

**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

## LEMBAR PERNYATAAN

Dengan ini saya menyatakan bahwa:

1. Pengerjaan dan penulisan Laporan Tugas Akhir ini dilakukan tanpa menggunakan bantuan yang tidak dibenarkan.
2. Segala bentuk kutipan dan acuan terhadap tulisan orang lain yang digunakan di dalam penyusunan laporan tugas akhir ini telah dituliskan dengan baik dan benar.
3. Laporan Tugas Akhir ini belum pernah diajukan pada program pendidikan di perguruan tinggi mana pun.

Jika terbukti melanggar hal-hal di atas, saya bersedia dikenakan sanksi sesuai dengan Peraturan Akademik dan Kemahasiswaan Institut Teknologi Bandung bagian Penegakan Norma Akademik dan Kemahasiswaan khususnya Pasal 2.1 dan Pasal 2.2.

Bandung, 10 September 2021

Bimo Adityarahman Wiraputra

NIM 13517004

## ABSTRAK

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

Oleh

BIMO ADITYARAHMAN WIRAPUTRA

NIM: 13517004

Dalam bidang robotika, pengembangan algoritma pengolahan data mentah dari sensor yang akurat merupakan hal yang penting agar robot dapat mengambil keputusan dan melakukan kontrol terhadap aktuator dengan akurat sehingga robot dapat bekerja dengan efisien dan aman. Pada konteks kontes robot sepak bola beroda, beberapa robot dalam tim dapat berkomunikasi untuk mendapatkan estimasi *world state* yang seakurat mungkin. Akan tetapi, keadaan di tempat kontes memungkinkan adanya gangguan komunikasi berupa latensi dan hilangnya pesan karena interferensi jaringan.

Pada tugas akhir ini telah dikembangkan suatu algoritma estimasi *world state* yang dapat memberikan hasil yang akurat pada konteks kontes robot sepak bola beroda bahkan dalam keadaan latensi komunikasi. Algoritma dikembangkan dari algoritma penapis Bayes berupa penapis Kalman dan penapis partikel dengan modifikasi untuk menangani data yang terlambat karena latensi.

Berdasarkan hasil pengujian menggunakan simulasi, didapatkan bahwa algoritma estimasi yang menggunakan AMCL berbasis partikel berisi posisi dan kecepatan untuk lokalisasi, penapis partikel dengan modifikasi penanganan data terlambat untuk estimasi gerakan bola dan gerakan teman, beserta koreksi waktu berdasarkan *timestamp* data menghasilkan akurasi yang paling baik dibandingkan dengan algoritma lain yang diuji coba.

Kata kunci: Penapis Bayes, *out of order measurement*, kontes robot sepak bola beroda

## KATA PENGANTAR

Puji syukur penulis panjatkan kepada Allah SWT karena hanya dengan rahmat dan karuniaNya, laporan tugas akhir yang berjudul Pelacakan Objek Dinamis Menggunakan Pengukuran Banyak Agen dengan Latensi Komunikasi pada Robot Kontes Sepak Bola Beroda ini dapat dirampungkan dengan baik dan tepat waktu. Penulis juga mengucapkan terima kasih kepada semua pihak yang telah terlibat dan membantu penulis dalam pengerjaan tugas akhir ini, termasuk tapi tidak terbatas padanya diantaranya:

1. Bapak Dr. Judhi Santoso, M.Sc. sebagai pembimbing penulis yang telah membimbing dan mengarahkan penulis dalam pengembangan solusi dan penulisan tugas akhir ini.
2. Bapak Ir. Rila Mandala, M.Eng., Ph.D. sebagai penguji penulis yang telah memberi kritik dan saran untuk tugas akhir ini sehingga dapat menjadi lebih baik lagi.
3. Ibu Ginar Santika Niwanputri, S.T., M.Sc., Bapak Adi Mulyanto, S.T., M.T., Bapak Nugraha Priya Utama, S.T., M.A., Ph.D., Ibu Latifa Dwiyanti, S.T., M.T., dan seluruh anggota tim tugas akhir lainnya yang telah mengatur dan mengarahkan penulis dan mahasiswa lainnya dalam pengerjaan dan penulisan tugas akhir ini.
4. Dosen-dosen Teknik Informatika dan dosen Institut Teknologi Bandung secara umum yang telah mendidik dan memberikan ilmu kepada penulis sehingga penulis dapat mengerjakan tugas akhir ini.
5. Kementerian Pendidikan dan Kebudayaan Indonesia yang telah membiayai penulis dalam menempuh pendidikan tinggi di Teknik Informatika ini.
6. Keluarga penulis yang selalu mendukung penulis di dalam pengerjaan tugas akhir ini maupun di luarnya.

7. Teman-teman penulis yang tergabung dalam Tim DAGOZILLA ITB yang telah mengajarkan penulis dalam bidang robotika dan membantu pengerjaan tugas akhir ini baik secara moril maupun materiil.
8. Teman-teman penulis di dalam atau luar Teknik Informatika yang telah menemani dan mendukung penulis selama pengerjaan tugas akhir ini.

Penulis berharap agar laporan tugas akhir ini dalam segala kekurangannya dapat memberi manfaat untuk seluruh pihak yang membaca maupun terlibat dalam pembuatannya.

Bandung, 10 September 2021

Penulis

## DAFTAR ISI

<b>ABSTRAK</b>	<b>iii</b>
<b>KATA PENGANTAR</b>	<b>iv</b>
<b>DAFTAR ISI</b>	<b>vi</b>
<b>DAFTAR GAMBAR</b>	<b>ix</b>
<b>BAB I PENDAHULUAN</b>	<b>1</b>
I.1 Latar Belakang	1
I.2 Rumusan Masalah	3
I.3 Tujuan	4
I.4 Batasan Masalah	4
I.5 Metodologi	4
I.6 Sistematika Pembahasan	5
<b>BAB II STUDI LITERATUR</b>	<b>6</b>
II.1 Robot Operating System	6
II.2 Pemodelan Robotika dan Komunikasi	8
II.2.1 Pergerakan Bola dan Robot	8
II.2.2 Metrik <i>Error</i> Pergerakan Objek	9
II.2.3 Distribusi Normal	10
II.2.4 Pemodelan <i>Error</i> Sensor	11
II.2.5 Pemodelan Latensi Komunikasi	12
II.3 Estimasi Keadaan Probabilistik	12
II.3.1 Teorema Bayes	13
II.3.2 Pemodelan Keadaan	13
II.3.3 Penapis Bayes	15
II.3.4 Penapis Kalman	16
II.3.5 Penapis Partikel	18
II.3.6 Penanganan <i>Out of Order Measurement</i>	19
II.4 Studi Terkait	20
<b>BAB III ANALISIS SOLUSI DAN ARSITEKTUR SISTEM</b>	<b>23</b>

III.1	Arsitektur Lingkungan Simulasi . . . . .	23
III.1.1	Simulator <i>World State</i> . . . . .	23
III.1.2	Simulator Sensor . . . . .	25
III.1.3	Simulator Komunikasi . . . . .	26
III.1.4	Aggregator dan Evaluator . . . . .	26
III.2	Analisis Studi Terdahulu . . . . .	27
III.3	Arsitektur Umum Solusi . . . . .	28
III.4	Implementasi Algoritma Lokalisasi . . . . .	30
III.4.1	Penapis Partikel Berbasis <i>Pose</i> . . . . .	31
III.4.2	Penapis Partikel Berbasis <i>Pose</i> dan <i>Twist</i> . . . . .	32
III.5	Estimasi Gerakan Bola . . . . .	33
III.5.1	Estimasi dengan Penapis Kalman . . . . .	33
III.5.2	Estimasi dengan Penapis Partikel . . . . .	34
III.6	Integrasi Data Bola Terlambat . . . . .	35
III.6.1	Estimasi dengan OOSM-PF . . . . .	35
III.6.2	Estimasi OOSM-PF dengan Data Bola Opsional . . . . .	36
III.6.3	Estimasi dengan OOSM-KF . . . . .	36
III.6.4	Estimasi OOSM-KF dengan Data Bola Kombinasi . . . . .	37
III.7	Koreksi Gerakan Robot Teman dengan Persepsi <i>Vision</i> . . . . .	37
<b>BAB IV EKSPERIMEN DAN ANALISIS . . . . .</b>		<b>39</b>
IV.1	Skenario Pengujian . . . . .	39
IV.2	Pengujian dan Analisis Algoritma Estimasi Dasar . . . . .	41
IV.2.1	Perbandingan Algoritma tanpa Penapis dan dengan Penapis . . . . .	41
IV.2.2	Perbandingan Algoritma Lokalisasi Penapis Partikel Berbasis <i>Pose</i> dan Penapis Partikel Berbasis <i>Twist</i> . . . . .	41
IV.2.3	Perbandingan Algoritma Estimasi Bola Menggunakan Penapis Kalman dan Penapis Partikel . . . . .	44
IV.2.4	Perbandingan Algoritma dengan Penyesuaian Waktu dan tanpa Penyesuaian Waktu . . . . .	44
IV.3	Pengujian dan Analisis Algoritma Estimasi Data Terlambat . . . . .	47



IV.3.1 Perbandingan Algoritma Estimasi Bola Menggunakan PF dan OOSM-PF . . . . .	47
IV.3.2 Perbandingan Algoritma Estimasi Bola tanpa Data Opsional dan dengan Data Opsional . . . . .	47
IV.3.3 Perbandingan Algoritma Estimasi Bola dengan OOSM- PF dan OOSM-KF . . . . .	52
IV.3.4 Perbandingan Algoritma Estimasi Bola OOSM-PF Biasa dan OOSM-KF dengan Kombinasi Data . . . . .	52
IV.3.5 Perbandingan Algoritma tanpa dan dengan Koreksi Posisi Robot Teman . . . . .	52
<b>BAB V PENUTUP . . . . .</b>	<b>59</b>
V.1 Simpulan . . . . .	59
V.2 Saran . . . . .	59
<b>DAFTAR PUSTAKA . . . . .</b>	<b>61</b>

## DAFTAR GAMBAR

Gambar I.1.1	Tim DAGOZILLA pada Kontes Robot Indonesia . . .	2
Gambar II.3.1	Hubungan $X, U, Z$ (Thrun dkk., 2010) . . . . .	14
Gambar III.1.1	Struktur modul simulasi . . . . .	23
Gambar III.1.2	Keadaan <i>world state</i> sesungguhnya . . . . .	24
Gambar III.1.3	Bacaan sensor masing-masing robot . . . . .	25
Gambar III.1.4	Latensi komunikasi . . . . .	26
Gambar III.1.5	Estimasi <i>world state</i> masing-masing robot . . . . .	26
Gambar III.3.6	Diagram alir hipotesis solusi . . . . .	30
Gambar IV.2.1	<i>Error world model</i> O dan P jaringan baik . . . . .	42
Gambar IV.2.2	<i>Error world model</i> P dan Q jaringan baik . . . . .	43
Gambar IV.2.3	<i>Error world model</i> Q dan R jaringan baik . . . . .	45
Gambar IV.2.4	<i>Error world model</i> R dan RB jaringan baik . . . . .	46
Gambar IV.3.5	<i>Error world model</i> R dan S jaringan baik . . . . .	48
Gambar IV.3.6	<i>Error world model</i> R dan S jaringan sedang . . . . .	49
Gambar IV.3.7	<i>Error world model</i> R dan S jaringan buruk . . . . .	50
Gambar IV.3.8	<i>Error world model</i> S dan SB jaringan baik . . . . .	51
Gambar IV.3.9	<i>Error world model</i> S dan SC jaringan baik . . . . .	53
Gambar IV.3.10	<i>Error world model</i> S dan SD jaringan baik . . . . .	54
Gambar IV.3.11	<i>Error world model</i> S dan T jaringan baik . . . . .	55
Gambar IV.3.12	<i>Error world model</i> S dan T jaringan sedang . . . . .	56
Gambar IV.3.13	<i>Error world model</i> S dan T jaringan buruk . . . . .	57

