PELACAKAN OBJEK DINAMIS MENGGUNAKAN PENGUKURAN BANYAK AGEN DENGAN LATENSI KOMUNIKASI PADA ROBOT KONTES SEPAK BOLA BERODA

Laporan Tugas Akhir

Disusun sebagai syarat kelulusan tingkat sarjana

Oleh

BIMO ADITYARAHMAN WIRAPUTRA

NIM: 13517004

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

September 2021

PELACAKAN OBJEK DINAMIS MENGGUNAKAN PENGUKURAN BANYAK AGEN DENGAN LATENSI KOMUNIKASI PADA ROBOT KONTES SEPAK BOLA BERODA

Laporan Tugas Akhir

Oleh

BIMO ADITYARAHMAN WIRAPUTRA

NIM: 13517004

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

Telah disetujui dan disahkan sebagai Laporan Tugas Akhir

di Bandung, pada tanggal 10 September 2021

Mengetahui,

Pembimbing,

Dr. Judhi Santoso, M.Sc.

NIP. 196302041989031002

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I

PENDAHULUAN

Bab ini memaparkan latar belakang yang mendasari penulisan tugas akhir ini; rumusan masalah, tujuan, dan batasan tugas akhir secara formal; metodologi pengerjaan tugas akhir ini; dan sistematika pembahasan untuk bab-bab berikutnya pada laporan tugas akhir ini.

I.1 Latar Belakang

Studi di bidang robotika merupakan salah satu bidang riset dalam lingkup intelijensi buatan yang memiliki aplikasi yang sangat luas dalam kehidupan manusia. Perkembangan robotika telah memiliki dampak yang sangat besar dalam perkembangan zaman, terutama di bidang industri manufaktur dimana robot memungkinkan pekerjaan kompleks untuk dilakukan dengan volume tinggi. Secara umum, robotika telah maupun sangat berpotensi untuk membantu menciptakan peralatan otomatis yang dapat mengerjakan pekerjaan kompleks dengan akurasi, kecepatan, ketahanan, maupun tingkat keamanan yang lebih tinggi dibandingkan dengan manusia di berbagai bidang lainnya, seperti industri pertahanan, pertambangan, sampai industri kesehatan.

Dalam studi dan pengembangan robotika, peneliti mengkaji dan mempelajari bagaimana robot dapat direkayasa untuk menyelesaikan pekerjaan yang kompleks. Lingkup studi ini pun mencakup area yang luas, dari bagaimana robot dapat berinteraksi dengan manusia sampai bagaimana robot dapat berkoordinasi dengan banyak robot lainnya dalam menyelesaikan pekerjaannya. Sebagai sistem fisik siber, robot yang baik dapat melakukan persepsi dan membaca keadaan dunia dengan akurat, melakukan komputasi dan pengambilan keputusan tingkat tinggi, dan mengeksekusi suatu aksi dengan presisi.

Salah satu inisiatif untuk membangkitkan perkembangan di bidang robotika ini adalah liga pertandingan Robocup, yang merupakan liga pertandingan

Gambar I.1.1. Tim DAGOZILLA pada Kontes Robot Indonesia

terbuka untuk tim pengembang dari universitas maupun organisasi lainnya untuk merekayasa robot-robot yang dapat ditandingkan antar tim. Diantara cabang liga yang ditandingkan dalam Robocup adalah kompetisi robot sepak bola beroda, dimana peserta mengembangkan tim robot-robot beroda yang harus berkoordinasi untuk mencetak gol di gawang lawan. Misi awal RoboCup saat didirikan adalah untuk mengumpulkan tim robot yang cukup maju untuk dapat mengalahkan tim manusia juara Piala Dunia pada tahun 2050. Inisiatif serupa juga ada di Indonesia dalam bentuk Kontes Robot Indonesia yang diselenggarakan oleh Kementrian Pendidikan dan Kebudayaan antar tim dari universitas-universitas Indonesia, seperti tim DAGOZILLA dari Institut Teknologi Bandung seperti pada gambar I.1.1.

Dalam rekayasa robotika, pembuatan modul persepsi yang akurat merupakan hal yang penting karena estimasi *world state* atau keadaan dunia merupakan masukan robot untuk melakukan pengambilan keputusan dan aksi, sehingga kualitas estimasi yang buruk dapat mengakibatkan pengambilan keputusan yang salah dan membahayakan keamanan robot itu sendiri, pengguna, maupun orang lain yang mungkin terlibat. Dalam konteks robot sepak bola beroda, pelacakan pergerakan robot sendiri (yang disebut juga lokalisasi) maupun pergerakan objek lain yaitu robot teman, bola, maupun robot musuh merupakan informasi yang penting yang digunakan di banyak level dari pengambilan keputusan dan strategi tim sampai fungsionalitas dasar seperti

mengejar, menggiring, mengoper, dan menembak bola.

Dalam kontes robot sepak bola, tidak terbacanya posisi suatu objek merupakan hal yang sering terjadi karena batasan jarak efektif sensor maupun halangan visual dari objek lain. Oleh karena itu, selain pengolahan data dari sensor yang dimiliki robot sendiri, pertukaran dan pengolahan data dari teman juga merupakan hal yang penting dalam persepsi *world state* untuk mendapatkan estimasi yang selengkap dan seakurat mungkin. Sayangnya, keterlambatan atau kegagalan penyampaian informasi antar robot anggota tim dapat terjadi karena inferensi jaringan, terutama di tempat kontes yang memiliki banyak penonton, sehingga algoritma pengolahan informasi harus didesain dengan baik untuk dapat tetap mengestimasi *world state* dengan baik tanpa ataupun dengan menggunakan data yang terlambat dari teman.

Telah terdapat banyak penelitian mengenai cara penggabungan persepsi keadaan dari banyak robot di dalam maupun luar konteks robot sepak bola beroda, dan ada beberapa studi modifikasi algoritma estimasi sensor dengan masukan data yang mungkin terlambat, akan tetapi masih sedikit penelitian yang membahas dan menguji coba penggabungan persepsi keadaan dari banyak robot dalam konteks robot sepak bola beroda yang mempertimbangkan latensi di komunikasi.

Dalam tugas akhir ini, dikembangkan dan diuji coba modul estimasi persepsi *world state* di lingkungan robot anggota tim untuk menangani kondisi komunikasi berlatensi pada konteks Kontes Robot Indonesia cabang sepak bola beroda.

I.2 Rumusan Masalah

Sesuai dengan latar belakang tersebut, masalah yang diselesaikan pada tugas akhir ini adalah mendesain, mengembangan, dan menguji coba algoritma estimasi persepsi *world state* menggunakan pertukaran informasi banyak robot dengan latensi komunikasi, spesifik pada kasus pengembangan robot sepak bola beroda di Kontes Robot Indonesia. Untuk menyelesaikan permasalahan tersebut, masalah dibagi menjadi beberapa submasalah:

1. Bagaimana memproses data dari sensor robot itu sendiri
2. Bagaimana memproses informasi yang diterima dari teman yang mungkin terlambat karena latensi komunikasi

I.3 Tujuan

Tujuan dari tugas akhir ini mengembangkan algoritma estimasi persepsi *world state* dari robot anggota tim yang memberikan estimasi yang akurat, bahkan dalam keadaan kondisi jaringan komunikasi yang berlatensi. Berdasarkan dari submasalah yang ada, terdapat juga subtujuan:

1. Mengembangkan algoritma pengolahan dan pelacakan data sensor robot sendiri yang akurat
2. Mengembangkan algoritma integrasi informasi yang mungkin terlambat dari teman yang menghasilkan estimasi yang akurat

I.4 Batasan Masalah

Batasan masalah yang diambil untuk membatasi lingkup penelitian di tugas akhir ini adalah:

1. *World state* yang diestimasi mencakup pergerakan posisi dan kecepatan robot anggota tim dan juga bola pada saat ini di lapangan.
2. Eksperimen dilakukan dalam lingkungan simulasi yang dikembangkan sendiri sesuai dengan pemodelan mengikuti keadaan nyata dalam tim DAGOZILLA ITB dan Kontes Robot Indonesia.
3. Algoritma yang dikembangkan tidak mencakup teknik dalam bidang *Machine Learning* maupun *Neural Network*.

I.5 Metodologi

Pengerjaan tugas akhir dibagi menjadi beberapa tahapan:

1. Pemodelan lingkungan nyata dan analisis solusi

   Pada tahap ini, dimodelkan lingkungan dunia nyata tempat robot bekerja, termasuk distribusi *error* dari sensor dan distribusi latensi

dari komunikasi. Dilakukan juga analisis masalah dan solusi untuk mendapatkan kerangka besar alur algoritma yang digunakan.

2. Pembuatan lingkungan simulasi

   Pada tahap ini, dibuat lingkungan simulasi yang dapat disambungkan dengan lingkungan kode robot untuk melakukan pengujicobaan dengan profil *error* dan latensi sesuai dengan pemodelan yang ada.

3. Pengembangan algoritma solusi dan evaluasi

   Pada tahap ini, diimplementasi dan diuji coba algoritma-algoritma hipotesis solusi yang dicetuskan untuk dibandingkan berdasarkan hasil evaluasinya dalam lingkungan simulasi, untuk mendapatkan algoritma yang paling akurat.

I.6 Sistematika Pembahasan

Selain Bab I Pendahuluan ini yang membahas mengenai latar belakang, rumusan masalah, dan metodologi, terdapat lima bab pada laporan tugas akhir ini dengan bahasannya masing-masing.

Bab II Studi Literatur membahas mengenai dasar teori yang didapatkan dari studi literatur yang digunakan untuk menyusun solusi dalam tugas akhir ini, baik dalam pemodelan dan pembuatan lingkungan simulasinya maupun solusi itu sendiri.

Bab III Analisis Solusi dan Arsitektur Sistem mendeskripsikan secara rinci masalah yang dihadapi, analisis dan desain solusi yang digunakan, juga rancangan umum arsitektur lingkungan simulasi tempat algoritma persepsi *world state* berjalan.

Bab IV Eskperimen dan Analisis menjelaskan implementasi simulasi, eksperimen beserta hasil evaluasi dan analisisnya.

Bab V Penutup merangkum dan menyimpulkan hasil eksperimen dan analisis solusi yang diuji coba, beserta saran untuk penelitian lanjutan mengenai topik terkait persepsi *world state* dalam keadaan latensi komunikasi.

