p-ISSN 2301–4156 | e-ISSN 2460–5719

Peningkatan Akurasi *Adaptive Monte Carlo Localization* Menggunakan *Convolutional Neural Network*

Riza Agung Firmansyah[1], Tri Arief Sardjono[2], Ronny Mardiyanto[3]

[1,2,3] Departemen Teknik Elektro Fakultas Teknologi Elektro dan Informatika Cerdas Institut Teknologi Sepuluh Nopember, Surabaya, 60111 INDONESIA (tel.: 031-5994251-54, ext. 1206 / 031-5947302; fax: 031-5931237, email: [1]7022211015@mhs.its.ac.id, [2]sardjono@bme.its.ac.id, [3]ronny@elect-eng.its.ac.id)

[Diterima: 30 Maret 2023, Direvisi: 14 Juni 2023]
Corresponding Author: Ronny Mardiyanto

INTISARI — Makalah ini menjelaskan peningkatan akurasi sistem lokalisasi *adaptive Monte Carlo localization* (AMCL) pada robot menggunakan *convolutional neural network* (CNN). Sistem lokalisasi pada sebuah robot merupakan proses pengenalan posisi suatu robot dalam lingkungan kerjanya. Sistem ini penting karena memungkinkan robot untuk melakukan navigasi dan pemetaan (*mapping*) secara efisien dan akurat. Tanpa lokalisasi yang tepat, robot tidak dapat beroperasi dengan efektif dan dapat mengalami masalah, seperti kehilangan arah atau menabrak objek. AMCL merupakan sistem lokalisasi yang populer dan banyak digunakan dalam robot. Metode ini menggunakan perubahan posisi robot dan perubahan pembacaan sensor *light detection and ranging* (LiDAR) sebagai masukannya. Pembacaan perubahan posisi robot rentan terhadap kesalahan akibat slip maupun deformasi roda. Ketidaktepatan pembacaan perubahan posisi robot menyebabkan prediksi posisi robot oleh AMCL menjadi kurang tepat, sehingga diperlukan sebuah perbaikan. Pada makalah ini, kebaruan yang diangkat adalah memberikan nilai kompensasi dari hasil AMCL, sehingga *error*-nya menjadi kecil. Nilai kompensasi tersebut didapatkan dari hasil pelatihan CNN, sehingga metode yang diusulkan tersebut dapat disebut AMCL+CNN. Masukan yang diberikan pada CNN adalah perubahan nilai *odometry* roda dan perubahan pembacaan jarak oleh sensor LiDAR. Keluaran CNN tersebut dibandingkan dengan data target berupa posisi aktual robot dari hasil pengamatan. Pelatihan jaringan dilakukan sebanyak 200 *epoch* untuk mendapatkan *validation loss* terkecil. Pengujian dilakukan pada sebuah robot yang telah dilengkapi dengan *robot operating system* (ROS). *Dataset* pelatihan dan pengujian yang diujikan berasal dari data *rosbag* saat robot berjalan di area pengujian. Hasil yang didapatkan algoritma AMCL+CNN memiliki *error* yang lebih kecil dibandingkan AMCL dan *Monte Carlo localization* (MCL) reguler pada skenario lurus dan belok. Hasil yang diperoleh juga lebih baik pada metrik *error* posisi saat dibandingkan dengan beberapa metode pembanding lain.

KATA KUNCI — Lokalisasi Robot, LiDAR, AMCL, CNN.

I. PENDAHULUAN

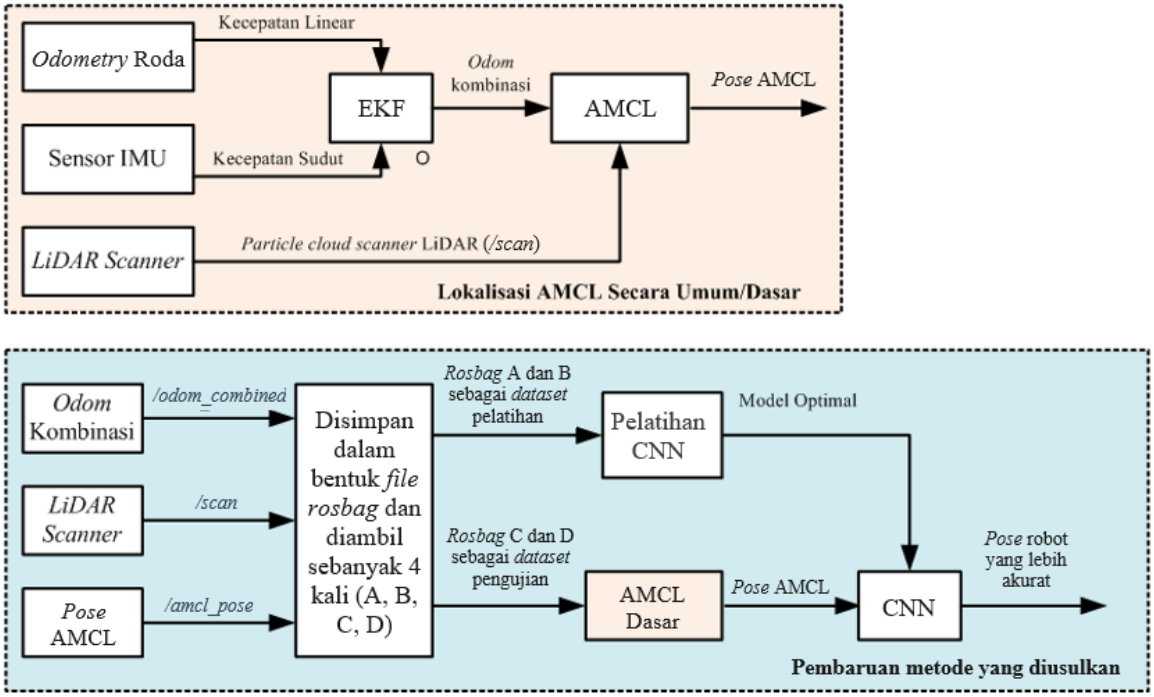
Mobile robot telah digunakan pada beberapa bidang, seperti pertanian [1], kesehatan [2], perhotelan [3], dan pariwisata [4]. Dalam melakukan pekerjaannya, robot memiliki peralatan yang terpasang, misalnya *gripper* [5], bak penampung pada robot AGV [6], dan juga alat ukur kesehatan manusia untuk mengukur tanda-tanda vital seseorang [7]–[9]. Saat melakukan pekerjaan, robot harus mengetahui posisi aktualnya agar dapat bekerja dengan benar. Jika tidak, robot tidak dapat bekerja sesuai rencana atau tugas yang diberikan sebelumnya [10]. Proses tersebut umumnya dikenal dengan sistem lokalisasi robot.

Untuk menunjang sistem lokalisasi di dalam ruangan, umumnya robot menggunakan sensor LiDAR 2D [10]–[12]. Sensor tersebut memiliki kemampuan *ranging* yang cukup bagus ke segala arah (360°). Seiring pesatnya perkembangan penelitian di bidang robotika, sensor LiDAR 3D juga banyak digunakan [13], [14]. Kemampuan *ranging* dari sensor LiDAR 3D jauh lebih baik daripada sensor LiDAR 2D, tetapi sensor LiDAR 3D memiliki harga yang cukup mahal di pasaran. Sensor berbasis visual seperti kamera RGB memiliki kemampuan untuk mengenali fitur di sekitar robot [15], [16]. Namun, kemampuan *ranging*-nya kurang bagus dibandingkan LiDAR, sehingga sensor ini masih perlu digabungkan dengan sensor lain. Selain itu, pengenalan fitur tidak dapat bekerja secara optimal pada area *featureless*. Sensor berbasis visual lain seperti kamera RGB-D memiliki kemampuan *ranging* berdasarkan kedalaman (*depth*) [17]–[19]. Namun, jangkauannya hanya terbatas di sekitar 87° pada arah horizontal.