# **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 intelegensi 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 Kementerian 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, mengembangkan, 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 pengujian 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 Eksperimen 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 multi agen 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 dites 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 penggunaanya 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 di antara 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  $(p, \theta)$  dimana komponen  $p$  menyatakan perpindahan posisi dan  $\theta$  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  $(p, \theta)$ . 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  $(p, \theta)$  yang diintegrasi terhadap waktu  $d$  akan menjadi *pose* dengan nilai

$$\left( \frac{v(i - i \cos \theta d)}{\theta}, \theta d \right) \quad (\text{II.1})$$

dimana vektor dua dimensi  $(x, y)$  di sini bisa diinterpretasi sebagai bilangan kompleks  $x + iy$ ,  $\cos \theta + i \sin \theta = (\cos \theta, \sin \theta)$  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* di antara 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

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

dimana  $S$  adalah massa objek dan  $|T_2(x) - T_1(x)|$  adalah jarak dari titik pada objek yang koresponden pada kedua *pose*. Dapat dilihat 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*  $(p, \theta)$ , maka titik  $x$  di *pose* pertama akan berkorespon dengan titik  $p + x \text{cis } \theta$ . Maka diturunkan persamaan jarak menjadi

$$\sqrt{\frac{1}{M} \int |p + x \text{cis } \theta - x|^2 dm} \quad (\text{II.3})$$

$$= \sqrt{|p|^2 + |\text{cis } \theta - 1|^2 I/M} \quad (\text{II.4})$$

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 di antara kerucut dan silinder,  $I/M$  bernilai di antara  $0.3r^2$  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  $\mu$  dan nilai variansi  $\sigma^2$  memiliki fungsi densitas

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

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 distribusi 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  $\mu$  dan matriks kovarian  $\Sigma$  yang bersifat simetrik dan positif semidefinit, didefinisikan fungsi densitas

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

dimana masing-masing komponen  $X_i$  sendiri memiliki distribusi normal dengan rata-rata  $\mu_i$  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

$$p(x | y) = Pr(X = x | Y = y) = \frac{p(x, y)}{p(y)} \quad (\text{II.7})$$

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

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

Terkait peluang kondisional, Teorema Bayes menyatakan bahwa

$$p(x | y) = \frac{p(y | x)p(x)}{p(y)} = \eta p(y | x)p(x) \quad (\text{II.9})$$

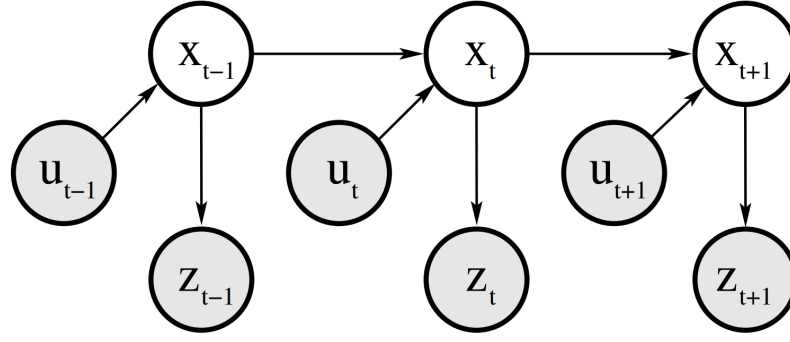
dimana  $\eta = p(y)^{-1}$  merupakan suatu nilai yang konstan untuk semua kemungkinan  $x$ . Karena jumlah dari semua nilai  $p(x | y)$  haruslah bernilai satu,  $\eta$  disebut juga faktor normalisasi dan dapat ditentukan kemudian sebagai jumlah dari nilai  $p(y | x)p(x)$  untuk semua kemungkinan 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

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

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

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

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

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*

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

dan model peluang pengukuran atau *measurement model*

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

### 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:      $\overline{bel}(x_t) = \int p(x_t | u_t, x_{t-1}) bel(x_{t-1}) dx_{t-1}$ 
4:      $bel(x_t) = \eta p(z_t | x_t) \overline{bel}(x_t) \quad \triangleright \eta$  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  $p(x_t | x_{t-1}, u_t)$ . Sedangkan integrasi informasi  $u$  atau perubahan distribusi dari  $\overline{bel}(x_t)$  ke  $bel(x_t)$  dapat ditentukan dari model pengukuran  $p(z_t | x_t)$  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. Dapat dilihat 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  $\mu$  dan  $\sigma^2$  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*  $\epsilon_t$