BAB II

STUDI LITERATUR

Bab ini memberikan pengetahuan prasyarat yang mendasari isi dari tugas akhir ini. Secara spesifik, bab ini membahas ROS sebagai *framework* yang digunakan; teknik pemodelan dalam robotika dan komunikasi; estimasi keadaan probabilistik berupa penapis Bayes, Kalman, dan partikel; dan studi terkait yang sudah ada pada topik estimasi keadaan multiagen maupun estimasi keadaan dengan latensi data.

II.1 Robot Operating System

Robot Operating System atau yang disingkat sebagai ROS merupakan *framework* atau *meta-operating system* yang sering digunakan dalam pengembangan perangkat lunak dalam bidang robotika. ROS dikembangkan untuk memenuhi kebutuhan yang ditemui dalam mengembangkan robot layanan skala besar, dan menyediakan berbagai fungsionalitas yang berguna dalam konteks pengembangan robotika tersebut, seperti sistem komunikasi pertukaran pesan antar proses atau yang disebut *node*, kontrol perangkat level rendah, abstraksi perangkat keras, manajemen *package* dan berbagai kakas dan algoritma yang berguna dalam robotika.

Berdasarkan Quigley dkk. (2009), filosofi pengembangan ROS bisa dibagi menjadi beberapa poin, yaitu:

1. *Peer-to-peer*

   Sistem yang dibangun menggunakan ROS terdiri dari beberapa *node*, yang mungkin berada di beberapa komputer berbeda, yang terhubung dan berkomunikasi pada saat *runtime* dalam suatu jaringan *peer-to-peer*. Jenis jaringan ini dianggap lebih baik dibanding jenis topologi menggunakan

2. Multilingual

ROS menyediakan antarmuka dalam beberapa bahasa seperti C++, Python, Octave, dan LISP sehingga pengembang dalam ROS dapat memilih untuk menggunakan bahasa yang sesuai kebutuhan untuk masing-masing *node*. Definisi pesan yang digunakan dalam pertukaran pesan menggunakan *interface definition language* (*IDL*) sederhana yang oleh ROS diterjemahkan menjadi definisi kelas dalam masing-masing bahasa pemrograman yang didukung.

3. Berbasis kakas

   Fungsionalitas kakas pada ROS tersedia dalam paradigma mikrokernel dan disebar dalam berbagai modul-modul terpisah dan modul *master* utama memiliki fungsionalitas sekecil mungkin agar program masih dapat berjalan. Dengan desentralisasi fungsionalitas ini, dianggap berkurangnya efisiensi program dapat dikompensasi dengan stabilitas dan kemudahan mengelola kompleksitas program.

4. Tipis

   Seperti pada kakasnya, ROS menganjurkan pengembangan algoritma dan fungsionalitas lainnya pada suatu *library* secara terpisah dengan ROS dengan memanfaatkan sistem *build* berbasis CMake, dan hanya menggunakan ROS sebagai fungsionalitas tipis untuk mengatur konfigurasi dan menyalurkan data. Dengan seperti ini, modul yang dikembangkan lebih mudah untuk ditest maupun digunakan ulang pada tempat lain.

5. Bebas dan sumber terbuka

   Framework ROS memiliki lisensi BSD sehingga pengembang dapat menggunakannya untuk proyek komersil dan non-komersil. ROS yang bersifat bebas dan sumber terbuka ini juga mendorong penggunanya untuk saling berkontribusi kepada ROS dan pengguna lainnya dalam bentuk bantuan *debugging*, pengajuan fitur, dan kakas sumber terbuka lainnya.

Pada praktisnya, pengembang ROS membangun program dalam struktur mikroservis dimana dibangun beberapa *node* dengan tanggung jawabnya masing-masing yang saling berkomunikasi menggunakan *message-passing*. Pertukaran pesan dilakukan di bawah suatu nama topik dan suatu jenis pesan, dimana suatu *node* dapat mengirimkan pesan di bawah topik tersebut, dan semua *node* yang mengikuti topik tersebut akan menerima dan memproses pesan tersebut dalam bentuk pemanggilan fungsi *callback*. Selain pertukaran pesan, fitur ROS yang sering digunakan adalah antarmuka *parameter server* untuk menyuplai parameter konfigurasi yang dibutuhkan oleh masing-masing *node*, dan *node-node* utilitas seperti perekaman dan putar balik pesan, visualisasi data pesan, pembangkitan otomatis dokumentasi, dan lain sebagainya.

II.2 Pemodelan Robotika dan Komunikasi

II.2.1 Pergerakan Bola dan Robot

Dalam pelacakan objek dinamis, hal yang penting untuk dilacak dari objek tersebut pada suatu waktu diantaranya adalah posisi, atau tempat objek itu berada, dan kecepatan, yang merupakan perubahan seketika dari posisi. Dalam kasus kontes robot beroda dimana objek seperti robot, dan bola berada pada bidang planar lapangan, posisi dan juga kecepatan dapat dimodelkan dengan vektor dua dimensi (x, y) dalam suatu sistem koordinat yang sudah ditentukan sebelumnya yang diukur pada pusat massa objek. Vektor posisi dan kecepatan tersebut dapat dimanipulasi secara matematis diantara dapat dijumlahkan seperti saat menghitung perpindahan atau resultan kecepatan, ataupun dikalikan dengan skalar seperti dalam menghitung perpindahan dari kecepatan dan waktu.

Untuk robot yang memiliki orientasi, digunakan istilah *pose* untuk menandakan keadaan posisi dan orientasi dari suatu robot. Seperti yang dijelaskan dalam Thrun dkk. (2010), dalam kasus dua dimensi, *pose* memiliki dua komponen (p, θ) dimana komponen v menyatakan perpindahan posisi dan θ menandakan perpindahan orientasi dari robot tersebut, terhadap suatu

sistem koordinat. Perpindahan dari dua *pose* dapat dianggap banyaknya translasi dan rotasi yang perlu dilakukan robot terhadap *pose* pertama untuk mencapai *pose* kedua. Karena arah perpindahan posisi yang dilakukan relatif terhadap orientasi dari robot, maka penjumlahan dari dua *pose* harus mempertimbangkan perubahan orientasi tersebut, dalam bentuk perputaran arah translasi kedua sebesar rotasi pertama. Cara kerja aritmatika yang unik inilah yang membuat manipulasi *pose* tidak bersifat komutatif, melainkan hanya asosiatif seperti pada konstruk matematis bernama grup.

Perubahan seketika dari *pose* disebut sebagai *twist*, yang terdiri dari kecepatan dan kecepatan sudut dari robot tersebut. Seperti *pose*, *twist* memiliki dua komponen yang sama (p, θ). Akan tetapi, karena perubahan posisi dan orientasi pada *twist* terjadi secara bersamaan, penjumlahan dua *twist* hanyalah penjumlahan komponen biasa. *Twist* juga dapat diintegrasi terhadap waktu untuk mendapatkan *pose*, dimana *twist* konstan (p, θ) yang diintegrasi terhadap waktu d akan menjadi *pose* dengan nilai

$$\left(\frac{v(i - i\mathrm{cis}\,\theta d)}{\theta}, \theta d\right) \tag{II.1}$$

dimana vektor dua dimensi (x, y) di sini bisa diinterpretasi sebagai bilangan kompleks $x + iy$, $\mathrm{cis}\,\theta$ menyatakan $\cos\theta + i\sin\theta = (\cos\theta, \sin\theta)$ dan i adalah $(0, 1)$ dengan perkalian vektor sebagai perkalian kompleks.

II.2.2 Metrik *Error* Pergerakan Objek

Dalam analisis algoritma estimasi pergerakan objek, dibutuhkan suatu metrik untuk menghitung *error* diantara nilai pergerakan estimasi dengan nilai pergerakan sesungguhnya. Pada posisi maupun kecepatan bola yang berupa vektor dua dimensi, secara natural dapat digunakan metrik *error* berupa jarak antara vektor yang bernilai $\sqrt{x^2 + y^2}$ dalam satuan meter. Untuk metrik estimasi pergerakan bola secara utuh dapat menggunakan penggabungan dari jarak posisi dan jarak kecepatan, seperti dengan menggunakan *root mean square* (*RMS*) dari keduanya.

Dalam menentukan metrik *error* untuk estimasi pergerakan robot yang memiliki orientasi, dapat digunakan definisi jarak antara dua *pose* dari Brégier dkk. (2018) dimana jarak antara *pose* dari dua *rigid body* didefinisikan sebagai

$$\sqrt{\frac{1}{M}\int |T_2(x) - T_1(x)|^2 dm} \tag{II.2}$$

dimana S adalah massa objek dan $|T_2(x) - T_1(x)|$ adalah jarak dari titik pada objek yang koresponden pada kedua *pose*. Perhatikan bahwa persamaan ini merupakan *RMS* dari jarak antara semua titik yang ada pada objek tersebut.

Karena objek robot hanya melakukan translasi dan rotasi di bidang dua dimensi, persamaan jarak tersebut dapat disederhanakan. Karena referensi koordinat dalam penentuan jarak tidak penting, tempatkan *pose* pertama di *origin* $(O, 0)$ dengan selisih *pose* (p, θ), maka titik x di *pose* pertama akan berkorespon dengan titik $p + x\mathrm{cis}\,\theta$. Maka diturunkan persamaan jarak menjadi

$$\sqrt{\frac{1}{M}\int |p + x\mathrm{cis}\,\theta - x|^2 dm} \tag{II.3}$$

$$= \sqrt{|p|^2 + |\mathrm{cis}\,\theta - 1|^2 I/M} \tag{II.4}$$

dengan mengekspansi ekspresi panjang kuadrat di dalam integral dan memperhatikan bahwa $\frac{1}{M}\int x dm$ sama dengan pusat massa objek yang bernilai 0, dan $\int |x|^2 dm = I$ adalah momen inersia dari objek. Untuk objek robot dengan distribusi massa kurang lebih diantara kerucut dan silinder, I/M bernilai diantara $0.3r^2$ sampai $0.5r^2$ dimana r adalah jari-jari objek.

II.2.3 Distribusi Normal

Keluarga distribusi normal atau distribusi Gaussian adalah salah satu distribusi probabilistik yang sering dipelajari dimana dengan nilai rata-rata μ dan nilai variansi σ^2 memiliki fungsi densitas

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right\} \tag{II.5}$$

dengan kasus spesial dimana $\mu = 0$ dan $\sigma^2 = 1$ disebut juga distribusi normal standar.

