

BAB II

DASAR TEORI

2.1 Tinjauan Pustaka

Tinjauan pustaka dilakukan untuk mengidentifikasi perkembangan penelitian yang berkaitan dengan sistem lokalisasi, fusi sensor, navigasi kendaraan otonom, serta perencanaan jalur pada mobile robot. Kajian terhadap penelitian terdahulu diperlukan untuk memahami metode yang telah digunakan, jenis sensor yang diintegrasikan, pendekatan estimasi posisi yang diterapkan, serta karakteristik algoritma perencanaan jalur yang digunakan pada kendaraan dengan keterbatasan gerak tertentu. Selain itu, tinjauan pustaka digunakan untuk melihat keterkaitan antara penelitian sebelumnya dengan sistem yang dikembangkan pada penelitian ini, khususnya pada penggunaan GNSS, IMU, odometri, *Extended Kalman Filter* (EKF), dan Smac Hybrid-A* pada Navigation2.

Beberapa penelitian sebelumnya telah membahas fusi GNSS, IMU, dan odometri untuk meningkatkan konsistensi estimasi posisi kendaraan, baik menggunakan EKF dua dimensi maupun model gerak kendaraan. Penelitian lain membahas navigasi berbasis waypoint serta penerapan algoritma perencanaan jalur yang mempertimbangkan karakteristik kinematika kendaraan Ackermann. Meskipun metode-metode tersebut memiliki keterkaitan dengan sistem yang dikembangkan, penelitian ini menerapkan arsitektur yang lebih terintegrasi melalui pembagian EKF lokal dan EKF global, validasi kualitas measurement, pengelolaan transformasi kerangka koordinat map-odom, serta penggunaan Smac Hybrid-A* sebagai global planner pada Navigation2. Dengan demikian, perbandingan terhadap penelitian terdahulu diperlukan untuk menunjukkan posisi penelitian, pendekatan yang digunakan, serta aspek pengembangan yang diterapkan pada sistem ADV. Ringkasan penelitian terdahulu dan target pembaruan pada penelitian ini disajikan pada **Tabel 2**.

Tabel 2. 1. Tinjauan Pustaka

Judul Penelitian	Keterangan	Target Pembaruan
------------------	------------	------------------

Judul Penelitian	Keterangan	Target Pembaruan
<i>Vehicle Localization Based on IMU, OBD2, and GNSS Sensor Fusion Using Extended Kalman Filter</i> (Teoh et al.)	Menggunakan OBD2 dan IMU untuk prediksi gerak, kemudian GNSS sebagai koreksi posisi dan orientasi melalui EKF [4].	Menerapkan dua estimator, yaitu EKF lokal pada frame odom dan EKF global pada frame map, dengan data yang telah melalui proses validasi.
<i>Experimental 2D Extended Kalman Filter Sensor Fusion for Low-Cost GNSS/IMU/Odometers Precise Positioning System</i> (Kaczmarek et al.)	Mengintegrasikan GNSS, IMU, dan odometer berbiaya rendah menggunakan EKF 2D untuk meningkatkan estimasi posisi dan arah [6].	Diterapkan pada ADV Ackermann dengan GNSS CUAV NEO 3, IMU Yahboom ROS 10-Axis, odometri ESC, serta evaluasi kualitas dan kontinuitas measurement.
<i>Sistem Navigasi Berbasis Modul Kompas dan GPS pada Prototipe Autonomous Car dengan Pengendali Steering Metode PI</i> (Muzakki et al.)	Menggunakan GPS, kompas, dan IMU untuk navigasi berbasis waypoint dengan kendali steering PI [8].	Menggunakan pose hasil fusi sensor untuk membentuk <i>global path</i> pada Navigation2, bukan hanya mengikuti waypoint tetap.
<i>Cost-Aware Kinematically Feasible Planning for Mobile and Surface Robotics</i> (Macenski et al.)	Mengembangkan Smac Planner dengan Cost-Aware A*, Hybrid-A*, dan State Lattice yang mempertimbangkan orientasi dan batas kinematika [10].	Menerapkan Smac Hybrid-A* model Dubins pada ADV Ackermann dan mengevaluasi waktu perencanaan, endpoint error, clearance, serta keberhasilan jalur.

2.2 Autonomous Delivery Vehicle, Sensor Lokalisasi, dan Odometri

Autonomous Delivery Vehicle (ADV) merupakan kendaraan bergerak otonom yang dirancang untuk membawa barang dari lokasi awal menuju lokasi tujuan dengan keterlibatan manusia yang minimum. Sistem ADV mengintegrasikan sensor, unit komputasi, lokalisasi, perencanaan jalur, dan pengendalian gerak agar kendaraan dapat beroperasi secara mandiri. Dalam konteks pengiriman jarak dekat, kendaraan pengiriman otonom dapat berupa robot darat berukuran relatif kecil yang membawa barang langsung kepada pengguna tanpa dikemudikan oleh petugas pengiriman [11]. Pada penelitian ini, ADV menggunakan sistem kemudi Ackermann dengan karakteristik nonholonomik sehingga kendaraan tidak dapat bergerak lateral maupun berputar pada satu titik.

Keadaan kendaraan direpresentasikan dalam bentuk pose yang terdiri atas posisi x , posisi y , dan orientasi $yaw \psi$ terhadap suatu kerangka koordinat. Informasi pose digunakan sebagai dasar untuk menentukan posisi dan arah hadap kendaraan selama proses navigasi. Lokalisasi dilakukan secara berkelanjutan menggunakan beberapa sumber pengukuran karena setiap sensor memiliki karakteristik yang berbeda. Penggabungan informasi lokal dan global diperlukan agar estimasi pose tetap kontinu sekaligus tetap terhubung dengan referensi posisi global [12].

Pada sistem yang dikembangkan, sumber informasi lokalisasi terdiri atas GNSS, IMU, dan odometri kendaraan. GNSS menyediakan informasi posisi global, IMU memberikan informasi orientasi dan kecepatan sudut, sedangkan odometri memberikan informasi gerak relatif kendaraan. Estimasi dibagi menjadi EKF lokal dan EKF global. EKF lokal menghasilkan estimasi gerak kontinu pada kerangka odom, sedangkan EKF global menghasilkan estimasi terhadap kerangka map menggunakan data yang telah melalui proses validasi. Hubungan transformasi sistem membentuk $map \rightarrow odom \rightarrow base_footprint$, dengan transformasi $map \rightarrow odom$ dikelola oleh modul *LocalizationCore* dan $odom \rightarrow base_footprint$ berasal dari estimasi lokal.

Pose hasil lokalisasi selanjutnya digunakan oleh Navigation2 untuk membentuk *global path*. Pada implementasi sistem, perencanaan jalur menggunakan Smac Hybrid-A* dengan model gerak Dubins untuk menyesuaikan jalur dengan karakteristik kendaraan Ackermann, sedangkan MPPI digunakan sebagai *local controller* pendukung.