$$x_t = A_t x_{t-1} + B_t u_t + \epsilon_t \quad (\text{II.14})$$

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

$$z_t = C_t x_t + \delta_t \quad (\text{II.15})$$

untuk  $z_t$  berdimensi  $k$ ,  $C_t$  berdimensi  $k \times n$ , dan  $\delta_t$  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:    $\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t$ 
3:    $\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$ 
4:    $K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$ 
5:    $\mu_t = \bar{\mu}_t + K_t (z_t - C_t \bar{\mu}_t)$ 
6:    $\Sigma_t = (I - K_t C_t) \bar{\Sigma}_t$ 
7:   return  $\mu_t, \Sigma_t$ 
8: end function
```

---

Berdasarkan hubungan di atas, algoritma penapis Kalman menghitung nilai dari parameter rata-rata  $\mu_t$  dan kovariansi  $\Sigma_t$  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  $H_t = h'(\bar{\mu}_t)$  dimana  $g'$  merupakan turunan parsial  $g$  terhadap  $x_{t-1}$ , didapatkan persamaan linear

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

$$h(x_t) \approx h(\bar{\mu}_t) + H_t (x_t - \bar{\mu}_t) \quad (\text{II.17})$$

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  $\mu$  dan dua titik di sekelilingnya untuk setiap dimensi yang ada.

Selain *EKF* dan *UKF*, karena popularitasnya, 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( $\chi_{t-1}, u_t, z_t$ )
2:    $\bar{\chi}_t = \bar{\chi}_t = \emptyset$ 
3:   for  $m = 1$  to  $M$  do
4:     sample  $x_t^{[m]} \sim p(x_t | u_t, x_{t-1}^{[m]})$ 
5:      $w_t^{[m]} = p(z_t | x_t^{[m]})$ 
6:     add  $(x_t^{[m]}, w_t^{[m]})$  to  $\bar{\chi}_t$ 
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  $\chi_t$ 
11:   end for
12:   return  $\chi_t$ 
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  $\bar{\chi}_t$  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  $\chi_t$ , 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 distribusi non-normal, 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  $\bar{\chi}_t$  ke dalam  $\chi_t$ .

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  $w_{\text{fast}}$  dan  $w_{\text{slow}}$  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 mengintegrasikan informasi tersebut. Terdapat beberapa studi terhadap modifikasi penapis Bayes dalam bentuk penapis Kalman ataupun penapis partikel agar dapat mengintegrasikan 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 diimplementasikan 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 Pahlani & 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 mengefisienkan 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 diintegrasikan 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 di antara 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 diintegrasikan 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 digerakkan 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.

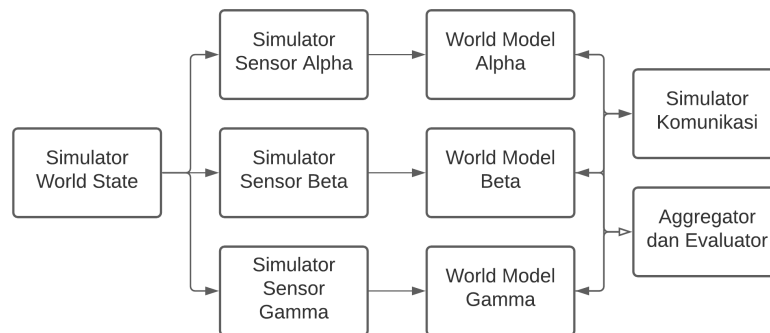


## BAB III

### ANALISIS SOLUSI DAN ARSITEKTUR SISTEM

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

#### III.1 Arsitektur Lingkungan Simulasi

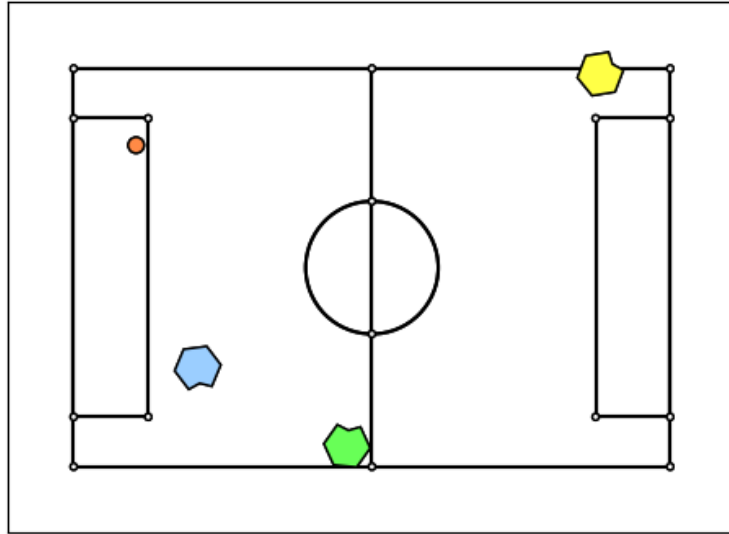


Gambar III.1.1. Struktur modul simulasi

Lingkungan simulasi dikembangkan 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.

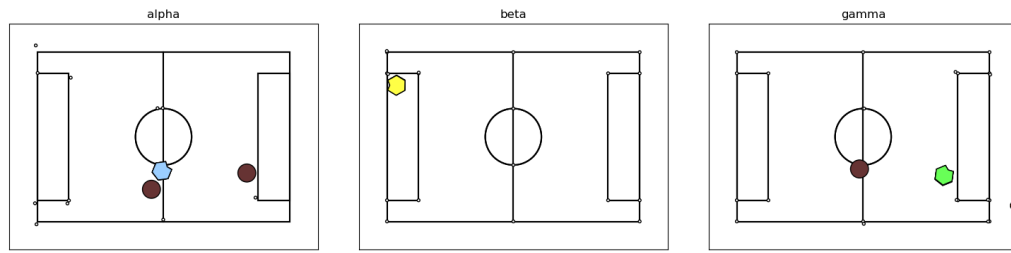
##### III.1.1 Simulator *World State*

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



Gambar III.1.2. Keadaan *world state* sesungguhnya

lapangan  $9 \times 6$  meter kuadrat sesuai spesifikasi pertandingan regional Kontes Robot Indonesia cabang sepak bola beroda tahun 2019. Terdapat dinding dengan jarak satu meter di sekitar lapangan untuk mencegah robot dan terutama bola untuk keluar dari lapangan. 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.



