PENGENALAN DAN PENCARIAN POSISI ROBOT DALAM PENCARIAN SUMBER KEBOCORAN GAS

Dhiemas R.Y. Sembor, Ferdian Jovan, dan M. Sakti Alvissalim

Fakultas Ilmu Komputer, Universitas Indonesia, Kampus Baru UI Depok, Jawa Barat, 16424, Indonesia

E-mail: dhiemas.y@.ui.ac.id

Abstrak

Implementasi algoritma *Particle Swarm Optimization* (PSO) pada robot untuk pencarian sumber kebocoran gas memerlukan informasi posisi dari setiap robot. Penggunaan kamera memungkinkan untuk mendapatkan informasi posisi robot yang lebih absolut dan akurat dibandingkan dengan perangkat pencari informasi posisi lainnya. *Paper* ini memaparkan suatu metode yang akan digunakan untuk mencari posisi robot. Metode yang diajukan menggunakan beberapa teknik pengolahan citra, seperti, *Color Filtering* dan *Blobs Filtering* guna mengenali dan memperoleh posisi robot. Berdasarkan eksperimen yang telah dilakukan, metode ini mampu menentukan posisi absolut robot sehingga ia dapat menyediakan informasi posisi robot dalam implementasi algoritma PSO.

Kata Kunci: *kamera, particle swarm optimization, posisi, sumber kebocoran gas, teknik pengolahan citra*

Abstract

Implementation of Particle Swarm Optimization (PSO) algorithm on the robot to search for the gas leakage source requires information of the position of each robot. The use of the camera allows the robot to obtain position information more accurate than the absolute position information of other search tools. This paper describes a method that will be used to locate the position of the robot. Proposed method uses several image processing techniques, such as, Color Filtering and Blobs Filtering in order to identify and obtain the robot position. Based on the experiments that have been performed, the method is capable of determining the absolute position of the robot so that it can provide position information of the robot in PSO algorithm implementation.

Keywords: *camera, image processing techniques, particle swarm optimization, positioning, sources of gas leakage*

1. Pendahuluan

Penelitian mengenai pencarian sumber kebocoran gas telah dilakukan oleh banyak peneliti. Salah satunya adalah dengan menggunakan algoritma *Particle Swarm Optimization (PSO)* yang dilakukan oleh Jatmino, dkk [1]. Sebelumnya, penelitian ini pernah dilakukan oleh Nugraha yang berupa pengembangan simulasi ruang pencarian 2D dengan mengimplementasikan algoritma PSO di dalamnya [2]. Setelah itu, Pambuko [3], dan Febrian [4] berusaha untuk memodifikasi algoritma PSO yang ada serta membangun simulasi ruang pencarian 3D. Penelitian yang dilakukan oleh peneliti sekarang ini adalah dalam rangka implementasi algoritma PSO pada ruang pencarian nyata, khususnya mencari posisi robot pada ruang pencarian.

Implementasi algoritma ini menggunakan banyak robot yang saling berkoordinasi satu sama lain. Robot-robot saling bertukar informasi mengenai posisi sumber gas yang terbaik menurut pengalamannya masing-masing. Selain itu, masing-masing robot juga perlu mengetahui posisinya sekarang ini agar dapat mencapai ke tempat sumber gas yang dicurigai berdasarkan pengalaman dan sugesti dari robot lainnya. Dengan demikian, informasi posisi yang dipakai di sini harus direpresentasikan dalam suatu sistem koordinat global yang dipahami oleh setiap robot. Permasalahannya adalah robot sangat sulit untuk menentukan posisinya sendiri. Salah satu pendekatan untuk mengatasi permasalahan ini adalah dengan memanfaatkan suatu sistem yang dibangun dengan tujuan untuk menentukan posisi suatu objek. Jadi, robot tinggal mengakses sistem tersebut untuk mengetahui posisinya berada sekarang ini.

Walaupun saat ini telah ada teknologi *Global Positioning System* (GPS) yang dapat mengetahui posisi suatu objek di muka bumi, GPS tidak dapat

diimplementasikan dalam penelitian ini. Hal ini disebabkan GPS memiliki keterbatasan dalam hal akurasi. Tingkat akurasi GPS saat ini berkisar antara 1 - 10 meter, bahkan bisa mencapai 30 meter [5]. Dengan kata lain, jika ada dua robot yang berjarak kurang dari 10 meter satu sama lain, maka kemungkinan GPS akan memberitahu bahwa posisi kedua robot tersebut pada koordinat yang sama. Dengan demikian, GPS tidak dapat diandalkan pada keadaan ruang pencarian yang kecil. Selain itu, keadaan ruang pencarianlah yang membuat GPS ini benar-benar tidak dapat dipakai, yaitu daerah berupa ruangan dengan dimensi yang bisa jadi kurang dari 10×10 meter. Oleh karena itu, peneliti membangun suatu sistem yang dapat menyediakan informasi posisi robot secara mandiri. Sistem ini dapat dibangun dengan memanfaatkan banyak kamera, di mana setiap kamera bertugas untuk mengawasi daerahnya masing-masing dari atas. Penggunaan banyak kamera dalam penelitian ini dimungkinkan karena ruang pencarian berupa ruangan tertutup sehingga kamera dapat diletakkan pada langit-langit ruangan.

Penggunaan banyak kamera dalam menentukan posisi robot bukan merupakan sesuatu yang mudah untuk dilakukan. Peneliti menemukan minimal tiga buah tantangan di dalamnya. Tantangan pertama adalah bagaimana cara mengidentifikasi robot melalui perangkat kamera?. Tantangan kedua adalah bagaimana cara membedakan robot satu sama lainnya?. Dalam penelitian ini, peneliti menggunakan tiga buah robot di mana robot ini identik satu sama lain. Hal ini merupakan suatu hal yang sulit dilakukan oleh kamera layaknya kita berusaha untuk membedakan tiga orang yang terlahir sebagai kembar identik. Tantangan ketiga adalah bagaimana cara menentukan posisi robot mengingat dalam penelitian ini digunakan banyak kamera?. Hal ini memerlukan koordinasi dan sinkronisasi antar kamera sehingga posisi robot yang didapatkan sesuai dengan yang ada di lapangan.