Susunan tersebut membentuk alur navigasi mulai dari pengindraan, estimasi pose, perencanaan jalur, hingga pengendalian gerak, sesuai dengan arsitektur umum sistem kendaraan otonom yang mengintegrasikan fungsi persepsi, lokalisasi, perencanaan, dan pengendalian [13].

2.3 Sensor GNSS CUAV NEO 3 dan Transformasi Koordinat

GNSS merupakan sistem penentuan posisi berbasis satelit yang menghasilkan informasi posisi dalam koordinat geografis berupa *latitude*, *longitude*, dan *altitude*. Pada penelitian ini digunakan modul CUAV NEO 3 dengan receiver u-blox NEO-M9N yang mendukung penerimaan beberapa konstelasi GNSS secara bersamaan yang tertera pada **Gambar 2.1**.



Gambar 2. 1 CUAV NEO 3

Receiver tersebut digunakan sebagai sumber informasi posisi global kendaraan untuk menghubungkan hasil lokalisasi dengan kerangka koordinat map. Spesifikasi receiver yang berkaitan langsung dengan kebutuhan sistem navigasi ditunjukkan pada **Tabel 2.2**. Receiver NEO-M9N memiliki akurasi posisi PVT tipikal 2,0 m CEP, akurasi kecepatan 0,05 m/s, dan laju pembaruan PVT maksimum 25 Hz

Tabel 2. 2 Spesifikasi CUAV NEO 3

Parameter	Spesifikasi
Tipe receiver	<i>Multi-constellation GNSS standard precision receiver</i>
Konstelasi	GPS, GLONASS, Galileo, BeiDou
Frekuensi sinyal	GPS L1 C/A, GLONASS L1OF, Galileo E1B/C, BeiDou B1I
Akurasi posisi PVT	2,0 m CEP
Akurasi kecepatan	0,05 m/s

Parameter	Spesifikasi
Laju pembaruan PVT	Maksimum 25 Hz
<i>Cold start</i>	24 s
<i>Hot start</i>	2 s
Sensitivitas <i>tracking/navigation</i>	-167 dBm

Kualitas data GNSS tidak hanya ditinjau dari koordinat posisi, tetapi juga dari kondisi solusi posisi dan ketidakpastian pengukurannya. Parameter seperti jumlah satelit, DOP, hAcc, serta status *fix* dapat digunakan sebagai indikator kelayakan pengukuran sebelum data digunakan oleh sistem lokalisasi. Nilai DOP yang lebih kecil menunjukkan geometri satelit yang lebih baik, sedangkan hAcc menggambarkan estimasi ketidakpastian horizontal dari receiver. Nilai tersebut merupakan indikator kualitas pengukuran dan tidak secara langsung menyatakan kesalahan aktual terhadap posisi kendaraan. Pada kendaraan *car-like*, pengukuran GNSS dapat digunakan sebagai sumber posisi global selama hubungan antara kerangka sensor dan kerangka kendaraan diperhitungkan dengan benar [15].

Koordinat *latitude* dan *longitude* tidak langsung digunakan sebagai koordinat pada peta Navigation2 karena masih dinyatakan dalam satuan sudut. Untuk area operasi lokal, perubahan koordinat geografis dapat diproyeksikan ke bidang lokal menggunakan pendekatan *equirectangular*. Pendekatan tersebut mengubah perubahan *longitude* dan *latitude* menjadi komponen arah timur (*East*) dan utara (*North*) dalam satuan meter [16]. Bentuk persamaan yang digunakan tertera pada persamaan (2.1) yaitu:

$$\begin{aligned} E_i &= R(\lambda_i - \lambda_0) \cos\left(\frac{\phi_i + \phi_0}{2}\right) \\ N_i &= R(\phi_i - \phi_0) \end{aligned} \quad (2.1)$$

Keterangan :

E_i = posisi lokal arah timur pada sampel ke- i (m);

N_i = posisi lokal arah utara pada sampel ke- i (m);

R = radius bumi yang digunakan dalam perhitungan (m);

ϕ_i = *latitude* sampel ke- i (rad);

λ_i = *longitude* sampel ke- i (rad);

ϕ_0 = *latitude* titik referensi (rad); dan

λ_0 = *longitude* titik referensi (rad).

Persamaan (2.1) menghasilkan posisi relatif terhadap titik referensi dalam bentuk koordinat *East-North*. Meskipun demikian, orientasi kerangka *East-North* belum tentu sama dengan orientasi kerangka lokal yang digunakan pada sistem navigasi. Oleh karena itu, diperlukan transformasi rotasi dan translasi untuk menyelaraskan koordinat lokal terhadap kerangka acuan yang digunakan. Transformasi tersebut dinyatakan pada Persamaan (2.2).

$$\begin{bmatrix} x_{L,i} \\ y_{L,i} \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} + \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} E_i \\ N_i \end{bmatrix} \quad (2.2)$$

Keterangan:

$x_{L,i}$ = posisi koordinat pada sumbu x dalam kerangka lokal yang telah diselaraskan pada sampel ke- i (m).

$y_{L,i}$ = posisi koordinat pada sumbu y dalam kerangka lokal yang telah diselaraskan pada sampel ke- i (m).

x_0 = posisi titik referensi awal pada sumbu x dalam kerangka lokal (m).

y_0 = posisi titik referensi awal pada sumbu y dalam kerangka lokal (m).

E_i = koordinat arah *East* (timur) pada sistem koordinat *East-North* untuk sampel ke- i (m).

N_i = koordinat arah *North* (utara) pada sistem koordinat *East-North* untuk sampel ke- i (m).

θ = sudut rotasi antara kerangka koordinat *East-North* dan kerangka koordinat lokal yang digunakan (rad).

i = indeks sampel atau urutan data pengukuran

Transformasi pada Persamaan (2.2) memungkinkan posisi GNSS yang semula dinyatakan terhadap arah geografis *East-North* diselaraskan dengan kerangka lokal yang digunakan dalam sistem navigasi. Dengan demikian, data GNSS dapat direpresentasikan dalam koordinat planar yang konsisten sebagai dasar bagi proses estimasi posisi kendaraan pada tahap berikutnya.

2.4 Sensor IMU Yahboom ROS 10-Axis

Inertial Measurement Unit (IMU) merupakan sensor yang digunakan untuk memperoleh informasi gerak dan orientasi melalui pengukuran percepatan dan kecepatan sudut. IMU berbasis *Micro-Electro-Mechanical Systems* (MEMS) umumnya menggunakan akselerometer dan giroskop tiga sumbu. Meskipun mampu menghasilkan data secara kontinu, sensor MEMS tetap memiliki sumber kesalahan seperti *bias*, derau, *scale factor error*, dan ketidaksejajaran sumbu. Pada giroskop, *bias* menjadi salah satu sumber kesalahan yang dapat memengaruhi kestabilan estimasi orientasi apabila tidak dikompensasi[17] untuk visualisasi IMU tertera seperti pada **Gambar 2.2**.



