

**IMPLEMENTASI POSE ESTIMATION UNTUK
PEMODELAN GERAK TARI TRADISIONAL INDONESIA
PADA ROBOT HUMANOID TARI ROBOTIS OP3**

SKRIPSI

Digunakan Sebagai Syarat Maju Ujian Diploma IV Politeknik Negeri Malang

Oleh:

AIDO LUTHFI AL HAKIM NIM.2141720136



**PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNOLOGI INFORMASI
POLITEKNIK NEGERI MALANG
JULI 2025**

lembar pengesahan

PERNYATAAN

Dengan ini saya menyatakan bahwa pada Skripsi ini tidak terdapat karya, baik seluruh maupun sebagian, yang sudah pernah diajukan untuk memperoleh gelar akademik di Perguruan Tinggi manapun, dan sepanjang pengetahuan saya juga tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini serta disebutkan dalam daftar sitasi/pustaka.

Malang, 16 Juli 2025

Aido Luthfi Al Hakim

ABSTRAK

Hakim, Aido Luthfi Al. “Implementasi *Pose Estimation* untuk Pemodelan Gerak Tari Tradisional Indonesia pada Robot *Humanoid* Tari Robotis OP3”. **Pembimbing:** (1) Wilda Imama Sabilla, S.Kom., M.Kom., (2) Ariadi Retno Tri Hayati Ririd, S.Kom., M.Kom.

Skripsi, Program Studi Teknik Informatika, Jurusan Teknologi Informasi, Politeknik Negeri Malang, 2025.

Perkembangan teknologi robotika membuka peluang bagi inovasi dalam berbagai bidang, termasuk pelestarian budaya melalui media robot *humanoid*. Salah satu penerapannya adalah pada Kontes Robot Seni Tari Indonesia (KRSTI), di mana robot diharuskan menampilkan gerakan tari tradisional secara natural dan sesuai dengan irama musik. Namun, pembuatan gerakan tari pada robot *humanoid* saat ini masih dilakukan secara manual sehingga tidak efisien. Penelitian ini mengembangkan sistem *pose estimation* untuk pemodelan gerak tari tradisional Indonesia pada robot *humanoid* ROBOTIS-OP3. Sistem menggunakan model *Metrabs* untuk deteksi pose 3D dan *DeciWatch* untuk penyempurnaan prediksi temporal. Pengujian pada gerakan T-pose dan tari Denok Semarangan menunjukkan penurunan MPJPE dari 107 mm menjadi 64,98 mm, dengan *loss* sebesar 0,1522 dan waktu pelatihan 50 jam menggunakan *custom training*. Sistem ini mendukung pelestarian budaya berbasis teknologi.

Kata Kunci: Pose Estimation, Robot Humanoid, OP3, Gerak Tari

ABSTRACT

Hakim, Aido Luthfi Al. “*Implementasi Pose Estimation untuk Pemodelan Gerak Tari Tradisional Indonesia pada Robot Humanoid Tari Robotis OP3*”. **Supervisor:** Wilda Imama Sabilla, S.Kom., M.Kom., **Co-Supervisor:** Ariadi Retno Tri Hayati Ririd, S.Kom., M.Kom.

Thesis, Information Engineering Study Program, Department of Information Technology, State Polytechnic of Malang, 2025.

The development of humanoid robots provides opportunities for cultural preservation through robotic dance performance. However, creating dance movements for humanoid robots is still mostly done manually, resulting in inefficiency and time-consuming processes. This research develops a pose estimation system for modeling Indonesian traditional dance movements on the ROBOTIS-OP3 robot. The system utilizes Metrabs for 3D pose detection and DeciWatch for temporal refinement. The output is converted into servo joint angles using Trigonometric Inverse Kinematics with motion retargeting. Tests on T-pose and Denok Semarangan dance show a reduction in MPJPE from 107 mm to 64.98 mm, with loss of 0.1522 and 50 hours of custom training. This system promotes technology-based cultural preservation.

Keywords: *Pose Estimation, Humanoid Robot, OP3, Dance Movement*

KATA PENGANTAR

Puji syukur kami panjatkan ke hadirat Allah SWT/Tuhan YME atas segala rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan skripsi dengan judul "*IMPLEMENTASI POSE ESTIMATION UNTUK PEMODELAN GERAK TARI TRADISIONAL INDONESIA PADA ROBOT HUMANOID TARI ROBOTIS OP3*". Skripsi ini disusun sebagai salah satu syarat untuk menyelesaikan studi program Diploma IV Program Studi Teknik Informatika, Jurusan Teknologi Informasi, Politeknik Negeri Malang.

Penulis menyadari bahwa tanpa adanya dukungan, bimbingan, dan kerja sama dari berbagai pihak, penyusunan laporan akhir ini tidak akan berjalan dengan baik. Oleh karena itu, dengan segala kerendahan hati, penulis ingin menyampaikan rasa terima kasih kepada:

1. Bapak Prof. Dr. Eng. Rosa Andrie Asmara, S.T., M.T., selaku Ketua Jurusan Teknologi Informasi.
2. Ibu Ely Setyo Astuti, S.T., M.T., selaku Ketua Program Studi D-IV Teknik Informatika.
3. Ibu Wilda Imama Sabilla, S.Kom., M.Kom., selaku Dosen Pembimbing Utama yang telah memberikan bimbingan, arahan, dan motivasi kepada penulis.
4. Ibu Ariadi Retno Tri Hayati Ririd, S.Kom., M.Kom., selaku Dosen Pembimbing Pendamping yang telah banyak membantu dalam proses penyusunan skripsi ini.
5. Bapak Indrazno Siradjuddin, S.T., M.T., Ph.D., selaku Kepala Laboratorium Robotika yang telah memberikan fasilitas dan tempat untuk melakukan riset serta turut membantu dalam pelaksanaan penelitian ini.
6. Seluruh tim robotika yang telah memberikan dukungan, bantuan, serta masukan yang sangat berharga selama proses penelitian berlangsung.
7. Dan seluruh pihak yang telah membantu dan mendukung kelancaran pembuatan laporan akhir ini dari awal hingga selesai, yang tidak dapat disebutkan satu per satu.

Penulis menyadari bahwa dalam penyusunan laporan ini masih terdapat

banyak kekurangan, baik dalam penyampaian materi, sistematika penulisan, maupun penggunaan bahasa. Oleh karena itu, penulis mengharapkan kritik dan saran yang bersifat membangun demi perbaikan di masa mendatang.

Akhir kata, penulis berharap laporan ini dapat bermanfaat bagi pembaca pada umumnya dan bagi penulis khususnya. Semoga karya ini menjadi awal yang baik dalam penerapan ilmu yang telah diperoleh selama masa studi.

Malang, 16 Juli 2025

Penulis

DAFTAR ISI

LEMBAR PENGESAHAN	ii
PERNYATAAN	iii
ABSTRAK	iv
<i>ABSTRACT</i>	v
KATA PENGANTAR	vi
DAFTAR ISI	viii
DAFTAR GAMBAR	xi
DAFTAR TABEL	xiii
DAFTAR LAMPIRAN	xiv
BAB I. PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	3
1.3 Batasan Masalah	3
1.4 Tujuan	3
1.5 Manfaat	4
BAB II. LANDASAN TEORI	5
2.1 Studi Literatur	5
2.2 Dasar Teori	10
2.2.1 Robot <i>Humanoid Robotis-OP3</i>	10
2.2.2 Dynamixel	11
2.2.3 <i>Robot Operating System</i>	13
2.2.4 Visi Komputer	13
2.2.5 <i>Deep Learning</i>	14
2.2.6 <i>Convolutional Neural Networks (CNN)</i>	15
2.2.7 <i>Transformer</i>	17

2.2.8	<i>Pose Estimation</i>	17
2.2.9	MeTRAbs	18
2.2.10	DeciWatch	19
2.2.11	Trigonometri	21
2.2.12	Matriks Rotasi	22
2.2.13	<i>Inverse Kinematics Transform</i>	23
2.2.14	<i>Mean Per Joint Position Error</i> (MPJPE)	24
BAB III. METODOLOGI PENENILITIAN		26
3.1	Waktu dan Tempat Penelitian	26
3.2	Diagram Alir Penelitian	26
3.3	Metode Pengumpulan Data	27
3.3.1	<i>Dataset AIST++</i>	27
3.3.2	Data Video Tari Tradisional Indonesia	29
3.4	Teknik Pengolahan Data	30
3.4.1	Pengolahan <i>Dataset</i>	30
3.4.2	<i>Training</i> dan <i>Fine-tuning</i> Model	32
3.4.3	Pengujian dan Evaluasi Model	32
3.4.4	Implementasi pada Robot Tari	32
3.5	Uji Coba Sistem	33
BAB IV. ANALISIS DAN PERANCANGAN SISTEM		34
4.1	Deskripsi Sistem	34
4.2	Analisa Kebutuhan Sistem	34
4.2.1	Kebutuhan Perangkat Keras	35
4.2.2	Kebutuhan Perangkat Lunak	35
4.2.3	Analisa Kebutuhan Fungsionalitas	36
4.3	Perancangan Sistem	37
4.3.1	Arsitektur Sistem	37

4.3.2	Alur Kerja Sistem	39
4.4	Perancangan Antarmuka Pengguna	42
4.5	Perancangan Modifikasi Hardware	43
BAB V. IMPLEMENTASI DAN PENGUJIAN		45
5.1	Implementasi Sistem	45
5.1.1	Data <i>Preprocessing</i>	45
5.1.2	Inisialisasi Dataset dan <i>DataLoader</i>	47
5.1.3	Pembuatan Model	48
5.1.4	Metode Perhitungan MPJPE dan Prosedur Pelatihan Model	52
5.1.5	Penyimpanan Model dan Evaluasi	53
5.1.6	Modifikasi Hardware	54
5.1.7	Implementasi pada Robot	55
5.1.8	Penyimpanan Gerakan dalam Format JSON	63
5.1.9	Eksekusi Gerakan pada Robot	64
5.2	Implementasi Website	66
5.3	Pengujian Fungsional Sistem	70
BAB VI. HASIL DAN PEMBAHASAN		72
6.1	Hasil dan Pembahasan Sistem	72
6.1.1	Hasil dan Pembahasan Implementasi pada Robot	72
6.1.2	<i>Motion Imitation</i> Pada Robot	73
6.1.3	Hasil dan Pembahasan Fungsional Sistem	76
6.1.4	Hasil dan Pembahasan Performa Model	78
BAB VII.KESIMPULAN DAN SARAN		83
7.1	Kesimpulan	83
7.2	Saran	84
DAFTAR PUSTAKA		85
LAMPIRAN		88

DAFTAR GAMBAR

Gambar 2.1	Robotis-OP3	11
Gambar 2.2	Sistem Robotis-OP3	11
Gambar 2.3	Dynamixel X-Series	12
Gambar 2.4	Perbedaan ML dan DL	15
Gambar 2.5	Ilustrasi Jaringan Saraf Tiruan pada Deep Learning	15
Gambar 2.6	Arsitektur AlexNet	16
Gambar 2.7	Hasil ILSVRC-2010 AlexNet	16
Gambar 2.8	Arsitektur jaringan MeTRAbs	19
Gambar 2.9	Keunggulan MeTRAbs	19
Gambar 2.10	Arsitektur DeciWatch	21
Gambar 3.1	Diagram Alir Penelitian	26
Gambar 3.2	<i>Dataset AIST++</i>	29
Gambar 3.3	Tari Denok Semarangan	29
Gambar 3.4	Diagram Alur Pengolahan Data	30
Gambar 4.1	Diagram Arsitektur Sistem	37
Gambar 4.2	Flowchart Perancangan Sistem	39
Gambar 4.3	Perancangan Antarmuka Sistem	42
Gambar 5.1	Tampilan fisik robot ROBOTIS-OP3 setelah modifikasi	54
Gambar 5.2	Visualisasi sistem koordinat robot	55
Gambar 5.3	Tampilan awal website sebelum interaksi pengguna	66
Gambar 5.4	Proses upload dan pemilihan video	67
Gambar 5.5	Tampilan proses estimasi pose 3D sedang berjalan	68
Gambar 5.6	Hasil visualisasi pose 3D setelah proses estimasi selesai	68
Gambar 5.7	Tampilan saat proses penyimpanan data sudut sendi	69
Gambar 5.8	Tampilan saat proses playback gerakan robot berjalan	69

Gambar 6.1	Grafik perubahan sudut sendi robot per frame pada lengan kiri, lengan kanan, dan kepala	72
Gambar 6.2	Visualisasi motion imitation pada gerakan tari tradisional: frame video asli, rangka hasil estimasi 3D, dan gerakan robot	74
Gambar 6.3	Visualisasi motion imitation untuk cakupan gerakan: frame video asli, rangka hasil estimasi 3D, dan gerakan robot	75
Gambar 6.4	Perbandingan nilai loss dan learning rate terhadap epoch (1–20)	79
Gambar 6.5	Perbandingan MPJPE antara input dan output pada sesi pelatihan pertama	79
Gambar 6.6	Perbandingan nilai loss dan learning rate terhadap epoch (21–45)	80
Gambar 6.7	Perbandingan MPJPE antara input dan output pada sesi pelatihan kedua	80
Gambar 6.8	Perbandingan koordinat X, Y, dan Z hasil estimasi pose 3D antara Metrabs (<i>Raw</i>) dan DeciWatch (<i>Filtered</i>)	81

DAFTAR TABEL

Tabel 2.1	Tabel Studi Literatur	8
Tabel 4.1	Spesifikasi Perangkat Keras Laptop	35
Tabel 4.2	Spesifikasi Perangkat Keras Robot	35
Tabel 4.3	Spesifikasi Perangkat Lunak	35
Tabel 4.4	Kebutuhan Fungsional Sistem	36
Tabel 5.1	Ringkasan Konfigurasi Pelatihan Model DeciWatch	51
Tabel 5.2	Rencana Pengujian Fungsional Sistem	70
Tabel 6.1	Hasil Pengujian Fungsional Sistem	76
Tabel 6.2	Ringkasan Hasil Pengujian Training Model DeciWatch	81
Tabel 6.3	Perbandingan Performa Model Custom dan Model Pre-trained	82

DAFTAR LAMPIRAN

Lampiran 1 Kuesioner <i>black-box testing</i>	88
Lampiran 2 Dokumentasi saat pengambilan data oleh anggota tim robotik	89

BAB I. PENDAHULUAN

1.1 Latar Belakang

Kemajuan teknologi semakin berkembang pesat dalam kehidupan manusia saat ini, terutama dengan adanya internet yang semakin luas dan mudah diakses. Hal ini membuka peluang besar bagi inovasi di berbagai bidang, termasuk pengembangan dalam bidang robotika. Salah satu cabang robotika yang menarik perhatian adalah robot *humanoid* yaitu robot yang memiliki struktur menyerupai manusia, seperti kepala, tangan, badan, dan kaki, serta mampu meniru sejumlah pergerakan manusia (Jalil, 2016).

Teknologi robotika di Indonesia sudah banyak diaplikasikan sebagai teknologi yang mempermudah pekerjaan manusia terutama dalam bidang industri. Tidak hanya itu, kompetisi robot di Indonesia juga sering diselenggarakan untuk mendorong inovasi dan kreativitas mahasiswa serta pelajar dalam bidang robotika. Salah satu acara yang paling dikenal adalah Kontes Robot Indonesia (KRI) yang diselenggarakan oleh Balai Pengembangan Talenta Indonesia (BPTI). Kontes ini rutin diadakan setiap tahun untuk mengukur kemampuan mahasiswa dalam mendesain dan mengendalikan robot sesuai tema dan tantangan tertentu. KRI memiliki dua divisi yang melibatkan robot *humanoid* yakni untuk melakukan misi bermain sepak bola pada Kontes Robot Sepak Bola Indonesia Humanoid (KRSBI-H) dan melakukan seni tari pada Kontes Robot Seni Tari Indonesia (KRSTI).

Robot *humanoid* yang digunakan pada KRSTI merupakan robot yang dapat melakukan tarian tradisional Indonesia dengan tema yang telah ditentukan. Dalam mencapai misinya, robot tentu harus menghayati dan bergerak sesuai dengan tarian dan musik aslinya (Imania et al., 2023). Pergerakan robot *humanoid* dalam melakukan tarian masih dilakukan secara manual dengan menggerakkan setiap sendi robot untuk setiap pergerakan tari. Hal tersebut menyebabkan sulitnya membuat gerakan untuk menerapkan gerakan yang diinginkan pada robot karena memakan banyaknya repetisi pada pergerakan serta pemborosan waktu.

Dalam ajang KRSTI, setiap robot dalam kompetisi ini harus menampilkan tarian terbaik sesuai dengan tema tarian yang telah ditentukan. Salah satu kriteria

penilaian utama adalah kemampuan robot untuk menari dan bergerak selaras dengan irama musik pengiring tema tarian tersebut. Selain itu, dua robot yang berpartisipasi harus dapat menari secara sinkron. Dalam ajang tersebut, Politeknik Negeri Malang telah beberapa kali berhasil lolos ke babak nasional. Namun, dalam pembuatan gerakan pada robot *humanoid*, khususnya untuk tarian, prosesnya masih dilakukan secara manual. Hal ini menyebabkan kesulitan dalam menciptakan gerakan yang sesuai dengan keinginan, karena membutuhkan banyak repetisi pergerakan serta memakan waktu yang cukup lama, sehingga kurang efisien.

Untuk mengatasi permasalahan tersebut, salah satu solusi yang dapat diterapkan adalah teknologi pose estimation. Teknologi ini memungkinkan robot *humanoid* untuk melakukan gerakan tarian secara otomatis melalui analisis visual dari gerakan manusia yang direkam menggunakan kamera. DeciWatch adalah salah satu metode canggih dalam estimasi pose manusia 2D/3D yang memanfaatkan pendekatan inovatif berbasis *transformers*. Dengan pendekatan *sample-denoise-recover*, DeciWatch secara efisien hanya menggunakan 10% frame video untuk estimasi, mengurangi *noise* pada data dengan DenoiseNet, dan merekonstruksi pose yang tidak diamati menggunakan RecoverNet yang memanfaatkan korelasi spasial dan temporal. Metode ini mampu menghasilkan estimasi pose manusia yang lebih akurat, presisi, dan halus, menjadikannya solusi ideal untuk menghasilkan gerakan robot *humanoid* yang lebih natural dan efisien.