Distribusi normal banyak dipelajari karena beberapa alasan. Salah satu alasan adalah karena distribusi ini memiliki banyak properti yang membuatnya mudah dimanipulasi dan digunakan, diantaranya adalah fungsi distribusi penjumlahan variabel-variabel acak independen berdistribusi normal, maupun perkalian dan konvolusi fungsi normal juga merupakan fungsi normal. Alasan lainnya adalah diobservasi juga berbagai variabel acak di berbagai pengujian ilmiah yang ternyata memiliki distribusi yang hampir mirip dengan normal. Selain itu juga, terbukti bahwa berdasarkan Teorema Limit Pusat, rata-rata dari banyak variabel acak distribusi identik independen akan mendekati distribusi normal. (DeGroot & Schervish, 2012)

Disebabkan oleh alasan-alasan tersebut, praktisnya distibusi normal banyak digunakan untuk memodelkan berbagai variabel acak di dunia nyata, seperti distribusi sampel dari populasi yang sangat besar maupun distribusi *error* pada suatu pengukuran atau kontrol sering diasumsikan memiliki distribusi normal.

Distribusi normal memiliki generalisasi ke dalam vektor acak yaitu kumpulan variabel acak yang direpresentasikan dalam bentuk vektor atau disebut juga distribusi normal multivariat. Dengan parameter vektor rata-rata μ dan matriks kovarian Σ yang bersifat simetrik dan positif semidefinit, didefinisikan fungsi densitas

$$p(x) = det(2\pi\Sigma)^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right\} \tag{II.6}$$

dimana masing-masing komponen X_i sendiri memiliki distribusi normal dengan rata-rata μ_i dan kovarian antara X_i, X_j sama dengan $\Sigma_{i,j}$.

II.2.4 Pemodelan *Error* Sensor

Beberapa sensor yang digunakan dalam kontes robot sepak bola beroda diantaranya adalah sensor odometri pada roda yang digunakan untuk

mengukur perpindahan dari robot, *IMU* untuk mengukur orientasi dari robot, dan kamera yang dapat mempersepsi posisi relatif dari berbagai jenis objek, seperti *landmark* di sekitar lapangan, bola, dan robot lainnya.

Dengan mengasumsikan *error* dari sensor memiliki distribusi normal independen, maka variansi dari *error* berkembang secara linear terhadap banyak *error* atau data yang terakumulasi. Oleh karena itu, selain pada bacaan absolut seperti orientasi, *error* dari bacaan seperti odometri memiliki distribusi normal dengan variansi senilai suatu konstanta dikali besar perpindahan sesungguhnya. Sedangkan *error* dari bacaan kamera atau *vision* memiliki variansi senilai suatu konstanta dikali jarak sesungguhnya dari objek tersebut.

II.2.5 Pemodelan Latensi Komunikasi

Keadaan latensi komunikasi dalam jaringan *wireless* maupun Internet sering dimodelkan sebagai *hidden Markov model* (*HMM*), dimana jaringan dimodelkan memiliki suatu *state* keadaan yang tidak dapat diobservasi secara langsung dan berubah-ubah seiring waktu yang mempengaruhi besarnya latensi dan peluang *packet loss* pada saat itu. Salvo Rossi dkk. (2006) menunjukkan bahwa latensi dan kemungkinan *packet loss* dapat dimodelkan menggunakan *HMM* dengan dua sampai empat *hidden state*. Selain peluang *packet loss*, masing-masing *state* tersebut memiliki parameter latensi komunikasi yang dimodelkan sebagai distribusi *shifted gamma*, yang merupakan distribusi gamma biasa ditambah suatu nilai minimum.

II.3 Estimasi Keadaan Probabilistik

Menurut Thrun dkk. (2010), penggunaan model probabilistik dalam mengestimasi *world state* seiring waktu menjadi lebih populer dibandingkan dengan penggunaan model perhitungan yang deterministik. Dengan memodelkan *world state* yang diketahui menjadi suatu distribusi peluang, hasil pengukuran *world state* menjadi lebih tahan terhadap efek *error* pada data pengukuran.

II.3.1 Teorema Bayes

Analog dengan definisi yang ada pada probabilitas kejadian, didefinisikan peluang kondisional

$$p(x \mid y) = Pr(X = x \mid Y = y) = \frac{p(x, y)}{p(y)} \tag{II.7}$$

yang digunakan untuk memodelkan peluang nilai x terjadi apabila nilai y terjadi. Dua variabel acak X dan Y disebut independen apabila untuk semua kemungkinan x dan y berlaku

$$p(x, y) = p(x)p(y) \quad atau \quad p(x \mid y) = p(x) \tag{II.8}$$

Terkait peluang kondisional, Teorema Bayes menyatakan bahwa

$$p(x \mid y) = \frac{p(y \mid x)\, p(x)}{p(y)} = \eta\, p(y \mid x)\, p(x) \tag{II.9}$$

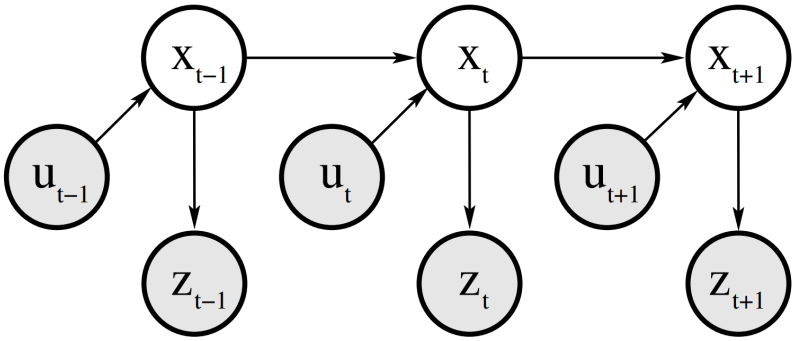
dimana $\eta = p(y)^{-1}$ merupakan suatu nilai yang konstan untuk semua kemungkinan x. Karena jumlah dari semua nilai $p(x \mid y)$ haruslah bernilai satu, η disebut juga faktor normalisasi dan dapat ditentukan kemudian sebagai jumlah dari nilai $p(y \mid x)\, p(x)$ untuk semua kemungkinan x.

Teorema Bayes sangat berguna untuk memperbarui kepercayaan terhadap distribusi peluang x setelah mengetahui terjadinya y apabila diketahui peluang terjadinya y dapat ditentukan untuk setiap kemungkinan x.

II.3.2 Pemodelan Keadaan

Dalam model probabilistik, *world state* disimbolkan sebagai suatu vektor acak X, dimana x_t menandakan *world state* pada waktu ke-t. Vektor acak ini mengandung berbagai variabel acak seperti posisi dan orientasi sesungguhnya dari robot maupun objek lain pada suatu waktu.

Seiring perubahannya terhadap waktu, informasi berkaitan dengan perubahan *world state* x_{t-1} ke x_t yang tersedia disebut sebagai data kontrol dan disimbolkan sebagai z_t. Contoh dari data kontrol misalnya perintah kecepatan



Gambar II.3.1. Hubungan X, U, Z (Thrun dkk., 2010)

pada roda robot ataupun bacaan perpindahan robot dari sensor odometri. Sedangkan informasi berkaitan keadaan *world state* x_t pada suatu waktu disebut sebagai data pengukuran dan disimbolkan sebagai u_t, seperti bacaan kompas atau persepsi *landmark*. Z dan U yang sebenarnya merupakan variabel acak juga memiliki hubungan dengan X seperti yang digambarkan pada II.3.1.

Perhatian dari estimasi keadaan probabilistik adalah mengestimasi nilai dari x_t diberikan informasi z dan u yang terjadi sebelumnya. Dalam persamaan matematika, hendak dihitung

$$bel(x_t) = p(x_t \,|\, z_{1:t}, u_{1:t}) \,, \tag{II.10}$$

. Adapun definisi distribusi lainnya dimana diprediksi nilai dari x_t sebelum mendapat hasil pengukurannya z_t,

$$\overline{bel}(x_t) = p(x_t \,|\, z_{1:t-1}, u_{1:t}) \,. \tag{II.11}$$

Mengasumsikan nilai X mengandung semua *world state* yang relevan terhadap sistem pada waktu sekarang, maka tidak ada informasi tambahan yang bisa diberikan oleh pengukuran z_t maupun keadaan sebelumnya x_{t-k} terhadap keadaan berikutnya x_{t+1} di luar apa yang terkandung pada x_t. Observasi ini disebut sebagai asumsi Markov. Oleh karena itu dalam pembangkitan nilai x_t dan z_t, dua distribusi peluang yang penting diperhatikan adalah model

peluang transisi keadaan atau *motion model*

$$p(x_t \mid x_{0:t-1}, z_{1:t-1}, u_{1:t}) = p(x_t \mid x_{t-1}, u_t) \tag{II.12}$$

dan model peluang pengukuran atau *measurement model*

$$p(z_t \mid x_{0:t}, z_{1:t-1}, u_{1:t}) = p(z_t \mid x_t) \tag{II.13}$$

II.3.3 Penapis Bayes

Algoritma II.1 Penapis Bayes

1: **function** BAYES_FILTER($bel(x_{t-1}), u_t, z_t$)
2: **for all** x_t **do**
3: $\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1})\, bel(x_{t-1})\, dx_{t-1}$
4: $bel(x_t) = \eta\, p(z_t \mid x_t)\, \overline{bel}(x_t)$ ▷ η menormalisasi nilai dari $bel(x_t)$
5: **end for**
6: **return** $bel(x_t)$
7: **end function**

Integrasi informasi z atau perubahan distribusi dari $bel(x_{t-1})$ ke $\overline{bel}(x_t)$ dapat ditentukan dari model transisi $p(x_t \mid x_{t-1}, u_t)$. Sedangkan integrasi informasi u atau perubahan distribusi dari $\overline{bel}(x_t)$ ke $bel(x_t)$ dapat ditentukan dari model pengukuran $p(z_t \mid x_t)$ menggunakan Teorema Bayes. Memanfaatkan fakta tersebut, dirumuskan algoritma penapis Bayes atau *Bayes filter algorithm* pada algoritma II.1. Baris tiga menunjukkan perhitungan nilai $bel(x_t)$ sebagai ekspansi dari penjumlahan nilai $p(x_t, x_{t-1})$ saat diketahui u_t untuk semua kemungkinan x_{t-1}, spesifik untuk kasus kontinu. Tahap ini sering disebut tahap prediksi. Baris empat memanfaatkan teorema Bayes untuk mendapatkan distribusi x_t setelah mendapatkan informasi dari z_t. Tahap ini sering disebut tahap pembaruan pengukuran.