Gambar 2. 2 IMU

Sensor yang digunakan pada penelitian ini adalah Yahboom ROS 10-Axis CMP10A. Modul tersebut menyediakan pengukuran percepatan, kecepatan sudut, dan medan magnet pada tiga sumbu serta mendukung pembentukan informasi orientasi. Karakteristik utama sensor yang relevan terhadap sistem navigasi ditunjukkan pada **Tabel 2.3** [18].

Tabel 2. 3 Spesifikasi IMU

Parameter	Spesifikasi
Model	CMP10A-10Axis Attitude Sensor
Akselerometer	3 sumbu
Girooskop	3 sumbu
Magnetometer	3 sumbu
Resolusi akselerometer	0,0005 g/LSB
Resolusi giroskop	0,061 °/s
Resolusi magnetometer	0,0667 mGauss/LSB

Berdasarkan **Tabel 2.3**, akselerometer digunakan untuk memperoleh percepatan pada sumbu X, Y, dan Z, sedangkan giroskop menghasilkan kecepatan sudut terhadap ketiga sumbu tersebut. Magnetometer menyediakan informasi medan magnet yang dapat digunakan sebagai referensi arah. Pada pergerakan kendaraan di bidang horizontal, kecepatan sudut sumbu Z merepresentasikan *yaw rate*, yaitu laju perubahan arah kendaraan, sedangkan nilai *yaw* menunjukkan arah hadap atau *heading*.

Salah satu proses penting pada pengolahan data giroskop adalah koreksi *bias*. Nilai kecepatan sudut terkoreksi diperoleh dengan mengurangi komponen *bias* dari hasil pengukuran giroskop. Hubungan koreksi giroskop dalam bentuk vektor dinyatakan pada [19]. Untuk komponen sumbu Z, persamaan tersebut dapat dituliskan pada persamaan (2.3) sebagai:

$$\omega_{z,k}^c = \omega_{z,k}^m - b_{\omega,z,k} \quad (2.3)$$

Keterangan:

$\omega_{z,k}^c$ =kecepatan sudut sumbu Z setelah koreksi *bias* pada sampel ke-*k* (rad/s);

$\omega_{z,k}^m$ =kecepatan sudut sumbu Z hasil pengukuran pada sampel ke-*k* (rad/s);

$b_{\omega,z,k}$ =*gyroscope bias* sumbu Z pada sampel ke-*k* (rad/s);

k =indeks sampel pengukuran.

Koreksi tersebut bertujuan mengurangi komponen kesalahan tetap pada keluaran giroskop sehingga kecepatan sudut lebih merepresentasikan rotasi sebenarnya. Dalam gerak planar, komponen sumbu Z menjadi penting karena perubahan orientasi kendaraan terutama terjadi terhadap sumbu vertikal.

Selain kecepatan sudut, orientasi dapat direpresentasikan menggunakan sudut Euler berupa *roll* (ϕ), *pitch* (θ), dan *yaw* (ψ), maupun dalam bentuk *quaternion*. Representasi *quaternion* banyak digunakan karena mampu merepresentasikan orientasi tiga dimensi secara konsisten. Hubungan antara sudut Euler dan *quaternion* dapat dinyatakan sebagai [20]:

$$\begin{aligned}
q_x &= \cos \frac{\psi}{2} \cos \frac{\theta}{2} \sin \frac{\phi}{2} - \sin \frac{\psi}{2} \sin \frac{\theta}{2} \cos \frac{\phi}{2}, \\
q_y &= \cos \frac{\psi}{2} \sin \frac{\theta}{2} \cos \frac{\phi}{2} + \sin \frac{\psi}{2} \cos \frac{\theta}{2} \sin \frac{\phi}{2}, \\
q_z &= \sin \frac{\psi}{2} \cos \frac{\theta}{2} \cos \frac{\phi}{2} - \cos \frac{\psi}{2} \sin \frac{\theta}{2} \sin \frac{\phi}{2}, \\
q_w &= \cos \frac{\psi}{2} \cos \frac{\theta}{2} \cos \frac{\phi}{2} + \sin \frac{\psi}{2} \sin \frac{\theta}{2} \sin \frac{\phi}{2}.
\end{aligned} \tag{2.4}$$

Keterangan:

q_x, q_y, q_z, q_w = komponen *quaternion* orientasi;

ϕ = sudut *roll* (rad);

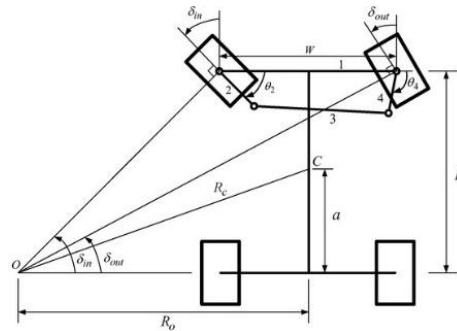
θ = sudut *pitch* (rad);

ψ = sudut *yaw* (rad).

Pada pergerakan kendaraan dua dimensi, *yaw* menjadi komponen orientasi utama karena menunjukkan arah hadap kendaraan, sedangkan kecepatan sudut sumbu Z menunjukkan laju perubahan arah tersebut. Oleh karena itu, informasi *yaw* dan kecepatan sudut sumbu Z menjadi dua keluaran IMU yang relevan untuk mendukung estimasi orientasi dan rotasi kendaraan.

2.5 Odometry Kendaraan Ackerman

Odometri merupakan metode untuk memperkirakan perubahan posisi dan orientasi kendaraan berdasarkan informasi gerak yang diperoleh selama kendaraan bergerak. Pada kendaraan dengan konfigurasi Ackermann, perubahan pose dipengaruhi oleh kecepatan longitudinal dan kondisi sudut kemudi yang ditunjukkan pada **Gambar 2.3**.



Gambar 2.3 Model Ackerman

Model kinematika Ackermann dapat merepresentasikan keadaan kendaraan menggunakan posisi x , y , dan orientasi ψ , dengan gerakan kendaraan ditentukan berdasarkan kecepatan dan geometri kemudinya [21]. Pada sistem yang dikembangkan,

kecepatan longitudinal diperoleh dari umpan balik sistem penggerak, sedangkan perubahan arah diperoleh dengan mempertimbangkan sudut kemudi serta geometri kendaraan. Kecepatan sudut kendaraan dinyatakan dengan persamaan pada (2.5) sebagai:

$$\omega_i = -\frac{v_i \text{sgn}(\delta_i)}{\frac{T}{2} + \frac{L}{\tan|\delta_i|}} \quad (2.5)$$

Keterangan:

ω_i = kecepatan sudut kendaraan pada sampel ke- i (rad/s)

v_i = kecepatan longitudinal kendaraan pada sampel ke- i (m/s)

L = *wheelbase* kendaraan (m)

T = *track width* kendaraan (m)

δ_i = sudut kemudi pada sampel ke- i (rad)

$\text{sgn}(\delta_i)$ = fungsi tanda untuk menentukan arah belok

Tanda negatif digunakan untuk menyesuaikan konvensi arah pada sistem, yaitu sudut kemudi positif menunjukkan belok kanan, sedangkan yaw positif pada ROS mengikuti arah berlawanan jarum jam.