Dengan pose estimation, titik-titik kunci pada tubuh manusia, seperti sendi dan anggota tubuh, dapat diidentifikasi dan diterjemahkan ke dalam data koordinat yang digunakan untuk memprogram gerakan robot. Proses ini tidak hanya mengurangi kebutuhan repetisi manual, tetapi juga meningkatkan efisiensi waktu dalam pembuatan gerakan. Selain itu, pose estimation memungkinkan robot untuk menghasilkan gerakan yang lebih halus dan natural, sehingga performa robot dalam menari dapat ditingkatkan secara signifikan. Dengan penerapan teknologi ini, diharapkan robot *humanoid* dapat lebih mudah memenuhi kriteria penilaian pada ajang seperti KRSTI.

1.2 Rumusan Masalah

Berdasarkan uraian pada latar belakang diatas, permasalahan utama yang akan diangkat dalam penelitian ini adalah sebagai berikut:

- Bagaimana memanfaatkan teknologi pose estimation untuk menganalisis gerakan manusia dan menghasilkan data pose dalam format 3D?
- Bagaimana mengaplikasikan data pose 3D yang dihasilkan oleh pose estimation untuk memprogram gerakan robot *humanoid* tari agar dapat meniru gerakan manusia?
- Bagaimana merancang aplikasi yang dapat mengintegrasikan proses perekaman gerakan dan implementasi gerakan pada robot *humanoid* tari dalam satu sistem berbasis website?
- Bagaimana mengevaluasi hasil *pose estimation* dan implementasi gerakan pada robot *humanoid* tari yang telah dibangun?

1.3 Batasan Masalah

Dari rumusan masalah diatas dapat dibuat suatu batasan-batasan masalah sebagai berikut:

- Robot diam di tempat tanpa pergerakan translasi atau rotasi pada bagian kaki.
- Sistem memanfaatkan *Robot Operating System (ROS)* dengan bahasa pemrograman Python sebagai media komunikasi dan pengendalian sebagian pergerakan robot.
- Pergerakan yang direplikasi hanya terbatas pada badan bagian atas, yaitu pergerakan lengan dan kepala, tanpa melibatkan sinkronisasi dengan musik.
- Aplikasi pendukung dibangun berbasis *website*, ditujukan untuk anggota tim robotik.
- *Dataset training* yang digunakan merupakan dataset *pose estimation* yang tersedia secara publik di internet, bukan data yang diambil secara mandiri.

1.4 Tujuan

Adapun tujuan dari dilakukannya penelitian ini adalah sebagai berikut:

- Mengembangkan teknologi *pose estimation* untuk menganalisis gerakan manusia dan menghasilkan data pose dalam format 3D secara akurat.

- Mengaplikasikan data pose 3D yang dihasilkan untuk memprogram gerakan robot *humanoid* tari sehingga dapat meniru gerakan manusia dengan presisi.
- Merancang dan membangun aplikasi berbasis website yang mengintegrasikan proses perekaman gerakan dan implementasi gerakan pada robot *humanoid* tari.
- Untuk mengetahui evaluasi hasil *pose estimation* dan implementasi gerakan pada robot *humanoid* tari yang telah dibangun.

1.5 Manfaat

Manfaat yang diharapkan dari penelitian ini adalah sebagai berikut:

- Memberikan kontribusi dalam pengembangan teknologi robotika di Indonesia, khususnya dalam bidang robot *humanoid* tari.
- Mempermudah proses pembuatan gerakan tarian robot *humanoid*, sehingga lebih efisien dalam waktu dan tenaga.
- Meningkatkan daya saing Politeknik Negeri Malang dalam ajang kompetisi robotika, khususnya dalam bidang robot *humanoid* tari.
- Menjadi referensi bagi peneliti selanjutnya dalam pengembangan teknologi *pose estimation* dan implementasi gerakan pada robot *humanoid* tari.

BAB II. LANDASAN TEORI

2.1 Studi Literatur

Penelitian ini didasarkan pada studi literatur yang relevan terhadap topik skripsi. Pada penelitian sebelumnya yang berjudul "*Pemanfaatan Estimasi Pose Gerak pada Penari Trance dalam Ritual Adat Ngalaksa*" bertujuan untuk mendeteksi pose gerak penari yang mengalami trance dalam ritual adat Ngalaksa di Desa Wisata Rancakalong, Sumedang, Jawa Barat. Penelitian ini menggunakan metode visi komputer berbasis GluonCV untuk analisis data pose gerak para penari. *Dataset* gerak tari dikumpulkan melalui observasi lapangan, wawancara dengan informan seperti Saehu dan peserta ritual, serta dokumentasi visual. Hasil penelitian menunjukkan bahwa model estimasi pose berhasil mendeteksi perbedaan gerak antara penari yang mengalami trance dan yang tidak, meskipun akurasi untuk penari trance cenderung lebih rendah dibandingkan penari lainnya. Contohnya, akurasi untuk penari trance perempuan adalah 0.537, lebih rendah dibandingkan dengan akurasi penari biasa yang mencapai 0.802 hingga 0.993. Temuan ini menyoroti perlunya pengembangan lebih lanjut untuk meningkatkan akurasi model dan mendukung pelestarian seni budaya nusantara dengan teknologi kecerdasan buatan.

Salah satu metode awal pada bidang estimasi pose adalah VNect yang merupakan hasil dari penelitian "*VNect: Real-time 3D Human Pose Estimation with a Single RGB Camera*" yang diperkenalkan pada tahun 2017. Metode ini mampu mengestimasi pose 3D tubuh manusia secara global dan konsisten secara temporal dari input kamera RGB tunggal secara *real-time*, metode ini berhasil menghasilkan model dengan akurasi *mean per joint position error* (MPJPE) 80.5mm. Dengan menggunakan jaringan konvolusional (CNN), VNect meregresikan posisi *joint* 2D dan 3D tanpa memerlukan *bounding box* yang ketat. Hasil regresi dioptimalkan melalui *kinematic skeleton fitting* untuk menghasilkan pose yang stabil secara temporal. Namun, metode ini rentan terhadap noise pada deteksi awal *joint* 2D yang dapat memengaruhi estimasi 3D (Mehta et al., 2017).

Metode lain yang lebih terkini adalah "*Deep High-Resolution Representation*

Learning for Human Pose Estimation” yang dirancang untuk mempertahankan resolusi tinggi sepanjang proses estimasi pose manusia. Dengan menggunakan *subnet* resolusi tinggi yang terhubung secara paralel dengan *subnet* resolusi rendah, HRNet memungkinkan pertukaran informasi multi-skala yang berulang. Hasilnya, HRNet memberikan akurasi tinggi pada beberapa *dataset*. Pada pengujian menggunakan metrik PCKh@0.5, HRNet mencapai nilai 90.3, yang menunjukkan performa luar biasa dalam estimasi pose manusia (Sun et al., 2019).

Di sisi lain, tak hanya untuk *text generation*, penggunaan *transformer* pada visi komputer juga sudah berkembang pesat. Dalam penelitian berjudul *”DeciWatch: A Simple Baseline for 10× Efficient 2D and 3D Pose Estimation”*, diperkenalkan kerangka kerja berbasis *transformer* yang inovatif bernama DeciWatch. Model ini dirancang untuk meningkatkan efisiensi estimasi pose manusia 2D/3D hingga 10 kali lipat dibandingkan metode tradisional tanpa mengorbankan akurasi. Hasil eksperimen menunjukkan bahwa DeciWatch menghasilkan estimasi yang lebih halus dan akurat dengan biaya komputasi yang jauh lebih rendah. Model ini divalidasi pada beberapa dataset benchmark, seperti Human3.6M dan AIST++, dengan pencapaian performa tinggi dalam akurasi serta efisiensi komputasi. Sebagai contoh, DeciWatch berhasil mencapai MPJPE sebesar 52,8 mm pada Human3.6M dengan hanya menggunakan 10% *frame* video, mengungguli model sebelumnya dan tetap mempertahankan biaya komputasi yang minimal (Zeng et al., 2022).

Pada topik yang sama dengan penelitian ini yaitu imitasi gerakan manusia pada robot humanoid terdapat penelitian berjudul *”Motion Imitation of a Humanoid Robot via Pose Estimation”*, memperkenalkan metode imitasi gerakan robot humanoid menggunakan estimasi pose 3D berbasis RGB-D. Penelitian ini memanfaatkan *sparse tensor*, *mask net*, dan *sparsity pruning* untuk mengurangi konsumsi memori dan waktu komputasi. Dengan menerapkan *motion retargeting* berbasis transformasi koordinat, robot NAO mampu meniru gerakan manusia secara akurat pada berbagai tugas, seperti meninju dan melambai (Meng et al., 2023). Eksperimen yang dilakukan menunjukkan bahwa metode ini menghasilkan imitasi gerakan yang stabil dan berkelanjutan. Untuk mengurangi getaran pada pergerakan

robot, digunakan *Kalman Filter* pada sudut sendi yang dihasilkan dari proses *retargeting*. Hasil uji coba pada *Berkeley MHAD dataset* menunjukkan tingkat kesalahan prediksi *Mean Per Joint Position Error (MPJPE)* sebesar 27.72 mm untuk semua aksi, dan 20.04 mm pada tujuh aksi pertama yang lebih terlihat penuh tanpa occlusion. Hal ini membuktikan bahwa metode yang diajukan dapat menghasilkan performa estimasi pose dan imitasi gerak yang akurat meskipun menggunakan perangkat RGB-D yang lebih ekonomis dibandingkan sistem *motion capture* konvensional. Dalam penelitian ini, robot NAO digunakan sebagai platform robotik, sedangkan dalam penelitian yang dilakukan oleh penulis, metode serupa diadaptasi dan diterapkan pada Robotis OP3, dengan menyesuaikan struktur mekanik dan aktuator yang berbeda namun dengan prinsip dasar pemetaan gerakan yang serupa.

Penelitian lain berjudul "*Robust Regression-Based Motion Perception for Online Imitation on Humanoid Robot*" mengusulkan metode *motion retargeting* berbasis *robust regression* untuk memetakan gerakan manusia ke robot humanoid secara langsung. Proses pemetaan sudut dilakukan dengan perhitungan trigonometri menggunakan transformasi vektor spasial, melibatkan perhitungan *shoulder pitch*, *shoulder roll*, *elbow yaw*, dan *elbow roll*. Pendekatan ini memproyeksikan posisi sendi manusia ke sistem koordinat robot untuk mengontrol pergerakan lengan secara *real-time*. Metode ini menjadi referensi dalam penelitian ini untuk diterapkan pada Robotis OP3, dengan penyesuaian pada struktur dan aktuator robot (Zhu, Qunfei, Wan, & Xia, 2017).

Selain itu, penelitian "*Mixed Reality Interface for Whole-Body Balancing and Manipulation of Humanoid Robot*" memperkenalkan antarmuka *mixed reality* (MR) untuk teleoperasi robot ROBOTIS-OP3. Dengan memanfaatkan teknologi *motion retargeting* berbasis perhitungan skala linier dan *inverse kinematics*, operator dapat mengontrol robot tanpa kontroler tambahan. Sistem ini juga memungkinkan pemantauan stabilitas robot melalui *augmented view* yang menampilkan posisi *center of mass* dan *balanced state basin* (Song et al., 2024).

Tabel 2.1 Tabel Studi Literatur

No	Judul	Metode Penelitian	Hasil Penelitian
1	" <i>Pemanfaatan Estimasi Pose Gerak pada Penari Trance dalam Ritual Adat Ngalaksa</i> " (Listiani & Rohaeni, 2024)	Estimasi pose menggunakan GluonCV, visi komputer, analisis data visual	Model estimasi pose berhasil mendekripsi perbedaan gerak antara penari trance dan tidak trance dengan akurasi 0.537 untuk penari trance perempuan, dan akurasi 0.802–0.993 untuk penari biasa.
2	" <i>Vnect: Real-time 3D Human Pose kinematic Estimation with a skeleton Single RGB Camera</i> " fitting (Mehta et al., 2017)	CNN,	Mengestimasi pose 3D secara global dan temporal dengan input kamera webcam, stabil tetapi rentan terhadap <i>noise</i> pada deteksi awal <i>joint</i> .
3	" <i>Deep High-Resolution Representation Learning for Human Pose Estimation</i> " Subnetworks) (Sun et al., 2019)	HRNet (Parallel Multi-resolution Subnetworks)	Akurasi tinggi pada estimasi pose dengan resolusi tinggi, unggul pada dataset COCO dan MPII dibandingkan metode Resnet.

No	Judul	Metode Penelitian	Hasil Penelitian
4	"DeciWatch: A Simple Baseline for 10× Efficient 2D and 3D Pose Estimation" (Zeng et al., 2022)	A estimasi pose menggunakan pendekatan <i>sample-denoise-recover</i> , <i>transformer</i> , dan <i>DenoiseNet</i> - <i>RecoverNet</i>	Model berhasil meningkatkan efisiensi estimasi pose hingga 10× lebih cepat dengan akurasi MPJPE 52,8 mm pada dataset Human3.6M, serta efisiensi komputasi yang signifikan.
5	"Motion Imitation of a Humanoid Robot via Pose Estimation" (Meng et al., 2023)	Sparse tensor, mask net, sparsity pruning, motion retargeting	Mengurangi konsumsi memori dan meningkatkan akurasi estimasi pose 3D pada robot humanoid NAO.
6	"Robust Regression-Based Motion Perception for Online Imitation on Humanoid Robot" (Zhu et al., 2017)	Robust regression, motion retargeting, trigonometri	Pemetaan sudut sendi manusia ke robot humanoid menggunakan transformasi vektor spasial dan perhitungan trigonometri secara <i>real-time</i> .

No	Judul	Metode Penelitian	Hasil Penelitian
7	"Mixed Reality Interface for Reality Whole-Body Balancing and manipulation of Humanoid Robot"	Mixed (MR), IK	Pengendalian robot humanoid ROBOTIS-OP3 menggunakan <i>Virtual Reality</i> . (Song et al., 2024)

2.2 Dasar Teori

2.2.1 Robot *Humanoid* Robotis-OP3

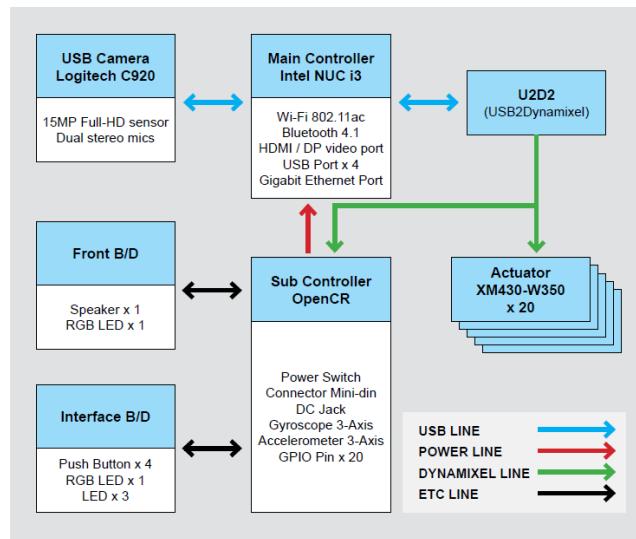
Robotis-OP3 adalah robot *humanoid* yang dikembangkan oleh Robotis, perusahaan yang dikenal dengan produk-produk robotika canggih untuk pendidikan, penelitian, dan otomasi. OP3 dirancang sebagai platform robot *humanoid* dengan kemampuan tinggi, yang sering digunakan dalam penelitian robotika, termasuk bidang pemrosesan visual, estimasi pose, dan kontrol gerakan.

Secara umum, robot ini merupakan robot *humanoid* yang dirancang untuk kompetisi robot sepak bola. Robot ini memiliki tinggi sekitar 51 cm dan berat 3,5 kg serta dilengkapi dengan 20 motor servo Dynamixel, terdiri dari 2 motor di kepala, 3 motor di setiap lengan, dan 6 motor di setiap kaki, yang mendukung gerakan artikulasi yang kompleks. Selain itu, OP3 memiliki kamera terintegrasi di bagian kepala untuk menangkap data visual, serta sensor IMU (*Inertial Measurement Unit*) yang berfungsi untuk menjaga keseimbangan dan membantu navigasi (Damar et al., 2023).



Gambar 2.1 Robotis-OP3

Sumber: (Robotis, 2023d)



Gambar 2.2 Sistem Robotis-OP3

Sumber: (Robotis, 2023d)

2.2.2 Dynamixel

Dynamixel adalah aktuator pintar yang dikembangkan oleh Robotis dan banyak digunakan pada sistem robotika modular, termasuk pada robot *humanoid* seperti ROBOTIS-OP3. Berbeda dengan servo konvensional, Dynamixel memiliki fitur komunikasi dua arah yang memungkinkan pengguna tidak hanya mengendalikan sudut atau kecepatan, tetapi juga membaca data seperti posisi saat ini, beban, suhu, dan status error secara real-time (Robotis, 2023a).

Dynamixel menggunakan protokol komunikasi serial berbasis *half-duplex* dengan dua metode utama, yaitu *TTL* dan *RS-485*. Pada ROBOTIS-OP3, komunikasi

dengan servo dilakukan menggunakan protokol *RS-485*, yang mendukung transmisi data jarak lebih jauh dan lebih stabil dibandingkan *TTL*. Setiap servo dihubungkan secara daisy-chain melalui satu jalur komunikasi, sehingga memudahkan perakitan dan pengendalian banyak motor secara simultan (Robotis, 2023b).

Untuk memudahkan integrasi dengan berbagai bahasa pemrograman, Robotis menyediakan *Dynamixel SDK*, sebuah pustaka perangkat lunak yang mendukung pengendalian servo menggunakan bahasa C, C++, Python, Java, dan MATLAB. SDK ini memungkinkan pengiriman instruksi dan pembacaan data melalui API yang kompatibel dengan berbagai sistem operasi seperti Windows, Linux, dan macOS (Robotis, 2023c).