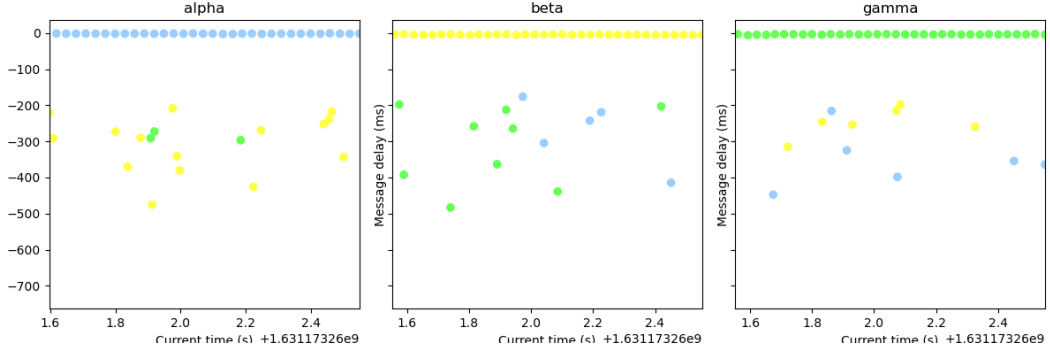
Gambar III.1.3. Bacaan sensor masing-masing robot

### 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 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.

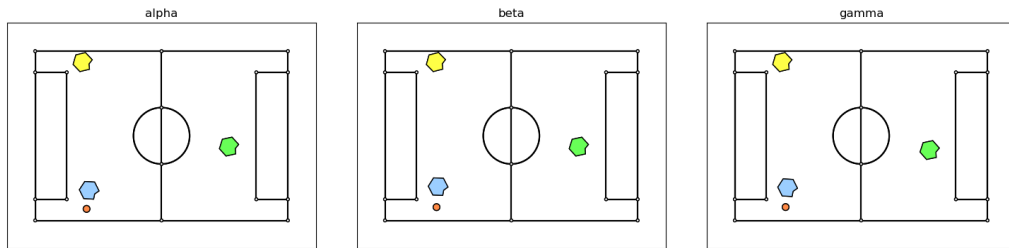
### III.1.3 Simulator Komunikasi



Gambar III.1.4. Latensi 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 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%.

### III.1.4 Agregator 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 seperti pada 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.

### 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 mengupdate posisi atau *pose* dari objek tersebut berdasarkan selisih waktu antara waktu sekarang dan *timestamp* masing-masing objek.

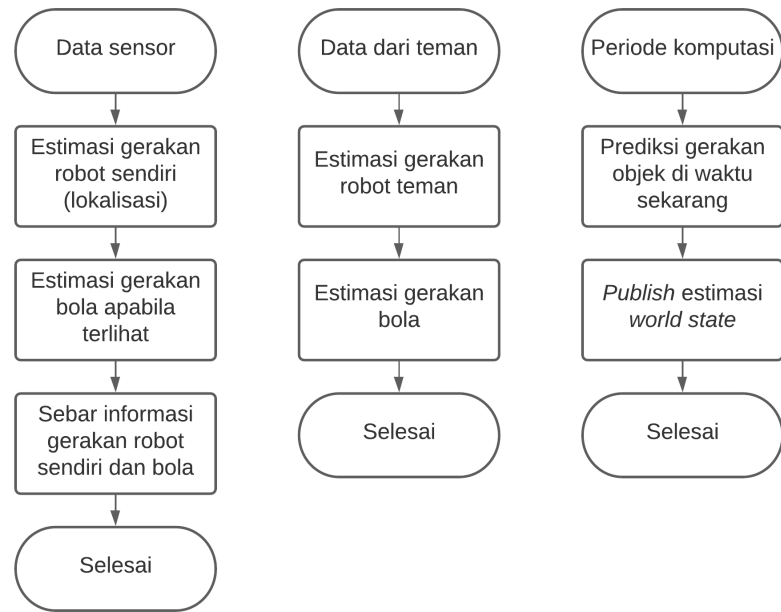
Karena walaupun dalam keadaan kegagalan komunikasi, robot masih harus menjalankan fungsionalitas dasarnya seperti navigasi setiap saat, maka masing-masing robot tim sebaiknya tetap memproses data sensornya masing-masing untuk mendapatkan informasi seperti gerakan robot sendiri, bola, dan mungkin objek 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 mengintegrasikan 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 *eventnya* 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 dituntut dalam segi konstrain waktu komputasi.



Gambar III.3.6. Diagram alir hipotesis solusi

### 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 Penapis Partikel 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( $\alpha \times \ln\_vis\_weight + \beta \times \ln\_ort\_weight$ )
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 diintegrasikan 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 Penapis Partikel 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 mengupdate *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 mengintegrasikan

*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  $\mu$  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  $4 \times 4$  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

$$x_t = A_t x_{t-1} + \epsilon_t \quad (\text{III.1})$$

dimana  $A_t$  adalah matriks  $4 \times 4$  *update* keadaan dimana posisi baru ditambahkan dengan kecepatan dikalikan selisih waktu, sedangkan  $\epsilon_t$  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  $2 \times 4$  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  $\delta_t$  yang sama atau *uniform* untuk semua posisi, maka digeneralisasi  $\delta_t$  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 *statenya* 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 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.6.3 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.6.4 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 robot teman dari suatu robot memiliki *error* yang cukup tinggi jika dibandingkan dengan hasil lokalisasi oleh teman tersebut menggunakan odometri, kompas, dan beberapa persepsi *landmark*, hasil persepsi tersebut lebih tersedia karena bersumber dari sensor sendiri jika dibandingkan dengan hasil lokalisasi teman yang harus melalui latensi jaringan, terutama apabila kondisi jaringan yang sedang buruk. Oleh karena itu, persepsi robot teman menjadi informasi yang berguna karena dapat digunakan untuk mengoreksi estimasi robot sendiri mengenai gerakan robot teman.