Nilai kecepatan sudut tersebut selanjutnya digunakan untuk memperbarui orientasi kendaraan secara diskrit seperti pada persamaan (2.6):

$$\psi_k = \psi_{k-1} + \omega_k \Delta t \quad (2.6)$$

Posisi kendaraan pada sumbu x dan y diperbarui menggunakan persamaan (2.7):

$$\begin{aligned} x_k &= x_{k-1} + v_k \cos(\psi_k) \Delta t \\ y_k &= y_{k-1} + v_k \sin(\psi_k) \Delta t \end{aligned} \quad (2.7)$$

Keterangan:

x_k = posisi kendaraan pada sumbu x pada sampel ke- k (m)

y_k = posisi kendaraan pada sumbu y pada sampel ke- k (m)

x_{k-1} = posisi kendaraan pada sumbu x pada sampel sebelumnya (m)

y_{k-1} = posisi kendaraan pada sumbu y pada sampel sebelumnya (m)

ψ_k = orientasi *yaw* kendaraan pada sampel ke- k (rad)

ψ_{k-1} = orientasi *yaw* kendaraan pada sampel sebelumnya (rad)

ω_k = kecepatan sudut kendaraan (rad/s)

v_k = kecepatan longitudinal kendaraan (m/s)

Δt = selang waktu antar-sampel (s)

Proses pembaruan tersebut menghasilkan estimasi gerak relatif kendaraan terhadap posisi sebelumnya. Karena odometri diperoleh melalui proses perhitungan secara berulang, ketidakakuratan pada informasi kecepatan, sudut kemudi, maupun parameter geometrik kendaraan dapat menyebabkan kesalahan posisi terakumulasi selama kendaraan bergerak [21].

2.5 Mini PC Intel NUC

Mini PC Intel NUC merupakan perangkat komputasi berukuran ringkas yang dapat digunakan sebagai unit pemrosesan utama pada sistem robot bergerak. Intel NUC memiliki kemampuan menjalankan sistem operasi berbasis Linux dan Robot Operating System (ROS), serta dapat digunakan sebagai komputer tingkat tinggi untuk mengelola proses navigasi dan komunikasi dengan mikrokontroler. Penerapan Intel NUC pada mobile robot juga digunakan pada platform Robotont 3 sebagai komputer utama yang menjalankan Ubuntu Linux 22.04 dan ROS serta berkomunikasi dengan mikrokontroler STM32 melalui antarmuka serial [22] seperti yang ditampilkan pada **Gambar 2.4**.



Gambar 2. 4 Mini PC NUC

Mini PC yang digunakan pada penelitian ini adalah Intel NUC 11 Pro NUC11TNHi5. Perangkat tersebut menggunakan prosesor Intel Core i5 generasi ke-11, memori RAM, penyimpanan berbasis NVMe, serta berbagai antarmuka komunikasi yang mendukung integrasi dengan sensor dan sistem kendali pada Autonomous Delivery Vehicle. Spesifikasi utama Mini PC Intel NUC yang digunakan ditunjukkan pada **Tabel 2.4**

Tabel 2. 4 Spesifikasi Jetson Orin Nano

Parameter	Spesifikasi
Model	Intel NUC 11 Pro NUC11TNHi5
Arsitektur	x86-64

Parameter	Spesifikasi
CPU	Intel Core i5-1135G7 Generasi ke-11
Core / Thread	4 Core / 8 Thread
Frekuensi Maksimum	4,2 GHz
GPU	Intel Iris Xe Graphics
Memori	16 GB RAM
Penyimpanan	128 GB NVMe SSD
Jaringan	Gigabit Ethernet dan Wi-Fi 6
Antarmuka	USB 3.2, USB-C/Thunderbolt, HDMI, Ethernet
Sistem Operasi	Ubuntu 22.04 LTS
Robot Middleware	ROS 2 Humble

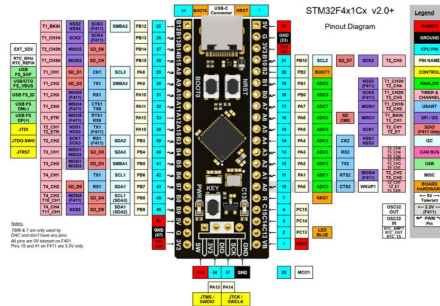
Berdasarkan **Tabel 2.4**, Intel NUC 11 Pro NUC11TNHi5 memiliki prosesor Intel Core i5-1135G7 dengan empat inti dan delapan *thread* yang mendukung pelaksanaan beberapa proses komputasi secara bersamaan. Kapasitas memori 16 GB digunakan untuk mendukung pemrosesan data selama sistem dijalankan, sedangkan media penyimpanan NVMe SSD 128 GB digunakan untuk menyimpan sistem operasi, perangkat lunak ROS 2, konfigurasi, serta data pengujian. Ketersediaan antarmuka USB, Ethernet, dan USB-C juga memungkinkan perangkat terhubung dengan sensor maupun perangkat kendali yang digunakan pada kendaraan.

Pada ADV yang dikembangkan, Mini PC Intel NUC digunakan sebagai unit komputasi utama untuk menjalankan ROS 2 serta mengolah data GNSS, IMU, dan odometri. Data tersebut selanjutnya digunakan pada proses estimasi posisi menggunakan Extended Kalman Filter, pembentukan transformasi koordinat, serta sistem navigasi menggunakan Navigation2. Mini PC juga menjalankan proses perencanaan jalur dan menghasilkan referensi gerak kendaraan, sedangkan fungsi pengendalian perangkat keras tingkat rendah dilakukan oleh mikrokontroler.

2.6 STM32F411CEU6

STM32F411CEU6 merupakan mikrokontroler 32-bit dari keluarga STM32F4 yang menggunakan inti prosesor Arm Cortex-M4. Mikrokontroler ini dilengkapi *Floating*

Point Unit (FPU) serta berbagai periferal komunikasi yang mendukung penerapan pada sistem tertanam dan kendali perangkat keras [23] yang ditunjukkan pada **Gambar 2.5**.



Gambar 2. 5 STM32F4

STM32F411CEU6 memiliki memori internal dan beberapa antarmuka komunikasi yang dapat digunakan untuk berinteraksi dengan unit komputasi, sensor, serta perangkat aktuator. Spesifikasi utama STM32F411CEU6 ditunjukkan pada **Tabel 2.5**.