Pada platform ROBOTIS-OP3, terdapat dua jenis seri Dynamixel yang digunakan, yaitu XM430-W350-R dan 2XL430-W250-T. Dynamixel XM430-W350-R digunakan pada bagian kaki dan lengan robot karena memiliki torsi yang cukup besar serta mendukung berbagai mode kontrol seperti posisi, kecepatan, dan arus. Selain itu, motor ini juga dilengkapi dengan fitur *current-based position control* yang memungkinkan pengendalian lebih presisi dan aman terhadap beban berlebih. Sementara itu, Dynamixel 2XL430-W250-T digunakan pada bagian kepala untuk menggerakkan dua sumbu sekaligus, yaitu *yaw* dan *pitch*, dalam satu modul motor. Modul ini memiliki dua port keluaran motor, sehingga lebih efisien dalam penggunaan ruang dan bobot, serta cocok untuk aplikasi di bagian kepala robot yang memerlukan mekanisme gerak ganda dengan dimensi yang lebih ringkas (Robotis, 2023d).



Gambar 2.3 Dynamixel X-Series
Sumber: (Robotis, 2023a)

2.2.3 *Robot Operating System*

Robot Operating System (ROS) adalah *framework* perangkat lunak untuk robotika yang menyediakan berbagai *library* dan alat untuk memfasilitasi pengembangan aplikasi robot. ROS memungkinkan komunikasi antar modul perangkat keras dan perangkat lunak melalui arsitektur berbasis *publish-subscribe* yang efisien, sehingga memudahkan integrasi berbagai komponen seperti sensor, aktuator, dan algoritma kontrol. Selain itu, ROS juga mendukung simulasi gerakan robot menggunakan platform seperti Gazebo, yang memungkinkan pengujian algoritma secara virtual sebelum diimplementasikan pada perangkat keras sebenarnya (Dhimas, Santoso, & Hartayu, 2024). Pada robot humanoid seperti ROBOTIS-OP3, ROS digunakan untuk berbagai fungsi, termasuk kontrol gerakan, visualisasi data sensor, pengolahan gambar, dan integrasi sistem secara keseluruhan, sehingga menjadi salah satu elemen kunci dalam pengembangan aplikasi robotik.

2.2.4 Visi Komputer

Visi Komputer adalah cabang ilmu komputer yang berfokus pada pengembangan algoritma dan teknik untuk memungkinkan komputer memahami dan mengekstrak informasi bermakna dari data visual seperti gambar, video, atau aliran data dari sensor visual lainnya. Bidang ini bertujuan untuk mensimulasikan kemampuan penglihatan manusia sehingga komputer dapat mengenali, menganalisis, dan membuat keputusan berdasarkan data visual (Marpaung et al., 2022). Teknik dasar dalam visi komputer meliputi deteksi tepi, segmentasi objek, pengenalan fitur, dan estimasi pose. Deteksi tepi digunakan untuk menemukan batasan objek dalam gambar dengan mengidentifikasi perubahan intensitas yang signifikan, seperti yang dilakukan oleh algoritma *Canny Edge Detection*. Segmentasi objek membagi gambar menjadi bagian-bagian yang lebih kecil atau objek tertentu untuk analisis lebih lanjut, menggunakan metode seperti *thresholding* atau *neural networks*. Pengenalan fitur bertujuan untuk mengekstrak elemen-elemen khas seperti sudut, garis, atau pola, yang penting untuk pelacakan atau pengenalan objek. Estimasi pose adalah proses menentukan posisi dan orientasi tubuh manusia atau objek dalam ruang 2D atau 3D, yang biasanya menggunakan

algoritma berbasis jaringan saraf tiruan atau model regresi.

Dalam robotika, visi komputer memainkan peran penting dalam berbagai aplikasi. Salah satunya adalah navigasi robot, di mana data visual digunakan untuk mengenali lingkungan sekitar dan menentukan jalur optimal. Selain itu, deteksi lingkungan memungkinkan robot untuk mengidentifikasi rintangan atau area tertentu, membantu dalam penghindaran tabrakan atau penyelesaian tugas. Visi komputer juga digunakan untuk mengenali gerakan manusia, yang menjadi dasar bagi robot *humanoid* dalam meniru gerakan tersebut. Salah satu penerapannya adalah dalam pemodelan gerakan kompleks, seperti gerakan tari atau aktivitas olahraga, yang membutuhkan analisis koordinasi tubuh secara terus-menerus (Kim et al., 2023).

2.2.5 Deep Learning

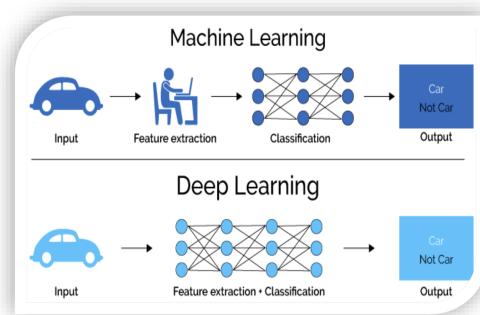
Dalam dunia teknologi saat ini sudah tidak asing lagi dengan *Artificial Intelligence* (AI). Perusahaan besar seperti Google, Microsoft, dan Meta saling berlomba untuk menciptakan *Large Language Model* (LLM) seperti *Gemma*, *Mistral*, dan *Llama*. LLM merupakan salah satu aplikasi utama dari *Deep Learning*, yang memungkinkan pemodelan teks secara kompleks dengan memahami konteks dan hubungan antar kata dalam skala besar.

Deep Learning (DL) atau Pembelajaran Mendalam adalah salah satu cabang dari pembelajaran mesin yang berfokus pada pemodelan abstraksi tingkat tinggi pada data. Hal ini dilakukan dengan menggunakan sekumpulan fungsi transformasi non-linear yang ditata secara berlapis-lapis (*deep neural networks*) dan mendalam (Ahmad, 2017). DL sangat baik diterapkan pada *supervised learning*, *unsupervised learning*, *semi-supervised learning*, maupun *reinforcement learning* untuk berbagai aplikasi seperti pengenalan citra, pengenalan suara, klasifikasi teks, dan lainnya.

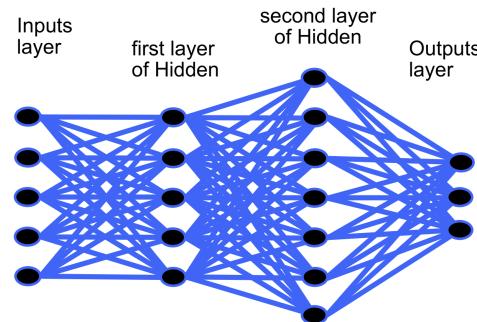
Model dalam DL pada dasarnya dibangun berdasarkan Jaringan Saraf Tiruan (*Neural Network*), yang risetnya telah berlangsung sejak era 1980-an. Namun, DL baru kembali menjadi populer dengan kemajuan teknologi komputer yang semakin cepat, terutama berkat teknologi *Big Data* seperti Hadoop dan Spark berbasis *multi-node cluster*, serta pemrosesan secara paralel berbasis GPU. Jika suatu jaringan

memiliki lebih dari tiga lapisan, maka jaringan tersebut dapat disebut sebagai *Deep Network* (Cholissodin et al., 2020).

DL memanfaatkan algoritma berbasis *backpropagation* untuk mengoptimalkan bobot jaringan. Dalam konteks visi komputer, DL telah digunakan secara luas untuk tugas seperti klasifikasi gambar, deteksi objek, segmentasi, dan estimasi pose manusia. Selain itu, DL juga menjadi tulang punggung bagi pengembangan LLM, yang memungkinkan analisis teks secara kompleks dan menghasilkan teks yang menyerupai bahasa manusia.



Gambar 2.4 Perbedaan ML dan DL
Sumber: (Cholissodin et al., 2020)



Gambar 2.5 Ilustrasi Jaringan Saraf Tiruan pada Deep Learning
Sumber: (Cholissodin et al., 2020)

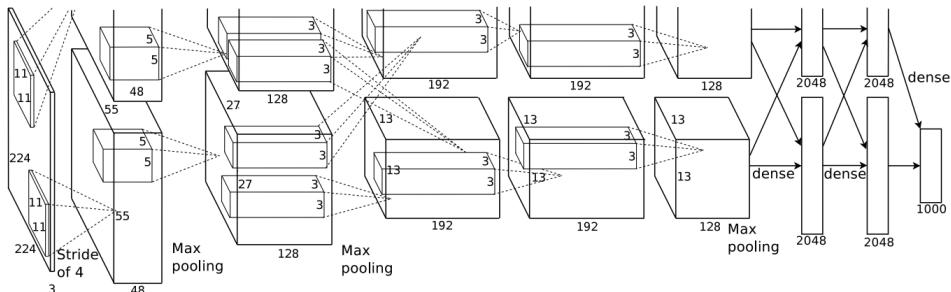
2.2.6 Convolutional Neural Networks (CNN)

Convolutional Neural Networks (CNN) merupakan arsitektur jaringan saraf tiruan yang dirancang khusus untuk memproses data visual. CNN bekerja dengan mengidentifikasi pola atau fitur dalam gambar melalui operasi konvolusi yang diikuti oleh proses *pooling*. Arsitektur ini menjadi dasar bagi berbagai metode modern

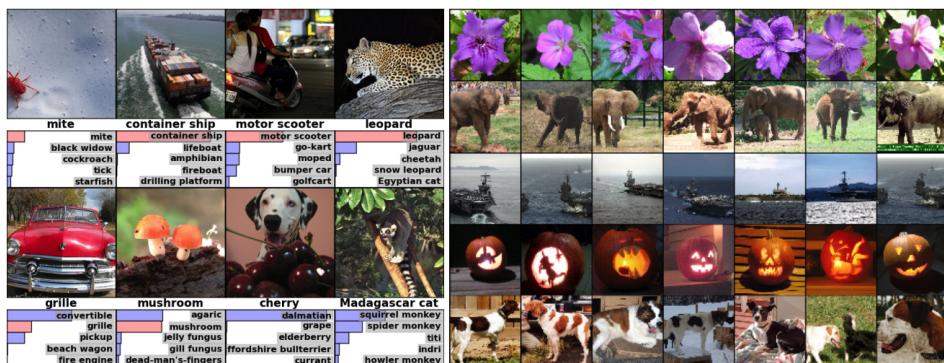
dalam tugas-tugas pengenalan citra, deteksi objek, dan estimasi pose manusia.

Salah satu arsitektur CNN yang populer adalah AlexNet. AlexNet diperkenalkan oleh Krizhevsky et al. pada tahun 2012 dan menjadi tonggak penting dalam perkembangan *deep learning* untuk tugas pengenalan citra. Arsitektur ini terdiri dari delapan lapisan, dengan lima lapisan konvolusional diikuti oleh tiga lapisan *fully connected* (FC). AlexNet menggunakan fungsi aktivasi ReLU untuk mempercepat konvergensi pelatihan, serta teknik *dropout* untuk mencegah *overfitting*. Selain itu, AlexNet memperkenalkan penggunaan GPU dalam pelatihan jaringan, yang secara signifikan mempercepat proses komputasi (Krizhevsky, Sutskever, & Hinton, 2012).

Keberhasilan AlexNet dalam memenangkan kompetisi *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC) pada tahun 2012 dengan peningkatan akurasi yang signifikan dibandingkan metode sebelumnya menjadikannya model yang mendasari pengembangan arsitektur CNN modern.



Gambar 2.6 Arsitektur AlexNet
Sumber: (Krizhevsky et al., 2012)



Gambar 2.7 Hasil ILSVRC-2010 AlexNet
Sumber: (Krizhevsky et al., 2012)

2.2.7 Transformer

Transformer adalah arsitektur berbasis *attention* yang pertama kali diperkenalkan oleh Vaswani et al. dalam makalah "Attention Is All You Need". Transformers dirancang untuk menggantikan arsitektur tradisional berbasis *recurrent neural networks (RNN)* dan *convolutional neural networks (CNN)* dalam berbagai tugas, khususnya dalam pemrosesan bahasa alami (*natural language processing*, NLP). Keunggulan utama Transformers terletak pada mekanisme *self-attention*, yang memungkinkan model untuk mempelajari hubungan antar elemen dalam data tanpa memerlukan pemrosesan berurutan, seperti pada RNN.

Dalam mekanisme *self-attention*, setiap elemen input dibandingkan dengan elemen lainnya untuk menentukan relevansi antar elemen. Mekanisme ini direpresentasikan dalam bentuk *query*, *key*, dan *value*, di mana relevansi dihitung sebagai nilai dot-product antara *query* dan *key*, yang kemudian digunakan untuk memberikan bobot pada elemen *value* (Vaswani et al., 2017). Pendekatan ini memungkinkan model untuk memahami hubungan jangka panjang (*long-range dependencies*) dalam data, sehingga cocok untuk data berurutan seperti teks atau video.

2.2.8 Pose Estimation

Pose estimation adalah proses untuk menentukan lokasi spasial (*keypoints*) tubuh manusia atau objek dalam representasi 2D atau 3D. Proses ini bertujuan untuk mendeteksi titik-titik penting, seperti kepala, siku, lutut, dan pergelangan tangan, yang membentuk kerangka tubuh manusia atau objek. Pada pendekatan modern, jaringan saraf konvolusional (*Convolutional Neural Network*, CNN) digunakan untuk menghasilkan peta panas (*heatmap*) 2D dari input berupa gambar atau video. *Heatmap* ini merepresentasikan probabilitas keberadaan *keypoints* pada setiap piksel gambar (Mehta et al., 2017). Untuk menghasilkan pose dalam ruang 3D, peta panas 2D digabungkan dengan informasi tambahan, seperti data kedalaman (*depth*) dari sensor atau rekonstruksi spasial menggunakan teknik regresi koordinat 3D. Pendekatan ini memungkinkan prediksi posisi *keypoints* secara akurat meskipun pada pose yang kompleks atau dalam lingkungan dengan sudut pandang kamera

yang tidak ideal.

Saat ini, *pose estimation* telah menjadi salah satu bidang utama dalam visi komputer, dengan berbagai aplikasi mulai dari animasi karakter, pengenalan gerakan manusia, hingga kontrol robot humanoid. Metode berbasis CNN, seperti OpenPose dan HRNet, telah menjadi standar untuk estimasi pose 2D dan 3D dengan akurasi tinggi. Selain itu, arsitektur modern seperti *Transformers* telah diperkenalkan dalam *pose estimation*, misalnya pada model VitPose. Pendekatan berbasis *Transformers* ini menggunakan mekanisme *self-attention* untuk menangkap hubungan spasial dan temporal antara *keypoints*, sehingga menghasilkan prediksi pose 3D yang lebih stabil dan akurat, terutama untuk data video (Xu et al., 2022). Keunggulan *Transformers* terletak pada kemampuannya untuk mempelajari konteks yang lebih luas dari data visual, membuatnya ideal untuk tugas yang memerlukan pemahaman gerakan berurutan, seperti dalam penelitian ini yang berfokus pada gerakan tari tradisional Indonesia.

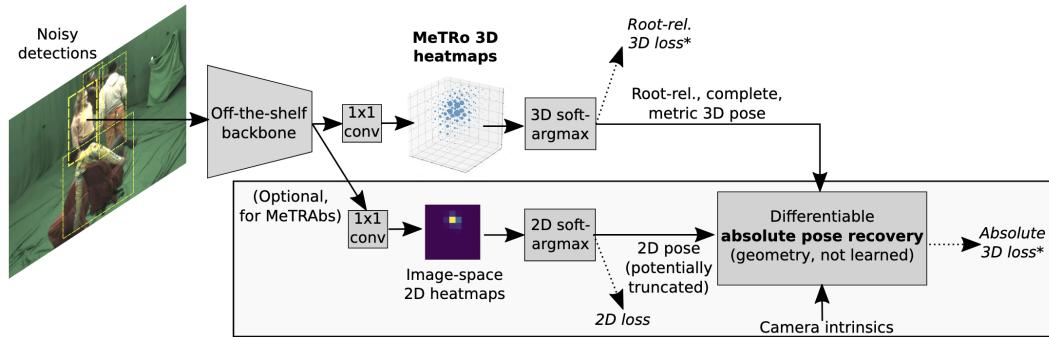
2.2.9 MeTRAbs

MeTRAbs (*Metric-Scale Truncation-Robust Heatmaps for Absolute 3D Human Pose Estimation*) adalah metode yang digunakan untuk memperkirakan pose 3D manusia secara langsung dari gambar RGB, bahkan ketika sebagian tubuh tidak terlihat sepenuhnya dalam gambar. Berbeda dengan metode sebelumnya yang hanya bisa memperkirakan posisi relatif antar sendi tubuh, MeTRAbs mampu memprediksi posisi setiap sendi dalam satuan meter bukan hanya piksel pada gambar.

Metode ini menggunakan representasi *heatmap* 3D, yaitu peta probabilitas untuk setiap titik sendi, yang dibuat dalam ruang tiga dimensi. Semua sumbu dari *heatmap* ini menggunakan skala metrik (meter), bukan piksel, sehingga hasilnya lebih realistik. Jaringan yang digunakan dalam MeTRAbs adalah jaringan konvolusional yang umum digunakan pada visi komputer, seperti ResNet-50, tanpa bagian tambahan yang rumit.

Selain memprediksi posisi 3D relatif antar sendi, MeTRAbs juga dapat memperkirakan posisi absolut tubuh manusia dalam ruang kamera. Ini dilakukan dengan cara menggabungkan prediksi posisi 3D dengan prediksi posisi 2D pada

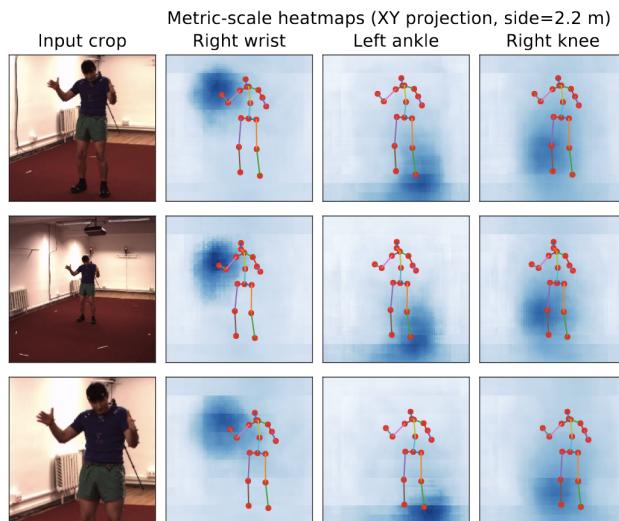
gambar, lalu menghitung posisi tubuh secara keseluruhan menggunakan rumus dari model kamera. Proses ini dilakukan secara otomatis dalam jaringan, sehingga pelatihan bisa dilakukan secara menyeluruh dan efisien.