Dalam penelitian ini terdapat batasan-batasan permasalahan yang dibuat oleh peneliti, antara lain: (1) Lingkungan robot dalam penelitian ini adalah sebuah ruangan tertutup sedemikian rupa sehingga intensitas cahaya yang masuk tidak memilki perbedaan yang besar setiap waktunya, (2) Robot adalah satu-satunya objek yang bergerak dalam lingkungan ini, dan (3) Lantai pada lingkungan mempunyai warna sedemikian rupa sehingga dapat dibedakan dengan warna objek robot.

Bagian ini akan menceritakan tentang landasan teori yang peneliti gunakan dalam penelitian ini. Pertama, peneliti akan menjelaskan tentang konsep posisi pada algoritma *Particle Swarm Optimization* (PSO) yang digunakan untuk mencari sumber kebocoran gas. Kemudian, peneliti akan menjelaskan teknik-teknik pengolahan citra yang digunakan.

Posisi pada *Particle Swarm Optimization* (PSO) pertama kali dicetuskan oleh Kennedy dan Eberhart [6][7] pada tahun 1995. Algoritma ini terinspirasi oleh perilaku koloni burung ketika mencari makanan. Pada koloni ini, setiap individu akan berusaha mencari makanan pada daerahnya masing-masing. Bukan hanya itu saja, individu burung tersebut juga saling bertukar informasi mengenai peluang adanya makanan di daerahnya masing-masing. Berdasarkan fenomena alam ini, dapat disimpulkan bahwa ketika mencari sumber makanan, perilaku burung dipengaruhi oleh dua komponen, yaitu komponen kognitif dan sosial. Komponen kognitif berasal dari informasi yang diperoleh burung atas usahanya sendiri. Komponen sosial berasal dari informasi yang diperoleh burung akibat komunikasi dengan kawan sesamanya. Pada implementasinya di PSO, koloni burung tersebut digantikan oleh sekumpulan robot yang bertugas untuk mencari sesuatu. Pada penelitian ini, PSO digunakan untuk mencari sumber gas.

Algoritma PSO dimodelkan dalam bentuk perhitungan matematis [8-10]. Robot yang digunakan untuk melakukan pencarian disebut sebagai partikel.

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (1)$$

persamaan di atas menjelaskan perhitungan posisi suatu partikel. Notasi $x_i(t)$ merupakan vektor posisi dari partikel ke-i pada waktu iterasi ke-t. Notasi $v_i(t+1)$merupakan vektor kecepatan partikel ke-i pada waktu iterasi ke-t. Adapun vektor kecepatan, $v_i(t)$, didapatkan dari persamaan berikut ini:

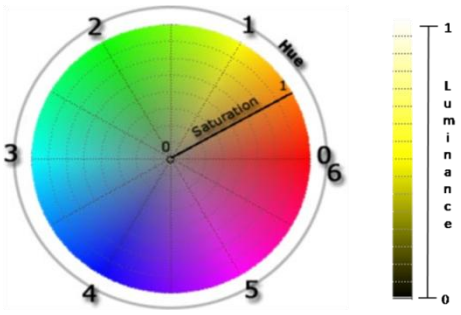
$$v_i(t+1) = \chi(v_i(t) + c_1 Rand()(p_l(t) - xit + c2Rand()(pgt - xi(t))) \quad (2)$$

χ merupakan faktor kontriksi yang mempunyai rentang nilai sebesar (0,1). $v_i(t)$ dan $x_i(t)$ masing-masing merupakan vektor kecepatan dan vektor posisi partikel ke-i (i = 1,2,3,...) pada waktu iterasi ke-t (t = 1,2,3,...). c_1 dan c_2 masing-masing merupakan faktor akselerasi yang mengatur distribusi dari pengaruh komponen kognitif dan sosial. $p_l(t)$ merupakan nilai vektor posisi optimum yang berhasil ditemukan oleh suatu partikel dari awal iterasi hingga iterasi ke-t dan biasa disebut sebagai *local best*. *Local best* merupakan komponen kognitif PSO. $p_g(t)$

merupakan nilai vektor posisi optimum yang berhasil ditemukan oleh sekumpulan partikel tersebut dari awal iterasi hingga iterasi ke-*t* dan biasa disebut sebagai *global best*. *Global best* merupakan komponen sosial PSO. $Rand()$ merupakan fungsi acak yang menghasilkan nilai antara 0 dan 1.

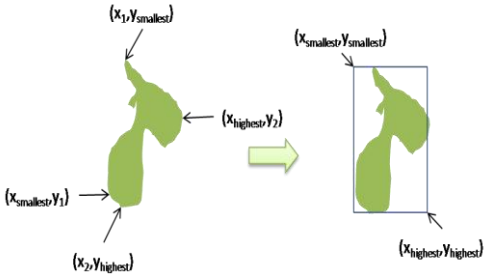
Berdasarkan Persamaan 1 dan 2, kita melihat bahwa algoritma PSO membutuhkan nilai posisi robot yang absolut untuk merepresentasikan vektor posisinya. Vektor-vektor posisi yang harus direpresentasikan dalam posisi absolut antara lain $(x_i(t), y_i(t))$ yang merupakan vektor posisi robot sekarang ini berada, *local best*, dan *global best*. Posisi berada dalam suatu sistem koordinat di mana setiap robot harus mengetahui sistem koordinat tersebut. Misalkan saja, titik asal (0,0) berada pada sudut kiri atas ruang pencarian dan sumbu-*X* merupakan panjang dari ruang pencarian, sumbu-*Y* merupakan lebar dari ruang pencarian.

Color Filtering adalah suatu teknik pengolahan citra yang yang dipakai untuk memanipulasi suatu citra berdasarkan warna spesifik [11]. Algoritma ini sudah banyak diimplementasikan oleh banyak orang. Salah satunya pada AForge.NET Framework [12-17]. Cara kerjanya adalah dengan membandingkan komponen warna setiap *piksel* citra dengan warna spesifik. Apabila warnanya sesuai dengan warna spesifik komponen warna *piksel* tersebut maka dibiarkan saja. Namun, bila warnanya tidak sesuai dengan warna spesifik maka komponen warna *piksel* tersebut diubah menjadi warna *background*, biasanya menjadi warna hitam.



Gambar 1. Ruang warna HSL.