Algoritma penapis digunakan untuk mengiterasi nilai dari $bel(x_t)$ seiring waktu, dan merupakan algoritma yang optimal mengasumsikan proses merupakan *hidden Markov model*. Perhatikan bahwa algoritma ini

mengharuskan pemodelan sendiri nilai dari model transisi, model pengukuran, dan distribusi awal $bel(x_0)$ yang akan digunakan.

Selain itu, karena nilai dari $bel(x_t)$ merupakan distribusi peluang yang umumnya berbentuk fungsi dari ruang kontinu ke bilangan riil yang tidak mungkin dimodelkan secara langsung dalam komputer, maka dalam aplikasinya dibutuhkan pendekatan untuk mendiskritkan distribusi tersebut. Pada praktisnya, distribusi peluang $bel(x_t)$ dimodelkan sebagai distribusi dengan jenis yang dapat diparameterasikan seperti pada penapis Kalman, atau distribusi tersebut didiskritkan seperti pada penapis histogram atau penapis partikel.

II.3.4 Penapis Kalman

Pada penapis Kalman atau *Kalman Filter* (*KF*), distribusi dari $bel(x_t)$ dan $\overline{bel}(x_t)$ diasumsikan memiliki distribusi normal dengan parameter μ dan σ^2 yang dicari pada setiap iterasi algoritmanya. Agar penapis Bayes normal bekerja, distribusi model transisi dan model pengukuran harus memiliki restriksi tambahan.

Pada variasi penapis Kalman yang paling sederhana, direstriksi bahwa (1) vektor acak x_t merupakan kombinasi linear dari x_{t-1}, u_t, dan faktor *error* ϵ_t

$$x_t = A_t x_{t-1} + B_t u_t + \epsilon_t \tag{II.14}$$

untuk x_t berdimensi n, u_t berdimensi m, A_t matriks berdimensi $n \times n$, dan B_t berdimensi $n \times m$, dan ϵ_t vektor acak normal dengan rata-rata nol dan kovariansi R_t; (2) z_t merupakan kombinasi linear dari x_t dan faktor *error* δ_t

$$z_t = C_t x_t + \delta_t \tag{II.15}$$

untuk z_t berdimensi k, C_t berdimensi $k \times n$, dan δ_t vektor acak normal dengan rata-rata nol dan kovariansi Q_t; dan (3) distribusi awal $bel(x_0)$ memiliki distribusi normal. Ketiga asumsi ini menjamin bahwa distribusi $bel(x_t)$ merupakan distribusi normal untuk semua waktu t yang akan datang.

Algoritma II.2 Penapis Kalman

1: **function** KALMAN_FILTER($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$)
2: $\quad \overline{\mu}_t = A_t\,\mu_{t-1} + B_t\,u_t$
3: $\quad \overline{\Sigma}_t = A_t\,\Sigma_{t-1}\,A_t^T + R_t$
4: $\quad K_t = \overline{\Sigma}_t\,C_t^T\,(C_t\,\overline{\Sigma}_t\,C_t^T + Q_t)^{-1}$
5: $\quad \mu_t = \overline{\mu}_t + K_t(z_t - C_t\,\overline{\mu}_t)$
6: $\quad \Sigma_t = (I - K_t\,C_t)\,\overline{\Sigma}_t$
7: $\quad$ **return** μ_t, Σ_t
8: **end function**

Berdasarkan hubungan di atas, algoritma penapis Kalman menghitung nilai dari parameter rata-rata μ_t dan kovariansi Σ_t dari distribusi normal $bel(x_t)$. Sehingga dapat diturunkan algoritma penapis Kalman seperti pada algoritma II.2.

Disebabkan restriksi linearitas pada model transisi dan pengukuran, algoritma penapis Kalman tidak dapat digunakan apabila restriksi linearitas tersebut tidak dipenuhi. Agar penapis Kalman dapat digunakan untuk kasus $x_t = g(u_t, x_{t-1}) + \epsilon_t$ dan $z_t = h(x_t) + \sigma_t$ dimana fungsi g dan h tidak linear, fungsi tersebut harus diaproksimasi terlebih dahulu agar menjadi linear.

Pada *Extended Kalman Filter*, fungsi dilinearkan menggunakan turunan parsial dari fungsi g dan h. Dengan menghitung $G_t = g'(u_t, \mu_{t-1})$ dan $H_t = h'(\overline{\mu}_t)$ dimana g' merupakan turunan parsial g terhadap x_{t-1}, didapatkan persamaan linear

$$g(u_t, x_{t-1}) \approx g(u_t, \mu_{t-1}) + G_t\,(x_{t-1} - \mu_{t-1}) \tag{II.16}$$

$$h(x_t) \approx h(\overline{\mu}_t) + H_t\,(x_t - \overline{\mu}_t) \tag{II.17}$$

Nilai A dan C pada penapis Kalman dapat digantikan dengan nilai G dan H.

Pada *Unscented Kalman Filter*, fungsi dilinearkan dengan mengambil sampel nilai pada fungsi g dan h, lalu membuat persamaan linear dengan melakukan regresi pada nilai masukan secara umum. Sampel diambil di rata-rata μ dan dua titik di sekelilingnya untuk setiap dimensi yang ada.

Selain *EKF* dan *UKF*, karena populeritasnya, sudah banyak modifikasi dari penapis Kalman lainnya seperti *Cubature Kalman Filter* atau *Rao-Blackwellised Unscented Kalman Filter*.

II.3.5 Penapis Partikel

Pada penapis partikel, distribusi peluang nilai dari $bel(x_t)$ direpresentasikan dengan menyimpan koleksi sejumlah nilai x_t atau disebut sebagai partikel, dimana distribusi dari partikel tersebut mengaproksimasi distribusi dari $bel(x_t)$ yang sesungguhnya. Semakin banyak partikel yang disimpan, semakin akurat algoritma penapis partikel ini, tetapi semakin besar juga sumber daya memori dan waktu yang dibutuhkan.

Algoritma II.3 Penapis Partikel

1: **function** PARTICLE_FILTER(χ_{t-1}, u_t, z_t)
2: $\overline{\chi}_t = \overline{\chi}_t = \emptyset$
3: **for** $m = 1$ to M **do**
4: sample $x_t^{[m]} \sim p(x_t \mid u_t, x_{t-1}^{[m]})$
5: $w_t^{[m]} = p(z_t \mid x_t^{[m]})$
6: add $(x_t^{[m]}, w_t^{[m]})$ to $\overline{\chi}_t$
7: **end for**
8: **for** $m = 1$ to M **do**
9: draw i with probability $\propto w_t^{[i]}$
10: add $x_t^{[i]}$ to χ_t
11: **end for**
12: **return** χ_t
13: **end function**

Algoritma penapis partikel digambarkan di algoritma II.3. Baris empat merupakan tahap prediksi dimana untuk setiap partikel sebelumnya dibangkitkan partikel sekarang menggunakan nilai kontrol. Partikel tersebut disimpan di himpunan $\overline{\chi}_t$ beserta evaluasi peluangnya atau *weight*nya berdasarkan pengukuran dan model pengukuran yang ada. Agar suatu partikel dengan peluang yang lebih besar memiliki lebih banyak representasi dalam himpunan χ_t, dilakukan sampling ulang terhadap x_t dengan peluang sebanding dengan hasil evaluasinya, pada baris delapan sampai sebelas.

Secara umum, penapis partikel ini merupakan pilihan paling populer dalam melakukan estimasi terhadap keadaan nonnormal, karena komputasinya yang mengandalkan sampling relatif lebih mudah dan tingkat akurasi terhadap kinerja algoritma dapat diatur dengan mudah melalui banyak partikel yang digunakan. Ada beberapa kondisi yang mengakibatkan ketidakakuratan pada algoritma ini, seperti saat terjadi konvergensi partikel di nilai yang salah, sehingga peningkatan pada algoritma ini diantaranya adalah dengan menambahkan faktor *error*tambahan pada tahap prediksi dan/atau memasukkan partikel baru di luar $\overline{\chi}_t$ ke dalam χ_t.

Penggunaan populer dari algoritma ini adalah pada algoritma *Monte Carlo localization* (*MCL*) dimana penapis partikel digunakan untuk kasus lokalisasi robot menggunakan data kontrol dari kontrol kecepatan atau data perpindahan dari sensor dan data pengukuran dari sensor peraba jarak atau kamera. Algoritma *Augmented Monte Carlo localization* (*AMCL*) merupakan modifikasi algoritma *MCL* biasa yang ditambahkan pemasukkan partikel baru atau *resampling* apabila hasil evaluasi partikel-partikel yang ada sekarang lebih buruk dari sebelumnya. *Resampling* ini dilakukan apabila dideteksi bahwa hasil pemberatan atau peluang partikel rata-rata lebih buruk dari sebelumnya, sehingga *resampling* dilakukan untuk memastikan distribusi partikel tidak menjadi konvergen sehingga masih bisa beradaptasi saat kualitas bacaan pengukuran membaik. Banyaknya partikel yang di*resample* pada setiap iterasinya bergantung pada rata-rata hasil evaluasi *weight*nya. Digunakan dua variabel w_{fast} dan w_{slow} yang mengikuti rata-rata *weight* tersebut lintas iterasinya dengan *rate* yang berbeda dimana rasio $w_{\text{fast}}/w_{\text{slow}}$ digunakan sebagai peluang suatu partikel tidak di*resample*.

II.3.6 Penanganan *Out of Order Measurement*

Latensi komunikasi dapat mengakibatkan diterimanya informasi dari teman yang terlambat atau *out of order* dibandingkan informasi dari teman yang sebelumnya ataupun informasi dari sensor robot sendiri. Penapis Bayes pada dasarnya tidak dapat menangani data yang terlambat tersebut, karena

menyalahi asumsi Markov dimana dibutuhkan informasi keadaan *world state* di masa lampau untuk dapat mengintegrasi informasi tersebut. Terdapat beberapa studi terhadap modifikasi penapis Bayes dalam bentuk penapis Kalman ataupun penapis partikel agar dapat mengintegrasi data yang terlambat, seperti mengestimasi distribusi *world state* lampau berdasarkan *world state* yang ada. Akan tetapi, pendekatan yang lebih sederhana dimana distribusi *world state* lampau disimpan dan digunakan untuk melakukan estimasi ulang dengan menyisipkan informasi yang baru didapat dapat digunakan apabila tidak ada batasan seperti konstrain memori.