Gambar 2.8 Arsitektur jaringan MeTRAbs

Sumber: (Sárándi et al., 2021)

Keunggulan MeTRAbs adalah kemampuannya untuk tetap memberikan hasil estimasi yang baik meskipun gambar hanya menampilkan sebagian tubuh saja seperti pada Gambar 2.8. Selain itu, metode ini tidak memerlukan informasi tambahan seperti panjang tulang atau jarak ke kamera saat pengujian.



Gambar 2.9 Keunggulan MeTRAbs

Sumber: (Sárándi et al., 2021)

2.2.10 DeciWatch

DeciWatch adalah sebuah kerangka kerja efisien yang dikembangkan untuk meningkatkan estimasi pose manusia 2D/3D pada video dengan mengurangi

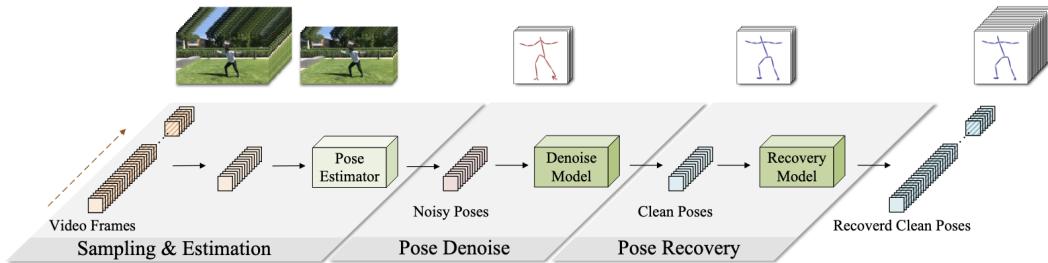
kebutuhan komputasi hingga 10 kali lipat tanpa mengorbankan akurasi (Zeng et al., 2022). Alih-alih memproses setiap frame dalam video, DeciWatch hanya mengambil sebagian kecil frame (sekitar 10%) melalui *uniform sampling*.

Arsitektur DeciWatch terdiri dari tiga tahap utama, yaitu *sample-denoise-recover*. Pada tahap pertama, beberapa frame video dipilih secara merata. Pose pada frame-frame ini kemudian diperkirakan menggunakan pose estimator yang ada. Namun karena prediksi ini biasanya mengandung *noise*, DeciWatch menggunakan DenoiseNet, sebuah jaringan berbasis *transformer*, untuk membersihkan hasil estimasi dari noise.

Setelah itu, tahap ketiga dilakukan oleh RecoverNet, jaringan *transformer* lain yang bertugas untuk merekonstruksi pose untuk frame-frame yang tidak diamati. RecoverNet memanfaatkan kontinuitas gerakan manusia dalam video untuk memperkirakan pose pada frame yang tidak diproses langsung. Pendekatan ini terbukti efektif dalam menghasilkan estimasi pose yang lebih halus dan konsisten, terutama pada gerakan yang cepat atau kompleks.

Keunggulan utama DeciWatch adalah efisiensinya. Karena hanya sebagian kecil frame yang dianalisis secara penuh, proses inferensi menjadi jauh lebih ringan, dengan konsumsi memori dan waktu komputasi yang lebih rendah. Selain itu, pendekatan ini juga memperhalus hasil estimasi pose karena mengurangi jitter atau getaran antar frame, menjadikannya sangat cocok untuk aplikasi video seperti pelacakan gerak, interaksi manusia-robot, dan pemodelan tari.

Eksperimen yang dilakukan pada berbagai *dataset* termasuk AIST++ menunjukkan bahwa DeciWatch tidak hanya unggul dalam efisiensi, tetapi juga mampu meningkatkan akurasi secara signifikan dibandingkan metode estimasi frame-by-frame. Kemampuannya dalam menangani gerakan cepat dan data dengan noise menjadikannya sangat relevan dalam konteks penelitian ini.



Gambar 2.10 Arsitektur DeciWatch
Sumber: (Zeng et al., 2022)

2.2.11 Trigonometri

Trigonometri adalah cabang matematika yang mempelajari hubungan antara sudut dan panjang sisi dalam segitiga (Maharaj, 2008). Dalam konteks robotika, khususnya pada perhitungan *inverse kinematics*, trigonometri digunakan untuk menentukan sudut artikulasi berdasarkan posisi kartesian dari titik-titik tertentu pada tubuh robot atau manusia. Konsep dasar trigonometri berpusat pada perbandingan sisi dalam segitiga siku-siku, seperti berikut:

$$\sin \theta = \frac{\text{depan}}{\text{miring}}, \quad \cos \theta = \frac{\text{samping}}{\text{miring}}, \quad \tan \theta = \frac{\text{depan}}{\text{samping}} \quad (2.1)$$

Pada aplikasi perhitungan sudut dari vektor posisi, fungsi *arctangent* sering digunakan untuk mengubah rasio sisi menjadi sudut. Fungsi ini merupakan kebalikan dari *tangent*, yang mengembalikan nilai sudut θ dari rasio $\frac{y}{x}$. Secara umum, perhitungan sudut menggunakan *arctangent* dapat dituliskan sebagai:

$$\theta = \arctan \left(\frac{y}{x} \right) \quad (2.2)$$

Namun, penggunaan fungsi arctan biasa memiliki keterbatasan karena tidak dapat membedakan kuadran dengan benar, sehingga rawan menghasilkan sudut yang salah jika tanda dari x atau y berbeda. Untuk mengatasi masalah ini, digunakan fungsi arctan 2, yaitu fungsi khusus yang mempertimbangkan tanda dari x dan y secara langsung, sehingga dapat menentukan sudut dengan tepat pada semua kuadran. Notasi umumnya adalah:

$$\theta = \arctan 2(y, x) \quad (2.3)$$

Fungsi arctan 2 menghasilkan sudut dalam satuan radian dengan rentang $-\pi$ hingga π , yang sesuai dengan rotasi penuh pada bidang kartesian. Hal ini sangat penting dalam aplikasi robotika untuk memastikan orientasi sudut sesuai dengan posisi aktual vektor di ruang dua dimensi maupun tiga dimensi. Penggunaan trigonometri seperti ini menjadi komponen utama dalam proses *motion retargeting* pada sistem robot humanoid, karena memungkinkan konversi dari data posisi menjadi sudut artikulasi yang dapat dijalankan oleh robot.

2.2.12 Matriks Rotasi

Matriks rotasi adalah alat matematis yang digunakan untuk memutar vektor dalam ruang dua atau tiga dimensi (Palazzolo, 1976). Dalam bidang robotika, matriks rotasi digunakan untuk mentransformasikan posisi dan orientasi dari satu sistem koordinat ke sistem lainnya, seperti dalam proses *motion retargeting* atau *inverse kinematics*. Rotasi dalam ruang tiga dimensi dapat dinyatakan dengan matriks 3×3 untuk masing-masing sumbu utama:

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta \\ 0 & \sin \theta & \cos \theta \end{bmatrix} \quad (2.4)$$

$$R_y(\theta) = \begin{bmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{bmatrix} \quad (2.5)$$

$$R_z(\theta) = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2.6)$$

Matriks ini sering digunakan secara berurutan untuk membentuk rotasi gabungan di ruang tiga dimensi sesuai urutan sumbu yang diinginkan. Dalam aplikasi robot humanoid, rotasi ini digunakan untuk mengubah orientasi lengan, kepala, atau bagian tubuh lain sesuai dengan data gerakan manusia.

2.2.13 Inverse Kinematics Transform

Inverse Kinematics Transform (IKT) adalah metode untuk menghitung sudut-sudut sendi robot berdasarkan posisi titik-titik tubuh manusia, seperti bahu, siku, dan pergelangan tangan, dengan memanfaatkan prinsip geometri dan trigonometri. Pada penelitian "*Robust Regression-Based Motion Perception for Online Imitation on Humanoid Robot*" (Zhu et al., 2017), IKT digunakan untuk mengontrol pergerakan lengan robot humanoid secara langsung dari data pose manusia.

Struktur lengan dianggap sebagai manipulator dua sendi (*two-link manipulator*) dengan empat derajat kebebasan (DOF). Untuk menyederhanakan perhitungan dan mengurangi kompleksitas komputasi, digunakan metode solusi geometris yang bersifat hierarkis dan intuitif. Pendekatan ini lebih efisien dibandingkan metode aljabar karena menghindari solusi yang tidak unik serta mempercepat proses komputasi.

Sebagai contoh, perhitungan sudut pada lengan kiri dilakukan secara bertahap. Pertama, sudut *shoulder pitch* (θ_{Pitch}) dihitung dari perubahan posisi pada sumbu X dan Z:

$$\theta_{\text{Pitch}} = \arctan 2(z_{\text{elbow}} - z_{\text{shoulder}}, x_{\text{elbow}} - x_{\text{shoulder}}) \quad (2.7)$$

Kemudian, sudut *shoulder roll* (θ_{Roll}) dihitung dari perubahan posisi pada sumbu Y terhadap proyeksi vektor lengan atas pada bidang XZ:

$$\theta_{\text{Roll}} = \arctan 2(y_{\text{elbow}} - y_{\text{shoulder}}, \sqrt{(x_{\text{elbow}} - x_{\text{shoulder}})^2 + (z_{\text{elbow}} - z_{\text{shoulder}})^2}) \quad (2.8)$$

Setelah mendapatkan sudut bahu, vektor lengan bawah diproyeksikan ke sistem koordinat lokal dengan rotasi menggunakan matriks rotasi. Dari situ, sudut *elbow yaw* dan *elbow roll* dihitung dengan prinsip trigonometri serupa. Proses ini memastikan konversi posisi Cartesian menjadi sudut artikulasi dilakukan secara efisien dan dapat berjalan secara *real-time*.

Metode IKT ini sangat sesuai untuk aplikasi *motion retargeting*, karena

memungkinkan robot humanoid untuk meniru gerakan manusia secara langsung tanpa proses optimisasi yang berat secara komputasi.

2.2.14 Mean Per Joint Position Error (MPJPE)

Mean Per Joint Position Error (MPJPE) adalah metrik evaluasi yang umum digunakan dalam estimasi pose manusia 3D. MPJPE mengukur rata-rata jarak Euclidean antara posisi hasil prediksi dan posisi *ground-truth* dari setiap *joint* tubuh manusia dalam ruang 3D. Dengan membandingkan posisi yang diprediksi dengan posisi sebenarnya, MPJPE memberikan gambaran seberapa akurat model memetakan pose manusia (Wang et al., 2021).

Secara matematis, MPJPE dirumuskan sebagai berikut:

$$E_{\text{MPJPE}}(f, \mathcal{S}) = \frac{1}{N_S} \sum_{i=1}^{N_S} \|P_{f,\mathcal{S}}^{(i)} - P_{gt,f,\mathcal{S}}^{(i)}\|_2 \quad (2.9)$$

Pada rumus di atas, f merepresentasikan satu frame video, dan \mathcal{S} adalah kerangka tubuh (*skeleton*). $P_{f,\mathcal{S}}^{(i)}$ merupakan vektor posisi 3D prediksi dari *joint* ke- i , sedangkan $P_{gt,f,\mathcal{S}}^{(i)}$ adalah posisi 3D dari data *ground-truth*. Nilai N_S menyatakan jumlah total *joints*, biasanya sebanyak 17.

Misalkan terdapat 3 titik sendi (*joint*) dengan posisi prediksi dan *ground-truth* berikut:

- Joint 1: prediksi = (1, 2, 3), ground-truth = (1, 2, 2)
- Joint 2: prediksi = (2, 1, 0), ground-truth = (2, 2, 0)
- Joint 3: prediksi = (0, 0, 0), ground-truth = (1, 0, 0)

Jarak Euclidean untuk masing-masing joint:

$$d_1 = \sqrt{(1-1)^2 + (2-2)^2 + (3-2)^2} = \sqrt{1} = 1$$

$$d_2 = \sqrt{(2-2)^2 + (1-2)^2 + (0-0)^2} = \sqrt{1} = 1$$

$$d_3 = \sqrt{(0-1)^2 + (0-0)^2 + (0-0)^2} = \sqrt{1} = 1$$

$$\text{MPJPE} = \frac{1+1+1}{3} = 1.0 \text{ satuan jarak (misal dalam milimeter).}$$

Contoh ini menunjukkan bahwa jika rata-rata kesalahan jarak antara posisi

yang diprediksi dan posisi sebenarnya adalah 1 mm, maka MPJPE-nya adalah 1 mm. Nilai MPJPE yang lebih kecil menandakan prediksi pose yang lebih akurat.

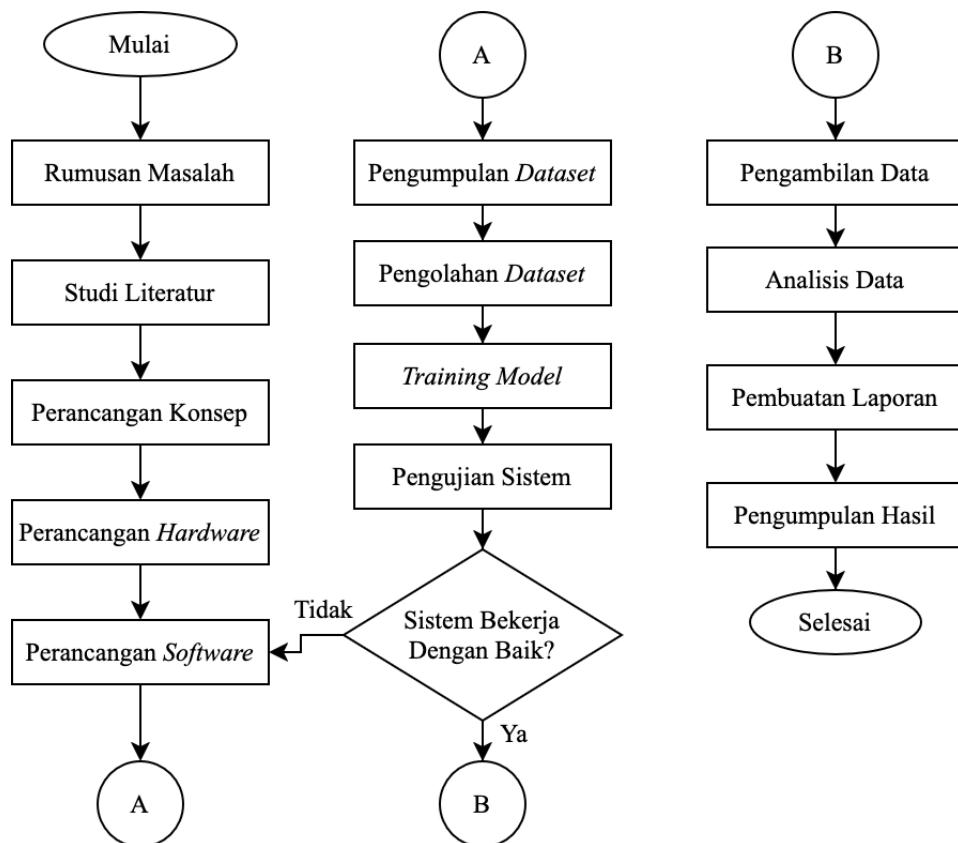
BAB III. METODOLOGI PENENILITIAN

3.1 Waktu dan Tempat Penelitian

Peneliti akan melaksanakan penelitian ini di Laboratorium Robotika Politeknik Negeri Malang (Polinema). Pemilihan lokasi ini dilakukan karena fasilitas laboratorium yang memadai untuk penelitian robotika serta keterlibatan tim Kontes Robot Tari Indonesia (KRSTI) Polinema, yang relevan dengan topik penelitian ini. Rentang waktu penelitian direncanakan mulai dari bulan Januari 2025 hingga Mei 2025. Pengujian sistem dan implementasi akan dilakukan di Laboratorium Robotika Polinema dengan dukungan tim Polinema Robotics.

3.2 Diagram Alir Penelitian

Untuk membuat sistem gerak robot berdasarkan *pose estimation*, dimulai dengan perancangan sistem kemudian diimplementasikan. Berikut ini pada 3.1 merupakan diagram alir dari penelitian yang dilakukan.



Gambar 3.1 Diagram Alir Penelitian

Penelitian ini dilaksanakan melalui serangkaian tahapan sistematis yang bertujuan untuk mengembangkan sistem pemodelan gerak tari tradisional Indonesia pada robot *humanoid* berbasis estimasi pose 3D. Proses dimulai dengan perumusan masalah, yaitu mengidentifikasi tantangan utama dalam mengubah gerakan tari manusia menjadi representasi gerak yang dapat diimplementasikan pada robot. Setelah masalah dirumuskan, dilakukan studi literatur untuk memperoleh pemahaman yang mendalam mengenai teori dan teknologi yang mendasari, seperti metode *pose estimation* (MeTRAbs, DeciWatch), pemrosesan dataset, dan pengendalian gerakan robot. Penelitian ini dilakukan melalui serangkaian tahapan terstruktur yang bertujuan untuk menghasilkan sistem pemodelan gerak tari tradisional Indonesia pada robot *humanoid* menggunakan metode estimasi pose. Alur penelitian diawali dengan identifikasi dan perumusan masalah, kemudian dilanjutkan dengan studi literatur untuk memperkuat dasar teori. Setelah itu dilakukan perancangan konsep dan perangkat lunak yang dibutuhkan, disusul oleh proses pengumpulan dan pengolahan data.

Model dilatih menggunakan data yang telah disiapkan, kemudian diuji untuk menilai kinerjanya. Apabila hasil pengujian belum sesuai, dilakukan evaluasi dan perbaikan sistem. Jika sistem telah berfungsi dengan baik, proses dilanjutkan dengan pengambilan data hasil, analisis, penyusunan laporan, dan pengumpulan hasil akhir.