Tabel 2. 5 Spesifikasi STM32F4

Parameter	Spesifikasi
CPU	Arm Cortex-M4 32-bit
Frekuensi maksimum	100 MHz
FPU	Single-precision
Flash Memory	512 KB
SRAM	128 KB
ADC	12-bit, 1 ADC
Timer	11 timer
USART	3
I ² C	3
SPI/I ² S	5
USB	USB OTG Full-Speed
GPIO	36
Tegangan operasi	1,7–3,6 V
Package	UFQFPN48

Berdasarkan **Tabel 2.5**, STM32F411CEU6 memiliki prosesor Arm Cortex-M4 dengan frekuensi maksimum 100 MHz serta FPU yang mendukung operasi numerik. Kapasitas Flash 512 KB dan SRAM 128 KB menyediakan ruang penyimpanan program dan data selama mikrokontroler bekerja. Periferal USART, I²C, SPI, timer, ADC, dan USB memungkinkan mikrokontroler melakukan komunikasi serta pengolahan data dari berbagai perangkat pada sistem kendaraan.

Pada *Autonomous Delivery Vehicle* yang dikembangkan, STM32F411CEU6 digunakan sebagai unit kendali tingkat rendah yang menghubungkan unit komputasi utama dengan perangkat aktuator. Mikrokontroler menangani komunikasi dan data umpan balik perangkat keras, kemudian informasi yang diperlukan diteruskan menuju Jetson Orin Nano untuk digunakan pada sistem navigasi. Dengan pembagian tersebut, Jetson berfungsi pada pemrosesan tingkat tinggi, sedangkan STM32F411CEU6 menangani proses yang berhubungan langsung dengan perangkat keras kendaraan.

2.7 Extended Kalman Filter

EKF merupakan metode estimasi keadaan yang digunakan pada sistem dengan model gerak atau model pengukuran yang bersifat nonlinear. EKF bekerja secara rekursif melalui proses prediksi dan koreksi dengan mempertimbangkan ketidakpastian model serta ketidakpastian pengukuran. Pada sistem lokalisasi, EKF dapat menggabungkan informasi dari beberapa sensor seperti GNSS, IMU, dan odometri untuk memperoleh estimasi keadaan kendaraan secara berkelanjutan [24].

Pada estimator, keadaan kendaraan dapat direpresentasikan dalam vektor keadaan yang mencakup posisi, orientasi, kecepatan linear, kecepatan sudut, dan percepatan linear. Vektor keadaan tersebut dinyatakan pada persamaan (2.8) sebagai:

$$\mathbf{x}_k = [x \ y \ z \ \phi \ \theta \ \psi \ v_x \ v_y \ v_z \ \omega_x \ \omega_y \ \omega_z \ a_x \ a_y \ a_z]^T \quad (2.8)$$

Keterangan:

$\mathbf{x}_k$ = vektor keadaan pada waktu ke- k

x, y, z = posisi pada masing-masing sumbu (m)

ϕ, θ, ψ = orientasi *roll*, *pitch*, dan *yaw* (rad)

v_x, v_y, v_z = kecepatan linear pada masing-masing sumbu (m/s)

$\omega_x, \omega_y, \omega_z$ = kecepatan sudut pada masing-masing sumbu (rad/s)

a_x, a_y, a_z = percepatan linear pada masing-masing sumbu (m/s²)

2.7.1 Tahap Prediksi

Tahap prediksi digunakan untuk memperkirakan keadaan kendaraan pada waktu berikutnya berdasarkan keadaan yang telah diperoleh sebelumnya. Secara umum, prediksi keadaan pada sistem nonlinear dapat dinyatakan dengan persamaan (2.9) sebagai:

$$\hat{\mathbf{x}}_k^- = f(\hat{\mathbf{x}}_{k-1}) \quad (2.9)$$

Keterangan:

$\hat{\mathbf{x}}_k^-$ = estimasi keadaan sebelum koreksi pada waktu ke- k

$\hat{\mathbf{x}}_{k-1}$ = estimasi keadaan pada waktu sebelumnya

$f(\cdot)$ = fungsi nonlinear model keadaan

Selain keadaan kendaraan, tingkat ketidakpastian hasil estimasi juga diperbarui melalui matriks *covariance*. Prediksi *covariance* dinyatakan pada persamaan (2.10) [24]:

$$\mathbf{P}_k^- = \mathbf{F}_k \mathbf{P}_{k-1} \mathbf{F}_k^T + \mathbf{Q}_k \quad (2.10)$$

Keterangan:

$\mathbf{P}_k^-$ = *covariance* keadaan hasil prediksi

$\mathbf{P}_{k-1}$ = *covariance* keadaan sebelumnya

$\mathbf{F}_k$ = matriks Jacobian model keadaan

$\mathbf{Q}_k$ = *process noise covariance*

Matriks $\mathbf{Q}$ merepresentasikan ketidakpastian pada model proses. Nilai tersebut digunakan untuk menggambarkan adanya perbedaan antara model matematis dan gerakan kendaraan sebenarnya. Proses prediksi juga memungkinkan estimator tetap memperbarui keadaan ketika pengukuran sensor belum tersedia. Kondisi ini penting pada sistem multisensor karena setiap sensor dapat memiliki laju pembaruan yang berbeda [25].

2.7.2 Tahap Koreksi

Tahap koreksi dilakukan ketika pengukuran sensor tersedia. Hasil prediksi dibandingkan dengan pengukuran untuk menentukan besarnya koreksi yang diberikan terhadap keadaan. Pengaruh pengukuran terhadap hasil estimasi ditentukan melalui *Kalman gain* seperti persamaan (2.11).

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^T (\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k)^{-1} \quad (2.11)$$

Keterangan:

K_k = *Kalman gain*

H_k = matriks Jacobian model pengukuran

R_k = *measurement noise covariance*

P_k^- = *covariance* keadaan hasil prediksi

Matriks R merepresentasikan ketidakpastian pengukuran sensor. Nilai *covariance* yang berbeda menyebabkan setiap sensor mempunyai tingkat pengaruh yang berbeda terhadap hasil estimasi. Dengan demikian, *Kalman gain* menentukan keseimbangan antara hasil prediksi dan pengukuran yang diterima.

Keadaan hasil prediksi kemudian diperbarui berdasarkan selisih antara pengukuran aktual dan pengukuran yang diperkirakan pada persamaan (2.12):

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k [\mathbf{z}_k - h(\hat{\mathbf{x}}_k^-)] \quad (2.12)$$

Keterangan:

$\hat{\mathbf{x}}_k$ = estimasi keadaan setelah koreksi

$\mathbf{z}_k$ = vektor pengukuran sensor

$h(\hat{\mathbf{x}}_k^-)$ = estimasi pengukuran berdasarkan keadaan hasil prediksi

K_k = *Kalman gain*

Setelah keadaan dikoreksi, tingkat ketidakpastian estimasi diperbarui menggunakan persamaan (2.13):

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_k^- \quad (2.13)$$

Keterangan:

$\mathbf{P}_k$ = *covariance* keadaan setelah koreksi

$\mathbf{I}$ = matriks identitas

K_k = *Kalman gain*

H_k = matriks model pengukuran

P_k^- = *covariance* keadaan hasil prediksi

Tahap prediksi dan koreksi berlangsung secara berulang selama estimator bekerja. Informasi gerak relatif dari odometri dan IMU dapat menjaga kontinuitas estimasi dalam jangka pendek, sedangkan pengukuran posisi absolut dari GNSS dapat digunakan untuk membatasi penyimpangan posisi dalam jangka panjang [26].