Sistem lokalisasi merupakan salah satu tantangan penting dalam penelitian robotika karena merupakan dasar sistem navigasi, *map building*, dan lain-lain [20]. Akurasi dan efisiensi sistem lokalisasi sangat berperan terhadap kinerja robot saat menjalankan pekerjaaanya. Oleh karena itu, perlu dilakukan sebuah penelitian mengenai peningkatan akurasi pengukuran dan efisiensi. Efisiensi yang dimaksud dapat merupakan efisiensi komputasi dan efisiensi harga. Peningkatan kinerja tersebut dapat dilakukan dengan menggunakan sensor berkualitas baik maupun menggunakan gabungan beberapa sensor atau *sensor fusion* [4], [21], [22].

Salah satu metode lokalisasi yang banyak digunakan adalah metode berbasis *particle filter*, yaitu *Monte Carlo localization* (MCL). Metode ini sangat efektif dalam memperkirakan lokasi robot, tetapi ada beberapa kelemahan yang perlu diperhatikan. MCL memerlukan banyak data sensor dan informasi lingkungan yang akurat dan *up-to-date* untuk menghasilkan estimasi yang tepat. Jika data sensor tidak tersedia atau tidak akurat, hasil dari MCL akan sangat tidak memuaskan. Selanjutnya, MCL juga sangat sensitif terhadap perubahan lingkungan, seperti perubahan tata letak atau penambahan objek baru. Ini berarti bahwa algoritma harus sering diperbarui untuk memastikan hasil yang akurat. MCL juga memerlukan banyak komputasi dan waktu untuk menghitung probabilitas lokasi yang akurat, sehingga dapat menyebabkan keterlambatan robot dalam merespons. Oleh karena itu, perlu ditemukan solusi alternatif untuk mengatasi kelemahan ini dalam implementasi MCL pada robot.

p-ISSN 2301–4156 | e-ISSN 2460–5719



Gambar 1. Diagram blok lokalisasi robot yang diusulkan.

Adaptive MCL (AMCL) adalah variasi dari algoritma MCL yang digunakan untuk menentukan posisi dan arah (*heading*) robot dalam lingkungan yang tidak diketahui. AMCL menggabungkan filter Kalman dengan MCL untuk menyesuaikan distribusi partikel dan mengurangi jumlah partikel yang diperlukan untuk mendapatkan akurasi yang sama. AMCL mengubah jumlah partikel yang digunakan secara adaptif sesuai dengan kondisi lingkungan yang berubah. Hal ini memungkinkan AMCL untuk mengoptimalkan jumlah partikel yang digunakan dan meningkatkan efisiensi. Secara umum, AMCL digunakan untuk menentukan posisi dan arah robot di lingkungan yang tidak diketahui dengan lebih cepat dan akurat dibandingkan dengan MCL biasa. AMCL menggabungkan teknik MCL dengan filter Kalman untuk meningkatkan akurasi dan *robustness* dari estimasi posisi robot. Namun, kedua algoritma tersebut masih dipengaruhi oleh potensi kesalahan pada saat melakukan proses *motion update*. *Motion update* tersebut didapatkan dari pembacaan perubahan posisi robot berdasarkan data sensor *odometry* roda. Kesalahan yang umum terjadi disebabkan oleh slip dan deformasi roda.

Teknik tentang lokalisasi robot menggunakan sensor LiDAR 2D sudah pernah diteliti [23]. Teknik tersebut menggunakan lokalisasi berbasis LiDAR dengan *occupancy grid map* yang digabungkan dengan kamera. Kamera digunakan untuk membaca teks yang telah dipasang pada area kerja robot. Pada lokalisasi dalam ruangan, hasilnya cukup baik. Teknik lain yang digunakan dalam lokalisasi robot adalah *2DLaserNet* [24]. Teknik tersebut menggunakan *convolutional neural network* (CNN) untuk mengidentifikasi ruangan, koridor, dan pintu di area kerjanya. Identifikasi dilakukan hanya berdasarkan pembacaan sensor LiDAR 2D. Penggunan sensor LiDAR 2D dengan jaringan saraf juga telah dilakukan dalam [25].

Peningkatan akurasi yang lain dalam pembacaan pada AMCL dapat dilakukan menggunakan *implicit representation-based Monte Carlo localization* (IR-MCL) [26]. Teknik tersebut melakukan estimasi posisi berdasarkan data dari sensor LiDAR 2D. Lingkungan kerja robot direpresentasikan menjadi bentuk *neural occupancy field* (NOF) yang dibuat menggunakan *artificial neural network* (ANN). NOF dibuat berdasarkan masukan berupa pemindaian oleh sensor LiDAR 2D dengan keluaran berupa probabilitas kesesuaian dari *occupancy map*. Teknik tersebut mampu menghasilkan estimasi posisi dan arah robot yang lebih akurat. Sistem lokalisasi robot menggunakan *machine learning* memiliki beberapa keunggulan dibandingkan dengan metode AMCL. Sistem *machine learning* memiliki kemampuan untuk mempelajari dan menyesuaikan diri dengan lingkungan secara otomatis, sehingga dapat menghasilkan estimasi lokasi yang lebih akurat dalam lingkungan yang berubah-ubah.

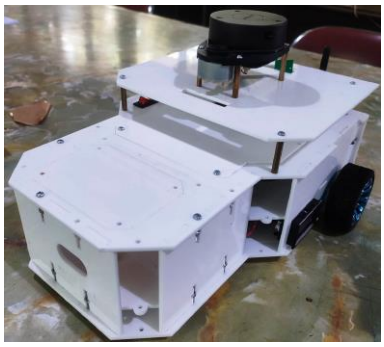
Makalah ini mengusulkan peningkatan akurasi sistem lokalisasi AMCL menggunakan CNN. Dalam metode yang diusulkan, CNN berfungsi untuk memprediksi kemungkinan kesalahan yang muncul pada posisi dan arah robot. Nilai prediksi kesalahan yang muncul akan digunakan untuk mengompensasi keluaran AMCL pada posisi sumbu x dan y, dan sumbu putar z (*yaw*). Jika posisi robot makin akurat, kinerja robot secara keseluruhan akan lebih baik. Metode yang diusulkan tersebut dijalankan pada sebuah robot yang telah dilengkapi dengan *robot operating system* (ROS).

Bagian selanjutnya dari makalah ini menjelaskan metodologi penelitian hingga hasil yang didapatkan. Pada Bagian II dibahas metodologi penelitian yang telah dilakukan, mulai dari perancangan robot, perancangan CNN, pengujian, dan metrik evaluasi. Bagian III membahas hasil dari pengujian yang telah dilakukan dan Bagian IV menyajikan kesimpulan berdasarkan hasil pengujian.

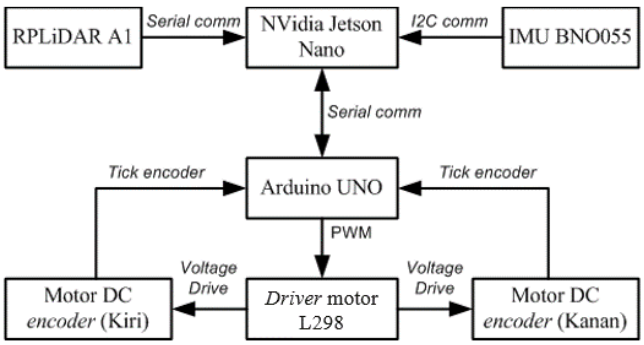
II. METODOLOGI

Dalam bagian ini, dijelaskan langkah-langkah untuk mencapai tujuan yang diinginkan. Langkah pertama adalah mempersiapkan *mobile robot* yang sudah terpasang ROS. Setelah itu, dipersiapkan *ROS node* yang menunjang sistem yang akan dibangun. Jaringan saraf tiruan dengan arsitektur CNN dibuat untuk menentukan jumlah partikel optimal berdasarkan masukan hasil pemindaian sensor LiDAR dan

p-ISSN 2301–4156 | e-ISSN 2460–5719



Gambar 2. Bentuk visual robot yang digunakan.