3.3 Metode Pengumpulan Data

Pengumpulan data dilakukan untuk memperoleh informasi yang diperlukan dalam mendukung pelaksanaan penelitian, terutama terkait pengembangan dan pengujian sistem estimasi pose. Data yang digunakan dalam penelitian ini terdiri dari dua jenis utama, yaitu *dataset* standar untuk pelatihan model dan data pengujian dari video tari tradisional Indonesia.

3.3.1 *Dataset AIST++*

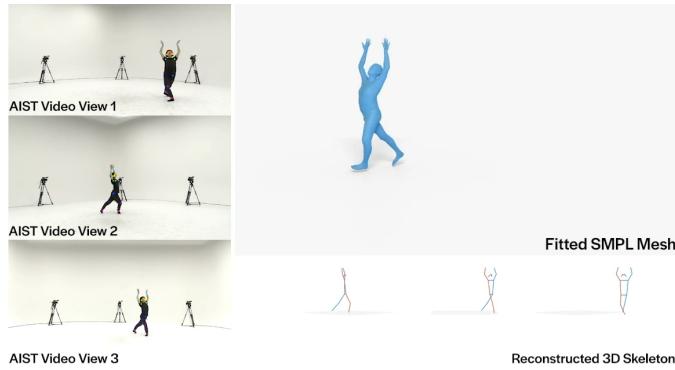
Penelitian ini menggunakan *dataset* AIST++, yang merupakan salah satu *benchmark dataset* standar untuk pelatihan dan evaluasi model estimasi pose manusia 3D, khususnya dalam konteks gerakan tari. AIST++ adalah *dataset* yang dirancang untuk menangkap gerakan tari yang kompleks dan beragam, berisi lebih

dari 10 juta frame video dengan anotasi pose 3D pada 60 fps (Li et al., 2021). *Dataset* ini mencakup berbagai gaya tarian, seperti hip-hop, ballet, breakdance, jazz, dan gaya lainnya, yang memberikan keragaman gerakan tubuh secara spasial dan temporal, sehingga sangat cocok untuk melatih model yang fokus pada dinamika gerakan yang cepat dan bervariasi.

Dalam penelitian ini, digunakan dua jenis data dari *dataset* AIST++ yakni hasil estimasi pose 3D dari video menggunakan model SPIN (*SMPLify-X In the Loop*) dan data *ground truth* pose 3D yang tersedia secara terbatas. Hal ini disebabkan oleh keterbatasan akses terhadap data video raw asli secara penuh, sehingga proses pelatihan model DeciWatch dilakukan menggunakan hasil prediksi SPIN sebagai data input dan data anotasi AIST++ sebagai target.

Data yang digunakan telah disusun dalam format .npz, yang berisi urutan pose 3D, parameter bentuk tubuh (shape), nama frame video, serta parameter kamera. Setiap file merepresentasikan satu sekuens gerakan dalam bentuk array dua dimensi, dengan jumlah frame yang bervariasi, rata-rata antara 300 hingga 600 frame per sekuens. Dataset ini dibagi menjadi dua bagian: data pelatihan (*train*) sebanyak 7.292 sekuens dengan total 5.916.474 frame, dan data validasi (*val*) sebanyak 3.840 sekuens dengan total 2.882.640 frame.

Penggunaan *dataset* AIST++ dalam penelitian ini dimaksudkan untuk melatih dan mengevaluasi model DeciWatch agar mampu mengenali dan menyempurnakan estimasi pose manusia dalam konteks gerakan tari. Kompleksitas dan keragaman data AIST++ menjadikannya relevan untuk mengembangkan model yang nantinya diharapkan dapat diadaptasikan pada representasi gerakan tari tradisional Indonesia yang memiliki karakteristik gerak yang dinamis dan kompleks.



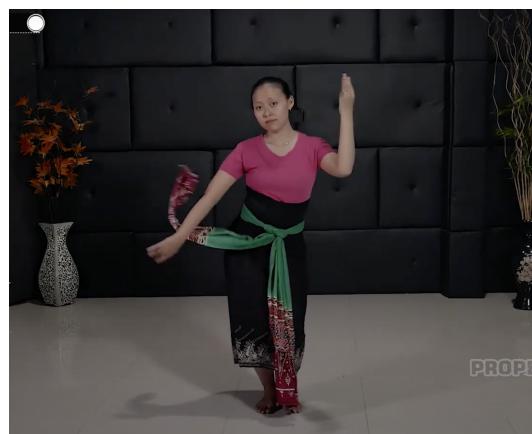
Gambar 3.2 *Dataset AIST++*
Sumber: (Li et al., 2021)

3.3.2 Data Video Tari Tradisional Indonesia

Untuk pengujian model, data yang digunakan berasal dari video tari tradisional Indonesia yang diambil dari platform seperti YouTube. Video yang dipilih harus memenuhi beberapa kriteria, yaitu:

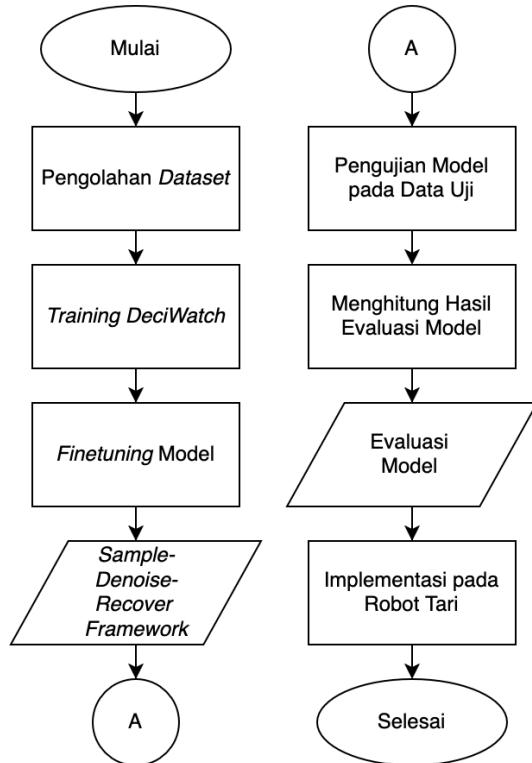
- Menampilkan gerakan tari secara jelas dan berkelanjutan.
- Menggunakan teknik pengambilan gambar dengan kamera statis untuk meminimalkan gangguan visual.
- Memuat berbagai jenis tari tradisional Indonesia yang representatif.

Video-video ini kemudian diproses untuk menghasilkan *keypoints* yang akan digunakan dalam evaluasi performa model estimasi pose. Sebagai contoh, gerakan tari seperti Denok Semarangan yang khas dan memiliki pola gerakan unik dapat menjadi bahan pengujian yang ideal untuk mengukur performa model.



Gambar 3.3 Tari Denok Semarangan
Sumber: YouTube

3.4 Teknik Pengolahan Data



Gambar 3.4 Diagram Alur Pengolahan Data

3.4.1 Pengolahan Dataset

Pada penelitian ini, *dataset* yang digunakan berasal dari AIST++ Human Motion Dataset, yang sudah tersedia dalam bentuk data pose 3D yang siap digunakan tanpa perlu proses anotasi atau pengambilan data tambahan. Dataset ini telah diolah oleh pengembang DeciWatch dan dilengkapi dengan anotasi *ground truth* serta hasil prediksi pose dari model SPIN. Struktur data AIST++ yang digunakan terdiri dari dua jenis data utama, yaitu:

- *Ground Truth Pose*: Berisi parameter koordinat 3D untuk 14 titik sendi tubuh.
- *Predicted Pose (SPIN)*: Berisi hasil prediksi pose dari model SPIN yang berupa koordinat 3D titik sendi.

Adapun langkah-langkah pengolahan *dataset* yang dilakukan dalam penelitian ini meliputi:

1. Pemilihan Data Latih dan Data Uji

Data dibagi menjadi dua bagian utama, yaitu data pelatihan (train) dan data

pengujian (val). File aist_gt_train.npz dan aist_gt_test.npz digunakan untuk *ground truth*, sedangkan aist_spin_train.npz dan aist_spin_test.npz digunakan untuk data prediksi SPIN.

2. Pemuatan dan Pemrosesan Struktur Data

Data dimuat menggunakan fungsi numpy.load() yang menghasilkan struktur dictionary berisi:

- imgname: Nama file dan indeks frame.
- pose: Parameter pose SMPL (72 dimensi).
- trans: Translasi gerakan 3D (3 dimensi).
- scaling: Faktor skala tubuh.
- joints_3d: Koordinat 3D untuk 14 titik sendi utama.

3. Pemetaan Keypoints

Data joints_3d pada AIST++ memiliki urutan 14 titik sendi sesuai format: *r ankle, rknee, rhip, lhip, lknee, lankle, rwrists, relbow, rshoulder, lshoulder, lelbow, lwrists, neck, headtop*. Urutan ini identik dengan skema *LSP-14* yang juga digunakan pada metode *Metrabs*, sehingga kompatibel dengan standar pemetaan pose yang umum digunakan dalam literatur visi komputer. Pemetaan ini digunakan sebagai acuan dalam proses *motion retargeting*, khususnya untuk menentukan titik-titik referensi pada perhitungan sudut sendi di robot humanoid.

4. Penyesuaian Input ke Model DeciWatch

Dataset ini kemudian digunakan sebagai input langsung ke model DeciWatch untuk proses prediksi pose berjangka waktu (temporal pose prediction). Karena DeciWatch membutuhkan input pose berurutan dalam bentuk sekuens, data pose diurutkan dan diproses dalam batch sesuai kebutuhan model.

Proses ini tidak melibatkan anotasi manual karena seluruh data sudah tersedia dalam format yang sesuai untuk keperluan pelatihan dan pengujian model. Pengolahan dilakukan untuk memastikan data tersusun sesuai kebutuhan input pada sistem yang dikembangkan.

3.4.2 Training dan Fine-tuning Model

Setelah dataset hasil prediksi pose 3D oleh model Metrabs berhasil dikumpulkan dan diproses, langkah selanjutnya adalah melakukan pelatihan dan *fine-tuning* model DeciWatch. Model ini bertujuan untuk meningkatkan akurasi estimasi pose secara temporal dengan memanfaatkan informasi antar-frame secara berurutan.

Model DeciWatch dilatih menggunakan data input berupa pose 3D dari Metrabs sebagai *frame-wise prediction*, kemudian disempurnakan melalui pembelajaran sekuensial untuk menghasilkan prediksi yang lebih halus dan akurat. Proses pelatihan dilakukan menggunakan *framework* PyTorch dengan bantuan GPU yang mendukung CUDA dan cuDNN untuk mempercepat komputasi. Tahap *fine-tuning* dilakukan dengan mengatur ulang parameter dan konfigurasi pelatihan agar model lebih sesuai dengan karakteristik data tari tradisional yang digunakan.

3.4.3 Pengujian dan Evaluasi Model

Model yang telah dilatih kemudian diuji menggunakan data uji yang berbeda dari data pelatihan untuk mengukur performanya. Evaluasi dilakukan dengan menghitung metrik *Mean Per Joint Position Error* (MPJPE), yaitu rata-rata jarak Euclidean antara koordinat prediksi dan *ground truth* dari setiap *keypoint* dalam ruang 3D. Perhitungan MPJPE dilakukan sesuai dengan rumus yang telah dijelaskan pada Bab 2 pada persamaan (2.9). Nilai MPJPE yang lebih rendah menunjukkan akurasi prediksi yang lebih baik.

Proses evaluasi dilakukan dengan bantuan pustaka Python seperti NumPy untuk perhitungan matematis dan Matplotlib untuk analisis visualisasi hasil. Seluruh eksperimen dilakukan di lingkungan Jupyter Notebook atau Google Colab untuk mendukung fleksibilitas dalam pemrosesan dan pengujian model.

3.4.4 Implementasi pada Robot Tari

Setelah model dievaluasi, tahap berikutnya adalah implementasi pada robot humanoid ROBOTIS-OP3 untuk menguji kemampuan model dalam menghasilkan gerakan yang natural berdasarkan estimasi pose. Implementasi ini dilakukan dengan menggunakan bahasa pemrograman Python dan C++, serta memanfaatkan

framework *Robot Operating System* (ROS) untuk mengelola komunikasi antara perangkat keras dan perangkat lunak. Untuk menerjemahkan koordinat pose ke dalam pergerakan robot, digunakan konsep *Inverse Kinematics* (IK) dengan dukungan pustaka Dynamixel SDK untuk mengontrol aktuator robot. Pengujian pergerakan robot juga dapat dilakukan di lingkungan simulasi menggunakan Gazebo dan Rviz sebelum diterapkan pada perangkat keras sebenarnya.

3.5 Uji Coba Sistem

Uji coba sistem dilakukan untuk mengevaluasi keandalan dan kesesuaian fungsional sistem dalam mereplikasi gerakan manusia menggunakan robot humanoid ROBOTIS-OP3. Proses uji coba mencakup dua jenis pengujian utama, yaitu pengujian fungsional berupa *black-box testing* dan pengujian performa dari model estimasi pose serta implementasi gerakan pada robot.

Secara umum, uji coba sistem difokuskan pada dua aspek utama berikut:

- Pengujian Fungsional Sistem (*Black-Box Testing*), yaitu pengujian terhadap alur kerja sistem secara keseluruhan, mulai dari input data gerak, pemrosesan melalui antarmuka berbasis *website*, hingga eksekusi gerakan pada robot. Pengujian ini memastikan bahwa seluruh komponen sistem dapat berfungsi dengan baik tanpa melihat detail proses di dalamnya.
- Pengujian Performa Estimasi dan Reproduksi Gerakan, meliputi:
 - *Akurasi prediksi pose temporal* dari model DeciWatch, dengan evaluasi menggunakan metrik *Mean Per Joint Position Error (MPJPE)* seperti pada persamaan (2.9).
 - *Kesesuaian hasil motion retargeting* pada robot ROBOTIS-OP3, yaitu evaluasi kualitas reproduksi gerakan robot terhadap urutan pose hasil prediksi serta kelancaran eksekusi pada saat proses *playback*.

Evaluasi ini dilakukan untuk memastikan sistem bekerja secara fungsional dan juga memiliki tingkat akurasi serta performa gerakan yang sesuai dengan kebutuhan pemodelan gerak tari tradisional Indonesia.

BAB IV. ANALISIS DAN PERANCANGAN SISTEM

4.1 Deskripsi Sistem

Sistem yang dikembangkan dalam penelitian ini merupakan sebuah aplikasi pemodelan gerak tari tradisional Indonesia dengan memanfaatkan teknologi *pose estimation* dan robot *humanoid* ROBOTIS-OP3. Sistem ini dirancang untuk mengekstraksi gerakan manusia dari video tari tradisional, menerjemahkannya ke dalam data koordinat 3D tubuh manusia, kemudian mengonversinya menjadi pergerakan fisik pada robot *humanoid*.

Proses utama dari sistem dimulai dari input berupa video tari tradisional Indonesia yang dianalisis menggunakan model *pose estimation* berbasis *deep learning*. Model awal menggunakan pendekatan Metrabs (METa-learning TRAnsformer for 3D human pose eStimation) untuk menghasilkan koordinat pose 3D dari setiap frame dalam video, kemudian disempurnakan dengan metode DeciWatch yang meningkatkan akurasi dan kelancaran gerakan melalui pendekatan *sample-denoise-recover*. Hasil estimasi berupa data *keypoints* 3D kemudian dikonversi menjadi sudut-sudut servo (*servo angles*) menggunakan algoritma *Inverse Kinematics* (IK), dan seluruh data gerak tersebut disimpan dalam format file seperti JSON. File ini dapat digunakan untuk input kontrol robot *humanoid* ROBOTIS-OP3 secara terpisah melalui sistem eksekusi gerak yang terintegrasi dengan ROS, sehingga memungkinkan robot untuk mereplikasi gerakan tari secara bertahap sesuai instruksi dalam file tersebut.

4.2 Analisa Kebutuhan Sistem

Dalam membangun sistem pemodelan gerak tari tradisional berbasis *pose estimation*, dilakukan analisa kebutuhan terhadap perangkat keras (*hardware*) dan perangkat lunak (*software*) yang digunakan dalam proses pengolahan data, pelatihan model, serta integrasi dengan robot *humanoid* ROBOTIS-OP3. Kebutuhan ini ditentukan berdasarkan tingkat kompleksitas komputasi dan kompatibilitas sistem.

4.2.1 Kebutuhan Perangkat Keras

Perangkat keras yang digunakan dalam penelitian ini terdiri dari laptop/PC untuk pengolahan data dan robot *humanoid* ROBOTIS-OP3 sebagai platform eksekusi gerakan. Spesifikasi perangkat keras yang dibutuhkan adalah sebagai berikut:

Tabel 4.1 Spesifikasi Perangkat Keras Laptop

No	Nama Perangkat Keras
1	Intel Core i5 Gen-10
2	NVIDIA GeForce RTX 3060 6GB Mobile
3	RAM 16GB
4	SSD 256GB

Tabel 4.2 Spesifikasi Perangkat Keras Robot

No	Nama Perangkat Keras
1	Intel Core i3-7100U
2	RAM 8GB
3	SSD 256GB
4	Dynamixel XM430-W350, 2XL430-W250

4.2.2 Kebutuhan Perangkat Lunak

Perangkat lunak yang digunakan dalam penelitian ini meliputi sistem operasi, bahasa pemrograman, dan berbagai pustaka (*library*) yang mendukung pengolahan data, pelatihan model, serta komunikasi dengan robot. Berikut adalah spesifikasi perangkat lunak yang dibutuhkan:

Tabel 4.3 Spesifikasi Perangkat Lunak

No	Nama	Keterangan
1	Sistem Operasi Laptop	Ubuntu 24.04 LTS
2	Sistem Operasi Robot	Linux Mint 16
3	Bahasa Pemrograman	Python 3.9, C++, Bash

4	Text Editor	Visual Studio Code, Nano
5	Pustaka Pemrograman	TensorFlow, PyTorch, NumPy, OpenCV, Matplotlib, Flask
6	Robot Operating System	ROS Kinetic, ROS 2 Humble
7	Perangkat Lunak Pendukung	Jupyter Notebook, Google Colab

4.2.3 Analisa Kebutuhan Fungsionalitas

Analisis kebutuhan fungsional merupakan proses identifikasi dan dokumentasi terhadap fitur dan fungsi spesifik yang harus dimiliki oleh sistem. Proses ini bertujuan untuk mendefinisikan tugas-tugas utama yang dilakukan oleh sistem, cara sistem berinteraksi dengan pengguna, serta bagaimana sistem merespons input yang diterima. Kebutuhan ini menjadi dasar dalam proses perancangan dan implementasi sistem secara keseluruhan, sebagaimana ditunjukkan pada Tabel 4.4.