II.4 Studi Terkait

Secara umum, teknik probabilistik seperti penapis Kalman maupun penapis partikel telah banyak diimplementasi dalam kasus pelacakan objek oleh satu agen dan telah menunjukkan tingkat keberhasilan yang cukup baik. Sedangkan dalam kasus pelacakan objek oleh banyak agen, terdapat banyak studi yang mencetuskan beragam ide yang sangat bervariasi dalam berbagai konteks. Dalam konteks kontes robot sepak bola sendiri, sudah terdapat beberapa studi yang mencoba menggabungkan data pelacakan objek maupun lokalisasi.

Robot-robot Stroupe dkk. (2001) melacak posisi bola menggunakan penapis Kalman dan distribusi hasilnya disebarkan ke robot lainnya, dimana distribusi normal dari semua robot akan digabungkan oleh masing-masing robot untuk menghasilkan suatu distribusi normal akhir. Asumsi dasar dari makalah ini adalah *error*nya bersifat normal independen, lokalisasi robot sempurna, dan waktu pengukuran bersamaan. Sedangkan Pinheiro & P. Lima (2004) mengalikan langsung distribusi normal masing-masing robot untuk mendapatkan fungsi objektif lokasi bola. Data digabungkan apabila hasil perkalian memiliki puncak probabilitas yang unik, dimana dibuat suatu rumus untuk menentukan apakah ini terjadi.

Ferrein dkk. (2005) membandingkan beberapa metode untuk menggabungkan observasi bola dari masing-masing robot, seperti merata-ratakan posisi pengamatan, menggunakan penapis Kalman menggunakan terhadap posisi

bola menggunakan data yang didapat masing-masing robot, menggunakan penapis partikel, maupun menggunakan penapis histogram yang digabungkan ke estimasi global menggunakan penapis Kalman. Hasilnya adalah penapis Kalman sederhana memiliki tingkat akurasi dan kecepatan yang paling baik.

Masing-masing robot Pahliani & P. Lima (2006) melakukan lokalisasi menggunakan *Markov localization*, lalu tergantung dari posisi robot, robot-robot yang ada membentuk beberapa kelompok kecil untuk bertukar informasi. Data dari robot di kelompok yang sama dibagikan, lalu menggunakan data yang didapat dari kelompok lain, hasil lokalisasi dari masing-masing robot diperbaiki secara Bayes. Setelah lokalisasi, deteksi objek juga dilakukan menggunakan alur dan kelompok yang sama.

Santos & P. Lima (2009) melacak bola menggunakan penapis partikel. Untuk mengefisiensikan pertukaran informasi antar robot, distribusi dalam bentuk partikel diubah dulu menjadi *gaussian mixture model* yang terdiri dari beberapa distribusi Gauss menggunakan algoritma *expectation maximization* yang bekerja mirip seperti algoritma *K means*. Ada juga ukuran persetujuan informasi antar robot menggunakan *covariance intersection* untuk menentukan apakah informasi baru diintegrasi dengan informasi robot sendiri atau tidak. Ahmad & P. U. Lima (2011) mengembangkan ide ini dengan membuat algoritma penggabungan distribusi dari masing-masing robot dimana dibangkitkan kumpulan partikel baru dengan melakukan *sampling* dari distribusi-distribusi yang ada berdasarkan tingkat kepercayaan distribusi tersebut sebagai masukan dari penapis partikel.

Ahmad, Tipaldi, dkk. (2013) mengumpulkan semua pengukuran posisi relatif semua robot terhadap robot lain, objek statis, maupun objek dinamis, dan mencari konfigurasi posisi semua objek yang meminimasi *error* dari pengukuran-pengukuran yang ada menggunakan *iterative local linearization* atau *least square minimization*. Kelebihan teknik ini diantara adalah beban komputasinya linear terhadap banyak robot, akan tetapi setiap robot harus menunggu sampainya data pengukuran dari semua robot lain.

Chang dkk. (2016) menggunakan *Extended Kalman Filter* untuk melacak sistem seluruh robot dan objek. *Update* dilakukan untuk masing-masing robot, sedangkan pengukuran diintegrasi berdasarkan robot apa saja yang terlibat dalam pengukuran tersebut. Teknik *multiple hypothesis tracking* digunakan untuk mengasosiasikan data pengukuran dengan posisi objek di hipotesis.

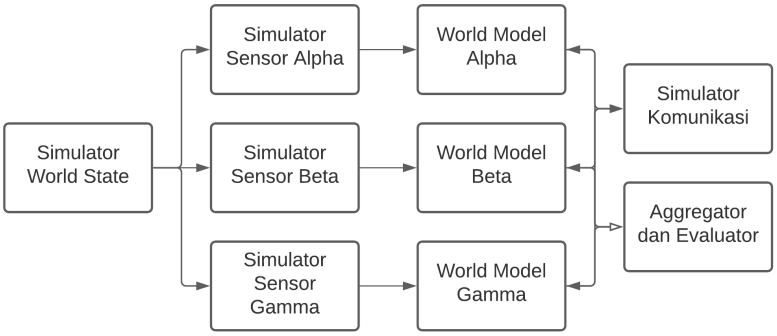
Ahmad, Lawless, dkk. (2017) menggunakan penapis partikel yang mencakup posisi semua robot ditambah objek yang diobservasi. Masing-masing bagian partikel dari robot digerakan dan dievaluasi secara terpisah, dan diurutkan ulang sehingga bagian partikel dengan evaluasi tinggi dari suatu robot dipasangkan dengan bagian partikel robot lain dengan evaluasi yang tinggi juga. Algoritma ini lalu memasangkan partikel bagian dari objek yang memaksimalkan evaluasi partikel bagian tersebut dengan partikel bagian robot dengan evaluasi tinggi juga. Teknik ini menyelesaikan masalah defisiensi partikel pada penapis partikel tanpa harus menambahkan banyak partikel secara eksponensial.

BAB III

ANALISIS SOLUSI DAN ARSITEKTUR SISTEM

Bab ini menjelaskan mengenai arsitektur lingkungan simulasi sebagai tempat uji coba, detil analisis masalah, desain arsitektur solusi, dan desain algoritma estimasi yang dapat digunakan.

III.1 Arsitektur Lingkungan Simulasi

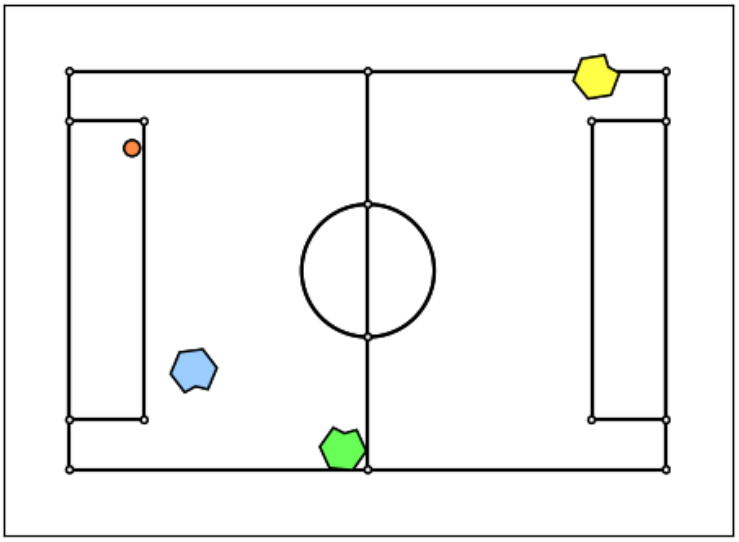


Gambar III.1.1. Struktur modul simulasi

Lingkungan simulasi dikembangan dalam *framework* ROS, dimana arsitektur lingkungan simulasi digambarkan di III.1.1 dimana masing-masing *node* berjalan dengan periode komputasi 30 ms yang didasari pada periode ketersediaan data sensor di keadaan nyata. Selain *node world model* yang bertugas untuk menampung algoritma estimasi *world state*, terdapat beberapa *node* simulator yang bertujuan untuk membangkitkan data pengujian untuk menganalisis akurasi algoritma estimasi.

III.1.1 Simulator *World State*

Modul simulator *world state* seperti pada III.1.2 bertujuan untuk membangkitkan nilai *world state* uji coba yang sebenarnya, yang mencakup pergerakan dari masing-masing robot tim dan bola. Simulasi dilakukan di lapangan 9×6 meter kuadrat sesuai spesifikasi pertandingan regional Kontes

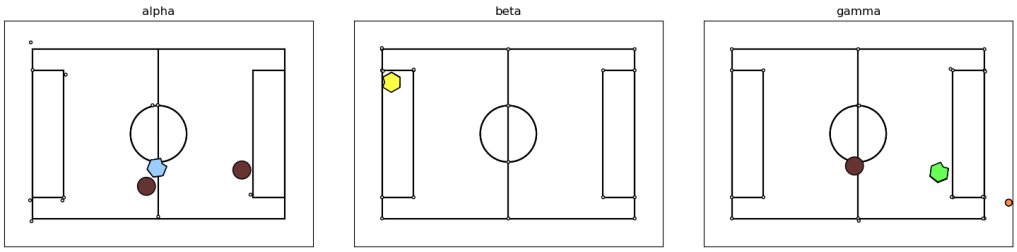


Gambar III.1.2. Keadaan *world state* sesungguhnya

Robot Indonesia cabang sepak bola beroda tahun 2019. Tiga robot tim yang disimulasikan, bernama alpha, beta, dan gamma, digerakan oleh suatu perencana gerakan sementara yang sekali-kali membangkitkan target *pose* yang harus dituju oleh robot terkait. Masing-masing robot lalu mendekati target *pose* masing-masing dengan gerakan lurus sekaligus dengan rotasi dengan kecepatan maksimum 2 meter per detik dan akselerasi maksimum 1.5 meter per detik kuadrat. Pergerakan bola dibangkitkan dengan terus meng*update* kecepatan bola setiap periode komputasi dengan perubahan kecepatan di sumbu x dan y mengikuti distribusi normal dengan variansi sebesar 0.16 meter per detik dikuadratkan dikali periode komputasi, dan kecepatan maksimal 2 meter per detik. Simulator *world state* juga mengecek apakah terjadi tabrakan antar robot, bola, ataupun dinding, dimana bola akan memantul sedangkan robot akan berhenti dan menghasilkan target *pose* baru untuk keluar dari tabrakan.