Gambar 3. Diagram blok koneksi antar komponen.

odometry roda. Setelah robot, *ROS framework*, dan model CNN siap, dilakukan pengumpulan *dataset* untuk proses pelatihan. *Dataset* tersebut diambil dalam bentuk *file rosbag* yang selanjutnya akan diekstrak secara *offline* untuk mendapatkan data pembacaan sensor. Proses pelatihan jaringan dilakukan berdasarkan data yang telah diekstraksi tersebut. Setelah model yang paling optimal diperoleh, dilakukan implementasi ke sistem lokalisasi robot yang telah dibuat. Nilai optimal diukur berdasarkan nilai *pose error* dan *error* arah yang lebih kecil dibandingkan metode MCL reguler dan AMCL *default* dari ROS. Proses dari metode yang diusulkan ditunjukkan pada Gambar 1. Perbedaan metode yang diusulkan dengan beberapa metode sebelumnya adalah penambahan sensor *inertial measurement unit* (IMU*)* dan *odometry* roda untuk menunjang sensor LiDAR yang digunakan [24], [25].

A. DESAIN PERANGKAT KERAS MOBILE ROBOT

Mobile robot yang digunakan dibuat dengan konfigurasi *differential drive mobile robot*. Robot dibuat dengan ukuran panjang 300 mm, lebar 200 mm, dan tinggi 100 mm. Roda memiliki diameter 65 mm, yang terpasang di bagian belakang robot. Jarak antar roda adalah 250 mm. Roda *castor* dipasang di bagian depan dengan jarak 170 mm terhadap sumbu roda belakang. Untuk menunjang sistem lokalisasi, sensor LiDAR 2D dipasang di atas *castor*. Bentuk robot yang digunakan diperlihatkan pada Gambar 2.

Robot yang dibuat memiliki beberapa peranti elektronik pendukung yang koneksi antarkomponennya ditunjukkan pada Gambar 3. NVIDIA Jetson Nano digunakan sebagai pemroses utama. Di dalamnya sudah terinstal ROS. *Pin general purpose input-output* (GPIO) pada Jetson Nano terhubung ke sebuah sensor IMU BNO055. Kedua peranti tersebut dihubungkan dengan komunikasi *inter-integrated circuit* (I2C). Sensor IMU tersebut terkoneksi ke dalam sistem ROS menggunakan *node /imu_node* yang menghasilkan *topic /imu_data*. Agar dapat mengendalikan aktuator, Arduino Uno dihubungkan ke NVIDIA Jetson Nano menggunakan komunikasi *universal serial bus* (USB). Dalam ROS, komunikasi tersebut dilakukan dengan menggunakan *node rosserial_arduino* (*/serial_node*). Arduino berperan untuk mengatur kecepatan motor DC sesuai perintah yang diberikan Jetson Nano melalui *topic /cmd_vel*. Sinyal *pulse width modulation* (PWM) dari Arduino dihubungkan ke *driver* motor L298, sehingga kecepatan motor dapat berubah-ubah. Saat motor berputar, sensor *rotary encoder* mengirimkan pulsa ke *pin* Arduino untuk diproses menjadi data kecepatan (*/left_speed* dan */right_speed*) dan jumlah putaran roda (*topic /left_ticks* dan */right_tick*).

B. DESAIN ROS FRAMEWORK

ROS merupakan *meta-operating system* yang dijalankan dalam robot yang telah dibuat. ROS memiliki beberapa buah *node* yang saling terhubung untuk menunjang kinerja robot. Setiap *node* dapat terhubung dengan *node* lain menggunakan *topic*. *Topic* berisikan pesan (*message*) yang mengandung data terkait pembacaan sensor, posisi robot, dan sebagainya. Dalam penelitian ini, digunakan beberapa *node default* atau bawaan ROS, antara lain */rpLiDAR_node*, */AMCL*, */movebase*, dan */SLAM_gmapping*. Selain *node default*, dibuat pula *node* lain yang menunjang kinerja robot, antara lain */ekf_odom_pub*, yang berfungsi menjalankan *extended Kalman filter* (EKF). *Node /datalogger* berfungsi untuk mengekstraksi data sensor yang sudah tersimpan terlebih dahulu dalam *rosbag*.

Node /rpLiDAR_node adalah salah satu *node* penting dalam sistem navigasi robot yang menggunakan ROS. RPLIDAR sendiri merupakan sebuah sensor LiDAR 2D yang dapat mengukur jarak dan sudut objek di sekitarnya dengan menggunakan sinar laser. Dalam ROS, *node /rpLiDAR_node* berfungsi untuk membaca data dari sensor RPLIDAR dan menghasilkan *topic /scan* yang nantinya akan digunakan oleh *node* lain dalam sistem ROS. *Topic /scan* yang dihasilkan oleh *node* RPLIDAR berisi data sudut dan jarak dari setiap titik di sekitar sensor LiDAR. Data tersebut dapat diakses dengan cara membaca *topic /scan* pada *child msg.ranges*[0-719], dengan nilai 0-719 tersebut merepresentasikan kenaikan setiap sudut 0,5°. Data ini kemudian direpresentasikan dalam bentuk *array*, sehingga dapat digunakan untuk melakukan pemetaan lingkungan atau navigasi robot secara otonom. Selain itu, data dari *topic /scan* juga dapat digunakan untuk menghindari rintangan atau objek di sekitar robot. Untuk menjalankan *node /rpLiDAR_node*, pengguna dapat menggunakan paket ROS (*ROS package*) yang telah disediakan oleh produsen RPLIDAR. Setelah dijalankan, *node /rpLiDAR_node* akan terus membaca data dari sensor dan menghasilkan *topic /scan* secara periodik sesuai dengan frekuensi yang ditentukan. Hal ini memungkinkan robot untuk terus memperbarui informasi tentang lingkungan sekitarnya dan mengambil keputusan yang tepat saat melakukan navigasi.

Node selanjutnya adalah *node /ekf_odom_pub*, yang menjalankan EKF pada ROS. EKF dapat digunakan untuk menggabungkan (*fusion*) data dari sensor *rotary encoder* dan IMU. Penggabungan kedua sensor tersebut bertujuan untuk perbaikan sistem *odometry* pada robot. *Rotary encoder* pada roda digunakan untuk mengukur jarak yang ditempuh oleh robot secara linear, sedangkan IMU digunakan untuk mengukur percepatan dan kecepatan sudut robot. Data dari kedua sensor ini digabungkan menggunakan filter EKF untuk memperkirakan posisi dan arah robot yang lebih akurat. Dalam *node* tersebut terdapat beberapa parameter yang perlu diatur, antara lain diameter roda yang digunakan, jarak antar roda atau *wheel base*, dan jumlah pulsa *encoder* saat robot bergerak satu meter.

p-ISSN 2301–4156 | e-ISSN 2460–5719

Gambar 4. Hasil *occupancy grid map* yang dihasilkan oleh *SLAM gmapping*.