Tabel 4.4 Kebutuhan Fungsional Sistem

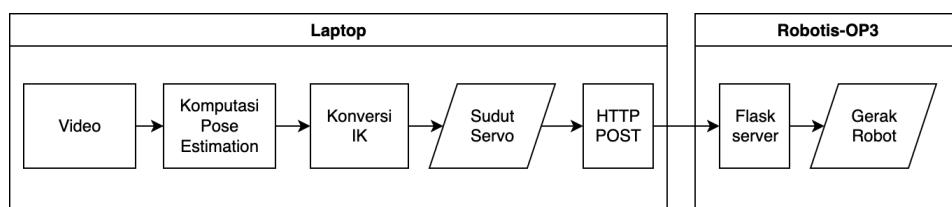
No	Fungsi	Deskripsi
1	Upload Video	Sistem menyediakan fitur bagi pengguna untuk mengunggah video tari tradisional Indonesia dalam format umum seperti MP4 atau AVI. Nama file akan tampil pada area log sebagai konfirmasi.
2	Estimasi Pose dan Filtering	Sistem melakukan proses estimasi pose tubuh manusia dalam bentuk koordinat 3D <i>keypoints</i> menggunakan model MeTRAbs dan DeciWatch. Hasil estimasi difilter untuk mengurangi noise pada prediksi temporal.
3	Visualisasi Skeleton 3D	Sistem menampilkan hasil estimasi pose dalam bentuk skeleton 3D yang divisualisasikan sesuai urutan frame video.

No	Fungsi	Deskripsi
4	Status Log	Sistem mencatat dan menampilkan status proses seperti upload, estimasi, dan penyimpanan pada area log agar pengguna dapat memantau jalannya sistem.
5	Penyimpanan Data	Sistem menyimpan hasil estimasi pose ke dalam file berformat JSON yang dapat digunakan untuk kontrol robot. File ini akan disimpan di direktori output.
6	Playback Robot	Sistem mengontrol robot ROBOTIS-OP3 untuk mereproduksi gerakan berdasarkan file JSON hasil estimasi. Robot akan bergerak sesuai urutan sudut artikulasi yang dihasilkan dari <i>motion retargeting</i> .

4.3 Perancangan Sistem

Perancangan sistem dilakukan untuk mendefinisikan bagaimana komponen-komponen dalam sistem saling berinteraksi dan berfungsi dalam keseluruhan proses pemodelan gerak tari tradisional menggunakan teknologi *pose estimation*. Sistem ini dirancang dengan pendekatan modular, mulai dari tahap akuisisi video hingga penyimpanan data gerakan dalam format yang dapat dieksekusi oleh robot *humanoid* ROBOTIS-OP3.

4.3.1 Arsitektur Sistem



Gambar 4.1 Diagram Arsitektur Sistem

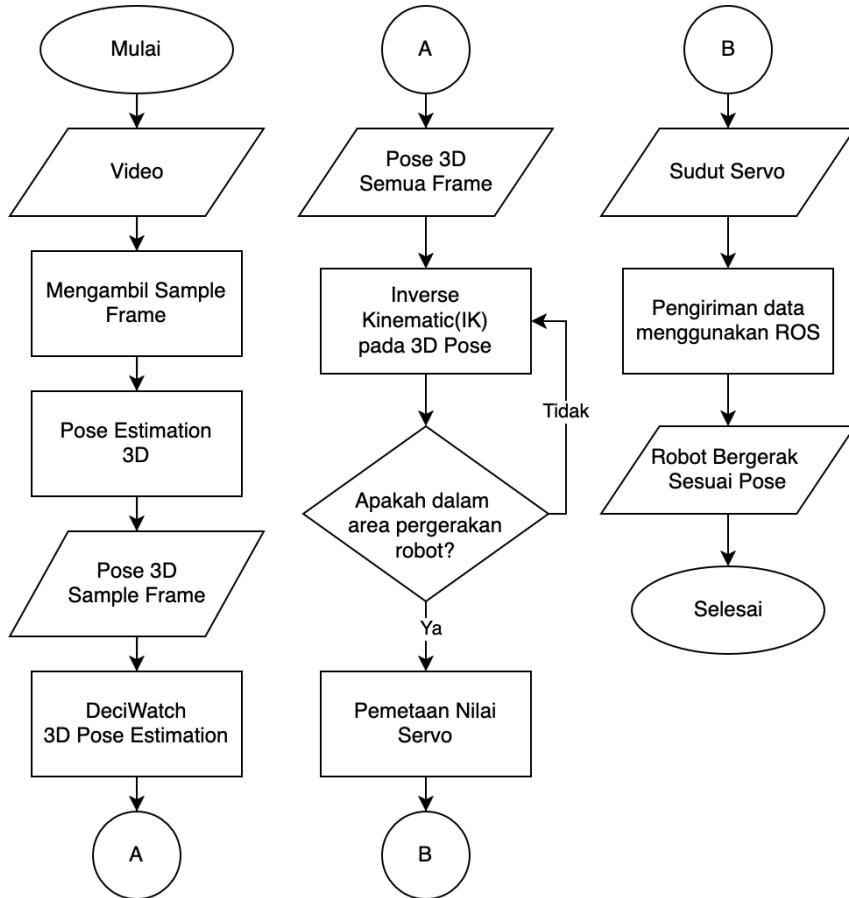
Gambar 4.1 menunjukkan arsitektur sistem yang dikembangkan untuk pemodelan gerak tari tradisional menggunakan teknologi *pose estimation*. Sistem

ini terdiri dari dua bagian utama, yaitu pemrosesan data pada laptop dan eksekusi gerakan pada ROBOTIS-OP3.

Pada sisi laptop, input berupa video tari diproses melalui modul komputasi estimasi pose untuk memperoleh koordinat 3D tubuh manusia. Data koordinat ini kemudian dikonversi menggunakan algoritma *Inverse Kinematics* (IK) menjadi sudut servo yang sesuai dengan struktur pergerakan robot. Sudut servo tersebut dikemas dalam format *JSON* dan dikirimkan ke robot melalui protokol komunikasi *HTTP POST* menggunakan *Flask client* di sisi laptop.

Sementara itu, di sisi robot ROBOTIS-OP3, data gerakan diterima oleh *Flask server*, kemudian disimpan dalam format file *JSON* sebagai rekaman gerak. Data ini digunakan untuk mengontrol aktuator robot sehingga menghasilkan gerakan yang sesuai dengan video input. Pendekatan ini memanfaatkan pemisahan beban komputasi, di mana proses komputasi berat dilakukan di laptop, sedangkan robot fokus pada eksekusi pergerakan. Metode ini juga memudahkan integrasi sistem dan pengujian karena menggunakan protokol *HTTP* yang lebih sederhana dan fleksibel dibandingkan komunikasi *socket*.

4.3.2 Alur Kerja Sistem



Gambar 4.2 Flowchart Perancangan Sistem

Gambar 4.2 menunjukkan alur kerja sistem yang dirancang dalam penelitian ini, dimulai dari input video hingga eksekusi gerakan oleh robot *humanoid* ROBOTIS-OP3. Berikut adalah penjelasan rinci tiap langkah dalam alur sistem:

1. Input Video

Sistem menerima masukan berupa video tari tradisional Indonesia dalam format umum seperti MP4.

2. Pengambilan Sampel Frame

Video diolah dengan mengambil sampel beberapa frame secara berkala untuk mengurangi beban komputasi dan menyesuaikan dengan kebutuhan input model.

3. Pose Estimation 3D

Setiap frame hasil sampel diproses menggunakan model *pose estimation*

berbasis Metrabs untuk menghasilkan prediksi koordinat 3D dari titik-titik persendian tubuh manusia. Sistem mendukung dua jenis model *skeleton*, yaitu *LSP-14* yang terdiri dari 14 titik persendian utama dan *COCO-19* yang terdiri dari 19 titik, sehingga dapat disesuaikan dengan kebutuhan pemetaan gerakan pada robot.

4. DeciWatch 3D Pose Estimation

Output pose 3D dari Metrabs kemudian diolah lebih lanjut menggunakan DeciWatch. Model ini melakukan prediksi berurutan (temporal) untuk memastikan kontinuitas dan kelancaran gerakan antar-frame, sehingga menghasilkan estimasi pose 3D yang lebih halus.

5. Inverse Kinematics (IK) pada 3D Pose

Seluruh pose 3D hasil prediksi kemudian dikonversi menjadi sudut sendi menggunakan metode *Inverse Kinematics Transform (IKT)*. Perhitungan ini dilakukan untuk mendapatkan sudut aktuator yang diperlukan agar robot dapat menirukan pose tersebut.

6. Validasi Area Pergerakan Robot

Sistem memeriksa apakah hasil perhitungan IK berada dalam jangkauan aktuator robot untuk menghindari servo bertabrakan. Jika tidak, maka pose akan disesuaikan agar tetap dalam batas gerak yang dapat dieksekusi oleh ROBOTIS-OP3 dengan cara membatasi sudut tersebut.

7. Pemetaan Nilai Servo

Setelah validasi, hasil IK dipetakan menjadi nilai sudut servo sesuai dengan konfigurasi servo pada robot.

8. Pengiriman Data ke Robot

Nilai sudut servo yang telah dihasilkan dari proses *motion retargeting* dikirimkan ke robot menggunakan metode komunikasi berbasis HTTP POST. Data sudut servo dikemas dalam format *JSON* dan dikirimkan melalui jaringan ke alamat IP robot.

9. Eksekusi Gerakan Robot

Robot ROBOTIS-OP3 akan mereplikasi gerakan sesuai dengan urutan pose yang telah dikirimkan, sehingga mampu merepresentasikan gerakan tari dalam

bentuk fisik secara otomatis.

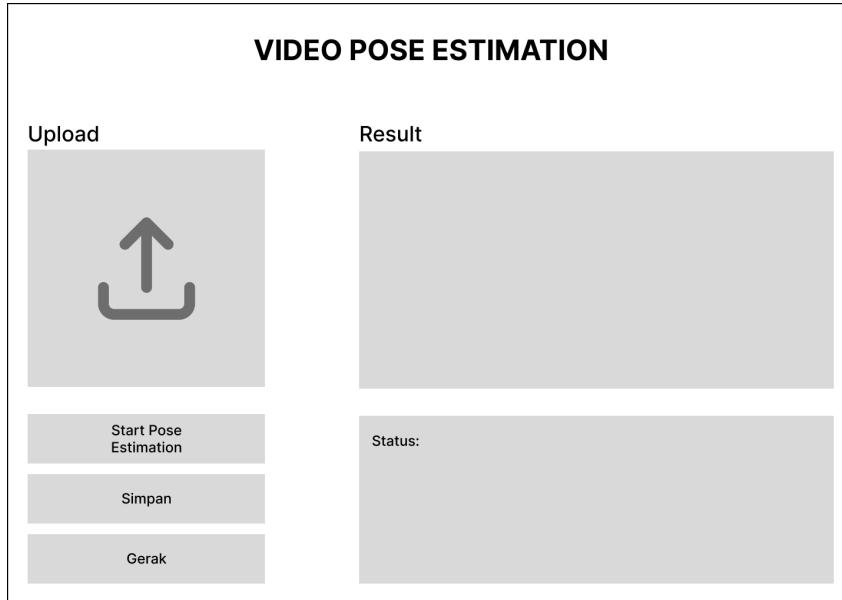
Dengan alur ini, sistem secara menyeluruh dapat mengonversi gerakan manusia dari video menjadi pergerakan robot yang menyerupai gerakan asli secara berurutan dan sinkron.

Potongan Kode 4.1 Pseudocode Struktur File JSON Gerakan

```
{  
    "frame_001": {  
        "joint_1": 45.2,  
        "joint_2": 30.8,  
        ...  
    },  
    "frame_002": {  
        "joint_1": 46.0,  
        "joint_2": 32.1,  
        ...  
    },  
    ...  
}
```

Format ini memudahkan sistem kendali robot dalam membaca data per frame secara teratur dan menjalankan gerakan yang telah disesuaikan dengan hasil *pose estimation*.

4.4 Perancangan Antarmuka Pengguna



Gambar 4.3 Perancangan Antarmuka Sistem

Gambar 4.3 menunjukkan rancangan awal antarmuka pengguna (UI) dari sistem yang dikembangkan. Antarmuka ini dirancang dengan tampilan yang sederhana dan intuitif untuk mendukung alur kerja mulai dari proses unggah video, estimasi pose 3D, hingga pengendalian gerak robot humanoid.

Di bagian kiri antarmuka terdapat elemen unggah video yang memungkinkan pengguna untuk memilih dan memasukkan video tari tradisional yang akan dianalisis. Setelah video berhasil diunggah, pengguna dapat memulai proses estimasi pose dengan menekan tombol "Start Pose Estimation". Proses ini akan mengaktifkan model *pose estimation* untuk menghasilkan koordinat 3D dari setiap titik sendi tubuh manusia dalam video.

Hasil estimasi kemudian divisualisasikan pada area di sisi kanan antarmuka dalam bentuk tampilan *3D skeleton*. Visualisasi ini memberikan gambaran postur dan gerakan tubuh dari objek dalam video secara menyeluruh berdasarkan estimasi pose yang telah dilakukan.

Setelah proses estimasi selesai, pengguna dapat menyimpan hasilnya dengan tombol "Simpan". Hasil yang disimpan berupa file JSON yang berisi informasi pose 3D dan sudut servo yang telah dihitung. Tombol "Gerak" digunakan

untuk mengirimkan data gerak tersebut ke robot ROBOTIS-OP3 melalui sistem komunikasi ROS, sehingga robot dapat menirukan gerakan berdasarkan data yang telah diproses.

Di bagian bawah kanan antarmuka terdapat area status yang menampilkan informasi proses secara langsung, seperti log atau keluaran terminal. Area ini memberikan umpan balik kepada pengguna terkait status estimasi, penyimpanan data, maupun proses pengiriman ke robot, sehingga memudahkan dalam memantau dan mengontrol seluruh jalannya sistem.

4.5 Perancangan Modifikasi Hardware

Robot humanoid ROBOTIS-OP3 yang digunakan dalam penelitian ini merupakan platform robot terbuka yang dilengkapi dengan aktuator servo pada berbagai sendi tubuh. Namun, dalam bentuk standarnya, konfigurasi pergerakan tangan pada OP3 masih terbatas. Secara default, masing-masing lengan robot hanya memiliki tiga *degree of freedom* (DoF), yaitu:

- 2 DoF pada bahu (*shoulder pitch* dan *shoulder roll*),
- 1 DoF pada siku (*elbow flexion*).

Konfigurasi ini cukup untuk gerakan dasar, namun belum mencukupi untuk mengekspresikan variasi gerakan tangan yang kompleks, seperti yang banyak ditemukan pada gerakan tari tradisional Indonesia yang menuntut rotasi pergelangan atau variasi gerak lengan secara lebih bebas.

Oleh karena itu, dilakukan modifikasi terhadap bagian tangan robot dengan menambahkan satu derajat kebebasan tambahan pada sendi siku berupa *elbow twist*. Dengan penambahan ini, lengan robot memiliki total empat derajat kebebasan sebagai berikut:

- 2 DoF pada bahu: *pitch* dan *roll*,
- 2 DoF pada siku: *flexion* dan *twist*.

Penambahan DoF pada siku memungkinkan lengan robot untuk melakukan gerakan rotasi aksial, seperti memutar lengan bawah atau menyesuaikan arah tangan, yang sangat penting dalam meniru ekspresi gerak tangan pada tari tradisional. Modifikasi ini dilakukan dengan menambahkan satu motor servo tambahan yang

dipasang secara koaksial pada sambungan siku, serta penyesuaian desain fisik dudukan servo agar tetap sesuai dengan struktur mekanik robot.

Modifikasi ini juga disertai dengan penyesuaian pada sistem kontrol dan inverse kinematics agar sistem dapat mengenali dan mengatur sudut baru tersebut. Dengan peningkatan fleksibilitas pada bagian lengan, robot memiliki kemampuan yang lebih tinggi dalam merepresentasikan gestur kompleks secara fisik dan artistik.

BAB V. IMPLEMENTASI DAN PENGUJIAN

5.1 Implementasi Sistem

5.1.1 Data *Preprocessing*

Pada tahap ini, proses *data preprocessing* dilakukan untuk menyiapkan data pose 3D yang akan digunakan dalam pelatihan dan evaluasi model DeciWatch. Dataset yang digunakan merupakan dataset AIST++ yang menyediakan anotasi gerakan tari dalam format koordinat 3D, namun karena keterbatasan akses terhadap data video asli, proses deteksi pose awal dilakukan menggunakan model SPIN (*SMPLify-X In the Loop*) yang memiliki struktur skeleton yang sama.

Model SPIN digunakan untuk mengekstraksi data pose 3D dari video tari yang tersedia dalam dataset AIST++. Hasil prediksi pose 3D dari SPIN kemudian dipasangkan dengan data *ground truth* yang tersedia untuk keperluan pelatihan dan evaluasi model DeciWatch. Pasangan data ini terdiri dari dua bagian: data input berupa hasil estimasi pose dari SPIN, dan data target berupa anotasi pose 3D asli dari dataset AIST++ yang tersedia secara terbatas.