Karena informasi hasil lokalisasi teman tersebut tetap lebih berharga dibandingkan persepsi melalui *vision*, digunakan OOSM-PF untuk mencampurkan data dari teman yang terlambat dengan persepsi *vision* yang selalu ada. Algoritma penapis partikel yang digunakan menyediakan dua jenis antarmuka fungsi estimasi, yaitu fungsi yang menerima data dari teman berupa *pose* dan *twist* secara absolut dan fungsi yang menerima data dari sensor berupa posisi relatif teman tersebut. Walaupun model transisinya sama, model pengukuran dari data teman membandingkan jarak nilai *pose* dan *twist* estimasi dan partikel, sedangkan model pengukuran dari data sensor sama seperti penapis partikel bola yang membandingkan posisi relatif robot

teman referensi dengan yang dilihat.

Sensor *vision* yang tidak dapat membedakan teman yang mana yang sedang dilihat mengakibatkan kebutuhan suatu algoritma yang dapat menentukan siapa robot teman yang dilihat berdasarkan posisinya. Dibuat matriks *cost* yang menyimpan *cost* dalam mencocokkan suatu persepsi teman dengan identitas robot teman tersebut. *Cost* berupa jarak kuadrat dari posisi absolut persepsi robot teman tersebut dengan estimasi posisi sesungguhnya suatu identitas robot teman berdasarkan estimasi yang sudah disesuaikan waktunya. Dari sini, masalah penyeleksian identitas teman dapat diselesaikan dengan algoritma seperti *Hungarian algorithm*, tetapi untuk kasus kemungkinan robot teman yang sedikit, dapat langsung *dibrute-force* untuk menentukan penyeleksian dengan jumlah *cost* minimum.



## BAB IV

### EKSPERIMEN DAN ANALISIS

Bab ini menjelaskan mengenai implementasi dan prosedur pengujian algoritma estimasi, perbandingan hasil dari pengujian, dan analisis dari hasil tersebut.

#### IV.1 Skenario Pengujian

Tujuan dari pengujian adalah untuk membandingkan kinerja dari algoritma estimasi yang dipilih dan mendapatkan algoritma estimasi yang paling baik dalam menyelesaikan masalah estimasi gerakan robot tim dan bola dalam kondisi latensi komunikasi.

Pengujian algoritma estimasi dilakukan di lingkungan simulasi yang dibangun pada *framework* ROS yang dibahas di bab sebelumnya. *Node-node* yang berfungsi untuk menjalankan simulasi dan estimasi yaitu *node* simulasi *world state*, simulasi sensor, *world model*, dan simulasi komunikasi ditulis dalam bahasa C++17 yang di *compile* menggunakan gcc dengan optimasi -O3, sedangkan *node* aggregator dan evaluator yang bertugas untuk mengagregasi dan mengambil statistik dari data, beserta melakukan visualisasi ditulis dalam bahasa Python 3 menggunakan *library* matplotlib. Beberapa *node* tambahan juga digunakan seperti *node timer* untuk menyelesaikan simulasi setelah waktu tertentu, dan *node* untuk menyimpan konfigurasi dan merekam eksekusi simulasi agar simulasi dapat ditampilkan ulang.

Berikut adalah spesifikasi *hardware* yang digunakan dalam pengujian di lingkungan simulasi:

1. OS Ubuntu 20.04.3 LTS
2. CPU Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz (12 CPUs)
3. RAM 16 GB

Pengujian dilakukan dengan mengambil statistik metrik *error* berupa jarak dari estimasi masing-masing robot dibandingkan dengan *world state* yang sesungguhnya. Masing-masing jenis *world model* dijalankan sebanyak lima kali tiga menit waktu berjalannya program, sehingga didapatkan sekitar  $5 \times 3 \times 60 \times 1000 : 30 = 30000$  banyak nilai per statistik yang akan dianalisis menggunakan ringkasan lima angkanya dan statistik lain seperti rata-rata.

Skenario pengujian dapat dijalankan dalam tiga jenis kondisi jaringan, yaitu:

1. Kondisi baik dengan 80% keadaan jaringan baik
2. Kondisi sedang dengan 50% keadaan jaringan baik
3. Kondisi buruk dengan 20% keadaan jaringan baik

Untuk perbandingan kinerja algoritma, diiterasi beberapa jenis *node world model* dengan variasi algoritma estimasi yang berbeda-beda dengan iterasi sebelumnya:

1. *world model* O dimana lokalisasi dan estimasi dilakukan dengan naif