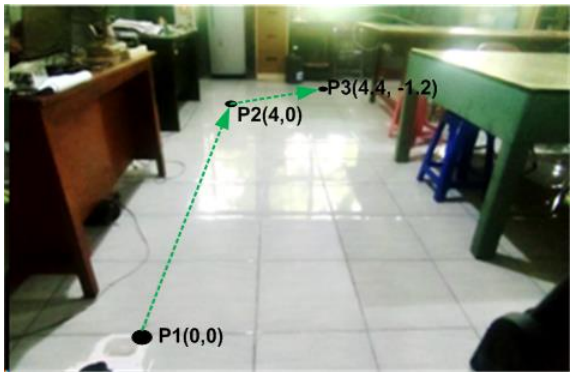
Untuk mendapatkan *occupancy grid map* dari ruangan, dijalankan algoritma *simultaneous localization and mapping* (SLAM) *gmapping*. Proses *SLAM gmapping* diawali dengan *data subscribing* dari sensor LiDAR dan posisi robot (dari *topic /ekf_odom_pub*). Selanjutnya data sensor diubah menjadi *gridmap* dengan menempatkan setiap partikel *scan* sensor LiDAR pada *grid* yang sesuai dengan posisi robot saat ini. *Grid* tersebut kemudian diperbarui untuk mencerminkan setiap *scan* baru yang diterima. *Grid* tersebut lalu dicocokkan dengan hasil *scan* sebelumnya. Hal ini digunakan untuk menentukan transformasi yang paling mungkin antara kedua *scan* tersebut saat memperbarui posisi robot. Selanjutnya, data *grid* peta diubah menjadi *occupancy grid* yang menyatakan probabilitas ruangan dalam *grid* tersebut ditempati atau tidak. Setelah peta selesai dibuat, *gmapping* menerbitkan peta yang dihasilkan melalui *topic* ROS yang sesuai, sehingga dapat digunakan oleh *node* lain dalam sistem. Hasil *occupancy grid map* ditunjukkan pada Gambar 4.

Node /AMCL bekerja dengan cara mengambil data dari sensor LiDAR dan peta *occupancy grid*, lalu menghasilkan *pose* (posisi dan arah) robot dalam lingkungan tersebut. Untuk menghasilkan *pose* robot, *node /AMCL* memerlukan data dari sensor LiDAR dan peta *occupancy grid*. Data dari sensor LiDAR digunakan untuk mendeteksi objek dan dinding yang ada di sekitar robot, sedangkan *occupancy grid map* digunakan sebagai referensi untuk menentukan posisi robot. Setelah data dari sensor LiDAR dan *occupancy grid map* dikumpulkan, *node /AMCL* melakukan pemrosesan data untuk menghasilkan *pose* robot. *Pose* tersebut dihasilkan dalam bentuk */amcl_pose* yang dapat digunakan untuk melakukan navigasi robot secara otonom.

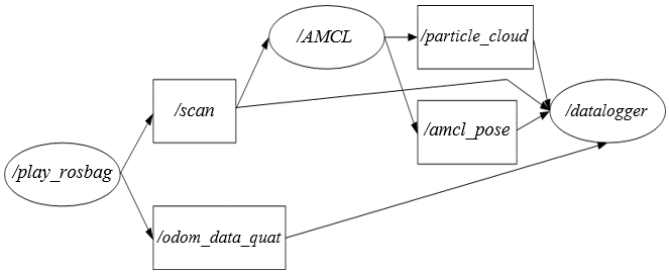
C. PENGAMBILAN DATASET ROSBAG

Setelah seluruh *node* terhubung, proses dilanjutkan dengan pengambilan *dataset* dalam bentuk file *rosbag*. *Rosbag* adalah alat bawaan di dalam ROS (*tools default*) yang digunakan untuk merekam data *topic* dalam sebuah *file*. Untuk merekam data menggunakan *rosbag*, pertama-tama perlu ditentukan *topic* yang akan direkam dan lama rekaman akan berlangsung. Kemudian, perekaman dapat dimulai dengan menjalankan perintah "*rosbag record*" dan memberikan nama *file* untuk menyimpan data rekaman. Setelah perekaman selesai, *file rosbag* dapat digunakan untuk memainkan kembali rekaman dengan perintah "*rosbag play*".

Dataset diambil menggunakan skenario menjalankan robot dari posisi start atau titik P1 menuju titik P2. Selanjutnya, robot berputar di tempat searah jarum jam sebesar 90°. Kemudian, robot dijalankan menuju titik P3 dan diputar lagi searah jarum jam sebesar 90°. Posisi titik-titik tersebut dalam area pengujian ditunjukkan pada Gambar 5. Dalam skenario tersebut, robot



Gambar 5. Area pengujian dalam pengambilan *file rosbag*.



Gambar 6. Koneksi antar-*node* saat mengekstraksi file *rosbag*.

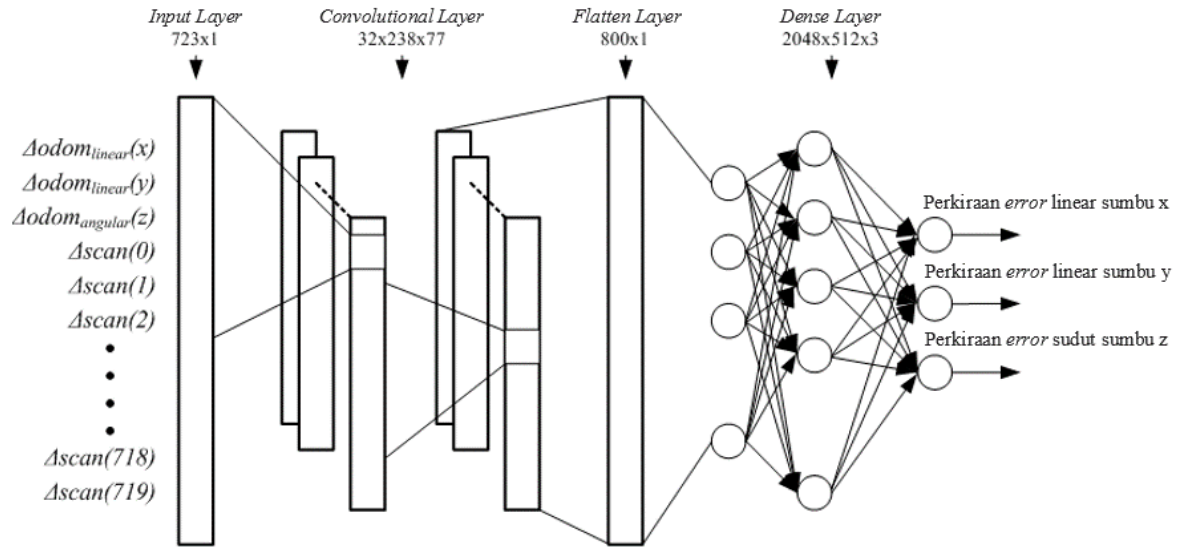
dijalankan dalam posisi maju lurus sebanyak dua kali dan belok sebanyak dua kali, sehingga dari skenario tersebut akan didapatkan data robot saat lurus dan belok. Data tersebut akan dijadikan sebagai acuan akurasi algoritma yang diusulkan. Tiap skenario dijalankan sebanyak empat kali dan tiap skenario disimpan dalam *rosbag* yang berbeda, sehingga didapatkan empat *rosbag* yang selanjutnya akan diekstrak menjadi *dataset* pelatihan dan *dataset* pengujian.

D. EKSTRAKSI ROSBAG UNTUK DATASET PELATIHAN

Proses pengumpulan *dataset* pelatihan merupakan salah satu tahap penting dalam membuat sebuah model *machine learning* yang berkualitas. *Machine learning* CNN ini menggunakan *dataset* pelatihan dan pengujian dalam bentuk teks dengan format CSV. Sementara itu, dari ROS, *dataset* tersimpan dalam bentuk *rosbag*, sehingga diperlukan sebuah konversi *rosbag* ke CSV agar *dataset* pelatihan maupun pengujian dapat digunakan. Berdasarkan kebutuhan tersebut, dibuat sebuah *node /datalogger*. *Node* tersebut mengonversi *rosbag* menjadi *file* CSV, mulai dari start hingga robot selesai melakukan pivot 90° di titik P3. Data yang disimpan adalah data dari *topic /amcl_pose*, */odom_combined*, dan */scan*. Koneksi antar-*node* saat menjalankan proses ini ditunjukkan pada Gambar 6.