III.1.2 Simulator Sensor

Terdapat tiga *node* simulator sensor untuk masing-masing robot yang mengikuti pesan *world state* dari *node* simulator *world state* dan menghitung hasil bacaan sensor dari robot tersebut, yang mencakup simulasi bacaan



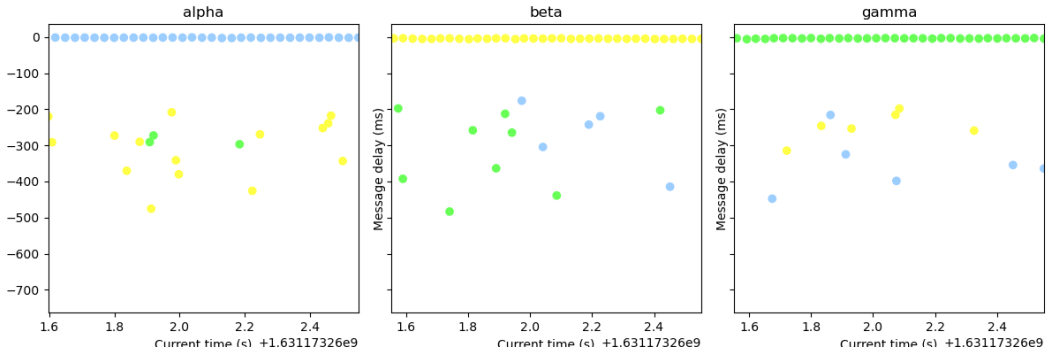
Gambar III.1.3. Bacaan sensor masing-masing robot

odometri berupa perpindahan *pose* dari robot terhadap *pose* di periode komputasi sebelumnya; simulasi bacaan kompas berupa orientasi absolut dari robot; dan simulasi persepsi *vision* dari kamera yang mencakup posisi relatif dari *landmark*, bola apabila terlihat, dan teman yang terlihat. *Landmark* merupakan 16 titik-titik statis yang tersebar di sekitar lapangan yang dapat dilihat oleh robot untuk membantu proses lokalisasi dari robot tersebut. Persepsi *vision* terbatas dalam suatu rentang penglihatan robot sebesar radius maksimal 4 meter yang juga terpengaruh oleh oklusi terhalangnya suatu objek oleh objek lain, seperti robot teman menghalangi persepsi bola atau *landmark* dibelakangnya, atau suatu peluang kecil dimana suatu objek yang harusnya terlihat gagal dideteksi oleh *vision* seperti pada III.1.3. *Vision* tidak dapat membedakan antara satu *landmark* dengan *landmark* lainnya ataupun satu teman dengan lainnya.

Masing-masing bacaan ditambahkan *error* dengan variansi yang proporsional terhadap bacaan sesungguhnya. Bacaan odometri memiliki *error* dengan variansi sebesar jarak perpindahan sebenarnya dikali sekitar $0,02$. Sedangkan bacaan *vision* memiliki *error* dengan variansi sebesar jarak objek sebenarnya dikali $0,0036$ untuk *error* sejajar dengan arah objek dan $0,0009$ untuk *error* tegak lurus arah objek.

III.1.3 Simulator Komunikasi

Node world model dapat menyiarkan pesan ke temannya, yang akan diterima oleh *node* simulator komunikasi dan disiarkan balik ke semua temannya setelah suatu latensi komunikasi dan peluang hilangnya pesan. Simulator komunikasi dimodelkan dengan *hidden Markov model* dengan dua *state*, yang

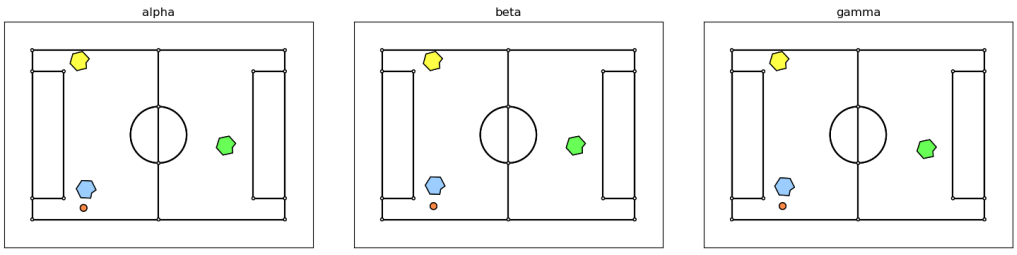


Gambar III.1.4. Latensi komunikasi

menandakan kondisi jaringan baik dengan latensi komunikasi berdistribusi *shifted gamma* dengan rata-rata 35 ms, minimum 20 ms, standar deviasi 10 ms, dan kemungkinan hilangnya pesan sebesar 5%; dan kondisi jaringan buruk dengan latensi dengan rata-rata 300 ms, minimum 150 ms, standar deviasi 100 ms, dan kemungkinan hilangnya pesan sebesar 65%.

Pengujian *world model* dibagi menjadi tiga skenario berdasarkan persentase keadaan dari jaringan, dengan skenario jaringan baik dengan persentase *state* jaringan baik sebesar 80%, skenario jaringan sedang dengan persentase jaringan baik sebesar 50%, dan skenario jaringan buruk dengan persentasi 20%. Latensi komunikasi dapat divisualisasikan seperti pada III.1.4.

III.1.4 Aggregator dan Evaluator



Gambar III.1.5. Estimasi *world state* masing-masing robot

Node-node aggregator dan evaluator mengikuti hasil estimasi dari *world model* yang ada dan nilai sebenarnya dari *world state* untuk melakukan evaluasi kinerja algoritma dan hal lain seperti visualisasi *world state* sesungguhnya, bacaan sensor masing-masing robot, hasil estimasi *world state* masing-masing

robot, dan latensi komunikasi yang sedang berlangsung yang digambarkan menggunakan *library* matplotlib dalam bahasa Python seperti pada gambar sebelumnya dan III.1.5. Evaluasi performa dari suatu algoritma diambil dari metrik *error* berupa jarak hasil estimasi dan keadaan sebenarnya dari masing-masing objek dimana nilai I/M dari robot bernilai 0.4. Statistik metrik *error* diambil dalam lima kali percobaan simulasi selama tiga menit dengan total sekitar $5 \times 3 \times 60 \times 33,3$ frekuensi komputasi untuk menghasilkan kumpulan data sebanyak sekitar 30000 nilai perkategori, yang diambil statistiknya seperti median, kuartil bawah, kuartil atas, dan rata-ratanya.

III.2 Analisis Studi Terdahulu

Dari studi yang sudah diteliti, didapat bahwa walaupun sudah cukup banyak studi yang membahas mengenai algoritma estimasi *world model* yang melacak posisi robot maupun bola di kontes robot sepak bola, hampir semua studi tersebut mengasumsikan bahwa komunikasi antar robot memiliki latensi yang dapat diabaikan. Studi yang melakukan pengujian pada kasus kegagalan komunikasi berjumlah sedikit dan juga hanya menguji kasus kegagalan komunikasi total pada suatu robot. Banyak algoritma yang dibahas dari studi sebelumnya tidak dapat bekerja secara efektif dalam keadaan latensi komunikasi yang signifikan, seperti teknik merata-ratakan data, menggabungkan distribusi Gaussian data, maupun melakukan minimasi *error* terhadap sistem. Algoritma Ahmad, Lawless, dkk. (2017) dibahas sebagai algoritma yang bersifat *online* yang tidak membutuhkan terkumpulnya informasi dari robot-robot lainnya sebelum suatu robot dapat mulai melakukan kalkulasi estimasi, tetapi studi tersebut tetap tidak menangani integrasi data yang terlambat dan di luar urutan.

Secara umum, di luar konteks kontes robot sepak bola beroda, walau terdapat banyak studi yang membahas penggabungan data dari banyak sensor dengan latensi dan keterlambatan, masih sedikit studi yang membahas penggabungan data pengukuran sensor dengan latensi pada sistem komputasi terdesentralisasi, dimana dapat dilakukan suatu pemrosesan data terlebih

dahulu sebelum dilakukannya komunikasi, dan hasil pemrosesan terjadi di masing-masing agen dengan hasil yang tidak harus sama.

III.3 Arsitektur Umum Solusi

Dalam pemrograman *node world model* dalam ROS yang berjalan berdasarkan fungsi *callback* yang dipanggil setiap *node* menerima pesan atau fungsi yang dijadwalkan untuk dipanggil dalam suatu frekuensi, terdapat tiga fungsi yang harus diimplementasi, yaitu fungsi saat *node* mendapat data sensor, saat *node* mendapat data dari teman, dan untuk setiap periode komputasi (30 ms).

Pertama, selain menyimpan estimasi gerakan untuk robot tim dan bola, *node* juga sebaiknya menyimpan *timestamp* dari data terakhir dari masing-masing objek yang diestimasi. Hal ini dikarenakan algoritma penapis yang digunakan pada dasarnya memproses data secara sekuensial terhadap waktu untuk menghasilkan estimasi pada waktu data yang terakhir. Oleh karena itu, estimasi yang disimpan oleh *node* juga merupakan estimasi pada *timestamp* dari data terakhir objek tersebut. Maka agar hasil estimasi yang dihasilkan *world model* faktual dengan waktu sekarang yang sesungguhnya, dilakukan penyesuaian dengan memprediksi gerakan objek pada waktu sekarang. Prediksi dilakukan dengan mengasumsikan kecepatan atau *twist* dari objek bernilai konstan dan meng*update* posisi atau *pose* dari objek tersebut berdasarkan selisih waktu antara waktu sekarang dan *timestamp* masing-masing objek.