Warna yang digunakan dalam *Color Filtering* dapat direpresentasikan dalam berbagai ruang warna. Ada beberapa ruang warna yang dikenal, antara lain RGB (*Red*, *Green*, *Blue*), HSL (*Hue*, *Saturation*, *Luminance*), YCbCr, dan sebagainya. Dalam penelitian ini, peneliti menggunakan ruang warna HSL (gambar 1) [18]. Hal ini disebabkan HSL sangat cocok untuk mengidentifikasi warna-warna dasar, di mana warna dasar ini digunakan dalam penelitian sebagai warna identifikasi robot. Selain itu, HSL menoleransi terhadap perubahan intensitas cahaya. Inilah yang menjadi keunggulan HSL dibandingkan dengan ruang warna lainnya. Representasi suatu warna dalam ruang warna HSL juga tergolong mudah.



Gambar 2. Segi empat pada *blob*.

Blobs Filtering adalah suatu teknik pengolahan citra yang digunakan untuk mengenali suatu objek secara utuh [19]. Teknik ini dipakai setelah citra dimanipulasi dengan menggunakan teknik *Color Filtering*. Cara kerjanya adalah dengan mengumpulkan *piksel-piksel* yang berwarna sama menjadi suatu peta (*map*). Warna yang dipakai adalah warna-warna selain warna *background*. Untuk setiap *piksel* berwarna sama yang berada dekat dengan kelompok piksel berwarna sama lainnya, maka ia akan dimasukkan ke dalam kelompok tersebut. Namun, apabila tidak ada kelompok yang dekat dengan piksel tersebut, maka ia akan dimasukkan ke dalam kelompok yang baru. Dengan kata lain, akan dibuat peta (*map*) baru yang memuat dirinya. *Map* di sini sering disebut dengan istilah *blob*.

Blobs Filtering sudah banyak diimplementasikan dan salah satunya pada AForge.NET *framework*. Pada *framework* ini, masing-masing *blob* mempunyai properti segi empat. Segi empat ini dibentuk dari dua titik (gambar 2). Titik pertama, yaitu titik sudut kiri atas, yang dibentuk dari nilai *x* terkecil ($x_{smallest}$) dan *y* terkecil ($y_{smallest}$) dari sekumpulan titik(x_i,y_j). Titik kedua, yaitu titik sudut kanan bawah, yang dibentuk dari nilai *x* terbesar ($x_{highest}$) dan *y* terbesar ($y_{highest}$) dari sekumpulan titik (x_i,y_j).

2. Metodologi

Robot yang digunakan dalam penelitian ini adalah Robot Al-Fath [20][21]. Setiap robot diberikan tanda satu sama lainnya (tabel I). Tanda berupa dua buah lingkaran dengan warna yang berbeda satu sama lain (gambar 3). Warna yang

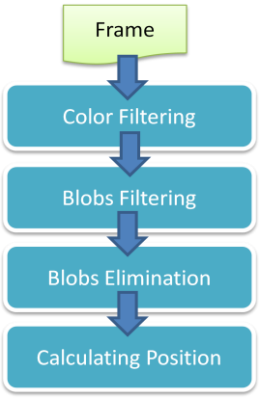
digunakan adalah warna merah, hijau, dan biru. Tanda ini ditempatkan pada bagian atas robot.

TABELI
TABEL TANDA PENGENAL PADA ROBOT

ID robot	Tanda
1	Biru – Hijau
2	Merah – Biru
3	Hijau – Merah

Gambar 3. Tanda pengenal pada robot 2.

Metode *robot tracking* menggunakan kombinasi teknik pengolahan citra di atas, yaitu *Color Filtering* dan *Blobs Filtering*. Umumnya pada metode ini, ada empat buah tahapan proses yang harus dilalui dalam setiap kali pengolahan suatu citra, yaitu *Color Filtering*, *Blobs Filtering*, *Blobs Elimination*, dan *Calculating Position*. Awalnya, setiap *frame* yang dihasilkan dari kamera diolah dengan menggunakan teknik *Color Filtering* (gambar 4). Oleh karena dalam penelitian ini digunakan tiga buah warna dasar (merah, hijau, dan biru), maka setiap *frame* melalui *Color Filtering* sebanyak tiga kali, yaitu *Color Filtering* untuk warna merah, *Color Filtering* untuk warna hijau, dan *Color Filtering* untuk warna biru.

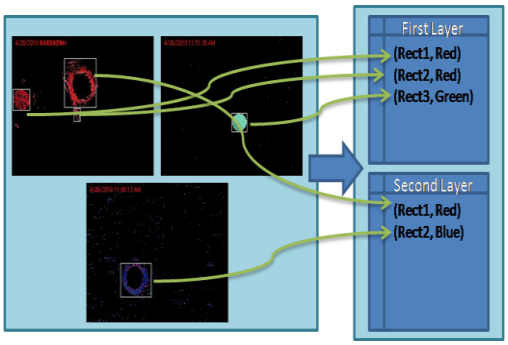


Gambar 4. Metode robot *tracking.*

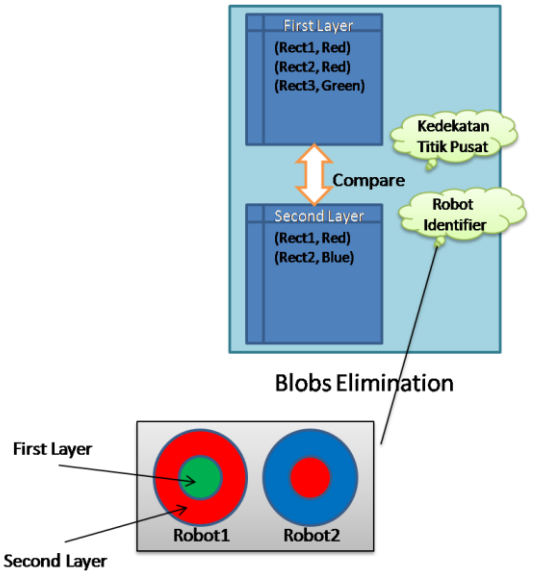
Setelah itu, hasil citra dari setiap *Color Filtering* diolah dengan teknik *Blobs Filtering* (gambar 5). Objek-objek (*blob*) yang ditemukan pada *Blobs Filtering* kemudian dipetakan ke dalam dua kategori, yaitu *First Layer* dan *Second Layer*. Kategori *First Layer* merepresentasikan objek yang dicurigai sebagai lingkaran kecil dari tanda robot. Kategori *Second Layer* merepresentasikan objek yang dicurigai sebagai lingkaran besar dari tanda robot. Parameter yang digunakan untuk membedakan kedua kategori tersebut adalah lebar dan tinggi *blob*. Jika lebar dan tinggi *blob* berada di bawah ambang batas nilai yang ditentukan, maka *blob* tersebut masuk ke dalam kategori *First Layer*. Namun, jika salah satu dari besaran lebar atau tinggi *blob* berada di atas ambang batas nilai yang ditentukan, maka *blob* tersebut masuk ke dalam kategori *Second Layer*.