Setiap data pose 3D disimpan dalam format .npz yang berisi beberapa array utama. Struktur file .npz ini dijelaskan sebagai berikut:

Struktur data dalam file .npz yang digunakan pada sistem ini memiliki beberapa *key* utama yang berisi informasi penting terkait pose 3D manusia untuk setiap sekuens gerakan. Setiap file menyimpan satu sekuens gerakan yang terdiri dari sejumlah frame, dengan format array sebagai berikut:

- *imgname* : Array berisi nama file atau jalur gambar dari frame-frame dalam sekuens. Tipe datanya adalah numpy.ndarray satu dimensi, dengan jumlah elemen sesuai panjang sekuens. Setiap elemen berupa string yang merepresentasikan nama atau jalur gambar dari masing-masing frame.
- *pose* : Array dua dimensi dengan bentuk (N, 72), di mana N adalah jumlah frame dalam sekuens. Setiap baris mewakili rotasi 24 sendi tubuh manusia dalam format axis-angle (masing-masing 3 dimensi, total 72). Data ini digunakan sebagai representasi postur tubuh untuk setiap frame.
- *shape* : Array dua dimensi dengan bentuk (N, 10), yang merepresentasikan

parameter bentuk tubuh manusia dalam model SMPL. Nilai-nilai ini menggambarkan variasi bentuk tubuh dan tetap konstan sepanjang sekuens dalam kebanyakan kasus.

- *joints_3d* : Array dua dimensi dengan bentuk (N, 3), yang menyimpan koordinat 3D dari titik pusat tubuh atau *root joint* pada setiap frame. Koordinat ini digunakan sebagai referensi posisi global dalam ruang 3D.
- *camera* : Array dua dimensi yang menyimpan parameter kamera per frame. Parameter ini digunakan dalam proses transformasi koordinat dari ruang kamera ke ruang dunia, serta untuk kebutuhan rendering ulang atau evaluasi terhadap data hasil prediksi.

Struktur ini berlaku untuk seluruh file .npz baik pada data pelatihan maupun validasi. Fleksibilitas format ini memungkinkan pemrosesan dan pelatihan model temporal seperti DeciWatch secara efisien dan modular berdasarkan sekuens per gerakan.

Dataset yang digunakan terdiri dari dua bagian utama, yaitu data pelatihan (*train*) dan data validasi (*val*). Pada bagian *train*, terdapat sebanyak 7.292 sekuens gerakan dengan total 5.916.474 frame. Sementara itu, bagian *val* terdiri dari 3.840 sekuens dengan total 2.882.640 frame. Rata-rata panjang setiap sekuens berkisar antara 300 hingga 600 frame, bergantung pada durasi dan kompleksitas gerakan dalam video.

Setiap sekuens menyimpan data dalam format .npz, yang berisi atribut seperti pose, shape, joints_3d, dan camera. Struktur ini memungkinkan pengolahan data secara terpisah per sekuens, sekaligus memudahkan pelatihan model berbasis waktu seperti DeciWatch yang membutuhkan urutan frame yang berurutan dan konsisten. Dengan struktur yang terorganisir dan skala data yang besar, model dapat dilatih untuk mengenali pola gerakan manusia secara lebih akurat dan stabil dari waktu ke waktu.

Jumlah frame yang besar dan cakupan gerakan yang beragam dalam dataset ini memberikan fondasi yang kuat bagi model DeciWatch untuk belajar pola gerakan manusia secara kompleks dan merepresentasikannya secara halus dalam prediksi pose 3D.

5.1.2 Inisialisasi Dataset dan *DataLoader*

Sebelum proses pelatihan dimulai, sistem terlebih dahulu memuat data pelatihan dan data uji dengan memanfaatkan kelas dataset yang telah disesuaikan dengan konfigurasi yang ditentukan. Proses ini dilakukan dengan memanggil fungsi *dataset_class* dan memberikan parameter seperti nama estimator (dalam hal ini menggunakan SPIN), jenis representasi tubuh (3D), serta bagian data yaitu *train* atau *test*. Pemanggilan ini akan menghasilkan dua objek yaitu *train_dataset* dan *test_dataset*.

Selanjutnya, kedua objek dataset tersebut dibungkus ke dalam *DataLoader* dari PyTorch untuk memfasilitasi pemrosesan data dalam bentuk mini-batch selama proses pelatihan dan evaluasi model. Untuk data pelatihan digunakan ukuran batch sebesar 512 dan opsi *shuffle* diaktifkan agar data tersaji secara acak pada setiap epoch. Sementara itu, pada data pengujian digunakan batch size 1 tanpa *shuffle*, agar urutan data tetap terjaga selama proses evaluasi.

Potongan Kode 5.1 Inisialisasi *dataset* dan *DataLoader* untuk pelatihan dan pengujian

```
train_dataset = dataset_class(
    dataset_name='aist',
    estimator='spin',
    return_type='3D',
    sample_interval=5,
    smpl_model_dir="data/smpl/",
    phase='train'
)

test_dataset = dataset_class(
    dataset_name='aist',
    estimator='spin',
    return_type='3D',
    sample_interval=5,
    smpl_model_dir="data/smpl/",
    phase='test'
)

train_loader = DataLoader(
```

```

        dataset=train_dataset,
        batch_size=512,
        shuffle=True,
        num_workers=4,
        pin_memory=True,
        worker_init_fn=worker_init_fn
    )

test_loader = DataLoader(
    dataset=test_dataset,
    batch_size=1,
    shuffle=False,
    num_workers=4,
    pin_memory=True,
    worker_init_fn=worker_init_fn
)

```

Penggunaan *DataLoader* ini sangat penting untuk mengoptimalkan performa proses pelatihan model, terutama dalam hal efisiensi pemuatan data dari penyimpanan ke memori. Pengaturan seperti jumlah *worker* dan aktivasi *pin memory* membantu meningkatkan kecepatan transfer data antar proses dan perangkat GPU.

5.1.3 Pembuatan Model

Pada tahap ini dilakukan perancangan dan konfigurasi arsitektur model untuk menyempurnakan prediksi pose 3D secara temporal. Model yang digunakan adalah DeciWatch, yaitu model prediktif berbasis *transformer encoder-decoder* yang dirancang untuk memproses urutan pose manusia dan menghasilkan prediksi yang lebih halus dan konsisten antar frame.

Potongan Kode 5.2 Potongan isi file config.yaml

```

MODEL:
  TYPE: network
  ENCODER_TRANSFORMER_BLOCK: 5
  ENCODER_EMBEDDING_DIMENSION: 128
  ENCODER_HEAD: 4
  ENCODER_RESIDUAL: true

```

```

DECODER_TRANSFORMER_BLOCK: 6
DECODER_EMBEDDING_DIMENSION: 128
DECODER_HEAD: 4
DECODER_RESIDUAL: true
DECODER_TOKEN_WINDOW: 5
SLIDE_WINDOW: true
SLIDE_WINDOW_SIZE: 51
SLIDE_WINDOW_Q: 10

```

Arsitektur model terdiri dari *transformer encoder-decoder*, dengan konfigurasi sebagai berikut:

- Encoder terdiri dari 5 blok transformer, dimensi embedding 128, 4 *attention head*, dengan koneksi residual.
- Decoder terdiri dari 6 blok transformer, embedding 128, dengan token window berukuran 5.
- Pendekatan *sliding window* digunakan untuk proses input sekuens dengan ukuran jendela 51 frame dan langkah pergeseran 10.
- Sampling frame dilakukan setiap 5 frame secara seragam.

Dataset yang digunakan adalah AIST++ dengan representasi tubuh 3D, menggunakan 14 titik sendi utama tanpa aktivasi SMPL.

Model diimplementasikan dengan PyTorch menggunakan kelas DeciWatch yang diturunkan dari nn.Module, seperti pada kode berikut:

Potongan Kode 5.3 Model DeciWatch

```

model = DeciWatch(input_dim=train_dataset.input_dimension,
                    sample_interval=5,
                    encoder_hidden_dim=128,
                    decoder_hidden_dim=128,
                    dropout=0.1,
                    nheads=4,
                    dim_feedforward=256,
                    enc_layers=5,
                    dec_layers=6,
                    activation="leaky_relu",
                    pre_norm=True,

```

```

recovernet_interJ_method="linear",
recovernet_mode="transformer").to('cuda')

```

Inisialisasi ini memanfaatkan nilai *input dimension* dari dataset pelatihan untuk menentukan jumlah fitur per frame yang akan diproses oleh model. Dengan parameter yang fleksibel dan bersifat modular, sistem ini memungkinkan penyesuaian cepat terhadap jenis data, panjang sekuens, atau kompleksitas gerakan yang berbeda pada skenario pelatihan lain di masa mendatang.

Setelah model berhasil diinisialisasi, langkah berikutnya adalah mendefinisikan fungsi loss dan algoritma optimisasi yang akan digunakan selama proses pelatihan. Pada sistem ini, digunakan kelas *DeciWatchLoss* yang dirancang khusus untuk menghitung selisih antara hasil prediksi model dengan data *ground truth* dalam bentuk pose 3D. Fungsi loss ini juga mendukung fitur tambahan seperti denoising dan regularisasi, yang dapat dikontrol melalui parameter konfigurasi.

Karena penelitian ini hanya difokuskan pada representasi pose dalam format 3D, maka fitur SMPL tidak diaktifkan (*smpl = False*). Parameter seperti bobot loss untuk denoise (*w_denoise*) dan koefisien regularisasi (*lamada*) diatur sesuai nilai pada file konfigurasi. Fungsi loss menggunakan kelas DeciWatchLoss dengan pengaturan:

- Bobot denoise: 1.0
- Koefisien regularisasi (λ): 5.0
- SMPL: Tidak digunakan (hanya pose 3D)

Potongan Kode 5.4 Inisialisasi fungsi loss dan optimizer

```

loss = DeciWatchLoss(w_denoise=1.0,
                     lamada=5.0,
                     smpl_model_dir="data/smpl/",
                     smpl=False)

optimizer = optim.Adam(model.parameters(), lr=0.001, amsgrad=True)

```

Setelah semua komponen siap, proses pelatihan dimulai dengan menggunakan kelas *Trainer*. Kelas ini mengelola seluruh proses pelatihan dan pengujian model, termasuk perhitungan loss, evaluasi pada data validasi, logging,

dan penyimpanan model. Metode run() dijalankan untuk memulai proses pelatihan berdasarkan konfigurasi yang telah ditentukan.

Potongan Kode 5.5 Pemanggilan kelas Trainer untuk pelatihan model

```
Trainer(train_dataloader=train_loader,
        test_dataloader=test_loader,
        model=model,
        loss=loss,
        writer=writer,
        optimizer=optimizer,
        cfg=cfg).run()
```

Tabel 5.1 Ringkasan Konfigurasi Pelatihan Model DeciWatch

Komponen	Nilai
Optimizer	Adam dengan AMSGrad
Learning rate	0,001
Jumlah epoch	20
Ukuran batch	512
Jumlah worker	4
Interval sampling	5 frame
Dimensi embedding encoder	128
Jumlah blok encoder	5
Dimensi embedding decoder	128
Jumlah blok decoder	6
Dropout	0,1
Fungsi aktivasi	Leaky ReLU
Ukuran jendela (<i>window size</i>)	51 frame
Metode interpolasi temporal	Linear
Representasi tubuh	3D (tanpa SMPL)

Dengan struktur pelatihan ini, seluruh pipeline mulai dari pemuatan data, inisialisasi model, perhitungan loss, hingga pelatihan dan evaluasi dilakukan

secara modular dan terintegrasi, sehingga memudahkan proses pengembangan serta eksperimen lanjutan.

5.1.4 Metode Perhitungan MPJPE dan Prosedur Pelatihan Model

Mean Per Joint Position Error (MPJPE) merupakan metrik utama yang digunakan untuk mengevaluasi kualitas estimasi pose 3D. MPJPE menghitung jarak rata-rata antara posisi titik sendi hasil prediksi model dan *ground truth* dalam ruang 3D. Semakin kecil nilai MPJPE, maka semakin akurat prediksi yang dihasilkan.

Perhitungan MPJPE dilakukan dengan menghitung *euclidean distance* antara setiap titik prediksi dan *ground truth* pada dimensi spasial (x, y, z), kemudian dirata-rata secara keseluruhan. Proses ini secara umum dituliskan sebagaimana pada rumus 2.9. Implementasi kode untuk perhitungan MPJPE dalam sistem ini adalah sebagai berikut:

Potongan Kode 5.6 Fungsi perhitungan MPJPE

```
def calculate_mpjpe(predicted, gt):
    mpjpe = torch.sqrt(((predicted - gt)**2).sum(dim=-1))
    mpjpe = mpjpe.mean(dim=-1)
    return mpjpe[~mpjpe.isnan()]
```

Prosedur pelatihan model dilakukan melalui kelas Trainer, yang menangani seluruh proses mulai dari pelatihan per epoch, evaluasi performa pada data validasi, penyimpanan model, serta pengaturan penurunan *learning rate* secara eksponensial. Loop pelatihan dilakukan selama 20 epoch, dan setiap epoch akan menghitung nilai loss serta melakukan evaluasi terhadap performa model.

Potongan kode berikut menunjukkan inti dari prosedur pelatihan model:

Potongan Kode 5.7 Loop pelatihan model DeciWatch

```
for epoch_num in range(0, 20):
    self.train()
    performance = self.evaluate()
    self.save_model(performance, epoch_num)

    # Penurunan learning rate
    self.lr *= 0.95
```

```
for param_group in self.optimizer.param_groups:
    param_group['lr'] *= 0.95
```

Selain itu, selama pelatihan berlangsung, nilai loss, waktu komputasi tiap *epoch*, serta nilai learning rate dicatat secara berkala dan divisualisasikan untuk analisis performa pelatihan secara keseluruhan.

5.1.5 Penyimpanan Model dan Evaluasi

Setelah setiap epoch, sistem menyimpan parameter model ke dalam file checkpoint.pth.tar melalui fungsi save_model(). Hal ini memungkinkan proses pelatihan untuk dilanjutkan (resumed) kapan pun diperlukan, serta memudahkan proses evaluasi performa model pada berbagai tahap pelatihan.

Fungsi save_model() akan menyimpan sebuah dictionary Python yang berisi:

- epoch: nomor epoch saat ini,
- state_dict: parameter bobot dari model DeciWatch,
- performance: skor evaluasi model (MPJPE atau metrik lain) pada epoch tersebut,
- optimizer: status parameter optimizer Adam, termasuk nilai learning rate dan momentum saat ini.

Potongan kode berikut menunjukkan cara penyimpanan model:

Potongan Kode 5.8 Penyimpanan checkpoint model setelah setiap epoch

```
def save_model(self, performance, epoch):
    save_dict = {
        'epoch': epoch,
        'state_dict': self.model.state_dict(),
        'performance': performance,
        'optimizer': self.optimizer.state_dict()
    }

    filename = os.path.join(self.logdir, 'checkpoint.pth.tar')
    torch.save(save_dict, filename)
```

Model disimpan dalam format file .pth.tar dengan nama checkpoint.pth.tar di dalam direktori log yang telah ditentukan sebelumnya (self.logdir). File ini bersifat

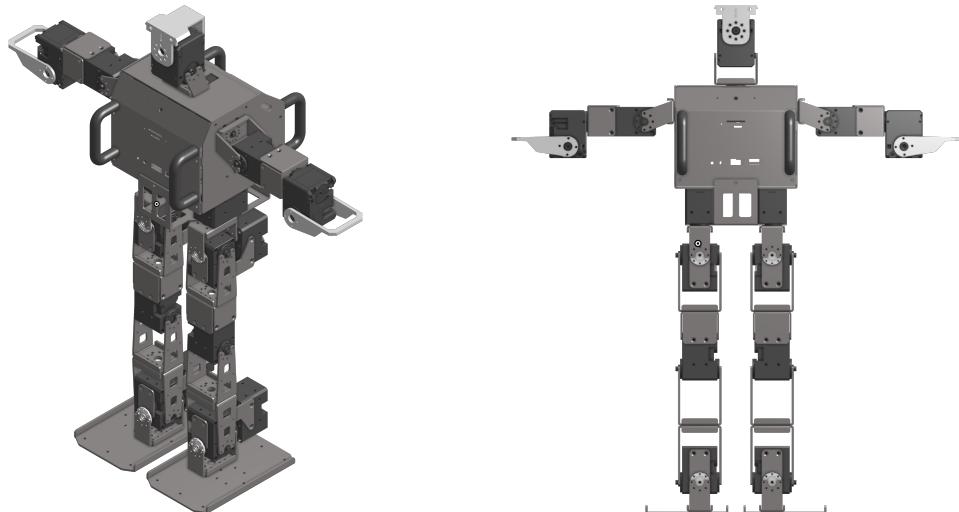
penting karena memungkinkan proses *resume training* apabila pelatihan terhenti, serta dapat digunakan kembali untuk keperluan inferensi atau evaluasi model di masa depan.

5.1.6 Modifikasi Hardware

Modifikasi ini diwujudkan dengan mengganti aktuator Dynamixel pada siku robot dengan servo tipe 2XL-430, yaitu servo ganda yang mendukung dua sumbu rotasi dalam satu unit. Servo ini digunakan untuk mengontrol gerakan *elbow flexion* dan *elbow twist* secara terintegrasi, sehingga lebih efisien secara ruang dan wiring.

Selain penggantian aktuator, dilakukan juga penyesuaian mekanik pada struktur lengan robot untuk menampung dimensi servo 2XL-430 yang lebih besar dibandingkan servo default. Modifikasi bentuk fisik ini dilakukan dengan mempertimbangkan keseimbangan berat dan distribusi massa pada kedua sisi lengan agar robot tetap stabil saat bergerak.

Gambar 5.1 menunjukkan tampilan fisik robot ROBOTIS-OP3 setelah proses modifikasi dilakukan. Terlihat posisi servo tambahan yang dipasang pada bagian siku dan penyusunan ulang rangka penopang servo untuk mengakomodasi dua derajat kebebasan siku.