Topic /amcl_pose digunakan untuk referensi atau data pembanding setelah proses koreksi dengan CNN selesai dilakukan. *Topic /odom_combined* digunakan untuk mengetahui perpindahan robot dalam waktu cuplik (*sampling*) tertentu. *Topic* tersebut berisi data posisi dan arah robot berdasarkan proses EKF dari sensor *odometry* roda dan sensor IMU. Dengan membandingkan nilainya pada waktu sekarang (t) dengan waktu sebelumnya (t-1), nilai perpindahan robot dapat dicari. Dalam *dataset* pelatihan yang dibuat, data yang disimpan hanya perpindahan robot terhadap sumbu x, sumbu y, dan perpindahan *angular* terhadap sumbu z atau *yaw*, sehingga dalam *dataset* pelatihan ini, nilai perpindahan robot memerlukan tiga kolom data. Persamaan perpindahan dinyatakan dalam (1) hingga (3) dengan nilai $OC(s,t)$ adalah nilai *topic /odom_combined* pada sumbu s. Notasi s dalam

p-ISSN 2301–4156 | e-ISSN 2460–5719



Gambar 7. Arsitektur *convolutional neural network* yang diusulkan.

persamaan tersebut menyatakan sumbu *odom* linear jika ditulis menggunakan huruf kecil, sedangkan huruf kapital menyatakan sumbu *angular*.

Topic /scan diperlukan untuk mencari selisih nilai pembacaan jarak tiap cuplik waktu. Selisih nilai yang muncul dapat merepresentasikan perpindahaan robot dalam sumbu x dan y. Nilai selisih ini menjadi masukan CNN karena diharapkan dapat mengoreksi nilai perpindahan robot yang didapatkan dari */odom_combined*. Jumlah selisih nilai *scan* (*Δscan*) dalam *dataset* sebanyak 720 kolom data. Nilai *Δscan* dapat dihitung dengan (4). Nilai *Δscan* adalah nilai selisih pembacaan jarak sensor LiDAR, sedangkan *n* adalah arah atau sudut pembacaan sensor LiDAR yang bernilai 0-719. Nilai *n* tersebut mewakili kenaikan setiap 0,5°. Nilai val_{scan} didapatkan dari pembacaan *topic /scan* pada pesan *msg.range*[*n*], sedangkan *t* adalah waktu saat ini dan *t*-1 adalah waktu pencuplikan sebelumnya.

$$\Delta odom_{linear}(x) = OC(x,t) - OC(x,t-1) \quad (1)$$

$$\Delta odom_{linear}(y) = OC(y,t) - OC(y,t-1) \quad (2)$$

$$\Delta odom_{angular}(z) = OC(Z,t) - OC(Z,t-1) \quad (3)$$

$$\Delta scan(n) = val_{scan}(n,t) - val_{scan}(n,t-1). \quad (4)$$

Arsitektur CNN yang dibuat merupakan *supervised machine learning*, sehingga dibutuhkan sebuah data target. Target yang digunakan adalah nilai selisih posisi dan arah aktual robot berdasarkan hasil pengamatan langsung dengan nilai */amcl_pose* yang didapatkan dari *file rosbag*.

E. DESAIN ARSITEKTUR CNN

Desain arsitektur CNN yang diusulkan dibuat menggunakan TensorFlow yang melibatkan beberapa tahap. Tahap pertama adalah inisialisasi lapisan (*layer*) yang berfungsi untuk memasukkan data latih ke dalam TensorFlow dan membuat lapisan-lapisan dasar dari CNN. Inisialisasi lapisan merupakan tahap penting karena memastikan bahwa semua data dan lapisan yang akan digunakan dalam proses pelatihan sudah tersedia dan siap digunakan. Tahap selanjutnya adalah membuat lapisan konvolusi (*convolutional layer*) yang berfungsi untuk menjalankan operasi konvolusi pada setiap bagian dari data masukan. Lapisan konvolusi memungkinkan ditangkapnya fitur-fitur penting dalam data masukan yang akan diteruskan ke lapisan selanjutnya.

Setelah lapisan konvolusi, dapat dibuat lapisan *pooling* yang akan mengurangi ukuran *feature map*, sehingga lebih mudah diteruskan ke lapisan selanjutnya. Proses *pooling* juga membantu mengurangi *overfitting* dan mempercepat proses pelatihan. Setelah proses *pooling*, harus dilakukan proses *flattening layer*, yaitu mengubah hasil *pooling* menjadi satu larik (*array*) yang lebih panjang. Proses *flattening layer* ini bertujuan untuk mempersiapkan data sebelum diteruskan ke lapisan *fully connected*. Jumlah *node* dalam lapisan ini adalah 800 *node*. Tahap terakhir adalah membuat lapisan *fully connected*, yaitu lapisan yang melakukan klasifikasi akhir dan memprediksi keluaran model. Lapisan *fully connected* ini terdiri atas dua lapisan tersembunyi dengan jumlah masing-masing adalah 2.048 dan 512 *node*. Sementara itu, lapisan keluaran terdiri atas tiga *node* yang masing-masing menghasilkan perkiraan *error* pada sumbu linear x dan y serta sumbu *angular* z. Arsitektur CNN yang digunakan ditunjukkan pada Gambar 7.

F. PELATIHAN JARINGAN

Untuk mendapatkan model yang optimal, CNN yang telah dibuat diberi *dataset* yang telah disiapkan untuk menjalankan proses pelatihan. *Dataset* tersebut selanjutnya dibagi menjadi *dataset* pelatihan dan *dataset* validasi untuk mengukur kinerja model selama pelatihan. Dalam pelatihan yang digunakan, *dataset* untuk pelatihan menggunakan 75% dari semua *dataset* dan 25% sisanya digunakan untuk *dataset* validasi. Sementara itu, *dataset* yang digunakan sebagai *dataset* pengujian merupakan *dataset* yang tidak diikutsertakan dalam proses pelatihan.

Setelah *dataset* siap untuk dibaca dalam tahap pelatihan, model dikompilasi dengan menggunakan *optimizer* Adam. Fungsi aktivasi yang digunakan adalah tanh karena sebaran *dataset* dan target berkisar antara -1 higga 1. *Loss function* dan matriks evaluasi yang digunakan adalah *mean squared error* (MSE). *Learning rate* yang digunakan adalah 0,01, yang merupakan nilai bawaan dari *optimizer* Adam. Proses pelatihan dilakukan hingga didapatkan bobot paling optimal. Dalam pelatihan ini, bobot dianggap optimal jika didapatkan nilai *validation loss*/MSE paling kecil.

G. METRIK EVALUASI

Untuk mendapatkan hasil kuantitatif mengenai *pose error* dari robot terhadap target, perlu dihitung *error* posisi total dari

p-ISSN 2301–4156 | e-ISSN 2460–5719

robot. Posisi robot direpresentasikan pada sumbu kartesian dengan asumsi nilai z = 0 karena robot berada di permukaan datar. *Error* posisi total dihitung berdasarkan selisih target dan posisi robot terhadap sumbu x dan y. Nilai ini dihitung menggunakan (5) hingga (7).

$$P_E(x,y) = \sqrt{(E_X)^2 + (E_Y)^2} \quad (5)$$

$$E_X = Act_X - Res_X \quad (6)$$

$$E_Y = Act_Y - Res_Y \quad (7)$$

dengan nilai $P_E(x,y)$ merupakan nilai *error* posisi total dalam satuan meter, E_X adalah *error* pada sumbu x, E_Y adalah *error* pada sumbu y, Act_X adalah nilai aktual robot pada sumbu x, Act_Y adalah nilai aktual robot pada sumbu y, serta nilai Res_X dan Res_Y merupakan hasil posisi robot dari algoritma yang sedang diujikan sesuai dengan sumbu yang bersangkutan. Algoritma yang diujikan antara lain MCL reguler, AMCL bawaan ROS, dan AMCL+CNN yang merupakan algoritma yang diusulkan.