2. *world model* P yang melakukan lokalisasi dengan penapis partikel *pose* dan estimasi bola menggunakan penapis Kalman
3. *world model* Q yang melakukan lokalisasi dengan penapis partikel *pose* dan *twist*
4. *world model* R yang mengestimasi bola menggunakan penapis partikel
5. *world model* RB yang tidak melakukan penyesuaian waktu saat mengirimkan hasil estimasi
6. *world model* S yang mengestimasi bola dengan OOSM-PF
7. *world model* SB yang mengestimasi bola dengan OOSM-PF dengan data opsional
8. *world model* SC yang mengestimasi bola dengan OOSM-KF

9. *world model* SD yang mengestimasi bola dengan OOSM-KF dengan data kombinasi

10. *world model* T yang mengoreksi lokasi teman menggunakan data *vision*

## **IV.2 Pengujian dan Analisis Algoritma Estimasi Dasar**

### **IV.2.1 Perbandingan Algoritma tanpa Penapis dan dengan Penapis**

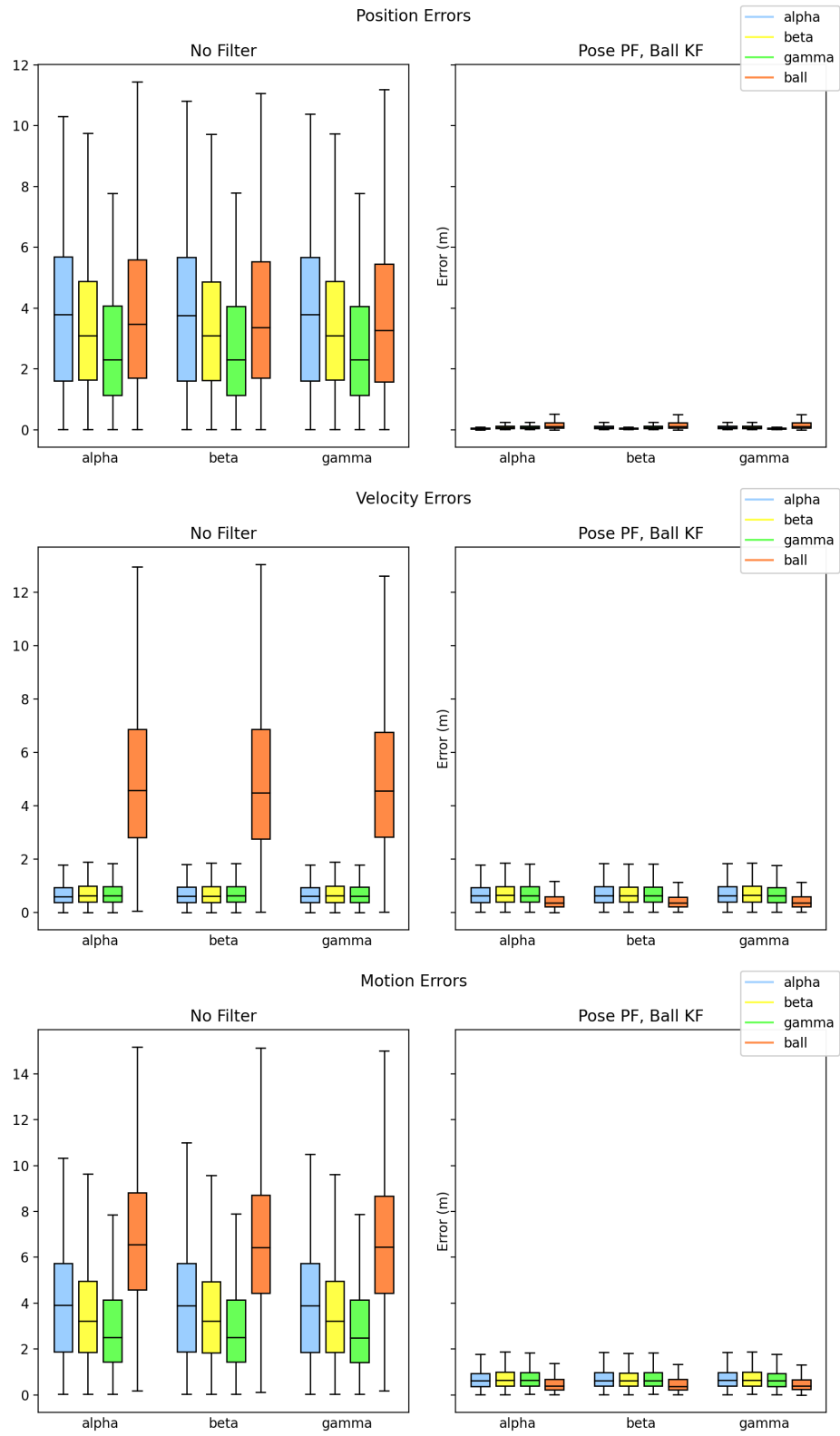
*World model* O yang melakukan lokalisasi dengan naif dengan mengakumulasi nilai odometri tanpa koreksi *vision* dan menggunakan data persepsi bola dengan mentah dibuat sebagai pembanding paling dasar dengan algoritma dengan penapis sederhana.

*World model* P menggunakan penapis partikel berbasis *pose* untuk lokalisasinya dan penapis Kalman untuk mengestimasi posisi bola. Data dari teman digunakan secara mentah dan data bola dimasukkan ke dalam penapis apabila *timestamp*-nya lebih baru. Dilakukan penyesuaian waktu dari *timestamp* data ke waktu saat ini saat hasil estimasi objek dikirimkan.

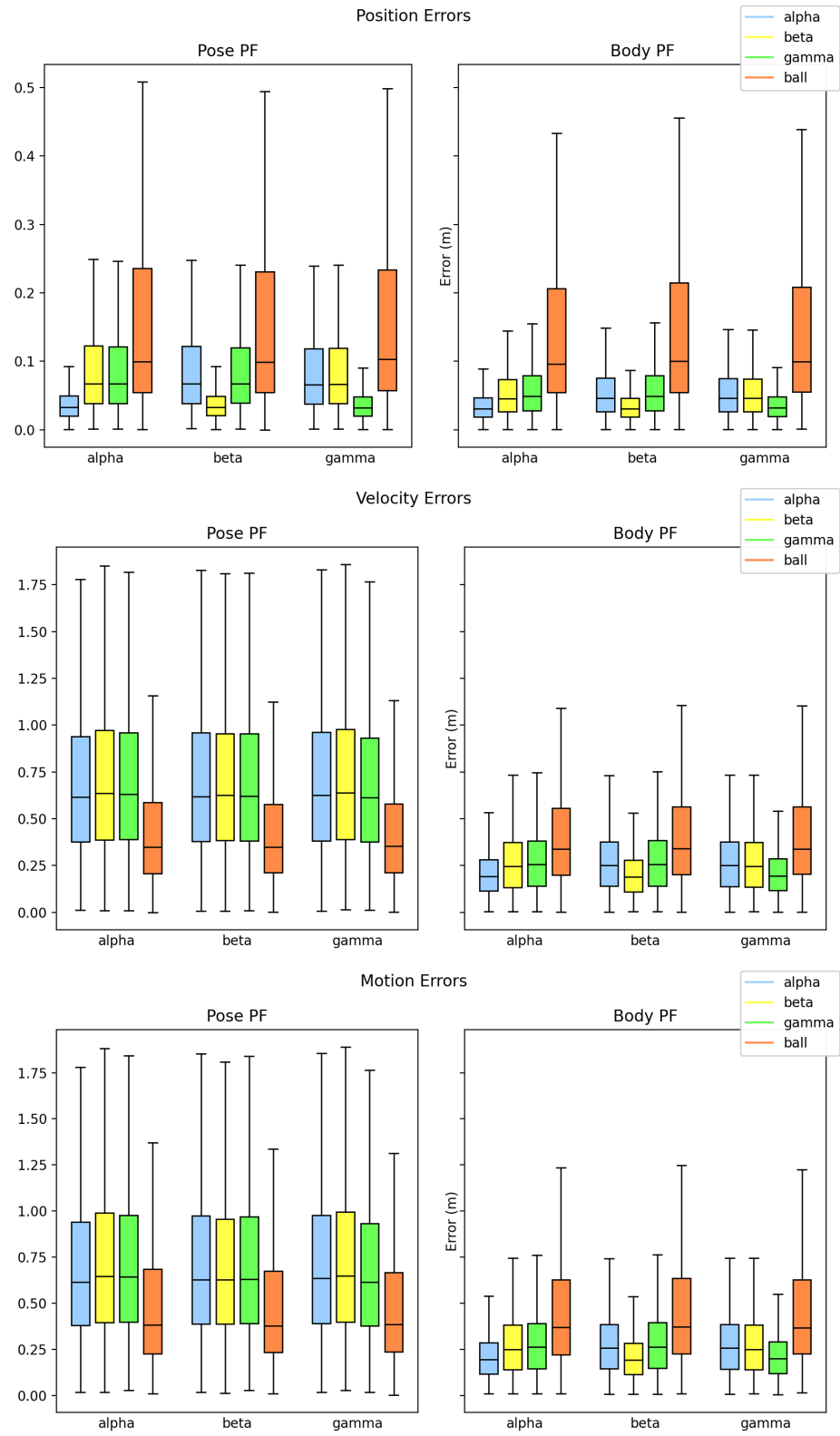
Dari grafik IV.2.1 dapat dilihat bahwa ada peningkatan yang sangat signifikan dalam akurasi lokalisasi robot. Dari diagonal lapangan yang sepanjang 10,82 meter, *world model* O mendapat *error* yang tersebar dalam rentang sebesar itu. Sehingga perbandingan ini menggambarkan seberapa berisiknya *error* sensor yang terakumulasi dalam bacaan odometri seperti sensor lainnya, sedangkan didapatkan bahwa tidak ada perbedaan yang signifikan dari estimasi kecepatan pada penapis partikel berbasis *pose* dan *noise* dari *error* mentah bacaan odometri. Untuk kecepatan bola, karena kecepatan diturunkan dari selisih posisi, masuk akal bahwa estimasi posisi yang lebih baik akan menghasilkan estimasi kecepatan yang lebih baik juga.

### **IV.2.2 Perbandingan Algoritma Lokalisasi Penapis Partikel Berbasis *Pose* dan Penapis Partikel Berbasis *Twist***

Dibandingkan dengan *world model* P, *world model* Q menggunakan penapis partikel berbasis *pose* dan *twist* untuk melakukan lokalisasi, dibanding *pose* saja.



Gambar IV.2.1. *Error world model O dan P jaringan baik*



Gambar IV.2.2. *Error world model P dan Q jaringan baik*

Pada grafik IV.2.2, didapatkan bahwa penapis partikel berbasis *pose* dan *twist* dapat memiliki performa yang lebih baik, terutama dalam mengestimasi *twist* dimana median *error* yang didapat sekitar tiga kali lipat lebih baik. Karena penapis partikel berbasis *pose* tidak melacak *twist* dalam partikelnya mengakibatkan penapis tidak mengetahui pergerakan jangka panjang dari robot dan hanya memaksimalkan informasi persepsi *landmark* dan kompasnya saja, sehingga tidak dapat seakurat penapis berbasis *pose* dan *twist*. Perhatikan juga dari grafik posisi bahwa peningkatan akurasi dari lokalisasi dapat meningkatkan akurasi dari estimasi gerakan bola, walaupun sedikit, karena estimasi bola bergantung pada hasil lokalisasi.

#### **IV.2.3 Perbandingan Algoritma Estimasi Bola Menggunakan Penapis Kalman dan Penapis Partikel**

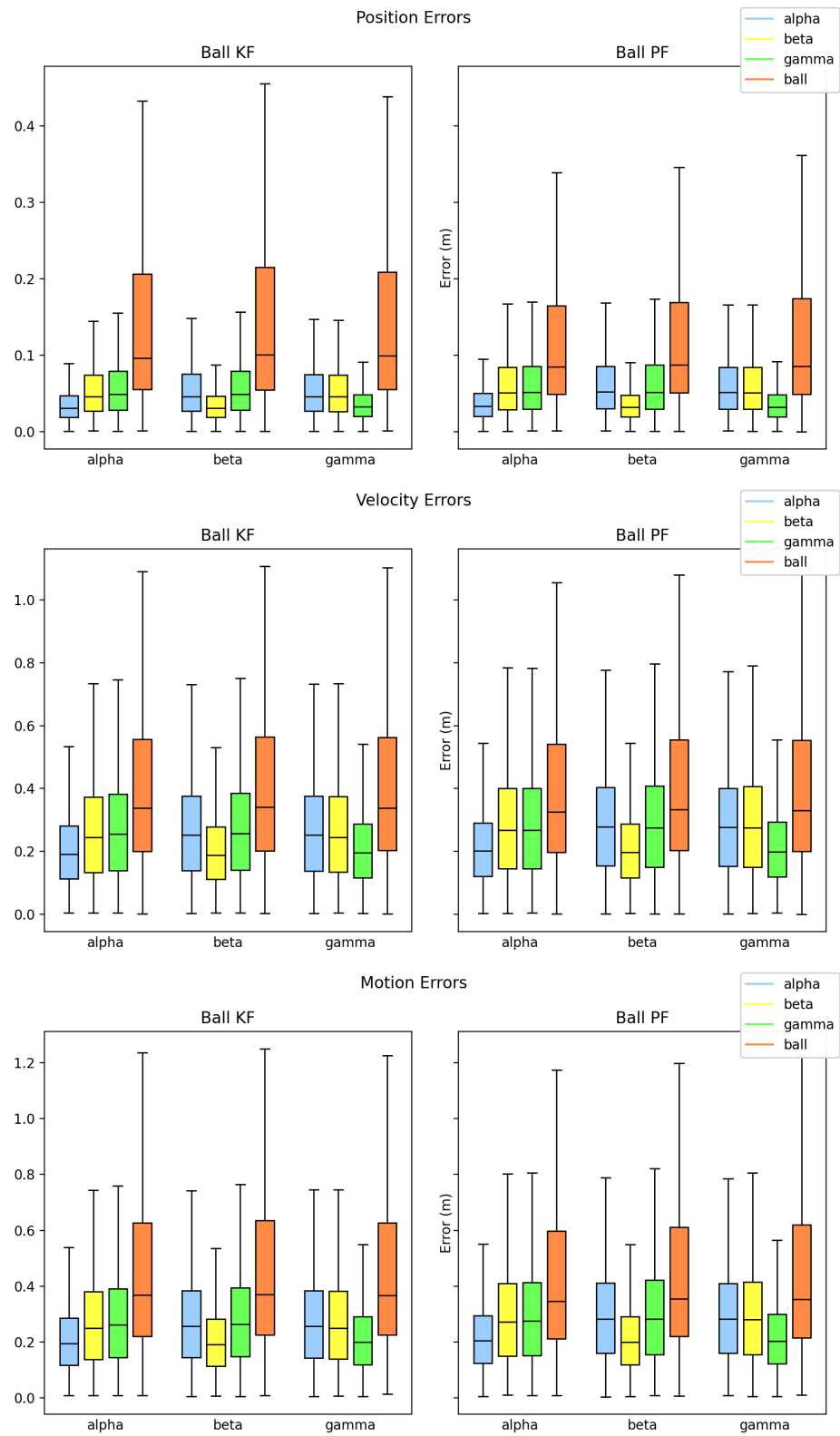
Dibandingkan dengan *world model* Q, *world model* R menggunakan penapis partikel untuk mengestimasi gerakan dari bola, daripada menggunakan penapis Kalman.