2.8 Robot Operating System 2 dan Navigation2

Robot Operating System 2 merupakan *framework* perangkat lunak yang digunakan untuk mengembangkan sistem robot secara modular. Setiap fungsi pada sistem dapat dijalankan melalui beberapa *node* yang saling bertukar data menggunakan mekanisme komunikasi seperti *topic*, *service*, dan *action*. Struktur tersebut memungkinkan proses pembacaan sensor, estimasi posisi, perencanaan jalur, serta pengendalian kendaraan dijalankan sebagai komponen yang terpisah tetapi tetap saling terhubung [27].



Gambar 2. 6 ROS

Pada sistem navigasi, ROS 2 menyediakan *framework* Navigation2 atau Nav2 yang digunakan untuk mengatur pergerakan robot dari posisi awal menuju posisi tujuan. Nav2 terdiri atas beberapa komponen utama, seperti *Planner Server* untuk menghasilkan jalur global, *Controller Server* untuk mengikuti jalur, *Behavior Tree Navigator* untuk mengatur proses navigasi, serta *costmap* sebagai representasi lingkungan yang digunakan selama proses perencanaan dan pengendalian [27].

Pada *Autonomous Delivery Vehicle* yang dikembangkan, ROS 2 digunakan sebagai penghubung antara data GNSS, IMU, odometri, hasil estimasi EKF, dan sistem Navigation2. Pose hasil lokalisasi digunakan sebagai informasi posisi kendaraan, kemudian Navigation2 membentuk jalur menuju *goal pose*. Global path pada penelitian ini dihasilkan menggunakan Smac Hybrid-A* yang selanjutnya dibahas pada subbab berikutnya

2.9 Peta dan Global Costmap

Peta pada sistem navigasi digunakan untuk merepresentasikan kondisi lingkungan yang menjadi ruang gerak kendaraan. Salah satu bentuk representasi lingkungan yang umum digunakan pada robot bergerak adalah *occupancy grid*, yaitu peta dua dimensi yang membagi lingkungan menjadi sejumlah sel. Setiap sel dapat merepresentasikan

area bebas (*free*), area yang ditempati hambatan (*occupied*), atau area yang belum diketahui (*unknown*) [28]. Representasi berbasis grid memungkinkan kondisi lingkungan diterjemahkan ke dalam bentuk numerik sehingga dapat digunakan sebagai dasar dalam proses lokalisasi dan perencanaan jalur.

Pada sistem navigasi, informasi lingkungan tersebut selanjutnya dapat direpresentasikan dalam bentuk *costmap*. Berbeda dengan *occupancy grid* yang terutama menunjukkan kondisi keterisian suatu sel, *costmap* memberikan nilai biaya pada area lingkungan sehingga planner dapat mempertimbangkan tingkat kelayakan suatu area untuk dilalui. Area bebas yang berada jauh dari obstacle mempunyai biaya yang lebih rendah, sedangkan area yang semakin dekat dengan obstacle memperoleh biaya yang semakin tinggi [29]. Dengan demikian, *costmap* tidak hanya digunakan untuk membedakan area yang dapat dan tidak dapat dilalui, tetapi juga untuk membentuk preferensi jalur berdasarkan jarak terhadap obstacle.

Pada implementasi *costmap*, nilai biaya disimpan dalam representasi data 8-bit tidak bertanda (*unsigned 8-bit integer*). Satu bit hanya mempunyai dua kemungkinan keadaan, yaitu 0 atau 1. Karena satu data *costmap* terdiri atas 8 bit, jumlah kombinasi nilai yang dapat direpresentasikan adalah 2 pangkat 8 maka 256 kemungkinan nilai. Penomoran data digital dimulai dari 0 sehingga rentang nilainya menjadi 0 sampai 255. Dengan demikian, angka 255 bukan diperoleh dari hasil perhitungan jarak terhadap obstacle, melainkan merupakan nilai maksimum yang dapat direpresentasikan oleh data 8-bit tidak bertanda.

Rentang nilai tersebut kemudian digunakan untuk membedakan kondisi dan tingkat biaya pada *costmap*. Nilai 0 merepresentasikan area bebas, nilai 1 sampai 252 digunakan sebagai biaya bertingkat pada daerah di sekitar obstacle, nilai 253 digunakan untuk daerah *inscribed*, dan nilai 254 merepresentasikan obstacle yang tidak dapat dilalui. Nilai 255 berada pada batas maksimum representasi 8-bit dan pada implementasi *Costmap2D* digunakan untuk menyatakan *NO_INFORMATION*, yaitu sel yang belum memiliki informasi lingkungan. Dengan pembagian tersebut, nilai 255 tidak digunakan sebagai biaya obstacle agar dapat dibedakan secara khusus dari sel yang memang diketahui sebagai obstacle.

Besarnya biaya suatu sel pada daerah di sekitar obstacle berkaitan dengan jarak sel tersebut terhadap obstacle terdekat. Jarak suatu sel p terhadap obstacle dapat dinyatakan menggunakan jarak Euclidean seperti pada Persamaan (2.14) [29].

$$d(p) = \min_{o \in O} \sqrt{(x_p - x_o)^2 + (y_p - y_o)^2} \quad (2.14)$$

Keterangan:

$d(p)$ = jarak sel p terhadap obstacle terdekat (m);

x_p = posisi sel yang dievaluasi pada sumbu x (m);

y_p = posisi sel yang dievaluasi pada sumbu y (m);

x_o = posisi obstacle pada sumbu x (m);

y_o = posisi obstacle pada sumbu y (m); dan

O = himpunan sel obstacle.

Berdasarkan Persamaan (2.14), setiap sel dapat dievaluasi berdasarkan jaraknya terhadap obstacle terdekat. Informasi jarak tersebut digunakan dalam pembentukan *inflation layer*, yaitu daerah di sekitar obstacle yang diberikan nilai biaya secara bertingkat. Sel yang berada semakin dekat dengan obstacle mempunyai biaya yang semakin tinggi, sedangkan biaya akan menurun ketika jarak sel terhadap obstacle bertambah [29].

Pada daerah di luar radius badan kendaraan, nilai biaya inflation dapat dinyatakan menggunakan fungsi eksponensial seperti pada Persamaan (2.15) [29].

$$C(d) = 252e^{-\alpha(d-r_i)} \quad (2.15)$$

Keterangan:

$C(d)$ = nilai biaya sel pada jarak d ;

252 = nilai maksimum pada zona inflation nonlethal;

α = cost scaling factor;

d = jarak sel terhadap obstacle (m); dan

r_i = inscribed radius kendaraan (m).