Setelah *blob-blob* yang ditemukan dipetakan ke dalam kedua kategori, selanjutnya masing-masing *blob* dicarikan pasangannya antara kategori satu dengan kategori lainnya. *Blob* yang tidak mempunyai pasangan nantinya akan dieliminasi. Inilah yang dinamakan dengan proses *Blobs Elimination* (gambar 6). Pembentukan pasangan ini ditentukan oleh dua syarat. Syarat pertama adalah kedekatan titik pusat. Setiap *blob* pada kategori *First Layer* diukur kedekatan titik pusatnya dengan setiap *blob* pada kategori *Second Layer*. Jika titik pusat kedua *blob* tersebut berdekatan, maka mereka direkomendasikan sebagai pasangan. Syarat kedua adalah kesamaan warna dengan tanda robot. Pasangan *blob* yang direkomendasikan atau dengan kata lain telah memenuhi syarat pertama, kemudian dicocokkan warnanya dengan tanda robot.

Blob yang berasal dari kategori *First Layer* dicocokkan warnanya dengan lingkaran kecil pada tanda robot (*First Layer*). *Blob* yang berasal dari kategori *Second Layer* dicocokkan warnanya dengan lingkaran besar pada tanda robot (*Second Layer*). Jika pasangan *blob* tersebut memiliki warna yang sama dengan tanda robot, maka pasangan ini resmi dijadikan sebagai pasangan. Dengan kata lain, objek robot yang memiliki warna tanda yang bersangkutan direpresentasikan oleh pasangan *blob* ini. Dari proses ini kita telah dapat membedakan robot satu sama lainnya.



Gambar 5. *Blobs filtering* pada robot *tracking*.



Gambar 6. *Blobs elimination* pada robot *tracking*.

Langkah selanjutnya adalah menentukan posisi robot atau dalam algoritma ini dinamakan sebagai *Calculating Position*. Posisi suatu robot dapat didekati dengan titik pusat pasangan *blob* yang merepresentasikannya. Misalkan (x_{pusat}, y_{pusat}) adalah titik pusat dari pasangan *blob* dalam satuan *piksel*, maka untuk mendapatkan posisi absolut robot di lapangan, (x_{abs}, y_{abs}), adalah sebagai berikut:

$$\begin{pmatrix} x_{abs} \\ y_{abs} \end{pmatrix} = S \times \begin{pmatrix} x_{pusat} \\ y_{pusat} \end{pmatrix} \quad (3)$$

$$S = \frac{panjang\ \ lapangan}{lebar\ \ frame} \quad (4)$$

Persamaan di atas hanya berlaku pada kasus kamera tunggal saja. Dengan kata lain, persamaan tersebut perlu dimodifikasi agar dapat berlaku pada kasus banyak kamera. Misalkan kamera ditempatkan seperti pada gambar 5, maka posisi robot (*x*,*y*) dapat ditentukan sebagai berikut:

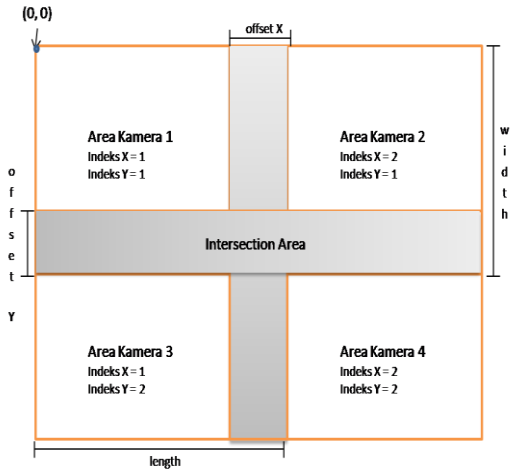
$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x_{abs} \\ y_{abs} \end{pmatrix} + \begin{pmatrix} (length - offset\ X) \times Indeks\ X \\ (width - offset\ Y) \times Indeks\ Y \end{pmatrix} \quad (5)$$

Namun, sepertinya sangat sulit untuk menempatkan kamera seperti pada gambar 7. Pasti terjadi pergeseran posisi kamera dari posisi yang diinginkan, entah itu bergeser ke arah kanan, ke arah kiri, ke arah atas, atau ke arah bawah. Oleh karena itu, persamaan 5 perlu disempurnakan lagi agar dapat mengakomodasi pergeseran posisi kamera dari posisi kamera yang diinginkan. Pergeseran ini dapat diukur sebagai nilai *error* dengan ε_X adalah *error* posisi kamera pada sumbu-*X* (horizontal) dan ε_Y adalah *error* posisi kamera pada sumbu-*Y* (vertikal). Dengan begitu, persamaannya menjadi sebagai berikut:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x_{abs} \\ y_{abs} \end{pmatrix} + \begin{pmatrix} (length - offset\ X) \times Indeks\ X \\ (width - offset\ Y) \times Indeks\ Y \end{pmatrix} - \begin{pmatrix} \varepsilon_X \\ \varepsilon_Y \end{pmatrix} \quad (6)$$

Ruang pencarian sumber kebocoran gas seluas 488 cm × 488 cm. Permukaan lantai langsung berupa lantai keramik. Di dalam ruang ini, objek yang ada hanyalah berupa robot. Kamera yang dibutuhkan untuk mengawasi seluruh daerah ruang pencarian sebanyak 12 kamera (gambar 8). Kamera yang digunakan berjenis *Web Cam*.

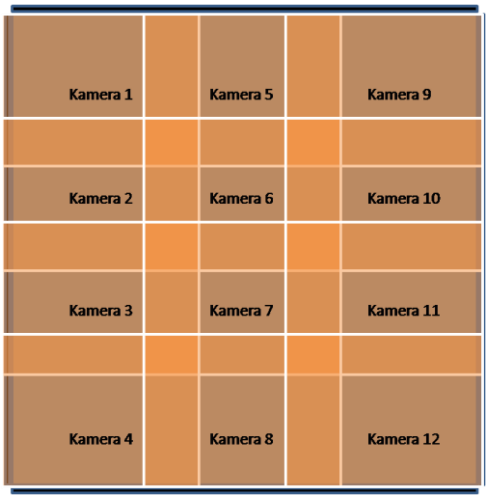
Skenario percobaan dilakukan dalam rangka untuk menguji kemampuan mengenali robot, kemampuan untuk membedakan robot satu dengan yang lainnya, dan akurasi dari posisi robot yang telah didapatkan. Skenario 1 menggunakan robot tunggal sebagai aktornya. Robot akan diperintahkan untuk berjalan lurus searah sumbu-*X* atau dengan kata lain robot akan berjalan menelusuri panjang dari lapangan.