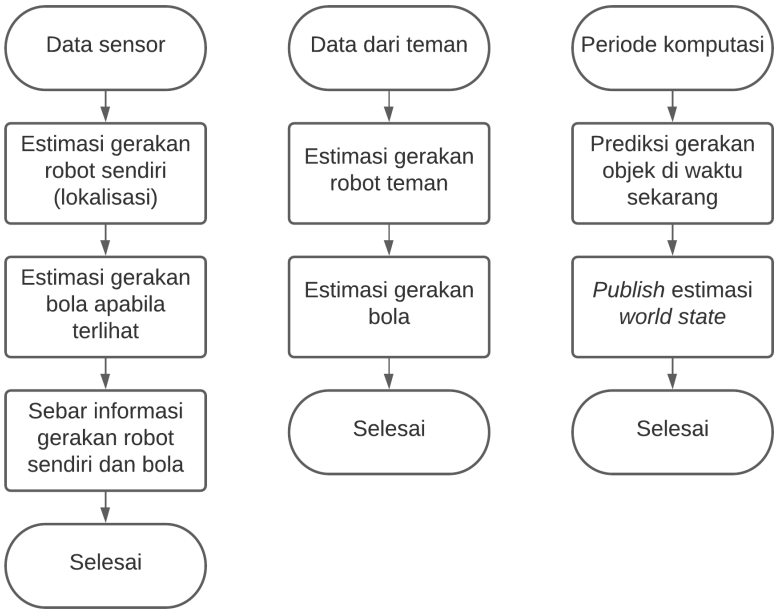
Karena walaupun dalam keadaan kegagalan komunikasi, robot masih harus menjalankan fungsionalitas dasarnya seperti navigasi setiap saat, maka masing-masing robot tim sebaiknya tetap memproses data sensornya masing-masing untuk mendapatkan informasi seperti gerakan robot sendiri, bola, dan mungkin objek di sekitarnya. Algoritma sebaiknya dapat berjalan secara *online* tanpa harus menunggu informasi dari robot lain terlebih dahulu sebelum dapat berjalan dengan akurat.

Data sensor robot sendiri yang mengandung informasi odometri, orientasi kompas, dan persepsi *landmark* merupakan sumber informasi yang paling

dapat diandalkan untuk menentukan *pose* atau gerakan secara umum dari robot sendiri. Oleh karena itu, dalam fungsi *callback* sensor sebaiknya melakukan estimasi gerakan robot sendiri atau lokalisasi. Selain itu, data persepsi *bola* yang merupakan satu-satunya sumber informasi mengenai gerakan bola juga sebaiknya diproses dan diestimasi apabila robot tersebut melihat bola. Hasil estimasi gerakan robot sendiri dan bola ini apabila terlihat disebarkan ke semua teman untuk dipakai ke dalam hasil estimasi dari *node world model* semua robot dalam tim. Pesan yang disebarkan tersebut mengandung *timestamp* dari waktu sensor robot agar teman yang menerima pesan tersebut dapat mengetahui kapan data terkandung dilihat dan memutuskan bagaimana cara untuk mengintegrasi data tersebut.

Selain menyimpan informasi gerakan dari robot teman, apabila teman tersebut menyebarkan informasi gerakan bola, informasi tersebut juga sebaiknya diintegrasikan ke dalam hasil estimasi robot sendiri, terutama apabila *timestamp* dari data tersebut lebih baru dari *timestamp* data bola terakhir yang dimiliki robot sendiri. Ini terjadi saat sensor *vision* yang dimiliki robot sendiri tidak melihat bola dalam radiusnya atau karena terjadi oklusi untuk cukup lama sehingga informasi dari teman yang melihat lebih baru dari terakhir robot sendiri dapat melihat bola. Selain itu, informasi dengan *timestamp* lebih lama dari informasi yang sudah dimiliki seperti informasi gerakan teman dapat diabaikan karena informasi yang lebih baru sudah mencakup dan kemungkinan lebih akurat dari informasi tersebut.

Diagram alir hipotesis solusi digambarkan di III.3.6. Walaupun terdapat beberapa fungsi *callback* yang dapat berjalan karena *event*nya masing-masing, pemanggilan fungsi *callback* oleh ROS dilakukan secara sinkronis satu-persatu sehingga tidak dibutuhkan pertimbangan mengenai *thread-safety* dan *mutex*. Dengan periode komputasi 30 ms, *callback* periode komputasi dan *callback* sensor akan dijalankan sekitar $33,3$ kali perdetik dan *callback* data dari teman bergantung pada kondisi jaringan komunikasi akan dijalankan rata-rata maksimal sebanyak $66,6$ kali perdetik, sehingga kinerja algoritma tidak terlalu



Gambar III.3.6. Diagram alir hipotesis solusi

dituntut dalam segi konstrain waktu komputasi.

III.4 Implementasi Algoritma Lokalisasi

Algoritma estimasi gerakan robot sendiri atau algoritma lokalisasi menggunakan informasi odometri atau perpindahan, orientasi dari kompas, dan persepsi *landmark* untuk mengestimasi dimana dan berapa kecepatan dari robot tersebut sekarang. Selain cara naif yang hanya mengagregasi perpindahan odometri, untuk menggunakan penapis Bayes, model pengukuran dari persepsi *landmark* yang tidak linear terhadap posisi robot sesungguhnya menjadikan penapis Kalman tidak dapat digunakan untuk melakukan lokalisasi ini. Sehingga penapis partikel lebih cocok digunakan untuk menyelesaikan permasalahan ini. Khususnya algoritma *AMCL* yang melakukan *resampling* dianggap sebagai algoritma yang tahan terhadap kemungkinan buruknya pengukuran.

Dalam praktisnya, informasi odometri dapat digunakan baik dalam model transisi maupun model pengukuran dari penapis partikel yang digunakan, sehingga menghasilkan dua variasi dari algoritma lokalisasi yang dapat digunakan, variasi *AMCL* yang berbasis *pose* atau variasi yang berbasis *pose*

dan *twist*.

III.4.1 Lokalisasi Berbasis *Pose*

Pada variasi pertama, partikel yang digunakan pada penapis partikel lokalisasinya mengandung informasi tentang *pose* sesungguhnya dari robot, sehingga estimasi yang dihasilkan *AMCL* juga berupa *pose* dari robot. Sedangkan *twist* dari robot diturunkan dengan mengambil perpindahan diantara hasil estimasi sekarang dengan hasil estimasi sebelumnya diturunkan terhadap selisih waktu.

Model transisi dari partikel yang ada menggunakan informasi odometri yang ada, dimana masing-masing partikel digerakan sesuai dengan perpindahan yang disampel di sekitar nilai odometri tersebut.

Algoritma III.1 Model pengukuran *pose*

```
1: function CALC_WEIGHT(pose, compass, landmarks, true_landmarks)
2:     ln_vis_weight = 0
3:     if size(landmarks) != 0 then
4:         vis_weight = 0
5:         for landmark in landmarks do
6:             landmark_weight = -infinity
7:             for true_landmark in true_landmarks do
8:                 rel_landmark = -pose + true_landmark
9:                 weight = exp(compare(landmark, rel_landmark))
10:                landmark_weight = max(landmark_weight, weight)
11:            end for
12:            vis_weight += landmark_weight
13:        end for
14:        ln_vis_weight = log(vis_weight / size(landmarks))
15:    end if
16:    ln_ort_weight = compare(pose.orientation, compass)
17:    return exp(α× ln_vis_weight + β× ln_ort_weight)
18: end function
```

Model pengukuran menggunakan informasi *landmark* dimana untuk masing-masing partikel, semua pengukuran *landmark* yang ada dibandingkan dengan semua posisi absolut *landmark* yang sebenarnya untuk diambil peluang terukurnya pengukuran tersebut berdasarkan profil *error* normal independen.

Diambil peluang pengukuran terbesar untuk masing-masing pengukuran yang berikutnya dirata-ratakan untuk masing-masing partikel. Selain itu, pengukuran dari kompas berdasarkan distribusi *error* normal dari orientasi sebenarnya juga digunakan. Kedua peluang dari pengukuran *landmark* dan dari pengukuran kompas diintegrasi dengan melakukan perkalian setelah masing-masing peluang dipangkatkan oleh suatu *weight* faktor pengukuran. Fungsi yang digunakan digambarkan pada III.1 dimana fungsi compare menghasilkan logaritma natural dari peluang dilakukannya pengukuran, yang berdasarkan persamaan distribusi normal hanyalah bernilai $-0,5$ dikali jarak kuadrat dibagi dengan variansi.

Hasil prediksi dari suatu iterasi estimasi adalah rata-rata dari partikel yang ada, dimana posisi dirata-ratakan sebagai vektor dan orientasi dirata-ratakan dengan mengambil arah dari rata-rata vektor unit dengan arah orientasi masing-masing partikel. Sedangkan *resampling* dilakukan dengan mengambil *pose* di sekitar rata-rata partikel tersebut ditambah *error* dengan distribusi normal.

III.4.2 Lokalisasi Berbasis *Pose* dan *Twist*

Pada variasi kedua, partikel yang digunakan mengandung informasi tentang *pose* dan juga *twist* dari robot, sehingga pada variasi ini tidak perlu diturunkan lagi *twist* dari robot secara terpisah karena sudah menjadi bagian dari hasil estimasi penapis.

Pada variasi ini nilai odometri tidak digunakan pada model transisi. Model transisi mengandalkan pada *twist* yang sudah dimiliki masing-masing partikel, dimana *twist* tersebut ditambahkan dengan *error* normal untuk membangkitkan *twist* baru dan *pose* baru dibangkitkan dengan meng*update* *pose* lama menggunakan *twist* yang baru dibangkitkan.

Model pengukuran variasi ini mirip dengan model pada variasi pertama dalam caranya menggunakan pengukuran *landmark* dan kompas. Di variasi ini, odometri juga digunakan pada model pengukuran dengan membandingkannya dengan perpindahan sebenarnya yang dapat didapatkan dari mengintegrasi

twist partikel dengan selisih waktu.

Rata-rata partikel diambil dengan cara yang sama untuk *pose* dari partikel dan analog untuk mendapatkan rata-rata dari *twist* partikel. *Resampling* juga dilakukan di sekitar *pose* dan *twist* partikel rata-rata dengan penambahan *error*.

III.5 Estimasi Gerakan Bola

Estimasi gerakan bola berdasarkan data persepsi bola yang didapat dari sensor sendiri atau teman lebih leluasa dibandingkan dengan lokalisasi. Hal ini dikarenakan bacaan posisi bola bersifat linear terhadap posisi bola sesungguhnya ditambah suatu *error* seperti yang ada pada asumsi penapis Kalman. Oleh karena itu, walaupun profil *error* sesungguhnya dari bacaan posisi bola tidak *uniform* untuk semua posisi sesungguhnya, dapat dimodelkan *error* normal *uniform* untuk dapat menggunakan penapis Kalman dalam melakukan estimasi gerakan bola. Karena bacaan posisi bola yang dilihat bersifat relatif, maka estimasi gerakan bola mengandalkan estimasi gerakan robot yang melakukan persepsi untuk mengubah bacaan posisi bola yang relatif ke ruang referensi absolut.