Berdasarkan Persamaan (2.15), nilai biaya berkurang secara eksponensial seiring bertambahnya jarak suatu sel terhadap obstacle. Pada representasi costmap yang dibahas., obstacle memiliki nilai biaya 254, sedangkan daerah kritis terhadap radius kendaraan direpresentasikan dengan nilai 253. Rentang biaya hingga 252 digunakan

untuk membentuk gradien biaya di luar daerah tersebut [29]. Mekanisme ini menyebabkan dua jalur yang sama-sama tidak mengalami collision tetap dapat memiliki tingkat biaya yang berbeda apabila memiliki jarak bebas atau *clearance* yang berbeda terhadap obstacle.

Global costmap selanjutnya digunakan oleh global planner sebagai representasi numerik kondisi lingkungan. Kandidat jalur yang melewati obstacle tidak dapat digunakan, sedangkan jalur yang berada semakin dekat dengan daerah berbiaya tinggi memperoleh biaya yang lebih besar. Dengan demikian, costmap membantu planner menghasilkan jalur yang tidak hanya menghubungkan pose awal dengan pose tujuan, tetapi juga mempertimbangkan keberadaan obstacle dan jarak bebas kendaraan terhadap lingkungan

2.10 Smac Hybrid-A* Planner

Smac Hybrid-A* merupakan algoritma *global path planning* pada Navigation2 yang dikembangkan untuk menghasilkan jalur dengan mempertimbangkan posisi, orientasi, biaya lingkungan, serta keterbatasan kinematika kendaraan. Berbeda dengan A* konvensional yang umumnya melakukan pencarian berdasarkan posisi pada grid, Hybrid-A* memasukkan orientasi kendaraan ke dalam keadaan pencarian sehingga lebih sesuai untuk kendaraan dengan karakteristik nonholonomik seperti sistem kemudi Ackermann [30].

Pada algoritma A*, pemilihan node dilakukan berdasarkan fungsi biaya persamaan (2.16):

$$f(n) = g(n) + h \quad (2.16)$$

Keterangan:

$f(n)$ = total estimasi biaya pada node n

$g(n)$ = biaya perjalanan dari titik awal menuju node n

$h(n)$ = estimasi biaya dari node n menuju titik tujuan

Hybrid-A* mengembangkan konsep tersebut dengan merepresentasikan keadaan kendaraan berdasarkan posisi dan orientasi seperti persamaan (2.17):

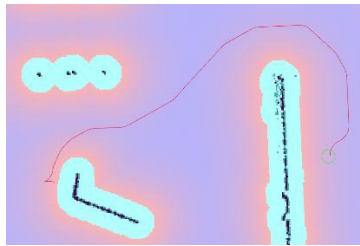
$$s = (x, y, \psi) \quad (2.17)$$

Keterangan: x = posisi kendaraan pada sumbu x (m)

y = posisi kendaraan pada sumbu y (m)

ψ = orientasi *yaw* kendaraan (rad)

Penggunaan orientasi sebagai bagian dari keadaan pencarian memungkinkan jalur yang dihasilkan mempertimbangkan perubahan arah dan keterbatasan radius putar kendaraan. Pendekatan tersebut penting pada kendaraan Ackermann karena kendaraan tidak dapat bergerak secara lateral maupun berputar pada satu titik seperti robot diferensial [31] seperti pada **Gambar 2.7**.



Gambar 2. 7 Planner

Dalam proses pencarian, Smac Hybrid-A* menggunakan *motion primitive* untuk membentuk kandidat pergerakan kendaraan. Kandidat tersebut dievaluasi berdasarkan arah gerak, batas radius putar, dan biaya pada *global costmap*. Parameter *minimum turning radius* membatasi kelengkungan jalur agar sesuai dengan kemampuan mekanik kendaraan, sedangkan diskretisasi orientasi digunakan untuk menentukan jumlah arah yang diperiksa selama proses pencarian [30].

Smac Hybrid-A* juga mempertimbangkan nilai biaya pada *costmap* serta melakukan pemeriksaan tumbukan terhadap bentuk kendaraan. Dengan demikian, jalur yang memiliki biaya tinggi atau berpotensi menyebabkan *footprint* kendaraan beririsan dengan hambatan dapat dihindari. Mekanisme *analytic expansion* dapat digunakan ketika kondisi geometris memungkinkan untuk membantu menghubungkan hasil pencarian menuju *goal pose* secara lebih efisien [30]. Pada *Autonomous Delivery Vehicle* yang dikembangkan, Smac Hybrid-A* digunakan sebagai *global planner* untuk membentuk jalur dari pose kendaraan menuju *goal pose* berdasarkan *global costmap*.

2.11 Parameter Evaluasi Sistem Navigasi

Parameter evaluasi digunakan untuk memberikan ukuran kuantitatif terhadap karakteristik data sensor, hasil estimasi *Extended Kalman Filter* (EKF), dan keluaran *global planner*.

2.11.1 Evaluasi Data Sensor

Karakteristik data sensor dapat ditinjau berdasarkan nilai rata-rata, sebaran data, dan penyimpangan terhadap nilai referensi. Nilai rata-rata digunakan untuk mewakili kecenderungan sejumlah sampel pengukuran dan dihitung menggunakan:

2.11.1 Parameter Statistik Data

Nilai rata-rata digunakan untuk menunjukkan kecenderungan pusat dari sejumlah data pengukuran. Secara matematis, nilai rata-rata dinyatakan pada persamaan (2.18) sebagai:

$$\bar{x} = \frac{1}{N} \sum_{i=1}^N x_i \quad (2.18)$$

Keterangan:

$\bar{x}$ = nilai rata-rata pengukuran

x_i = nilai pengukuran sampel ke- i

N = jumlah sampel

Selain nilai rata-rata, penyebaran data dapat dinilai menggunakan simpangan baku. Simpangan baku menunjukkan tingkat variasi sampel terhadap nilai rata-ratanya dan dinyatakan sebagai persamaan (2.19):

$$\sigma = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2} \quad (2.19)$$

Keterangan:

σ = simpangan baku

x_i = nilai pengukuran sampel ke- i

$\bar{x}$ = nilai rata-rata pengukuran

N = jumlah sampel

Nilai simpangan baku yang kecil menunjukkan bahwa data cenderung terkonsentrasi di sekitar nilai rata-rata, sedangkan nilai yang lebih besar menunjukkan penyebaran data yang lebih tinggi.

Besarnya penyimpangan antara hasil pengukuran atau estimasi terhadap suatu nilai pembandingan dapat dinilai menggunakan *Root Mean Square Error* (RMSE). RMSE

memberikan ukuran keseluruhan error dengan memberikan pengaruh yang lebih besar terhadap penyimpangan yang bernilai besar seperti persamaan (2.20) [32].