Gambar 7. Contoh penempatan kamera.

Skenario 2 hampir sama dengan skenario 1. Skenario ini masih menggunakan robot tunggal sebagai aktornya. Robot akan diperintahkan berjalan lurus searah sumbu-*Y* atau dengan kata lain robot akan berjalan menelusuri lebar dari lapangan.

Berbeda dengan skenario 1 dan skenario 2, skenario 3 sudah menggunakan ketiga robot sebagai aktornya. Ketiga robot tersebut akan diperintahkan untuk berjalan secara acak di lapangan.



Gambar 8. Penempatan kamera pada ruang pencarian.

3. Hasil dan Pembahasan

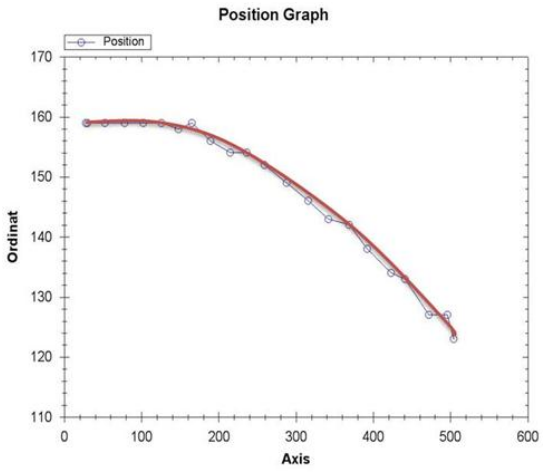
Skenario 1 dapat kita lihat pada gambar 9, grafik posisi yang terbentuk sudah hampir menyerupai grafik posisi yang sebenarnya. Robot berjalan dari posisi awal di titik (28, 159) dan berhenti di titik (504, 123). Dari kedua titik ini, besar pergeseran robot pada sumbu-Y dari awal hingga akhir yang terukur adalah sebesar 36 cm. Pada kenyataannya di lapangan, robot bergeser sekitar 30 cm dari awal hingga akhir. Berarti, *error* posisi yang tercipta ketika robot menelusuri panjang lapangan maksimal 6 cm. Selain itu, apa yang dapat kita lihat pada gambar 7 adalah kemampuan dalam pengenalan robot. Metode Robot *Tracking* yang diajukan dinilai cukup berhasil untuk mengenali robot. Hal ini dapat dilihat pada grafik bahwa tidak ada nilai posisi robot yang sebesar (-1, -1). Posisi (-1, -1) merupakan posisi yang sengaja dibuat oleh peneliti untuk menandakan bahwa robot yang dicari tidak dapat ditemukan.

TABELII
TABEL TINGKAT PENGENALAN ROBOT PADA SKENARIO 1

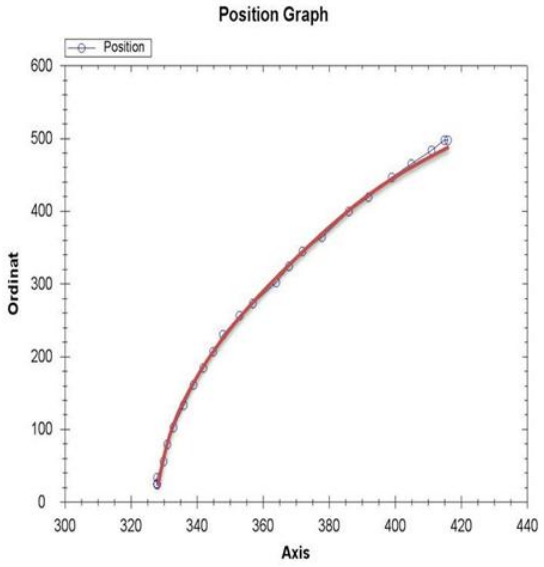
Kamera	Citra	Sukses	Gagal	Salah
1	100	82	16	2
2	100	100	0	0
5	100	91	4	5
6	100	98	2	0
7	100	100	0	0
8	100	100	0	0
9	100	94	5	1
10	100	96	4	0
11	100	100	0	0
12	100	100	0	0
Rata-rata (%)	100	96.1	3.1	0.8

Tingkat pengenalan robot lebih rincinya ditampilkan pada tabel II. Jumlah citra yang diambil sebagai sampel analisis adalah sebanyak 1000 buah yang berasal dari 10 buah kamera dengan masing-masing sebanyak 100 buah. Citra ini merupakan citra yang berhasil ditangkap oleh kamera. Citra ini kemudian dimanipulasi oleh program yang telah dibuat sedemikian hingga objek robot yang berhasil dikenali ditandai dalam citra ini. Kolom 'Sukses' dalam tabel ini menunjukkan jumlah citra di mana program berhasil mengenali robot di dalamnya dengan benar. Kolom 'Gagal' menunjukkan jumlah citra di mana program gagal dalam mengenali robot di dalamnya. Kolom 'Salah' menunjukkan jumlah citra di mana program salah dalam mengenali robot di dalamnya, seperti *noise* yang dianggap sebagai robot. Seperti yang terlihat pada tabel tersebut, tingkat pengenalan robot sangat tinggi yaitu sebesar 96.1 persen.

Skenario 2 dapat kita lihat pada gambar 10, grafik posisi yang terbentuk sudah hampir menyerupai grafik posisi yang sebenarnya. Robot berjalan dari posisi awal di titik (328, 24) dan berhenti di titik (416, 497). Dari kedua titik ini, besar pergeseran robot pada sumbu-*X* dari awal hingga akhir yang terukur adalah sebesar 88 cm. Pada kenyataannya di lapangan, robot bergeser sekitar 83 cm dari awal hingga akhir. Berarti, *error* posisi yang tercipta ketika robot menelusuri lebar lapangan maksimal 5 cm. Selain itu, apa yang dapat kita lihat pada gambar 8 adalah kemampuan dalam mengenali robot. Metode Robot *Tracking* yang diajukan, dinilai cukup berhasil untuk mengenali robot. Hal ini dapat dilihat pada grafik bahwa tidak ada nilai posisi robot yang sebesar (-1, -1). Posisi (-1, -1) merupakan posisi yang sengaja dibuat oleh peneliti untuk menandakan bahwa robot yang dicari tidak dapat ditemukan.