Error arah menunjukkan selisih antara target arah dengan arah aktual. Arah ini diukur pada sumbu z atau sumbu *yaw* dari robot. Hal ini dilakukan karena robot diasumsikan berada pada permukaan datar. Nilai *error* arah didapatkan menggunakan (8).

$$H_E = Act_{YAW} - Res_{YAW} \quad (8)$$

dengan nilai H_E adalah *error* arah pada sumbu z atau *yaw* dalam satuan derajat (°), nilai Act_{YAW} merupakan nilai *yaw* aktual dari hasil pengamatan, sedangkan nilai Res_{YAW} merupakan nilai *yaw* hasil penerapan algoritma yang sedang diujikan.

III. HASIL DAN PEMBAHASAN

Dalam bagian ini, dijelaskan hasil pengujian sistem lokalisasi yang diterapkan pada robot. Pengujian diawali dengan membuat *occupancy grid map* dan selanjutnya menjalankan algoritma lokalisasi MCL reguler dengan variasi jumlah partikel. Jumlah partikel yang diujikan adalah 1.000 hingga 5.000, dengan interval 1.000 partikel. Lalu, algoritma AMCL bawaan dijalankan dengan beberapa rentang nilai partikel, 100 sampai 5.000, dan terakhir dilakukan pengujian algoritma AMCL+CNN, yang merupakan metode yang diusulkan dalam makalah ini. Selain itu, hasil pengujian juga dibandingkan dengan hasil penelitian sebelumnya yang mengusulkan IR-MCL [26]. Beberapa pengujian tersebut kemudian dibandingkan untuk mengetahui tingkat rata-rata *error* yang terjadi. Makin kecil nilai *error* yang muncul, makin akurat atau baik sistem lokalisasi.

A. PEMBUATAN OCCUPANCY GRID MAP DAN ROSBAG

Untuk menguji sistem lokalisasi, terlebih dahulu dibuat *occupancy grid map* dari area kerja yang digunakan. Peta tersebut dibuat menggunakan algoritma *SLAM gmapping*. *SLAM gmapping* bekerja dengan men-*subscribe* data sensor LiDAR (dari *topic /scan rpLiDAR*) dan pose robot (dari *topic /ekf_odom_pub*). Agar diperoleh hasil peta yang baik, robot dijalankan secara perlahan, karena ada potensi slip jika kecepatan terlalu tinggi. Dalam pengujian ini, robot dijalankan dengan kecepatan 0,1 m/s.

Dalam proses pembuatan peta, robot mula-mula diletakkan pada posisi P1 pada koordinat (0,0). Selanjutnya, robot digerakkan secara manual menggunakan *node /teleop_twist_keyboard*. Robot dijalankan ke seluruh bagian ruangan hingga seluruh sudut ruangan terjangkau pemindaian oleh LiDAR. Setelah selesai, peta disimpan menggunakan *node /map_saver*. Peta yang telah dibuat ditunjukkan pada Gambar 5. Peta disimpan dengan */map_server* tersebut digunakan dalam pengujian berikutnya.

Perekaman *rosbag* dilakukan dengan cara menjalankan robot pada posisi P1. Sebelumnya, perintah "*rosbag record*" harus dijalankan terlebih dahulu. Selanjutnya robot dijalankan menuju titik P2 dan melakukan gerakan berputar di tempat sebesar 90°. Robot kemudian dijalankan menuju posisi titik P3 dan dilanjutkan dengan melakukan gerakan berputar di tempat sebesar 90°. Selanjutnya, perekaman *rosbag* dihentikan dan dilanjutkan dengan perintah penyimpanan *rosbag*. Penyimpanan *rosbag* ini dilakukan sebanyak empat kali dengan setiap pengujian masing-masing diberi nama "A", "B", "C", dan "D".

TABEL I
ERROR POSISI DAN *ERROR* ARAH PADA MCL REGULER

Jumlah Partikel	*Error* Posisi (m)		*Error* Arah (°)	
	Lurus	*Belok*	*Lurus*	*Belok*
1.000	0,0277	0,0236	12,49	12,28
2.000	0,0558	0,0826	13,98	7,79
3.000	0,0243	0,0139	12,60	11,45
4.000	0,0249	0,0164	12,62	10,80
5.000	0,0287	0,0155	12,68	11,79
Rata-rata	0,0323	0,0304	12,87	10,82

B. PENGUJIAN DENGAN MCL REGULER

Setelah *occupancy grid map* dibuat, untuk pengujian *localization system* digunakan peta sesuai Gambar 5. Pengujian dilakukan dengan menjalankan *rosbag* yang telah tersimpan. Tiap *rosbag* diputar ulang dan secara bersamaan menjalankan algoritma MCL reguler. MCL reguler ini dibuat dengan cara mengatur *node /AMCL* bawaan ROS agar tidak melakukan pembaruan (*update*) partikel dan menggunakan partikel dengan jumlah tetap. Partikel yang digunakan berjumlah 1.000 hingga 5.000, dengan interval 1.000 partikel, sehingga pengujian ini mendapatkan lima variasi jumlah partikel, seperti disajikan pada Tabel I.

Pada pengujian dengan skenario lurus, jumlah partikel 3.000 memberikan hasil *error* posisi terkecil, yaitu 0,0243 m di skenario lurus dan 0,0139 m di skenario belok. Sementara itu, *error* posisi terbesar dihasilkan oleh jumlah partikel 2.000, yaitu 0,0558 m di skenario lurus dan 0,0826 m di skenario belok. Namun, jumlah partikel 2.000 menghasilkan *error* arah terkecil saat skenario belok, yaitu 7,79°, sedangkan saat skenario lurus, *error* arah terkecil terjadi saat jumlah partikel 1.000.

Berdasarkan pengujian pada MCL reguler, rata-rata *error* posisi yang terjadi pada saat robot berjalan lurus adalah 0,0323 m. Nilai *error* ini diambil pada saat robot berjalan dari P1 menuju P2 dan dari P2 menuju P3. Rata-rata *error* arah yang terjadi dalam skenario tersebut adalah 12,87°. Nilai *error* posisi dan *error* arah pada saat belok diambil saat robot melakukan gerakan belok secara pivot pada P2 dan P3 sebesar 90°. *Error* posisi saat belok adalah 0,0304 m dan *error* arahnya adalah 10,82°.

C. PENGUJIAN DENGAN AMCL DEFAULT ROS

Dalam pengujian AMCL bawaan ROS, jumlah partikel diatur dengan rentang 100-5.000 partikel. Pengujian data dilakukan sebanyak empat kali pada *file rosbag* yang berbeda. Pengujian dilakukan dengan prosedur yang sama dengan pengujian MCL reguler. Namun, dalam pengujian ini dilakukan pembaruan jumlah partikel dalam *node /AMCL*.

p-ISSN 2301–4156 | e-ISSN 2460–5719

TABEL II
ERROR POSISI DAN *ERROR* ARAH PADA AMCL BAWAAN ROS

Rosbag	*Error* Posisi (m)		*Error* Arah (°)	
	Lurus	*Belok*	*Lurus*	*Belok*
A	0,0262	0,0112	11,50	10,75
B	0,0262	0,0106	11,50	10,41
C	0,0229	0,0381	12,81	15,87
D	0,0434	0,0322	13,18	17,90
Rata-rata	0,0297	0,0230	12,25	13,73

Tabel II menunjukkan hasil pengujian AMCL *default* atau bawaan ROS. *Error* posisi rata-rata dari keempat *rosbag* adalah 0,0297 m pada skenario lurus dan 0,023 m pada skenario belok, sedangkan nilai *error* arah adalah 12,25° pada skenario lurus dan 13,73° pada skenario belok.