Pada grafik IV.2.3, terdapat peningkatan kecil dalam penggunaan penapis partikel dibanding dengan menggunakan penapis Kalman. Hal ini tampaknya disebabkan karena pemodelan pengukuran pada penapis Kalman yang tidak eksak sama dengan profil *error* dari *vision* yang membedakan *error* sejajar dan tegak lurus dengan besar tergantung dari jarak objek, jika dibandingkan dengan penapis partikel yang dapat dengan eksak memodelkan profil *error* tersebut.

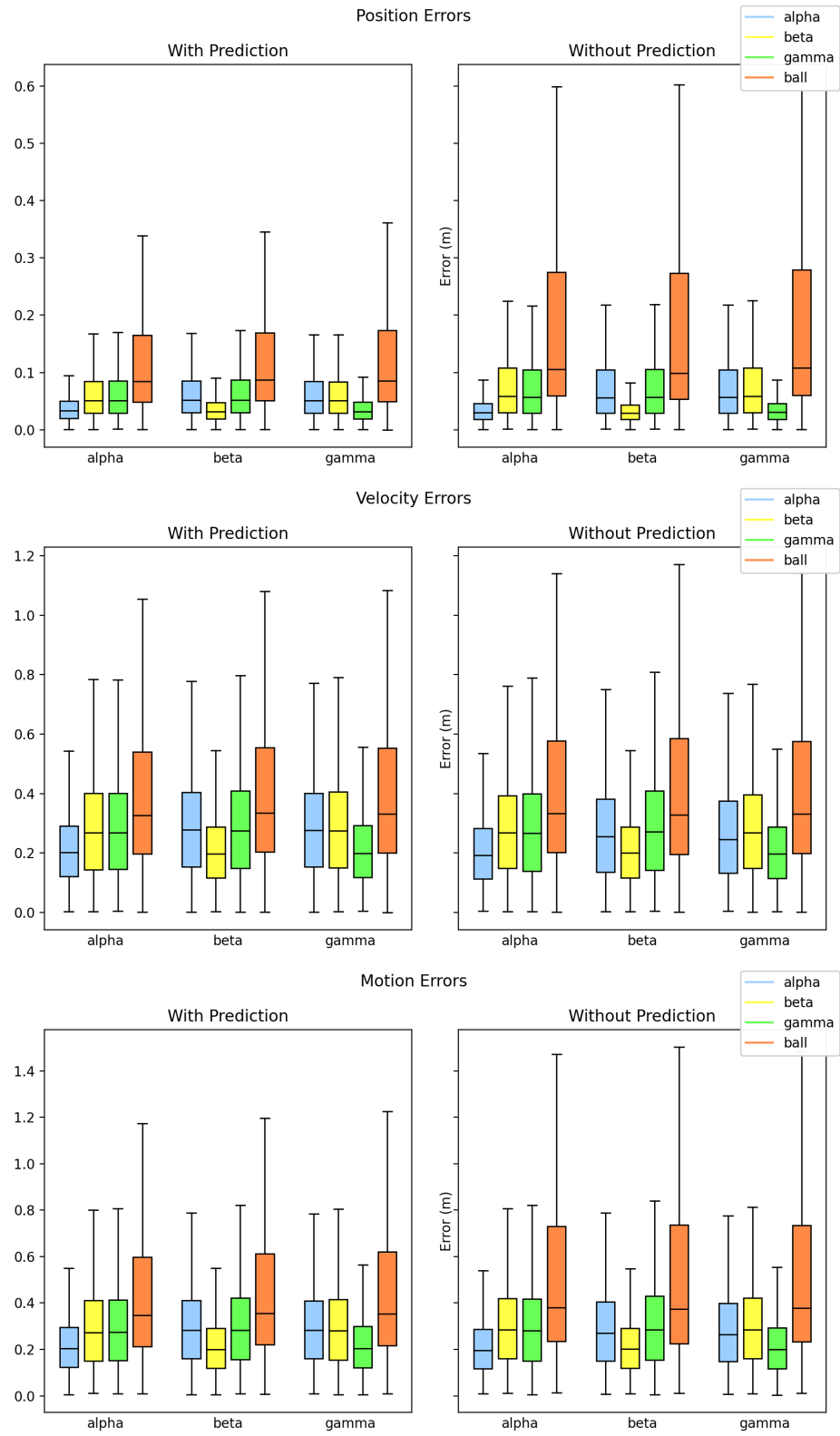
#### **IV.2.4 Perbandingan Algoritma dengan Penyesuaian Waktu dan tanpa Penyesuaian Waktu**

Dibandingkan dengan *world model* R, *world model* RB dibuat tanpa melakukan penyesuaian waktu antara *timestamp* data estimasi dan waktu sekarang saat mengirimkan hasil estimasi ke evaluator.

Pada grafik IV.2.4, hasil estimasi posisi yang tidak dikompensasi cukup lebih buruk jika dibandingkan dengan estimasi dengan kompensasi waktu. Maka dapat dilihat kompensasi waktu dapat membantu untuk meningkatkan akurasi



Gambar IV.2.3. *Error world model Q dan R jaringan baik*



Gambar IV.2.4. *Error world model* R dan RB jaringan baik



estimasi untuk objek seperti robot teman yang mungkin terlambat karena latensi atau bola yang mungkin tidak dipersepsi atau datanya terlambat karena latensi.

### **IV.3 Pengujian dan Analisis Algoritma Estimasi Data Terlambat**

#### **IV.3.1 Perbandingan Algoritma Estimasi Bola Menggunakan PF dan OOSM-PF**

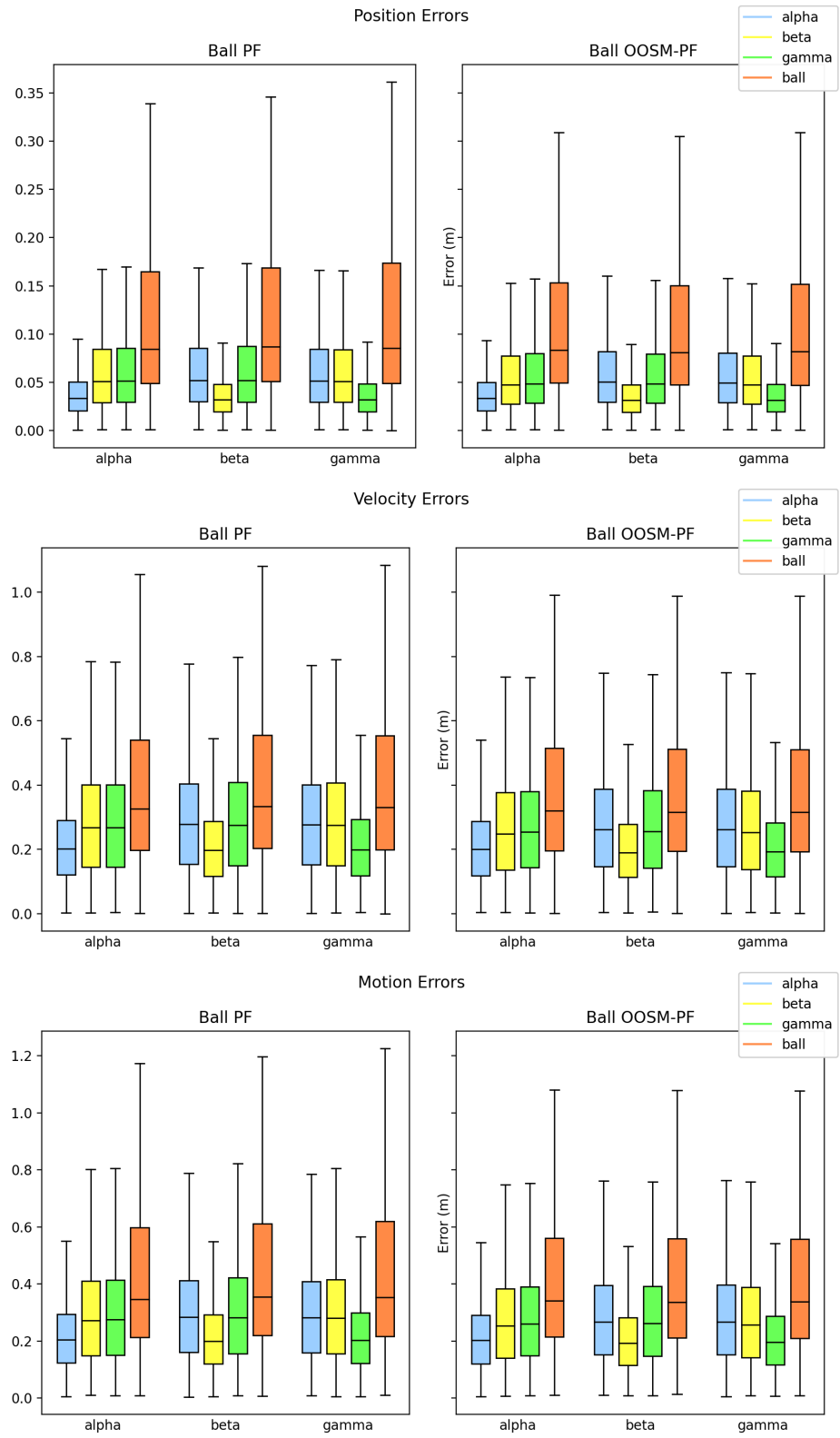
Dibandingkan dengan *world model* R, *world model* S menggunakan modifikasi OOSM-PF untuk dapat mengintegrasikan data bola teman yang terlambat, daripada dengan PF biasa.

Pada grafik IV.3.5, terdapat peningkatan pada hasil estimasi posisi bola pada *world model* yang menggunakan OOSM-PF saat jaringan baik, karena robot dapat mengintegrasikan bacaan dari dua atau lebih robot yang melihat bola pada saat yang bersamaan untuk meningkatkan banyak data yang diintegrasikan dan meningkatkan akurasi. Pada grafik IV.3.6 dan IV.3.7, peningkatan ini semakin berkurang, karena pada saat robot dapat mengintegrasikan data bola yang bersamaan dari dua robot atau lebih, data tersebut sudah cukup lama dan tidak lagi terlalu relevan terhadap hasil yang diestimasi sekarang.

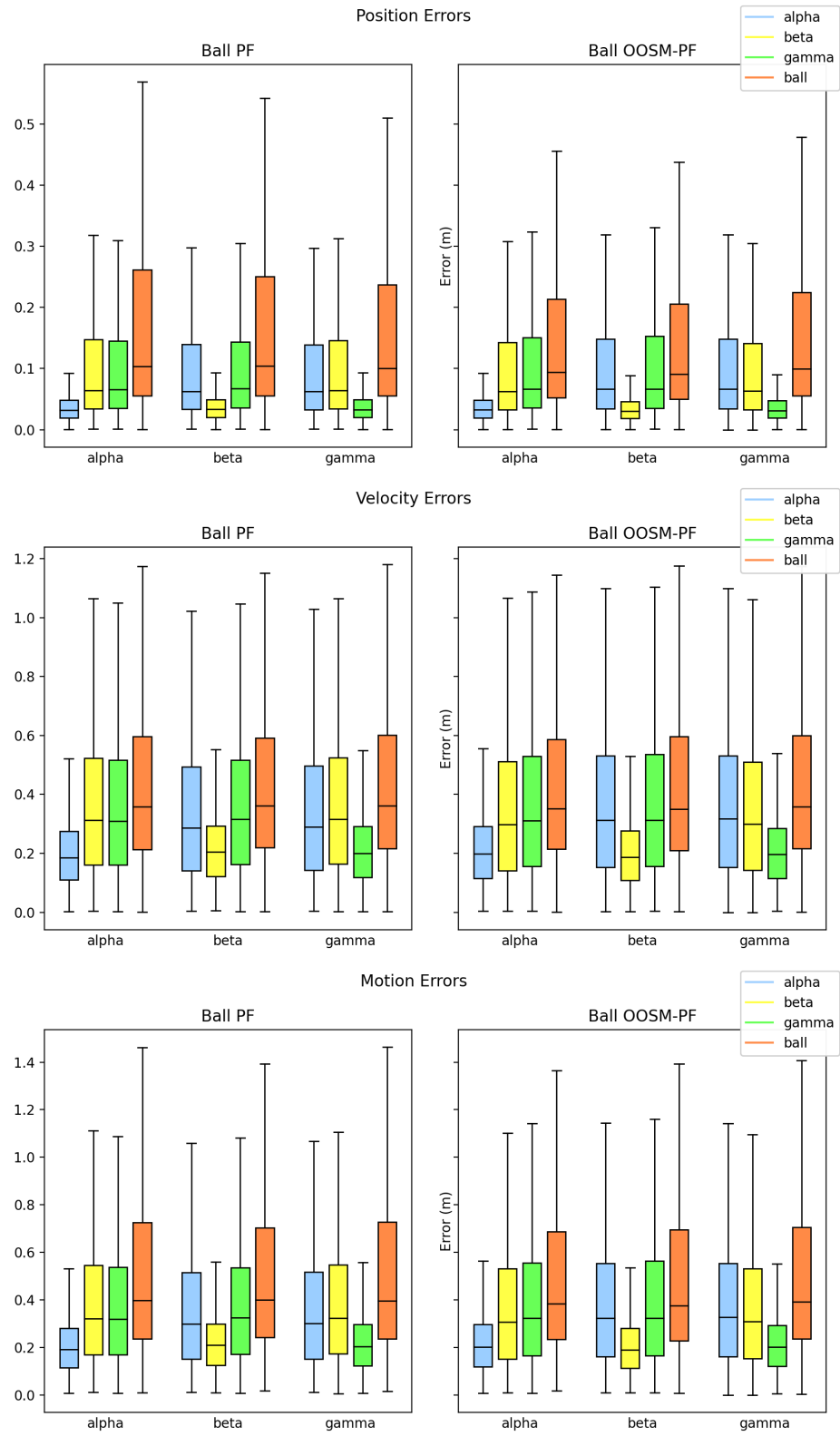
#### **IV.3.2 Perbandingan Algoritma Estimasi Bola tanpa Data Opsional dan dengan Data Opsional**

Dibandingkan dengan *world model* S, *world model* SB menggunakan OOSM-PF yang menerima data sensor yang dapat tidak melihat bola, daripada harus mengandung data bola yang terlihat.

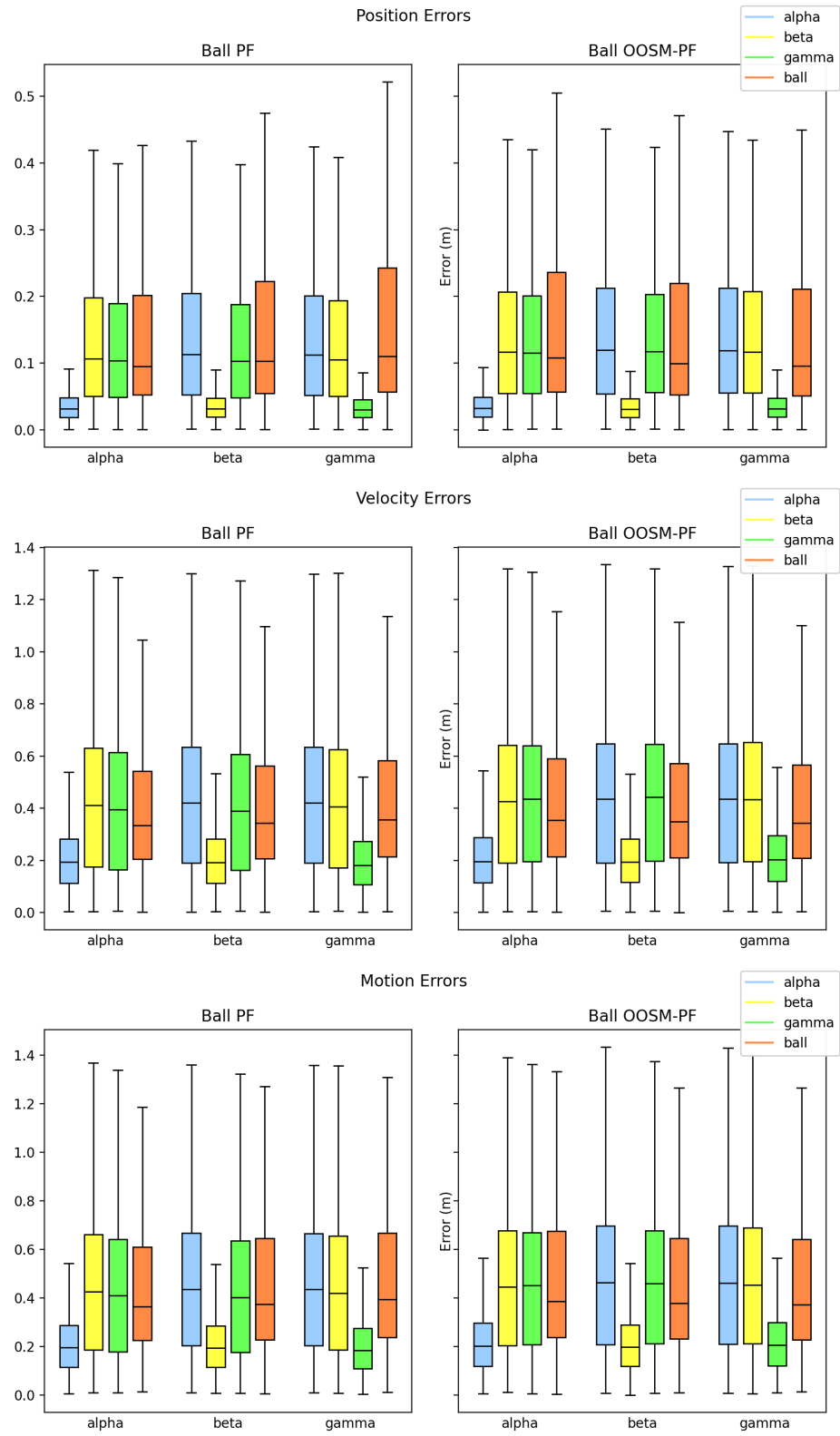
Pada grafik IV.3.8, tidak ada perubahan yang signifikan dari integrasi data yang tidak *null* untuk estimasi posisi dan malah memburuk untuk hasil estimasi kecepatan. Ini tampaknya disebabkan karena algoritma ini tidak mempertimbangkan saat bola tidak terlihat karena oklusi dari objek, sehingga saat bola tidak terlihat, partikel yang ada di luar lingkaran radius penglihatan punya berat yang lebih tinggi dan punya kecenderungan memiliki kecepatan yang lebih tinggi dalam menjauhi robot keluar dari lingkaran.



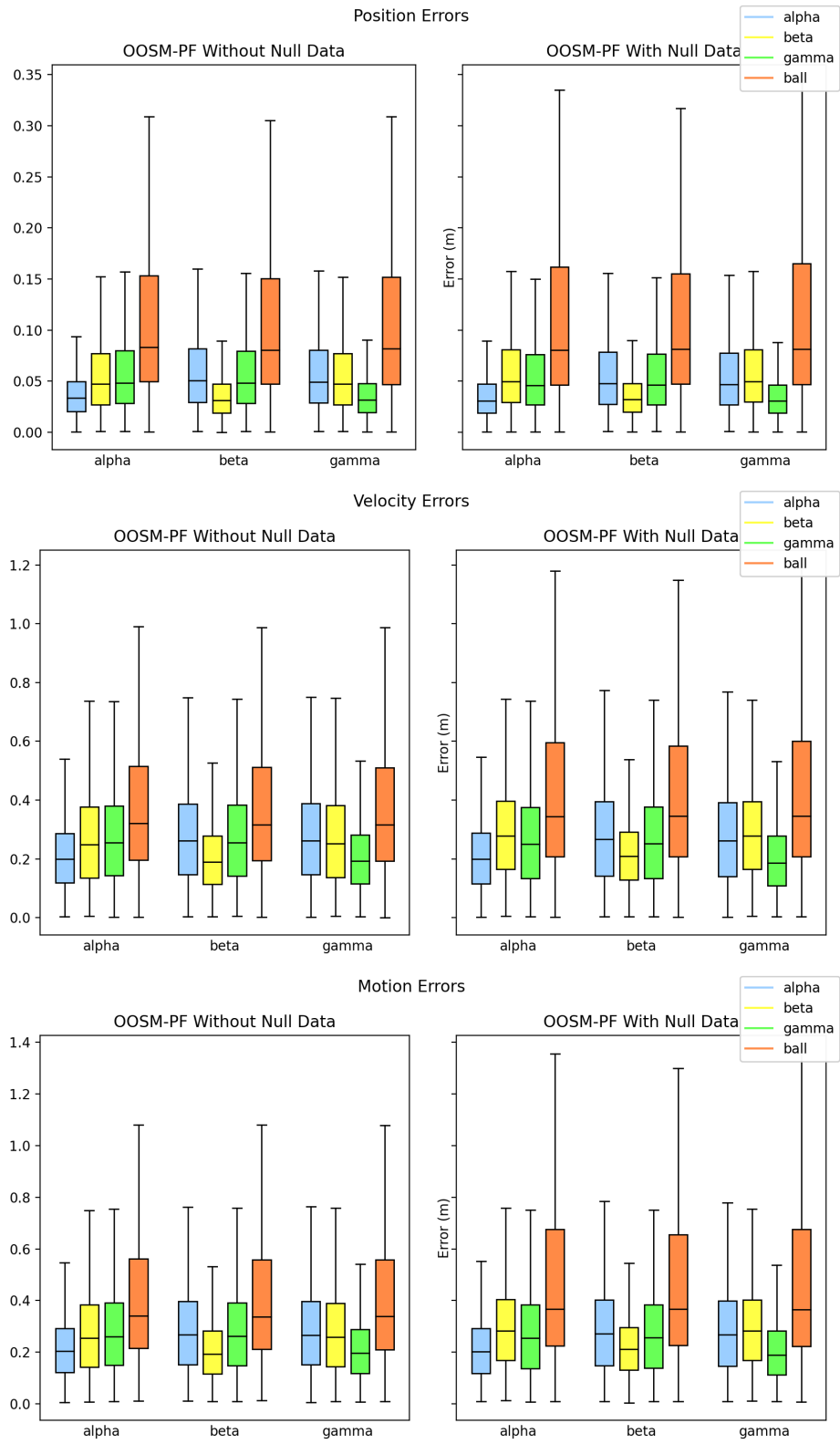
Gambar IV.3.5. *Error world model* R dan S jaringan baik



Gambar IV.3.6. *Error world model* R dan S jaringan sedang



Gambar IV.3.7. *Error world model R dan S jaringan buruk*



Gambar IV.3.8. *Error world model S dan SB jaringan baik*

### **IV.3.3 Perbandingan Algoritma Estimasi Bola dengan OOSM-PF dan OOSM-KF**

Dibandingkan dengan *world model S*, *world model SC* menggunakan OOSM-KF yang berdasarkan algoritma penapis Kalman, daripada algoritma OOSM-PF yang berdasarkan algoritma penapis partikel.