Gambar 9. Grafik posisi pada skenario 1.



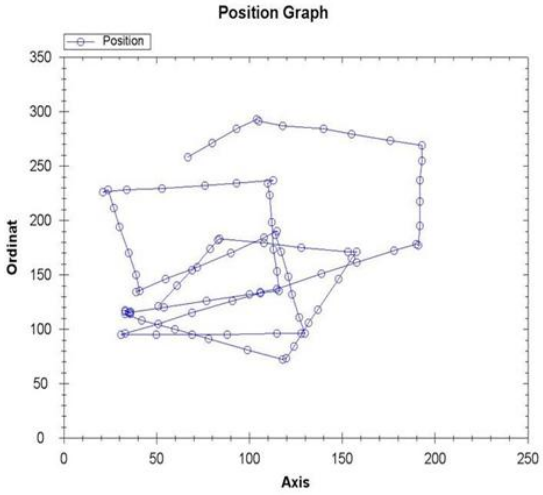
Gambar 10. Grafik posisi robot pada Skenario 2.

Tingkat pengenalan robot lebih rincinya ditampilkan pada tabel III. Jumlah citra yang diambil sebagai sampel analisis adalah sebanyak 1000 buah yang berasal dari 10 buah kamera dengan masing-masing sebanyak 100 buah. Citra ini merupakan citra yang berhasil ditangkap oleh kamera. Citra ini kemudian dimanipulasi oleh program yang telah dibuat sedemikian hingga objek robot yang berhasil dikenali ditandai dalam citra ini. Kolom 'Sukses' dalam tabel ini menunjukkan jumlah citra di mana program berhasil mengenali robot di dalamnya dengan benar. Kolom 'Gagal' menunjukkan jumlah citra di mana program gagal dalam mengenali robot di dalamnya. Kolom 'Salah' menunjukkan jumlah citra di mana program salah dalam mengenali robot di dalamnya, seperti *noise* yang dianggap sebagai robot. Seperti kita lihat pada tabel tersebut, tingkat pengenalan robot cukup tinggi yaitu sebesar 91.4 %.

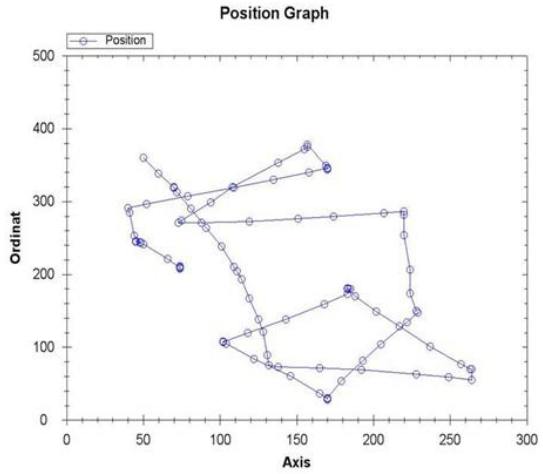
TABEL III
TABEL TINGKAT PENGENALAN ROBOT PADA SKENARIO 2

Kamera	Citra	Sukses	Gagal	Salah
1	100	100	0	0
2	100	100	0	0
5	100	22	77	1
6	100	100	0	0
7	100	98	2	0
8	100	100	0	0
9	100	95	5	0
10	100	100	0	0
11	100	100	0	0
12	100	99	0	1
Rata-rata (%)	100	91.4	8.4	0.2

Skenario 3 terlihat pada gambar 11 yaitu Robot 1 (Biru-Hijau) dapat dikenali dengan baik oleh program yang telah dikembangkan. Robot 1 selalu dapat diketahui posisinya dimanapun ia berada. Robot 1 berjalan dari posisi awal (67, 258) dan berakhir di posisi (51, 121). Begitu pula pada gambar 12, Robot 2 (Merah-Biru) dapat dikenali dengan baik oleh program yang telah dikembangkan. Robot 2 selalu dapat diketahui posisinya dimanapun ia berada. Robot 2 berjalan dari posisi awal (50, 360) dan berakhir di posisi (74, 211). Namun pada gambar 13, Robot 3 (Hijau-Merah) hampir dapat dikenali dengan baik oleh program yang telah dikembangkan karena pernah ada keadaan di mana posisinya sebesar (-1, -1). Robot 3 hampir selalu dapat diketahui posisinya dimanapun ia berada. Robot 3 berjalan dari posisi awal (29, 57) dan berakhir di posisi (84, 168).



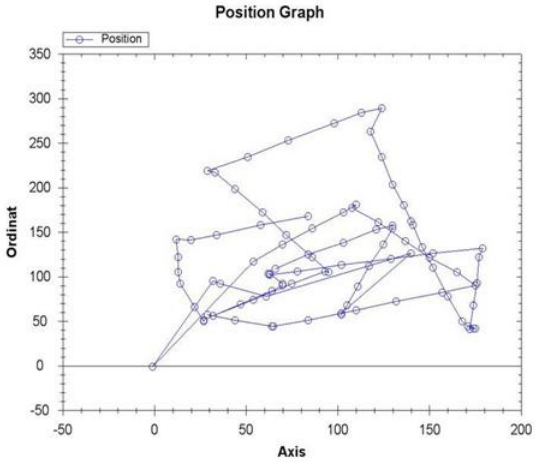
Gambar 11. Grafik posisi robot 1 pada Skenario 3.



Gambar 12. Grafik posisi robot 2 pada skenario 3.

Tingkat pengenalan robot lebih rincinya ditampilkan pada tabel IV. Jumlah citra yang

diambil sebagai sampel analisis adalah sebanyak 1200 buah yang berasal dari 12 buah kamera dengan masing-masing sebanyak 100 buah. Citra ini merupakan citra yang berhasil ditangkap oleh kamera. Citra ini kemudian dimanipulasi oleh program yang telah dibuat sedemikian hingga objek robot yang berhasil dikenali ditandai dalam citra ini.



Gambar 13. Grafik posisi robot 3 pada skenario 3.