Berdasarkan data tersebut, AMCL memberikan nilai posisi robot yang lebih baik jika dibandingkan dengan MCL. Hal ini ditunjukkan oleh Tabel II, dibuktikan dengan nilai rata-rata *error* posisi yang makin kecil. Namun, untuk *error* arah, justru AMCL memberikan nilai *error* yang makin besar pada saat belok, sedangkan saat lurus nilai *error* arah hanya turun di bawah 1°. Kedua data yang didapatkan dari MCL dan AMCL diperbaiki menggunakan penambahan CNN, lalu kedua data tersebut akan dijadikan sebagai pembanding dengan data yang dihasilkan oleh AMCL+CNN.

D. PENGUJIAN DENGAN AMCL+CNN

Pengujian AMCL+CNN dilakukan dengan mengoreksi nilai keluaran AMCL agar mendekati nilai aktual berdasarkan data yang telah dilatihkan sebelumnya. Proses pelatihan dilakukan dengan model seperti yang telah dijelaskan pada subbagian II.E dan II.F. *Dataset* untuk pelatihan memiliki data sebanyak 200 dengan 723 masukan dan 3 keluaran. Pelatihan jaringan CNN dilakukan sebanyak sepuluh kali dengan variasi jumlah *epoch* maksimal. Jumlah *epoch* yang diujikan mulai dari 50 hingga 500 dengan interval 50. Pada pelatihan yang dilakukan, nilai *validation loss* menjadi metrik acuan yang diamati. Nilai *validation loss* yang makin kecil menunjukkan hasil pelatihan yang lebih baik. Hasil terbaik dalam pelatihan tersebut adalah pada nilai 200 *epoch*, yang menghasilkan *validation loss* sebesar 0,0148.

Bobot paling optimal dari pelatihan tersebut lalu diujikan pada *dataset* pengujian. *Dataset* ini memiliki jumlah data sebanyak 114 dengan 723 masukan dan 3 keluaran. Keluaran pada *dataset* pengujian ini dijadikan sebagai referensi posisi dan arah dari robot. Sementara itu, keluaran dari algoritma AMCL+CNN merepresentasikan posisi dan arah aktual dari robot. Berdasarkan kedua nilai tersebut, nilai *error* posisi dan *error* arah dapat diketahui.

Berdasarkan data pembanding yang telah didapatkan, dilakukan komparasi antardata pengujian. Dari Tabel III dan Tabel IV, terlihat bahwa nilai *error* posisi dan *error* arah AMCL+CNN dalam skenario lurus dan belok selalu lebih kecil daripada MCL reguler dan AMCL ROS. Hasil *error* posisi rata-rata dari keempat *rosbag* dengan algoritma AMCL+CNN adalah 0,0219 m pada skenario lurus dan 0,0186 m pada skenario belok, sedangkan untuk *error* arah, pada skenario lurus didapatkan nilai 7,95° dan pada skenario belok didapatkan nilai 8,76°.

Selain itu, digunakan juga hasil pengujian yang telah dilakukan [26]. Dalam pengujian tersebut, dilakukan pengujian estimasi posisi robot menggunakan *Monte Carlo Localization + exploiting text spotting* (NMCL) dan IR-MCL. Jika

TABEL III
ERROR POSISI DAN *ERROR* ARAH PADA AMCL+CNN

Rosbag	*Error* Posisi (m)		*Error* Arah (°)	
	Lurus	*Belok*	*Lurus*	*Belok*
A	0,0076	0,0059	4,03	4,06
B	0,0076	0,0075	4,03	3,95
C	0,0255	0,0280	11,74	12,17
D	0,0470	0,0328	12,00	14,87
Rata-rata	0,0219	0,0186	7,95	8,76

TABEL IV
PERBANDINGAN NILAI RATA-RATA *ERROR* PADA TIAP ALGORITMA

Algoritma	**Rata-Rata *Error* Posisi (m)**	**Rata-Rata *Error* Arah (°)**
MCL reguler	0,0314	11,85
AMCL ROS	0,0264	12,99
NMCL [26]	0,1369	2,37
IR-MCL [26]	0,0687	1,19
AMCL+CNN	0,0202	8,35

dibandingkan dengan metode tersebut, metode yang diusulkan, yaitu AMCL+CNN, memberikan *error* posisi terkecil, yaitu 0,0202 m (rata-rata skenario lurus dan belok). Algoritma NMCL dalam pengujian menghasilkan nilai *error* posisi sebesar 0,1369 m, sedangkan IR-MCL menghasilkan *error* posisi sebesar 0,0687 [26]. Namun, nilai *error* arah dari AMCL+CNN menjadi *error* arah terbesar, sehingga masih berpotensi untuk diperbaiki lagi.

IV. KESIMPULAN

Peningkatan akurasi algoritma AMCL menggunakan CNN untuk robot pada kondisi dalam ruangan telah berhasil direalisasikan. Penerapan AMCL+CNN yang telah dilakukan berhasil menurunkan nilai *error* jika dibandingkan dengan AMCL bawaan maupun MCL reguler. Pada skenario berjalan lurus, *error* posisi yang dihasilkan turun menjadi 0,022 m jika dibandingkan dengan AMCL bawaan dan MCL reguler, yang masing-masing sebesar 0,0297 m dan 0,0323 m. Sementara itu, *error* arah saat berjalan lurus turun menjadi 7,95° dari nilai AMCL bawaan, sebesar 12,25°, dan MCL reguler, sebesar 12,87°. Pada skenario belok, *error* posisi yang dihasilkan turun menjadi 0,0186 m jika dibandingkan dengan AMCL bawaan dan MCL reguler, yang masing-masing sebesar 0,023 m dan 0,0304 m. *Error* arah saat berjalan lurus turun menjadi 8,76° dari nilai AMCL bawaan, sebesar 13,73°, dan MCL regular, sebesar 10,58°. Dengan nilai *error* yang makin kecil, kinerja robot saat menjalankan sistem navigasi maupun pekerjaan lain akan semakin optimal. Jika dibandingkan dengan metode AMCL, NMCL, dan IR-MCL, metode yang diusulkan memiliki *error* posisi yang lebih baik.

KONFLIK KEPENTINGAN

Penulis menyatakan bahwa selama melaksanakan penelitian dan penulisan artikel ilmiah dengan judul "Peningkatan Akurasi *Adaptive Monte Carlo Localization* Menggunakan *Convolutional Neural Network*" ini, tim penulis tidak memiliki konflik kepentingan dengan pihak mana pun.

KONTRIBUSI PENULIS

Pembuatan robot, Riza Agung Firmansyah; pengujian, Riza Agung Firmansyah; analisis, Riza Agung Firmansyah; analisis lanjutan, Tri Arief Sardjono dan Ronny Mardiyanto; penulisan

makalah, Riza Agung Firmansyah, Tri Arief Sardjono, dan Ronny Mardiyanto.

REFERENSI

[1] F. Rovira-Más, V. Saiz-Rubio, dan A. Cuenca-Cuenca, "Augmented Perception for Agricultural Robots Navigation," *IEEE Sens. J.*, Vol. 21, No. 10, hal. 11712–11727, Mei 2021, doi: 10.1109/JSEN.2020.3016081.

[2] S. Karmore dkk., "IoT-Based Humanoid Software for Identification and Diagnosis of Covid-19 Suspects," *IEEE Sens. J.*, Vol. 22, No. 18, hal. 17490–17496, Sep. 2022, doi: 10.1109/JSEN.2020.3030905.

