November 2014; pp. 14311434.

[4]. T. F. Villa, F. Gonzales, B. Milijevic, Z. D. Ristovski, and L. Morawska, "An Overview of Small Unmanned Aerial Vehicles for Air Quality Measurements: *Present Applications and Future Prospectives," Sensors*, vol. 16, (7), pp. 1072, 2016.

[5]. N. M. P. Kakalis and Y. Ventikos, "Robotic Swarm Concept for Efficient Oil Spill Confrontation," *Journal of Hazardous Materials* Volume 154, Issues 13, 15 June 2008, Pages 880-887.

[6]. K. A. Ghamry, M. A. Kamel, and Y. Zhang, "Multiple UAVs in Forest Fire Fighting Mission Using Particle Swarm," *Optimization 2017 International Conference on Unmanned Aircraft Systems (ICUAS).* June 13-16, 2017, Miami, FL, USA.

[7]. M. Turduev, Y. Atas, P. Sousa, V. Gazi, and L. Marques, "Cooperative Chemical Concentration Map Building Using Decentralized Asynchronous Particle Swarm Optimization Based Search by Mobile Robots," *The 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems* October 18-22, 2010, Taipei, Taiwan.

[8]. S. P. Lloyd, "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, 28 (2): 129137, 1982.

[9]. J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage Control for Mobile Sensing Networks, *Automatica*," vol. 20, no. 2, pp. 243255, 2004.

[10]. A. A. Adepegba, S. Miah, and D. Spinello, "Multi-Agent Area Coverage Control Using Reinforcement Learning," *Proceedings of the Twenty-Ninth International Florida Artificial Intelligence Research Society Conference*, 2016.

[11]. Dirafzoon, S. Emrani, S. M. A. Salehizadeh, and M. B. Menhaj, "Coverage control in unknown environments using neural networks," *Springer Science+Business Media* B.V., 2011.

[12]. Carron, M. Todescato, R. Carli, L. Schenato, and G. Pillonetto, ”Multi-agents Adaptive Estimation and Coverage Control using Gaussian Regression,” 2015 *European Control Conference (ECC)*, July 15-17, 2015. Linz, Austria.
[13]. M. Todescato, A. Carron, R. Carli, G. Pillonetto, and L. Schenato, ”Multi-robots Gaussian Estimation and Coverage Control: From Client-server to Peer-to-peer Architectures,” *Automatica* 80 (2017) 284294.
[14]. J. Snape, J. van den Berg, S. J. Guy, and D. Manocha, ”The Hybrid Reciprocal Velocity Obstacle,” *IEEE Transactions on Robotics*, Vol. 27, No. 4, August 2011.
[15]. J. van den Berg, M. Lin, and D. Manocha, “Reciprocal velocity obstacles for real-time multi-agent navigation,” *in Proc. IEEE Int. Conf. Robot. Autom., Pasadena*, CA, May 2008, pp. 19281935.
[16]. P. Fiorini and Z. Shiller, “Motion planning in dynamic environments using velocity obstacles,” *Int. Journal of Robotics Research*, vol. 17, no. 7, pp. 760772, 1998.
[17]. A. Breitenmoser, M. Schwager, J. C. Metzger, R. Siegwart and D. Rus, ”Voronoi Coverage of Non-Convex Environments with A Group of Networked Robots,” *IEEE International Conference on Robotics and Automation*, 2010.
[18]. P. E. Hart, N. J. Nilsson, and B. Raphael, ”A Formal Basis for the Heuristic Determination of Minimum Cost Paths,” *IEEE Transactions on Systems Science and Cybernetics SSC4*, 1968.

Yaqub Aris Prabowo received a bachelor’s degree in electrical engineering and master’s degree in control engineering and intelligent systems from the School of Electrical Engineering and Informatics in Bandung Institute of Technology. He is currently a Ph.D. student in electrical engineering and informatics in Bandung Institute of Technology. His research interests include swarm intelligence, robotics, embedded control systems, and artificial intelligence.

Bambang Riyanto Trilaksono received the graduate degree from Electrical Engineering in Bandung Institute of Technology, master and Ph.D. degree from Electrical Engineering, Waseda University, Japan. Currently, he is a professor in Control and Computer System Research Group in Bandung Institute of Technology. His research interests include optimal control, robust control, intelligent control & intelligent systems, discrete event systems, control applications, robotics, and embedded control systems.