Kolom 'Sukses' dalam tabel IV menunjukkan jumlah citra di mana program berhasil mengenali robot di dalamnya dengan benar. Kolom 'Gagal' menunjukkan jumlah citra di mana program gagal dalam mengenali robot di dalamnya. Kolom 'Salah' menunjukkan jumlah citra di mana program salah dalam mengenali robot di dalamnya, seperti *noise* yang dianggap sebagai robot. Seperti kita lihat pada tabel tersebut, tingkat pengenalan robot cukup tinggi yaitu sebesar 95%. Tingkat ke-'Gagal'-an hanya sebesar 5.75% sedangkan tingkat ke-'Salah'-an sangat kecil sekali, tidak sampai dengan 1%.

TABEL IV
TABEL TINGKAT PENGENALAN ROBOT PADA SKENARIO 3

Kamera	Citra	Sukses	Gagal	Salah
1	100	98	12	0
2	100	97	3	0
3	100	97	3	0
4	100	100	0	0
5	100	82	18	0
6	100	86	13	1
7	100	80	20	0
8	100	100	0	0
9	100	100	0	0
10	100	100	0	0
11	100	100	0	0
12	100	100	0	0
Rata-rata (%)	100	95	5.75	0.083

Berdasarkan ketiga skenario yang telah diujicobakan, ada beberapa hal yang akan penulis analisis terhadap metode Robot *Tracking* yang telah diajukan. Hal-hal tersebut antara lain tingkat akurasi posisi robot dan tingkat pengenalan robot.

Pada tingkat akurasi posisi, berdasarkan hasil skenario 1 dan 2 (gambar 9 dan 10), peneliti melihat bahwa *error* posisi yang tercipta tidak terlalu buruk. *Error*-nya maksimal sebesar 5 - 6 cm dari posisi yang sebenarnya. Ada beberapa faktor yang menyebabkan *error* seperti ini masih terjadi, antara lain pengukuran nilai variabel $\varepsilon_X$dan $\varepsilon_Y$yang masih belum tepat dan belum sempurnanya rumus posisi yang digunakan. Nilai $\varepsilon_X$dan $\varepsilon_Y$yang dipakai dirasa masih belum tepat. Hal ini disebabkan peneliti hanya mengambil empat buah titik sampel di daerah irisan untuk menghitung kedua nilai *error* tersebut. Jumlah titik ini peneliti rasa masih kurang untuk menghitung nilai *error*.

Persamaan 6 yang digunakan sebagai rumus posisi juga dirasakan belum sempurna. Seperti pada dugaan awal, rumus ini masih belum bisa mengatasi masalah penempatan kamera yang tidak sejajar dengan arah sumbu-*X* dan sumbu-*Y* yang telah ditetapkan. Rumus ini hanya mengatasi masalah pergeseran bidang kamera, sedangkan perputaran bidang kamera belum dapat diatasi. Pada penelitian ini, seluruh kamera tidak dapat ditempatkan dengan sempurna. Hal ini disebabkan oleh sulitnya penempatan kamera secara sempurna, yaitu sesuai dengan perencanaan. Dengan demikian, rumus posisi yang digunakan sudah tidak akurat lagi.

Selain kedua hal di atas, masih ada satu faktor yang menyebabkan *error* posisi. Faktor tersebut terdapat pada kameranya sendiri. Setiap kamera pastinya mempunyai titik fokus, di mana objek yang berada tepat di titik fokus pasti akan mempunyai citra yang sama dengan dirinya. Namun ketika posisi objek tersebut sudah tidak berada di titik fokus, maka citra objek yang dihasilkan agak sedikit berbeda. Begitu pula halnya dengan robot. Bagian tanda pengenal robot akan terlihat tidak berada pada posisi sebenarnya ketika robot menjauhi titik fokus kamera. Tentu saja hal ini akan memengaruhi akurasi posisi robot, di mana dalam penelitian ini posisi robot ditentukan berdasarkan posisi tanda pengenalnya.

Pada tingkat pengenalan robot, berdasarkan skenario 1, 2, dan 3, kita dapat melihat bahwa program yang telah dikembangkan dapat mengenali robot dengan baik, baik di lapangan itu hanya ada robot tunggal maupun banyak robot. Walaupun pada Skenario 3, robot sesekali tidak terdeteksi posisinya oleh kamera. Hal ini ditunjukkan dengan posisi (-1, -1) pada grafik posisi.

Secara rata-rata, tingkat pengenalan robot mencapai hingga 94.17%. Daerah yang paling

rawan untuk tidak terdeteksinya robot adalah daerah pinggir lapangan, di mana pada daerah tersebut terdapat bayangan dari dinding lapangan. Hal ini disebabkan kombinasi warna *filter* yang dipakai masih belum baik. Kombinasi warna *filter* yang digunakan belum bisa mengakomodasi pengenalan robot pada daerah bayangan.

Selain itu jika dilihat lebih dalam, ada kalanya robot ini tidak dapat terdeteksi oleh kamera. Biasanya hal ini terjadi ketika robot baru memasuki daerah kamera yang bersangkutan. Dugaan awal peneliti mengarah ke konfigurasi warna *filter* yang masih belum tepat. Namun terasa aneh ketika robot tersebut didiamkan pada posisi tersebut, kamera dapat mendeteksi keberadaan robot tersebut, walaupun terkadang robot juga sempat tidak terdeteksi dalam beberapa *milisecond.*

Bagaimanapun juga, kasus di atas tidak akan membawa dampak dalam masalah pencarian posisi robot. Hal ini disebabkan kasus tersebut terjadi di daerah irisan, di mana ada kamera lain yang juga memantau daerah tersebut. Selama uji coba, robot selalu dapat terdeteksi di daerah irisan, baik itu oleh satu kamera atau dua kamera. Dengan demikian, posisi robot selalu dapat diketahui pada kasus ini.

4. Kesimpulan

Berdasarkan hasil eksperimen yang telah dilakukan, kita dapat melihat bahwa tujuan penelitian ini telah tercapai. Kita dapat mengetahui posisi robot dalam ruang pencarian melalui perangkat kamera. Selainitu, robot juga dapat mengetahui posisinya berada pada saat tersebut dengan cara mengakses program yang telah dikembangkan.