III.5.1 Estimasi dengan Penapis Kalman

Penapis Kalman memodelkan distribusi gerakan bola sebagai distribusi normal multivariat dengan vektor rata-rata μ empat dimensi yang merepresentasikan komponen posisi bola pada sumbu x, kecepatan pada sumbu x, posisi pada sumbu y, dan kecepatan pada sumbu y, juga matriks kovarian 4×4 dengan urutan komponen yang sama.

Karena informasi bola hanya berupa pengukuran, tidak ada variable transisi yang digunakan dalam estimasi gerakan bola. Sehingga persamaan transisi yang digunakan hanyalah

$$x_t = A_t x_{t-1} + \epsilon_t \tag{III.1}$$

dimana A_t adalah matriks 4×4 *update* keadaan dimana posisi baru ditambahkan dengan kecepatan dikalikan selisih waktu, sedangkan ϵ_t adalah vektor acak *error* empat dimensi. Variansi dari komponen *error* kecepatan adalah konstanta dikali selisih waktu. Karena perubahan posisi adalah perubahan kecepatan dikali selisih waktu, maka didapatkan variansi komponen *error* posisi adalah konstanta dikali selisih waktu pangkat tiga dan kovarian antara komponen *error* posisi dan kecepatan adalah konstanta dikali selisih waktu kuadrat.

Pada persamaan pengukuran $z_t = C_t x_t + \delta_t$, z_t adalah vektor dua dimensi berisi pengukuran posisi absolut bola pada sumbu x dan y. C_t adalah matriks pengukuran berdimensi 2×4 yang bernilai 1 di komponen posisi yang terkait dan 0 di sisanya. Walaupun distribusi *error* pengukuran bola tergantung dari posisi robot yang melihat dan jarak bola dengan robot, karena dibutuhkan suatu vektor acak δ_t yang sama atau *uniform* untuk semua posisi, maka digeneralisasi δ_t sebagai vektor dua dimensi dengan matriks kovariansi Q_t merupakan matriks identitas dikali suatu konstanta.

III.5.2 Estimasi dengan Penapis Partikel

Estimasi gerakan bola dengan penapis partikel mirip dengan algoritma lokalisasi variansi dua, dimana sampel berisi posisi dan kecepatan dari bola dan tidak ada kalkulasi terpisah untuk menentukan kecepatan dari bola. Tidak ada variabel transisi yang dapat digunakan, jadi model transisi hanyalah membangkitkan kecepatan baru di sekitar kecepatan partikel dan meng*update* posisi berdasarkan kecepatan baru dan selisih waktu seperti pada lokalisasi variansi dua.

Model pengukuran menggunakan persamaan yang sama dengan pengukuran menggunakan *landmark*, dimana posisi robot yang melakukan persepsi mengandalkan hasil lokalisasi untuk menghasilkan posisi relatif bola referensi untuk dibandingkan dengan posisi relatif bola yang dilihat.

III.6 Integrasi Data Bola Terlambat

Cara integrasi data bola dari teman yang paling dasar adalah dengan mengintegrasinya ke dalam penapis bola yang digunakan, sama seperti dengan pemrosesan data sensor, apabila dilihat bahwa *timestamp* dari data tersebut lebih baru daripada data terakhir yang sudah diproses. Kekurangan dari metode ini adalah apabila robot sendiri dan satu atau lebih teman melihat bola pada saat yang sama, informasi yang didapat dari teman tersebut tidak akan pernah digunakan karena data sensor dari robot sendiri hampir pasti sampai lebih cepat dibandingkan dengan data dari teman yang ditambahkan dengan latensi komunikasi. Salah satu cara untuk menangani hal ini adalah dengan menggunakan modifikasi penapis untuk *out of order measurement* atau (*OOSM*).

III.6.1 Estimasi dengan *OOSM-PF*

Karena tidak ada konstrain waktu ataupun memori yang signifikan, penanganan data terlambat dapat dilakukan menggunakan penyimpanan dan penyisipan data pengukuran. Pada penapis partikel *OOSM-PF*, untuk batas jumlah yang telah ditentukan sebelumnya, beberapa data pengukuran dan *state* estimasi yang terakhir disimpan secara terurut berdasarkan *timestamp* datanya. Sistem penyimpanan dapat menggunakan tipe data seperti *map* terurut yang menyimpan datanya secara terurut berdasarkan *key*nya yang diimplementasi di atas tipe data seperti *red-black tree*. Dalam kasus penapis partikel, *state* yang dimaksud adalah kumpulan partikel dalam penapis dan nilai dari w_fast dan w_slow.

Apabila ditemui data terlambat yang lebih lama dari data terlama yang disimpan, maka data tersebut dibuang dan dikembalikan hasil estimasi terakhir. Apabila data tersebut lebih baru, cari data paling baru yang disimpan yang lebih lama dari data terlambat tersebut. Lalu simpan data terlambat tersebut dengan menghitung *state*nya menggunakan algoritma penapis partikel biasa menggunakan *state* sebelumnya dari data paling baru tersebut. *Update* juga semua *state* dari data yang disimpan yang ada setelah

data yang baru dimasukan, dan buang data terlama yang disimpan apabila sudah melebihi jumlah maksimal data yang disimpan.

Kompleksitas waktu dan ruang dari algoritma ini berkembang secara linear terhadap batas jumlah data yang disimpan. Akan tetapi karena data yang sudah melewati beberapa periode komputasi semakin lama tidak relevan, maka batas jumlah ini tidak perlu diatur dengan besar.

III.6.2 Estimasi dengan *OOSM-KF*

OOSM-KF adalah algoritma seperti *OOSM-PF* akan tetapi menggunakan algoritma penapis Kalman sebagai dasarnya. Perbedaan dari algoritma ini adalah *state* yang disimpan hanyalah berupa representasi distribusi normal multivariat berupa vektor rata-rata dan matriks kovariansi, sehingga kompleksitas waktu dan ruangnya setaraf lebih efisien dibandingkan *OOSM-PF*.

III.6.3 Estimasi *OOSM-PF* dengan Data Bola Opsional

Karena kegagalan persepsi bola oleh robot juga sebenarnya memberikan informasi mengenai posisi bola dimana kegagalan posisi bola yang sebenarnya kemungkinan berada di luar radius penglihatan robot, maka dapat dibuat variasi dari penapis partikel bola yang dapat menerima data bola secara opsional. Pemanggilan penapis partikel tanpa adanya informasi bola yang menandakan bahwa kemungkinan bola berada di luar radius penglihatan mengakibatkan algoritma model pengukuran meninggikan *weight* partikel yang berada di luar radius penglihatan dan mengurangi *weight* partikel yang berada di dalam radius penglihatan.

Algoritma ini hanya mungkin berfungsi dengan baik menggunakan modifikasi data *OOSM*. Ini disebabkan karena jika digunakan cara integrasi dasar yang hanya mempertimbangkan *timestamp*, maka robot hanya akan mempertimbangkan data dari sensor sendiri saja walaupun bola tidak terlihat, karena informasi dari teman yang hampir pasti terlambat walaupun mungkin teman tersebut melihat posisi bola sehingga informasinya lebih berguna dibandingkan informasi robot sendiri yang tidak melihat bola.

III.6.4 Estimasi *OOSM-KF* dengan Data Bola Kombinasi

Salah satu cara konvensional untuk menggabungkan data posisi bola dari banyak robot adalah dengan merata-ratakan data tersebut. Variansi *OOSM-KF* dengan data bola kombinasi adalah variansi *OOSM-KF* bola dimana apabila diterima dua atau lebih data dengan *timestamp* yang sama dari beberapa robot, maka data tersebut tidak akan diproses satu-persatu, melainkan dengan dirata-ratakan dan diproses dengan sekaligus. Variansi ini menggunakan penapis Kalman karena rata-rata posisi absolut bola yang digunakan dengan kemungkinan bola dilihat lebih dari satu robot, sehingga penapis partikel yang membutuhkan satu robot yang melihat bola untuk menentukan profil *error* dari bacaan tidak dapat digunakan.

III.7 Koreksi Gerakan Robot Teman dengan Persepsi *Vision*

Walaupun hasil persepsi

DAFTAR PUSTAKA

Ahmad, A., Lawless, G., dkk. (2017). An online scalable approach to unified multirobot cooperative localization and object tracking. *IEEE transactions on robotics* 33(5), pp. 1184–1199.

Ahmad, A. & Lima, P. U. (2011). Multi-Robot Cooperative Object Tracking Based on Particle Filters. *ECMR*. Citeseer, pp. 37–42.

Ahmad, A., Tipaldi, G. D., dkk. (2013). Cooperative robot localization and target tracking based on least squares minimization. *2013 IEEE International Conference on Robotics and Automation*. IEEE, pp. 5696–5701.

Brégier, R. dkk. (2018). Defining the Pose of any 3D Rigid Object and an Associated Metric. *International Journal of Computer Vision* 126.

Chang, C. dkk. (2016). Exploiting Moving Objects: Multi-Robot Simultaneous Localization and Tracking. Vol. 13. 2, pp. 810–827.

DeGroot, M. H. & Schervish, M. J. (2012). *Probability and statistics*. Pearson Education.

Ferrein, A. dkk. (2005). Comparing sensor fusion techniques for ball position estimation. *Robot Soccer World Cup*. Springer, pp. 154–165.

Pahliani, A. & Lima, P. (2006). Improving Self Localization and Object Localization by a Team of Robots Using Sensor Fusion. *Proc. of CONTROLO*.

Pinheiro, P. & Lima, P. (2004). Bayesian sensor fusion for cooperative object localization and world modeling. *Proc. 8th Conference on Intelligent Autonomous Systems*. Citeseer.

Quigley, M. dkk. (2009). ROS: an open-source Robot Operating System.

Salvo Rossi, P. dkk. (2006). Joint end-to-end loss-delay hidden Markov model for periodic UDP traffic over the Internet. *IEEE Transactions on Signal Processing* 54(2), pp. 530–541.

Santos, J. & Lima, P. (2009). Multi-robot cooperative object localization. *Robot Soccer World Cup*. Springer, pp. 332–343.

Stroupe, A. W. dkk. (2001). Distributed sensor fusion for object position estimation by multi-robot systems. *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*. Vol. 2, 1092–1098 vol.2.

Thrun, S. dkk. (2010). *Probabilistic robotics*. MIT Press.