[3] P.-Y. Yang, T.-H. Chang, Y.-H. Chang, dan B.-F. Wu, "Intelligent Mobile Robot Controller Design for Hotel Room Service with Deep Learning Arm-Based Elevator Manipulator," *2018 Int. Conf. Syst. Sci., Eng. (ICSSE)*, 2018, hal. 1–6. doi: 10.1109/ICSSE.2018.8520030.

[4] B.P.E.A. Vasquez, R. Gonzalez, F. Matia, dan P. De La Puente, "Sensor Fusion for Tour-Guide Robot Localization," *IEEE Access*, Vol. 6, hal. 78947–78964, Des. 2018, doi: 10.1109/ACCESS.2018.2885648.

[5] Y. Peng dkk., "Research Progress of Urban Dual-Arm Humanoid Grape Harvesting Robot," *2021 IEEE 11th Annu. Int. Conf. CYBER Technol. Automat. Control, Intell. Syst. (CYBER)*, 2021, hal. 879–885. doi: 10.1109/CYBER53097.2021.9588266.

[6] D. Shi, H. Mi, E.G. Collins, dan J. Wu, "An Indoor Low-Cost and High-Accuracy Localization Approach for AGVs," *IEEE Access*, Vol. 8, hal. 50085–50090, Mar. 2020, doi: 10.1109/ACCESS.2020.2980364.

[7] R.A. Firmansyah, Y.A. Prabowo, dan T. Suheta, "Thermal Imaging-Based Body Temperature and Respiratory Frequency Measurement System for Security Robot," *Przegląd Elektrotechniczny*, Vol. 98, No. 6, hal. 126-130, 2022, doi: 10.15199/48.2022.06.23.

[8] R.M. Ñope-Giraldo dkk., "Mechatronic Systems Design of ROHNI-1: Hybrid Cyber-Human Medical Robot for COVID-19 Health Surveillance at Wholesale-Supermarket Entrances," *2021 Glob. Med. Eng. Phys. Exch./Pan Amer. Health Care Exch. (GMEPE/PAHCE)*, 2021, hal. 1–7. doi: 10.1109/GMEPE/PAHCE50215.2021.9434874.

[9] A. Carlucci, M. Morisco, dan F. Dell'Olio, "Human Vital Sign Detection by a Microcontroller-Based Device Integrated into a Social Humanoid Robot," *2022 IEEE Int. Symp. Med. Meas., Appl. (MeMeA)*, 2022, hal. 1–6. doi: 10.1109/MeMeA54994.2022.9856407.

[10] L. Garrote, T. Barros, R. Pereira, dan U.J. Nunes, "Absolute Indoor Positioning-aided Laser-based Particle Filter Localization with a Refinement Stage," *IECON 2019 - 45th Annu. Conf. IEEE Ind. Electron. Soc.*, 2019, hal. 597–603. doi: 10.1109/IECON.2019.8927475.

[11] M.-A. Chung dan C.-W. Lin, "An Improved Localization of Mobile Robotic System Based on AMCL Algorithm," *IEEE Sens. J.*, Vol. 22, No. 1, hal. 900–908, Jan. 2022, doi: 10.1109/JSEN.2021.3126605.

[12] S.J. Dignadice dkk., "Application of Simultaneous Localization and Mapping in the Development of an Autonomous Robot," *2022 8th Int. Conf. Control Automat., Robot. (ICCAR)*, 2022, hal. 77–80. doi: 10.1109/ICCAR55106.2022.9782658.

[13] A. Ehambram, L. Jaulin, dan B. Wagner, "Hybrid Interval-Probabilistic Localization in Building Maps," *IEEE Robot., Autom. Lett.*, Vol. 7, No. 3, hal. 7059–7066, Jul. 2022, doi: 10.1109/LRA.2022.3181371.

[14] K. Żywanowski, A. Banaszczyk, M.R. Nowicki, dan J. Komorowski, "MinkLoc3D-SI: 3D LiDAR Place Recognition with Sparse Convolutions, Spherical Coordinates, and Intensity," *IEEE Robot., Autom. Lett.*, Vol. 7, No. 2, hal. 1079–1086, Apr. 2022, doi: 10.1109/LRA.2021.3136863.

[15] W. Shen, Y. Jia, J. Zhu, dan X. Qian, "A Fast Monocular Visual–Inertial Odometry Using Point and Line Features," *2022 7th Int. Conf. Signal, Image Process. (ICSIP)*, 2022, hal. 591–595. doi: 10.1109/ICSIP55141.2022.9886829.

[16] J. Li dan A. Hamdulla, "A Research of Visual-Inertial Simultaneous Localization and Mapping," *2022 3rd Int. Conf. Pattern Recognit., Mach. Learn. (PRML)*, 2022, hal. 143–150. doi: 10.1109/PRML56267.2022.9882205.

[17] J. Yuan, S. Zhu, K. Tang, dan Q. Sun, "ORB-TEDM: An RGB-D SLAM Approach Fusing ORB Triangulation Estimates and Depth Measurements," *IEEE Trans. Instrum., Meas.*, Vol. 71, hal. 1–15, 2022, doi: 10.1109/TIM.2022.3154800.

[18] J. Liu, X. Li, Y. Liu, dan H. Chen, "RGB-D Inertial Odometry for a Resource-Restricted Robot in Dynamic Environments," *IEEE Robot., Automat. Lett.*, Vol. 7, No. 4, hal. 9573–9580, Okt. 2022, doi: 10.1109/LRA.2022.3191193.

[19] R. Long dkk., "RGB-D SLAM in Indoor Planar Environments with Multiple Large Dynamic Objects," *IEEE Robot., Automat. Lett.*, Vol. 7, No. 3, hal. 8209–8216, Jul. 2022, doi: 10.1109/LRA.2022.3186091.

[20] Y. Chen dkk., "Submap-Based Indoor Navigation System for the Fetch Robot," *IEEE Access*, Vol. 8, hal. 81479–81491, Apr. 2020, doi: 10.1109/ACCESS.2020.2991465.

[21] K. Tian dan K. Mirza, "Sensor Fusion for Octagon – an Indoor and Outdoor Autonomous Mobile Robot," *2022 IEEE Int. Syst. Conf. (SysCon)*, 2022, hal. 1–5. doi: 10.1109/SysCon53536.2022.9773827.

[22] C. Li, S. Wang, Y. Zhuang, dan F. Yan, "Deep Sensor Fusion Between 2D Laser Scanner and IMU for Mobile Robot Localization," *IEEE Sens. J.*, Vol. 21, No. 6, hal. 8501–8509, Mar. 2021, doi: 10.1109/JSEN.2019.2910826.

[23] N. Zimmerman dkk., "Robust Onboard Localization in Changing Environments Exploiting Text Spotting," *2022 IEEE/RSJ Int. Conf. Intell. Robots, Syst. (IROS)*, 2022, hal. 917–924. doi: 10.1109/IROS47612.2022.9981049.

[24] B. Kaleci, K. Turgut, dan H. Dutagaci, "2DLaserNet: A Deep Learning Architecture on 2D Laser Scans for Semantic Classification of Mobile Robot Locations," *Eng. Sci., Technol. Int. J.*, Vol. 28, hal. 1-13, Apr. 2022, doi: 10.1016/j.jestch.2021.06.007.

[25] G. Spampinato, A. Bruna, I. Guarneri, dan D. Giacalone, "Deep Learning Localization with 2D Range Scanner," *2021 7th Int. Conf. Automat. Robot., Appl. (ICARA)*, 2021, hal. 206–210. doi: 10.1109/ICARA51699.2021.9376424.

[26] H. Kuang dkk., "IR-MCL: Implicit Representation-Based Online Global Localization," *IEEE Robot., Automat. Lett.*, Vol. 8, No. 3, hal. 1627–1634, Mar. 2023, doi: 10.1109/LRA.2023.3239318.