Adapun permasalahan penelitian yang telah dituliskan oleh peneliti pada bagian sebelumnya telah terjawab. Dimulai dari permasalahan bagaimana cara mengidentifikasi objek robot melalui kamera. Hal ini terjawab dengan cara menambahkan tanda pengenal pada robot. Lalu, peneliti mengembangkan suatu metode untuk mengenali objek robot tersebut. Hasil dari eksperimen yang telah dilakukan adalah robot dapat dikenali dengan menggunakan metode tersebut dengan baik walaupun terkadang di beberapa daerah, robot sulit untuk dikenali. Tingkat pengenalan robot hingga mencapai 94.17%.

Kemudian berlanjut ke permasalahan bagaimana cara membedakan objek robot tersebut satu sama lainnya. Hal ini dilakukan dengan memberikan kombinasi warna yang berbeda pada masing-masing tanda pengenal robot. Hasilnya berdasarkan eksperimen yang telah dilakukan, robot dapat dibedakan satu sama lainnya dengan baik dengan menggunakan metode yang diajukan. Dari metode yang dibuat, jumlah robot yang dibedakan adalah sebanyak $N(N-1)^{n-1}$ buah, di mana$N$ adalah jumlah warna yang digunakan dan n adalah jumlah lapisan *layer* yang dipakai. Jadi, program yang dikembangkan dalam penelitian ini dapat membedakan robot sampai enam buah karena menggunakan tiga buah warna dan dua lapisan pengenal.

Permasalahan bagaimana menentukan posisi absolut dari suatu robot juga terjawab. Berdasarkan hasil eksperimen yang telah dilakukan, posisi robot mempunyai kesalahan sebesar 5 - 6 cm. Penelitian ini berhasil menjawab permasalahan penentuan posisi robot melalui banyak kamera. Posisi robot dihitung dengan menggunakan persamaan 6.

Namun di atas semua itu, metode yang dibuat sangat sulit untuk mendapatkan hasil yang optimal. Hal ini disebabkan banyaknya parameter yang harus diukur dengan baik, seperti konfigurasi warna *filter*, *error* kamera, dan sebagainya. Penentuan nilai parameter yang baik akan menjamin hasil yang baik pula. Namun sayangnya hal tersebut sangat memakan waktu yang mungkin cukup lama.

Referensi

[1] W. Jatmiko, K. Sekiyama, & T. Fukuda, "A PSO-Based Mobile Robot for Odor Source Localization in Dynamic Advection-Diffusion with Obstacles Environment: Theory Simulation and Measurement," *IEEE Computational Intelligence Magazine*, vol. 2, pp. 37 – 51, 2007.

[2] A. Nugraha, "Modifikasi Particle Swarm Optimization untuk Pencarian Banyak Sumber Asap dengan Visualisasi," B.S Thesis, Department of Computer Science, Faculty of Computer Science, Universitas Indonesia, Indonesia, 2008.

[3] W. Pambuko, "Modifikasi Particle Swarm Optimization untuk pencarian banyak sumber asap dengan Open Dynamic Engine," Ph.D Thesis, Department of Computer Science Master, Faculty of Computer Science, Universitas Indonesia, Indonesia, 2009.

[4] A. Febrian, "Implementasi Model Robot Al-Fath, Pembatasan Gerak Robot Bermuatan, dan Main Spread 2 pada Perangkat Simulasi Robot Pencari Sumber Asap," Ph.D Thesis, Department of Computer Science Master,

Faculty of Computer Science, Universitas Indonesia, Indonesia, 2009.

[5] Maps Gps Info, GPS Accuracy - How Accurate is it?, Maps Gps Info, http://www.maps-gps-info.com/gps accuracy.html, retrieved March 24, 2010.

[6] R.C. Eberhart and J. Kennedy, "A New Optimizer Using Particle Swarm Theory" *In Proceeding ofthe 6th International Sysmposium on Micromachine and Human Science*, pp. 39-43, 1995.

[7] R.C. Eberhart and J. Kennedy, "Particle Swarm Optimization" *In Proceeding ofthe IEEE International Joint Conference on Neural Networks*, pp. 1942-1948, 1995.

[8] A.P. Engelbrecht, *Fundamentals of Computational Swarm Intelligence*, Wiley, Hoboken, NJ, 2005.

[9] R. Efendi, "Modifikasi Algoritma Particle Swarm Optimization untuk Pencarian Multi Sumber Gas: Simulasi dan Analisis," Ph.D Thesis, Department of Computer Science Master, Faculty of Computer Science, Universitas Indonesia, Indonesia, 2008.

[10] A. Nugraha, W. Jatmiko, J. Perkasa, "Modifikasi Particle Swarm Optimization untuk Pencarian Banyak Sumber Gas," *Jurnal Ilmu Komputer dan Informasi*, vol. 1, pp.1-8, 2008.

[11] Answer Corporation, Color Filter, Answer Corporation, http://www.answers.com/topic/Color-filter-, retrieved March 24, 2010.

[12] AForge.NET, AForge.NET Framework, AForge.NET, http://www.aforgenet.com/framework/, retrieved February 10, 2010.

[13] AForge.NET, Framework Samples - Image Processing, AForge.NET, http://www.aforgenet.com/framework/samples/image processing.html, retrieved March 24, 2010.

[14] MSDN, Microsoft Visual Studio, Microsoft, http://msdn.microsoft.com/en-us/vstudio/default.aspx, retrieved February 10, 2010.

[15] AForge.NET, Framework Documentation, AForge.NET, http://www.aforgenet.com/framework/docs/, retrieved March 24, 2010.

[16] AForge.NET, Framework Samples – Video, AForge.NET, http://www.aforgenet.com/framework/samples/video.html, retrieved March 24, 2010.

[17] MSDN, MSDN Library, Microsoft, http://msdn.microsoft.com/en-us/library/default.aspx, retrieved March 24, 2010.

[18] ChaosPro, HSL*Color*space, ChaosPro, http://www.chaospro.de/documentation/html/paletteeditor/Colorspacehsl.htm, retrieved March 24, 2010.

[19] RoboRealm, Blob Filter, RoboRealm, http://www.roborealm.com/help/Blob Filter.php, retrieved March 24, 2010.

[20] D. Vektor and R. Prasetya, *Hardware Documentation of Track Mobile Robot Alfathvrss*, Depok, 2010.

[21] Tim Robotika Fakultas Teknik dan Lab Robotika Fakultas Ilmu Komputer Universitas Indonesia, Hardware Prototype untuk Pengembangan Swarm Robot untuk Mendeteksi Kebocoran Gas dan Mendeteksi Bom di Gedung/pabrik di Indonesia, Faculty of Computer Science, Universitas Indonesia, Indonesia, 2009.