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^N e_i^2} \quad (2.20)$$

Keterangan:

$RMSE$ = *Root Mean Square Error*

e_i = error atau residual pada sampel ke- i

N = jumlah sampel

Nilai e_i merupakan selisih antara nilai yang dievaluasi dengan nilai pembanding. RMSE yang semakin kecil menunjukkan penyimpangan yang semakin rendah. Apabila nilai pembanding berasal dari *ground truth* independen, RMSE dapat digunakan untuk menggambarkan ketelitian terhadap referensi. Sebaliknya, apabila pembanding bukan *ground truth*, RMSE lebih tepat digunakan untuk menunjukkan tingkat perbedaan atau konsistensi antardata.

2.11.2 Parameter Evaluasi Estimasi Keadaan

Pada metode estimasi berbasis Kalman Filter, hasil estimasi tidak hanya berupa nilai keadaan, tetapi juga tingkat ketidakpastian yang direpresentasikan melalui matriks *covariance*. Elemen diagonal matriks tersebut merupakan *variance* dari masing-masing keadaan sehingga simpangan bakunya dapat diperoleh melalui persamaan (2.21):

$$\sigma_i = \sqrt{P_{ii}} \quad (2.21)$$

Keterangan:

σ_i = simpangan baku keadaan ke- i

P_{ii} = elemen diagonal matriks *covariance*

Nilai σ_i memberikan informasi mengenai tingkat ketidakpastian estimasi. Nilai yang lebih kecil menunjukkan tingkat ketidakpastian yang lebih rendah, sedangkan nilai yang meningkat menunjukkan bertambahnya ketidakpastian keadaan. Representasi ketidakpastian melalui *covariance* merupakan bagian penting dalam proses estimasi berbasis EKF [24].

Selain ketidakpastian, aspek temporal data perlu diperhatikan karena pengukuran pada sistem multisensor dapat diterima pada waktu yang berbeda. Umur suatu data terhadap waktu evaluasi dapat dinyatakan sebagai persamaan (2.22):

$$a_i = t_{\text{now}} - t_i \quad (2.22)$$

Keterangan:

a_i = umur data (*measurement age*) (s)

t_{now} = waktu saat evaluasi

t_i = *timestamp* data

Nilai umur data yang kecil menunjukkan informasi yang lebih baru, sedangkan nilai yang besar menunjukkan adanya keterlambatan antara waktu pengukuran dan waktu data digunakan.

Kontinuitas digunakan untuk menunjukkan tingkat ketersediaan data selama suatu periode pengamatan. Secara umum, kontinuitas dapat dihitung menggunakan persamaan (2.23):

$$C = \frac{N_{\text{available}}}{N_{\text{total}}} \times 100\% \quad (2.23)$$

Keterangan:

C = kontinuitas data (%)

$N_{\text{available}}$ = jumlah sampel yang tersedia

N_{total} = jumlah seluruh sampel pengamatan

Nilai kontinuitas yang mendekati 100% menunjukkan bahwa data tersedia pada sebagian besar periode pengamatan. Kestabilan perubahan posisi juga dapat dianalisis dari perpindahan antara dua sampel yang berurutan. Perubahan posisi maksimum dinyatakan sebagai persamaan (2.24):

$$J_{\text{max}} = \max_i \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2} \quad (2.24)$$

Keterangan:

J_{max} = perubahan posisi maksimum antarsampel (m)

x_i, y_i = posisi pada sampel ke- i

x_{i-1}, y_{i-1} = posisi pada sampel sebelumnya

Nilai perubahan posisi maksimum dapat digunakan untuk mengidentifikasi adanya lonjakan pada keluaran estimasi. Nilai yang kecil menunjukkan perubahan posisi yang

lebih kontinu, sedangkan nilai yang besar menunjukkan adanya perubahan posisi yang lebih tajam.

2.11.3 Parameter Evaluasi Perencanaan Jalur

Kinerja algoritma perencanaan jalur dapat dievaluasi berdasarkan kemampuan menghasilkan jalur secara efisien, mencapai titik tujuan, dan mempertahankan jarak terhadap hambatan. Parameter yang umum digunakan antara lain waktu perencanaan, error posisi akhir, tingkat keberhasilan, dan jarak minimum terhadap obstacle [33].

Waktu perencanaan menunjukkan durasi yang diperlukan algoritma sejak proses pencarian dimulai hingga menghasilkan keluaran pada persamaan (2.25):

$$t_{\text{plan}} = t_{\text{finish}} - t_{\text{start}} \quad (2.25)$$

Keterangan:

t_{plan} = waktu perencanaan

t_{start} = waktu mulai proses perencanaan

t_{finish} = waktu selesai proses perencanaan

Nilai waktu perencanaan yang lebih kecil menunjukkan bahwa algoritma mampu menghasilkan solusi dalam waktu yang lebih singkat. Kesesuaian titik akhir jalur terhadap posisi tujuan dapat dinilai menggunakan *endpoint error*. Nilai tersebut diperoleh dari jarak Euclidean antara titik akhir jalur dan titik tujuan oleh persamaan (2.26):

$$e_{\text{goal}} = \sqrt{(x_N - x_g)^2 + (y_N - y_g)^2} \quad (2.26)$$

Keterangan:

e_{goal} = *endpoint error* (m)

x_N, y_N = koordinat titik akhir jalur

x_g, y_g = koordinat titik tujuan

Nilai *endpoint error* yang lebih kecil menunjukkan bahwa titik akhir jalur berada semakin dekat dengan posisi tujuan yang ditentukan. Keberhasilan algoritma dalam menghasilkan solusi pada sejumlah pengujian dapat dinilai melalui *success rate* dengan persamaan (2.27):

$$SR = \frac{N_{\text{success}}}{N_{\text{test}}} \times 100\% \quad (2.27)$$

Keterangan:

$SR = \text{success rate (\%)}$

N_{success} = jumlah pengujian yang berhasil

N_{test} = jumlah seluruh pengujian

Nilai *success rate* menunjukkan tingkat keberhasilan algoritma dalam memperoleh solusi pada kondisi pengujian yang diberikan [33]. Selain keberhasilan mencapai tujuan, jarak jalur terhadap hambatan menjadi salah satu aspek dalam menilai kelayakan jalur. Jarak minimum terhadap hambatan dapat dinyatakan sebagai persamaan (2.28):

$$d_{\min} = \min\{D_i\} \quad (2.28)$$

Keterangan:

$d_{\min}$ = jarak minimum terhadap hambatan (m)

D_i = jarak titik jalur ke- i terhadap hambatan terdekat

Nilai $d_{\min}$ yang lebih besar menunjukkan bahwa jalur mempunyai jarak bebas yang lebih besar terhadap hambatan. Parameter jarak minimum dapat digunakan bersama informasi ukuran kendaraan dan representasi lingkungan untuk menilai kelayakan jalur yang dihasilkan [33]