Pada grafik IV.3.9, didapatkan *error* estimasi posisi maupun kecepatan yang lebih buruk. Hal ini kemungkinan disebabkan oleh alasan yang sama dengan perbandingan kinerja PF dan KF dimana KF tidak dapat dengan sempurna memodelkan profil *error* dari *vision*, apalagi saat menggabungkan data dari sudut pandang dua atau lebih robot yang berbeda dengan arah distribusi *error* yang berbeda juga.

### **IV.3.4 Perbandingan Algoritma Estimasi Bola OOSM-PF Biasa dan OOSM-KF dengan Kombinasi Data**

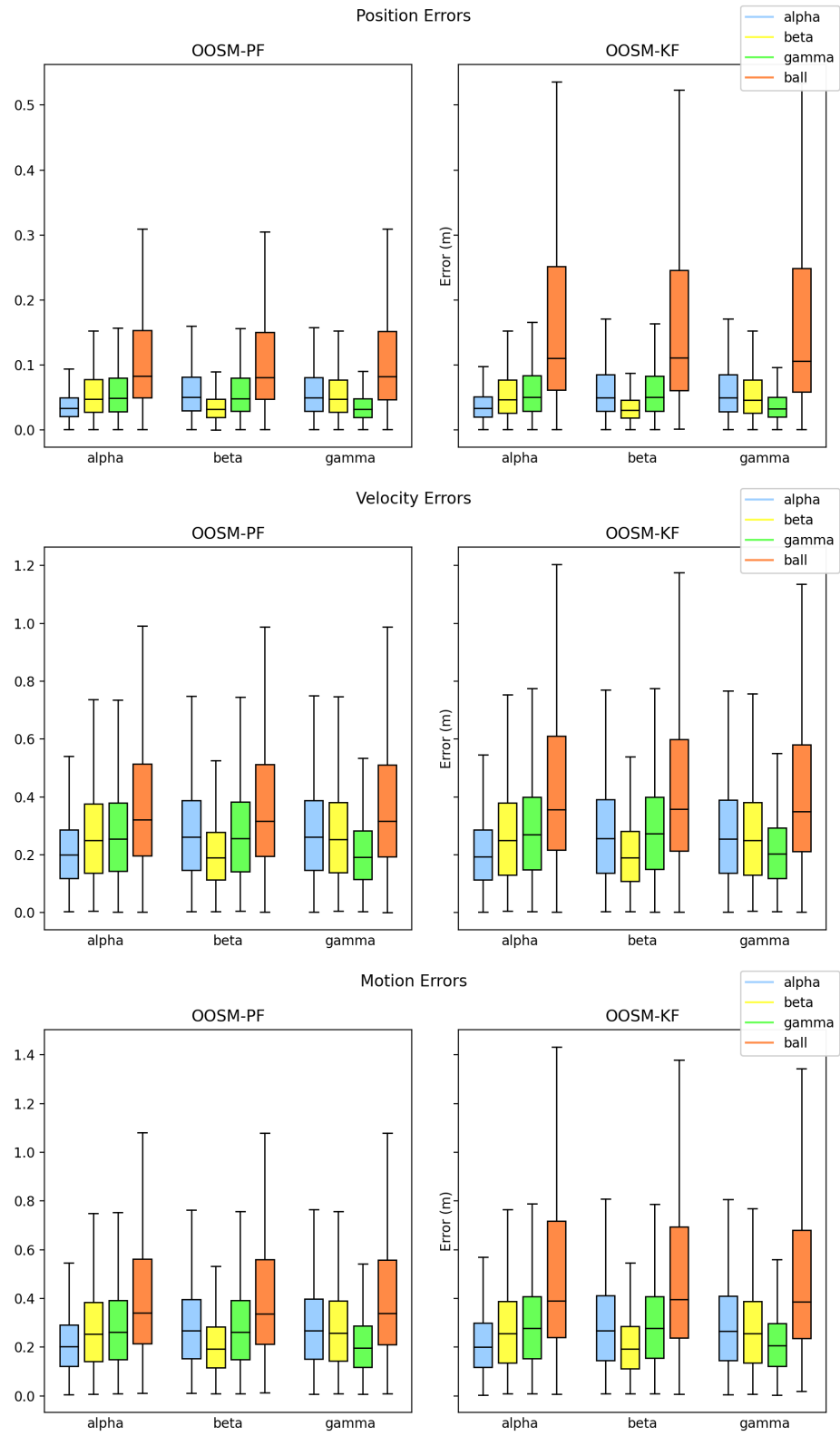
Dibandingkan dengan *world model S*, *world model SD* menggunakan OOSM-KF yang menggabungkan beberapa persepsi bola pada waktu yang sama dengan merata-ratakan posisinya.

Pada grafik IV.3.10, didapatkan *error* estimasi posisi maupun kecepatan yang lebih buruk, dan kinerja *world model SD* yang tidak jauh dari SC, yang kemungkinan dikarenakan oleh alasan yang sama dimana penggunaan OOSM-KFnya tidak cocok untuk profil *error vision* yang ada.

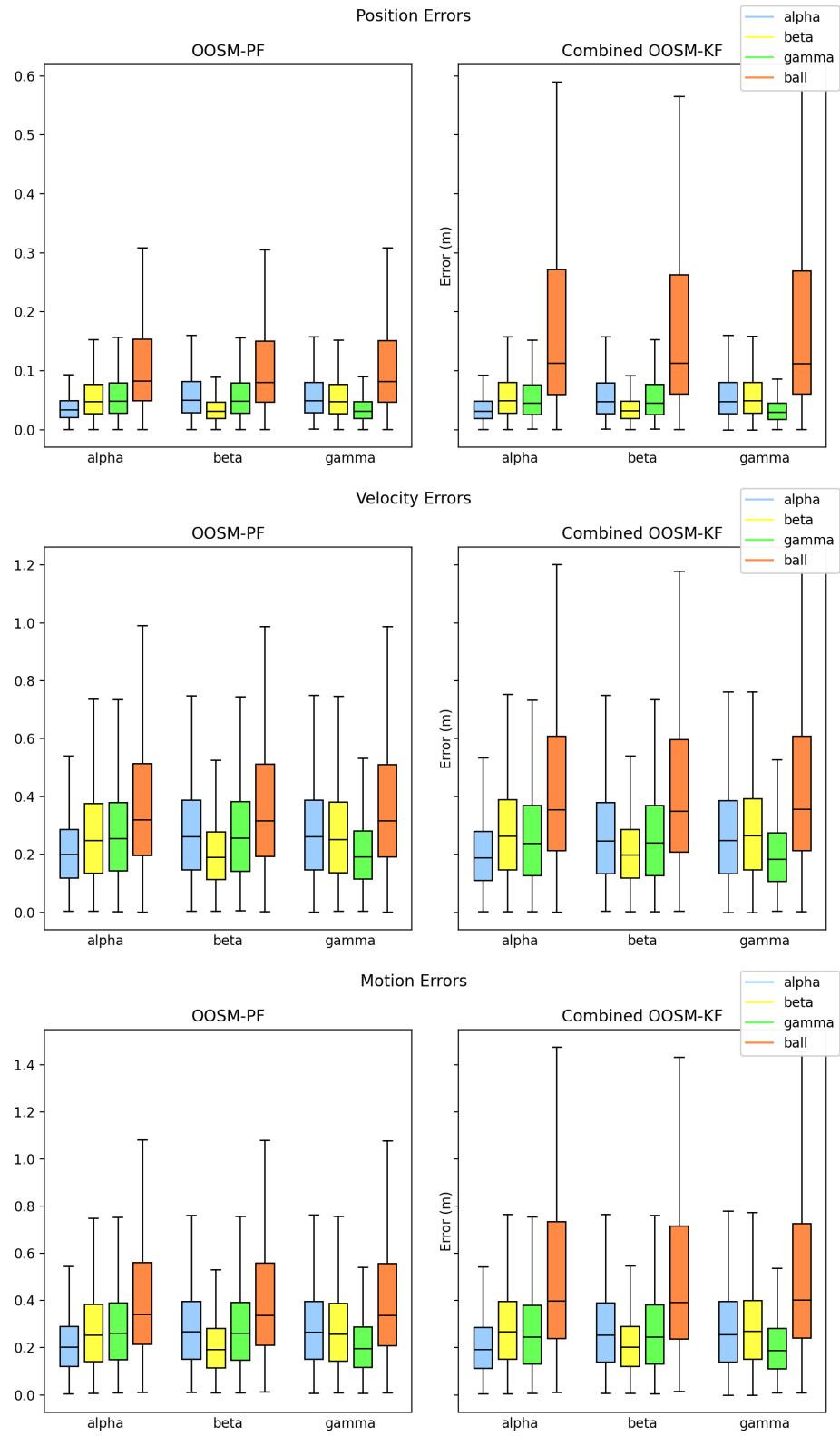
### **IV.3.5 Perbandingan Algoritma tanpa dan dengan Koreksi Posisi Robot Teman**

Dibandingkan dengan *world model S*, *world model T* mengoreksi posisi teman yang diterima dari komunikasi menggunakan persepsi sendiri dari *vision*.

Pada grafik IV.3.11, hasil estimasi posisi dan kecepatan dari robot teman sedikit memburuk. Hal ini dikarenakan hasil persepsi *vision* yang lebih *noisy* dibandingkan dengan hasil lokalisasi yang menggunakan beberapa sensor berbeda. Sedangkan pada grafik IV.3.12 dan IV.3.13, koreksi robot teman dapat membantu meningkatkan akurasi estimasi posisi dan kecepatan karena

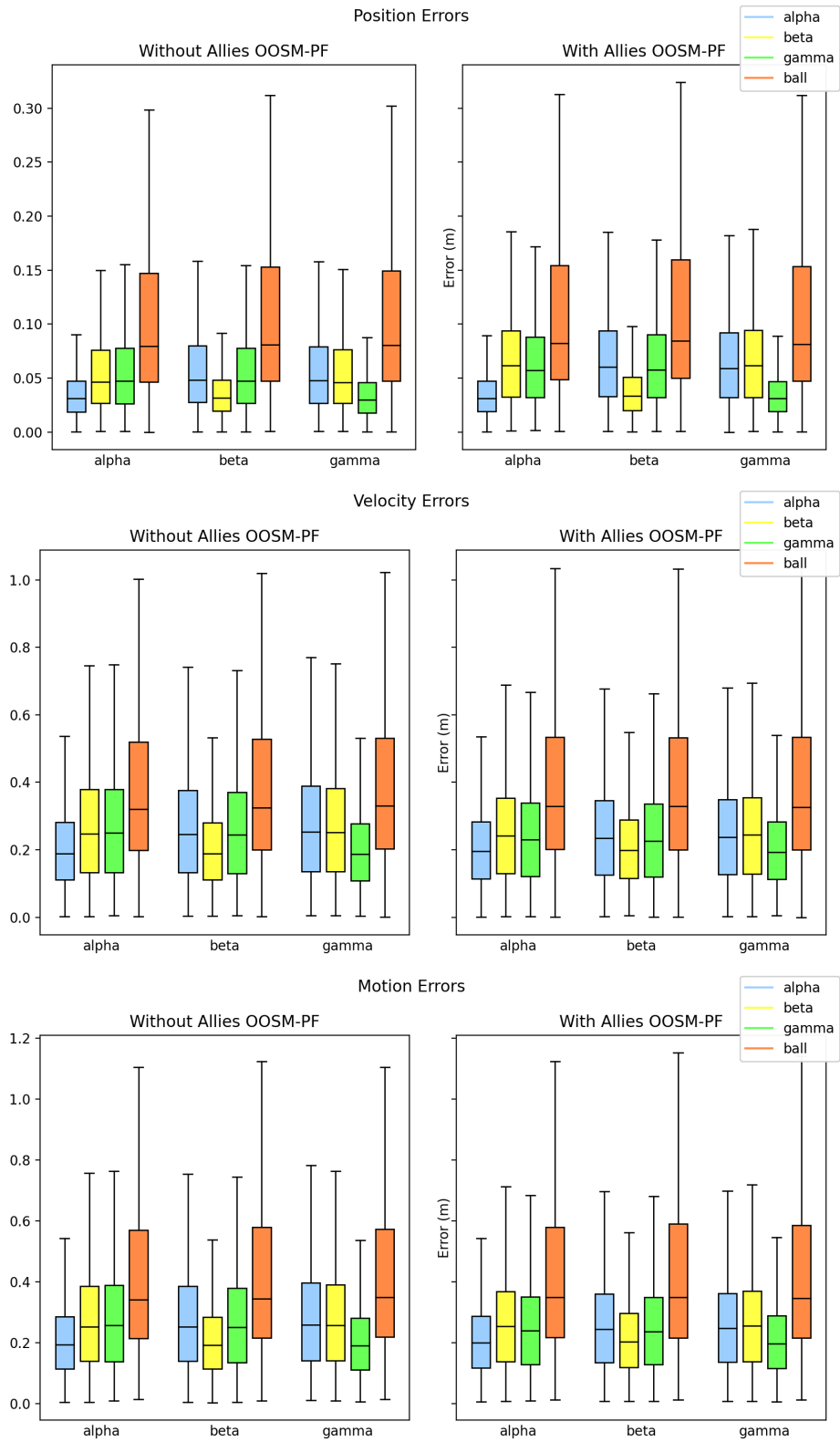


Gambar IV.3.9. *Error world model S dan SC jaringan baik*

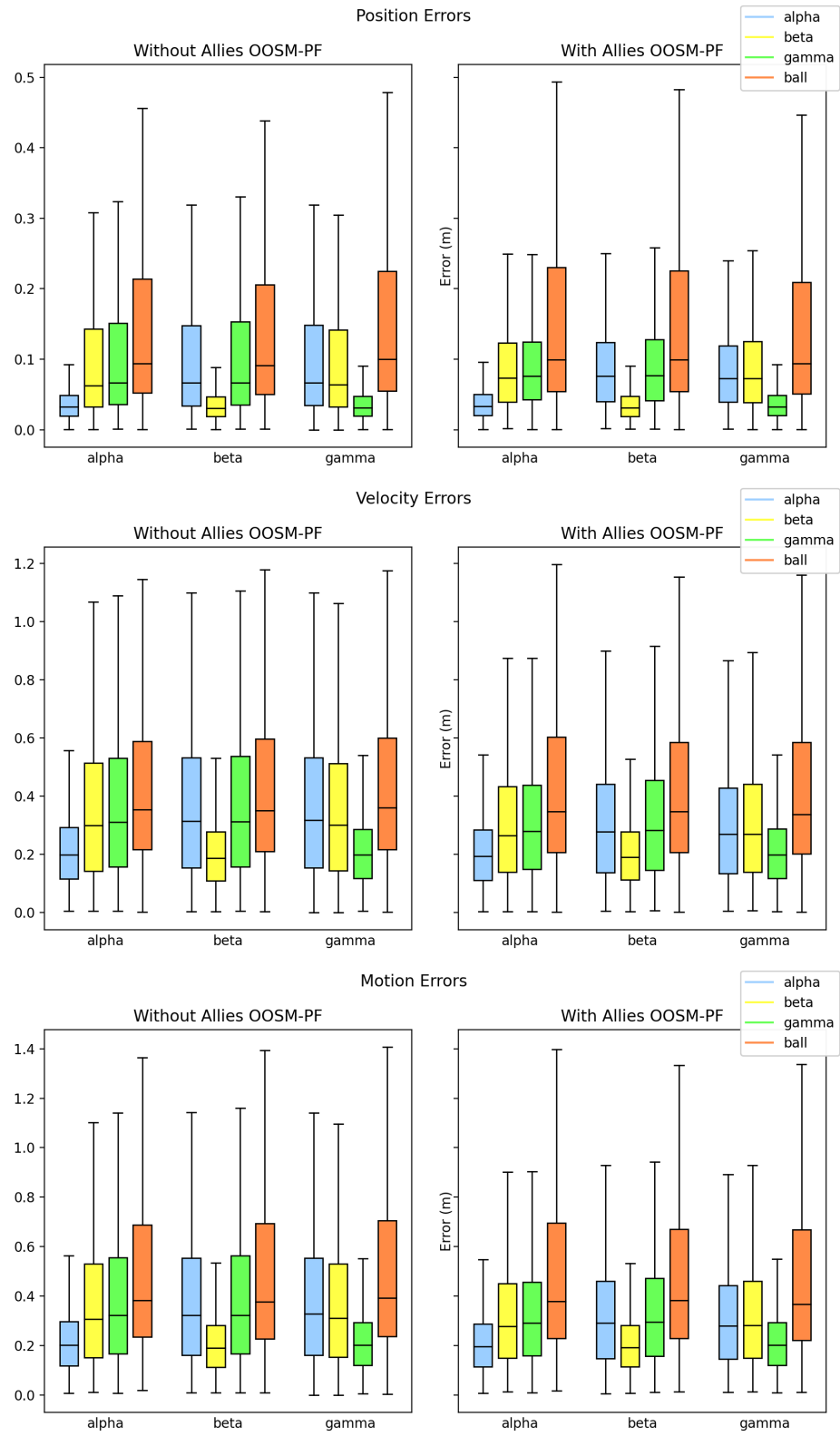


Gambar IV.3.10. *Error world model* S dan SD jaringan baik

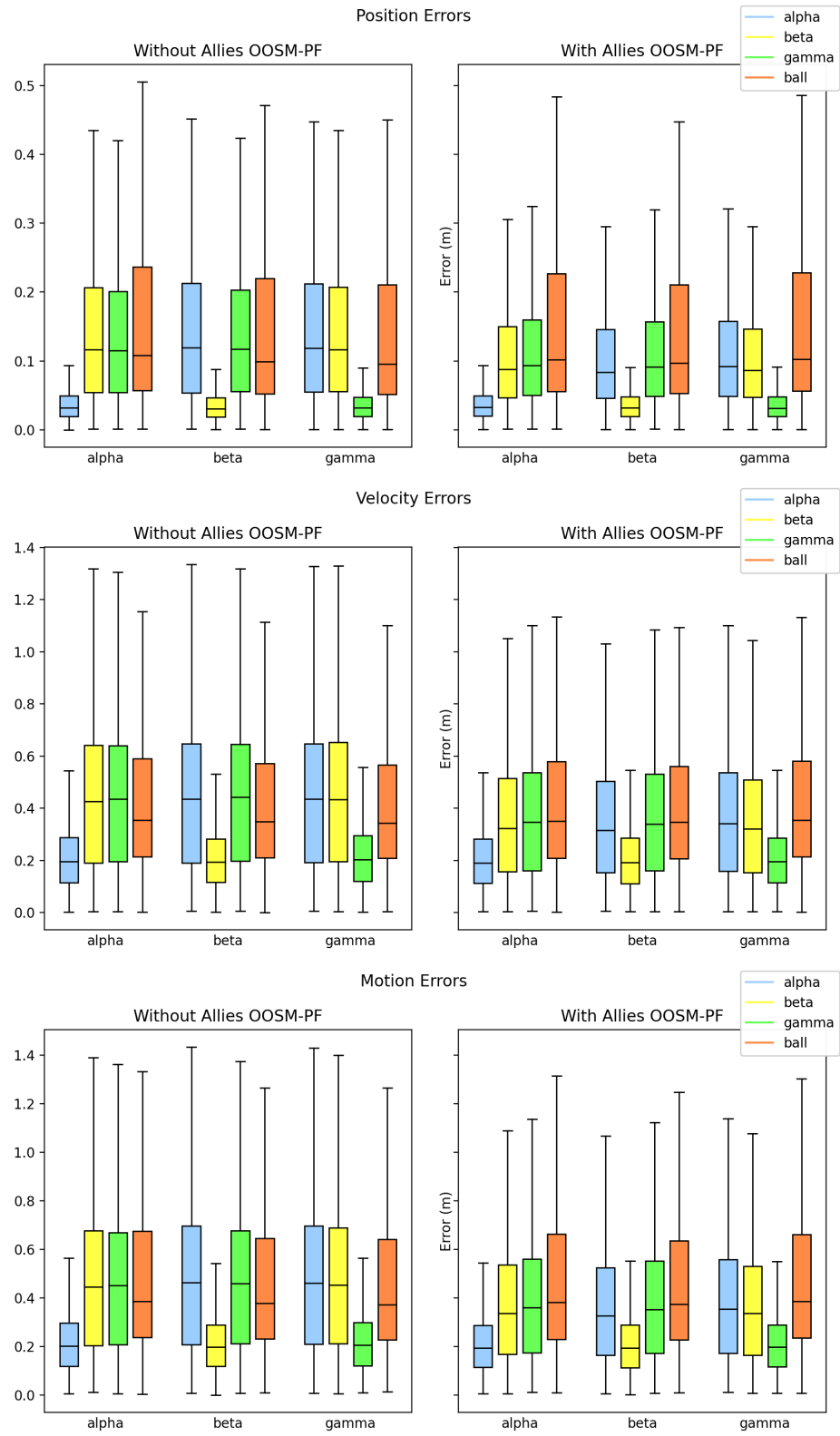




Gambar IV.3.11. *Error world model S dan T jaringan baik*



Gambar IV.3.12. *Error world model S dan T jaringan sedang*



Gambar IV.3.13. *Error world model S dan T jaringan buruk*

persepsi *vision* yang lebih tersedia dibandingkan menunggu informasi dari robot teman yang terlambat.

## BAB V

### PENUTUP

Bab ini memaparkan hasil-hasil simpulan yang dapat ditarik berdasarkan uji coba beserta saran yang dapat diambil untuk pengembangan-pengembangan berikutnya.

#### V.1 Simpulan

Berdasarkan hasil eksperimen yang diperoleh, didapatkan bahwa algoritma estimasi gerakan robot tim dan bola yang paling akurat adalah algoritma yang:

1. Menggunakan algoritma AMCL berbasis *pose* dan *twist* untuk mengestimasi gerakan robot sendiri
2. Menggunakan algoritma penapis partikel dengan penanganan data terlambat untuk mengestimasi gerakan bola
3. Menggunakan algoritma penapis partikel dengan penanganan data terlambat untuk mengestimasi dan mengoreksi gerakan robot teman
4. Mengoreksi hasil estimasi berdasarkan selisih waktu sekarang dengan *timestamp* data terakhir

Konfigurasi-konfigurasi tersebut mempunyai kinerja yang lebih baik dibandingkan dengan alternatif seperti menggunakan AMCL berbasis *pose* saja atau estimasi gerakan bola menggunakan penapis Kalman.

#### V.2 Saran

Saran yang dapat diterapkan untuk pengembangan-pengembangan berikutnya diantaranya adalah:

1. Memvalidasi hasil uji coba dengan percobaan langsung di lingkungan nyata dengan robot di lapangan atau menggunakan data yang diambil dari uji coba robot di lapangan

2. Melakukan koreksi posisi robot teman menggunakan persepsi *vision* terhadap teman tersebut
3. Menyebarkan seluruh distribusi estimasi dibandingkan hanya estimasinya saja melalui komunikasi
4. Menerapkan berbagai variasi modifikasi penapis Kalman yang ada

## 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.