(a) Tampilan Robotis OP3 - Sisi Isometrik (b) Tampilan Robotis OP3 - Sisi Depan

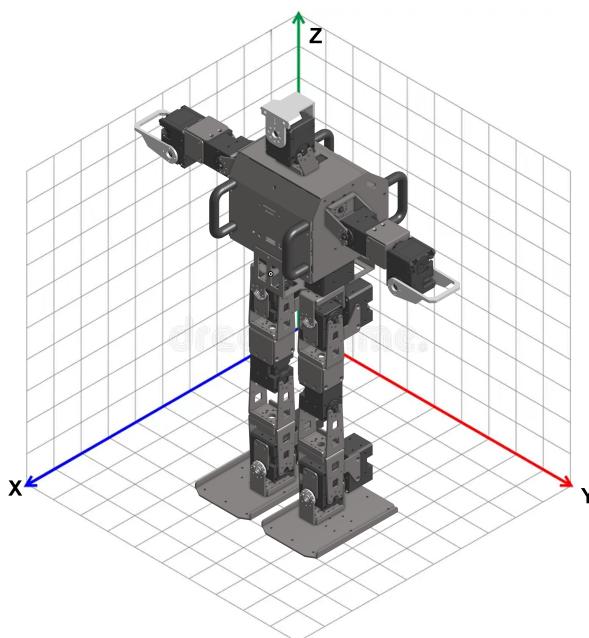
Gambar 5.1 Tampilan fisik robot ROBOTIS-OP3 setelah modifikasi

Setelah pemasangan, dilakukan kalibrasi awal untuk memastikan sudut nol

pada sumbu fleksi dan rotasi sesuai dengan orientasi default robot. Modifikasi fisik ini bertujuan agar robot mampu melakukan gerakan lengan yang lebih kompleks dan proporsional, sehingga mendukung representasi gerakan tari tradisional secara lebih realistik.

5.1.7 Implementasi pada Robot

5.1.7.1 Transformasi Koordinat



Gambar 5.2 Visualisasi sistem koordinat robot

Karena terdapat perbedaan struktur mekanik, panjang sambungan, jumlah derajat kebebasan, serta sistem koordinat antara lengan manusia dan lengan robot ROBOTIS-OP3, maka koordinat 3D hasil estimasi pose tidak dapat langsung diterapkan secara langsung ke sistem kontrol robot. Untuk itu, diperlukan proses transformasi koordinat agar data pose manusia dapat diadaptasikan ke dalam sistem koordinat robot.

Transformasi yang dilakukan pada penelitian ini tidak melibatkan perubahan skala atau translasi global, melainkan hanya penyesuaian orientasi sumbu agar sesuai dengan definisi sumbu pada robot ROBOTIS-OP3. Pada sistem koordinat hasil estimasi pose, orientasi sumbu mengikuti konvensi kamera dengan:

- Sumbu X sebagai kiri-kanan,

- Sumbu Y sebagai atas-bawah,
- Sumbu Z sebagai depan-belakang.

Sedangkan pada sistem koordinat robot ROBOTIS-OP3, orientasi sumbu diubah menjadi:

- Sumbu X menjadi maju-mundur,
- Sumbu Y menjadi kiri-kanan,
- Sumbu Z menjadi atas-bawah.

Proses transformasi dilakukan dengan menukar posisi sumbu sesuai kebutuhan agar sistem koordinat manusia dapat dipetakan dengan benar ke sistem koordinat robot.

Selain penyesuaian orientasi sumbu, transformasi juga dilakukan untuk memindahkan posisi skeleton pose ke titik referensi yang diketahui, yaitu pada pusat koordinat (0,0,0). Hal ini bertujuan untuk menghilangkan translasi global atau pergeseran posisi dari pose manusia, sehingga seluruh gerakan hanya dihitung relatif terhadap titik tengah yang berupa bahu. Dengan menjadikan skeleton berada di titik nol, sistem hanya memproses perubahan orientasi dan sudut pergerakan, tanpa mempengaruhi posisi absolut di ruang global. Hal ini sesuai dengan implementasi pada robot, di mana pergerakan dilakukan secara *in-place* dengan robot tidak berpindah tempat, hanya menggerakkan lengan dan kepala.

Hubungan transformasi koordinat untuk memetakan titik-titik tubuh manusia ke sistem koordinat robot dirumuskan sebagai berikut:

$$J'_i = \mathbf{A} \cdot (J_i - J_1), \quad i = 1, 2, 3 \quad (5.1)$$

$$J'_i = \mathbf{A} \cdot (J_i - J_4), \quad i = 4, 5, 6 \quad (5.2)$$

$$J'_i = \mathbf{A} \cdot (J_i - J_7), \quad i = 7, 8 \quad (5.3)$$

Dengan penjelasan sebagai berikut:

- J_i merupakan koordinat titik kunci tubuh manusia dalam sistem koordinat

dunia, sedangkan J'_i adalah koordinat hasil transformasi dalam sistem koordinat robot.

- Matriks \mathbf{A} adalah matriks rotasi yang dibentuk berdasarkan bidang segitiga yang menghubungkan bahu kiri (J_1), bahu kanan (J_4), dan panggul sebagai referensi.

Adapun penomoran titik kunci tubuh adalah sebagai berikut:

- J_1, J_2, J_3 : Titik pada lengan kiri, secara berurutan meliputi *shoulder left*, *elbow left*, dan *wrist left*.
- J_4, J_5, J_6 : Titik pada lengan kanan, secara berurutan meliputi *shoulder right*, *elbow right*, dan *wrist right*.
- J_7, J_8 : Titik pada kepala, secara berurutan meliputi *neck*, *head*.

Transformasi ini dilakukan untuk memastikan bahwa seluruh pergerakan tubuh manusia dapat dipetakan secara tepat ke dalam sistem koordinat robot, dengan mempertimbangkan perbedaan orientasi sumbu dan struktur anatomi antara manusia dan robot. Proses ini memungkinkan perhitungan sudut sendi dilakukan secara konsisten dan sesuai dengan konfigurasi kinematika robot.

Proses transformasi koordinat tidak hanya melibatkan pergeseran titik referensi, tetapi juga pembentukan matriks rotasi A yang berfungsi untuk menyelaraskan orientasi tubuh manusia dengan sistem koordinat robot. Matriks ini dibentuk berdasarkan tiga vektor ortonormal yang merepresentasikan arah sumbu lokal tubuh manusia, yaitu arah kiri-kanan, depan-belakang, dan atas-bawah.

Pembuatan matriks A dilakukan dengan langkah-langkah sebagai berikut:

1. Hitung vektor sumbu z sebagai arah tegak lurus tubuh dari panggul(J_9) ke pertengahan bahu:

$$z = \frac{(J_1 + J_4)}{2} - J_9 \quad (5.4)$$

Normalisasi vektor z menjadi vektor unit:

$$z_{\text{axis}} = \frac{z}{||z||} \quad (5.5)$$

2. Hitung vektor sumbu y sebagai vektor dari bahu kanan ke bahu kiri:

$$y = J_1 - J_4 \quad (5.6)$$

Normalisasi vektor y menjadi vektor unit:

$$y_{\text{axis}} = \frac{y}{\|y\|} \quad (5.7)$$

3. Hitung vektor sumbu x sebagai hasil perkalian silang antara y_{axis} dan z_{axis} :

$$x_{\text{axis}} = y_{\text{axis}} \times z_{\text{axis}} \quad (5.8)$$

Normalisasi vektor x menjadi vektor unit:

$$x_{\text{axis}} = \frac{x_{\text{axis}}}{\|x_{\text{axis}}\|} \quad (5.9)$$

4. Lakukan re-orthogonalization pada y_{axis} agar tetap tegak lurus terhadap z_{axis} dan x_{axis} :

$$y_{\text{axis}} = z_{\text{axis}} \times x_{\text{axis}} \quad (5.10)$$

Normalisasi kembali:

$$y_{\text{axis}} = \frac{y_{\text{axis}}}{\|y_{\text{axis}}\|} \quad (5.11)$$

5. Susun ketiga vektor ortonormal menjadi matriks rotasi A sebagai berikut:

$$\mathbf{A} = \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{bmatrix} \quad (5.12)$$

di mana $x_{\text{axis}} = [x_1, x_2, x_3]^T$, $y_{\text{axis}} = [y_1, y_2, y_3]^T$, dan $z_{\text{axis}} = [z_1, z_2, z_3]^T$.

Matriks \mathbf{A} ini digunakan untuk mentransformasikan koordinat dari sistem global (kamera) ke sistem lokal tubuh yang telah disesuaikan dengan orientasi robot. Proses ini memastikan bahwa orientasi sumbu tubuh manusia selaras dengan sistem koordinat robot, sehingga perhitungan sudut sendi dapat dilakukan secara konsisten

dan akurat.

5.1.7.2 Konversi Pose 3D ke Sudut Sendi

Setelah koordinat pose 3D diselaraskan ke dalam sistem koordinat robot, langkah selanjutnya adalah menghitung sudut-sudut sendi berdasarkan arah vektor lengan atas, lengan bawah dan kepala. Sudut yang dihitung mencakup dua sudut pada bahu (*pitch* dan *roll*), dua sudut pada siku (*twist* dan *flexion*), dan 3 sudut pada kepala (*pan*, *tilt*, *roll*). Proses ini memanfaatkan fungsi trigonometri invers dan operasi aljabar vektor.

Vektor lengan atas dan lengan bawah dapat didefinisikan sebagai:

$$\vec{V}_{\text{up}} = J'_{\text{elbow}} - J'_{\text{shoulder}} = \begin{bmatrix} v_x^{up} \\ v_y^{up} \\ v_z^{up} \end{bmatrix} \quad (5.13)$$

$$\vec{V}_{\text{low}} = J'_{\text{wrist}} - J'_{\text{elbow}} = \begin{bmatrix} v_x^{low} \\ v_y^{low} \\ v_z^{low} \end{bmatrix} \quad (5.14)$$

Sudut *Shoulder Pitch* pada lengan kiri dinotasikan sebagai θ_{LSPitch} , dihitung menggunakan fungsi $\arctan 2$ terhadap komponen $-x$ dan $-z$ dari vektor lengan atas. Penggunaan tanda negatif pada kedua komponen dilakukan untuk menyelaraskan arah rotasi dengan orientasi sumbu pada robot, sehingga pergerakan yang dihasilkan sesuai dengan konfigurasi servo pada lengan kiri. Rumus perhitungan θ_{LSPitch} adalah sebagai berikut:

$$\theta_{\text{LSPitch}} = \arctan 2(-v_x^{up}, -v_z^{up}) \quad (5.15)$$

Sudut *Shoulder Roll* pada lengan kiri dinotasikan sebagai θ_{LSRoll} . Perhitungan sudut ini digunakan untuk mengetahui seberapa jauh lengan kiri bergerak ke samping (ke atas atau ke bawah). Sudut dihitung menggunakan fungsi $\arctan 2$ sebagai berikut:

$$\theta_{LSRoll} = \arctan 2 \left(\sqrt{(v_x^{up})^2 + (v_z^{up})^2}, v_y^{up} \right) \quad (5.16)$$

Perhitungan sudut θ_{LEYaw} dan θ_{LERoll} memerlukan transformasi koordinat lanjutan, yaitu dari sistem koordinat bahu kiri ke sistem koordinat siku kiri. Transformasi ini diperlukan agar perhitungan sudut di siku dilakukan secara lokal terhadap orientasi lengan atas. Hal tersebut didapatkan dengan melakukan rotasi pada vektor lebgan bawah dengan dua matriks rotasi berurutan, yaitu matriks R_y dan matriks R_x untuk mengoreksi orientasi terhadap sudut $\theta_{LSPitch}$ dan θ_{LSRoll} .

$$\vec{V}'_{low} = R_x \cdot R_y \cdot \vec{v}_{low} \quad (5.17)$$

Transformasi ini menghasilkan vektor lengan bawah dalam kerangka koordinat baru yang telah disejajarkan dengan orientasi lengan atas, sehingga sudut $-\theta_{LEYaw}$ dan $-\theta_{LERoll}$ dapat dihitung dengan lebih akurat.

Matriks rotasi R_y dan R_x masing-masing dituliskan sebagai berikut:

$$R_y(-\theta_{LSPitch}) = \begin{bmatrix} \cos(-\theta_{LSPitch}) & 0 & \sin(-\theta_{LSPitch}) \\ 0 & 1 & 0 \\ -\sin(-\theta_{LSPitch}) & 0 & \cos(-\theta_{LSPitch}) \end{bmatrix} \quad (5.18)$$

$$R_x(-\theta_{LSRoll}) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(-\theta_{LSRoll}) & \sin(-\theta_{LSRoll}) \\ 0 & -\sin(-\theta_{LSRoll}) & \cos(-\theta_{LSRoll}) \end{bmatrix} \quad (5.19)$$

Tanda negatif pada setiap sudut digunakan untuk menyesuaikan arah rotasi dari sistem koordinat manusia ke sistem koordinat robot, agar sesuai dengan pergerakan fisik servo pada ROBOTIS-OP3.

Sudut *Elbow Twist* pada lengan kiri dinotasikan sebagai $\theta_{LETwist}$, dihitung dari komponen vektor lengan bawah hasil rotasi. Perhitungan dilakukan menggunakan fungsi $\arctan 2$ sebagai berikut:

$$\theta_{LETwist} = \arctan 2 \left(v_x'^{low}, v_z'^{low} \right) \quad (5.20)$$

Sedangkan sudut *Elbow Flexion* pada lengan kiri dinotasikan sebagai θ_{LEFlex} , dihitung berdasarkan sudut antara vektor lengan atas dan vektor lengan bawah dengan menggunakan rumus dot product sebagai berikut:

$$\theta_{LEFlex} = \arctan 2 \left(-\sqrt{(v_x'^{low})^2 + (v_z'^{low})^2}, v_y'^{low} \right) \quad (5.21)$$

Perhitungan ini digunakan untuk menentukan seberapa besar lengan bawah menekuk (*flexion*) dan berputar (*twist*) relatif terhadap lengan atas dalam kerangka lokal yang telah disejajarkan.

Implementasi dari proses ini dilakukan pada kode program dengan memanfaatkan operasi vektor sederhana, seperti perkalian silang, dot product, dan normalisasi, yang diikuti dengan fungsi trigonometri arctan 2 dan arccos. Langkah ini memastikan sudut yang diperoleh sesuai dengan batas pergerakan fisik servo pada robot humanoid ROBOTIS-OP3.

Potongan Kode 5.9 Implementasi perhitungan sudut sendi sesuai rumus

```
aup_L = J_elbow - J_shoulder
alow_L = J_wrist - J_elbow

LShoulderPitch = np.arctan2(-aup_L[0], -aup_L[2]) # (X, Z)
LShoulderRoll = np.arctan2(np.sqrt(aup_L[0]**2 + aup_L[2]**2),
                           aup_L[1])
                     )

Ry = np.array([
    [np.cos(-LShoulderPitch), 0, np.sin(-LShoulderPitch)],
    [0, 1, 0],
    [-np.sin(-LShoulderPitch), 0, np.cos(-LShoulderPitch)]
])

Rx = np.array([
    [1, 0, 0],
    [0, np.cos(-LShoulderRoll), np.sin(-LShoulderRoll)],
    [0, -np.sin(-LShoulderRoll), np.cos(-LShoulderRoll)]
])
```

```

alow_L_prime = Rx @ Ry @ alow_L
LElbowPitch = np.arctan2(-alow_L_prime[0], -alow_L_prime[2])
LElbowRoll = np.arctan2(
    -np.sqrt(alow_L_prime[0]**2 + alow_L_prime[2]**2),
    alow_L_prime[1]
)

ElbowFlexion = -np.degrees(angle_between_vectors(auJ_L, alow_L))

```

Begitu juga dengan bagian kepala, untuk mendapatkan sudut orientasi kepala, pertama-tama didefinisikan vektor kepala sebagai selisih antara posisi titik kepala dan titik leher:

$$\vec{V}_{\text{head}} = J'_{\text{head}} - J'_{\text{neck}} = \begin{bmatrix} v_x^{\text{head}} \\ v_y^{\text{head}} \\ v_z^{\text{head}} \end{bmatrix} \quad (5.22)$$

Sudut *Head Pan* dinotasikan sebagai θ_{HPan} , yang merepresentasikan rotasi mendatar kepala (gerakan menengok ke kiri dan kanan). Sudut ini dihitung menggunakan fungsi arctan 2 terhadap komponen y dan x dari vektor kepala:

$$\theta_{\text{HPan}} = \arctan 2 \left(v_y^{\text{head}}, v_x^{\text{head}} \right) \quad (5.23)$$

Sudut *Head Roll* dinotasikan sebagai θ_{Hroll} , yang menunjukkan kemiringan kepala ke samping (memiringkan kepala kiri atau kanan). Perhitungan dilakukan dengan:

$$\theta_{\text{Hroll}} = -\arctan 2 \left(v_y^{\text{head}}, v_z^{\text{head}} \right) \quad (5.24)$$

Sedangkan sudut *Head Tilt* atau *Head Pitch*, dinotasikan sebagai θ_{HTilt} , digunakan untuk mendeskripsikan gerakan kepala mendongak atau menunduk. Perhitungan dilakukan sebagai berikut:

$$\theta_{\text{HTilt}} = \arctan 2 \left(v_x^{\text{head}}, v_z^{\text{head}} \right) \quad (5.25)$$

Perhitungan ketiga sudut ini dilakukan dalam kerangka lokal tubuh yang telah disesuaikan dengan orientasi robot, sehingga pergerakan kepala yang dihasilkan dapat sesuai dengan arah rotasi servo pada bagian kepala ROBOTIS-OP3.

Potongan Kode 5.10 Implementasi perhitungan sudut kepala

```
ahead = J_head - J_neck # vektor kepala
HeadPan = (np.arctan2(ahead[1], ahead[0]))
HeadRoll = -(np.arctan2(ahead[1], ahead[2]))
HeadTilt = (np.arctan2(ahead[0], ahead[2]))
```

5.1.8 Penyimpanan Gerakan dalam Format JSON

Setelah seluruh sudut sendi berhasil dihitung untuk setiap frame pose 3D, data gerakan kemudian disimpan dalam format berkas JSON. Penyimpanan ini bertujuan untuk memudahkan proses pemutaran ulang gerakan pada robot, serta sebagai format pertukaran data yang sederhana dan mudah diakses. Pada tahap ini, setiap frame hasil konversi disusun dalam struktur data Python *dictionary*, dengan penamaan kunci yang sesuai dengan nama sendi pada robot. Contoh pemetaan sudut sendi ke nama kunci JSON dapat dilihat berikut:

Potongan Kode 5.11 Pemetaan hasil sudut ke format JSON

```
frame_result = {
    "r_sho_pitch": angles_R["ShoulderPitch"],
    "l_sho_pitch": angles_L["ShoulderPitch"],
    "r_sho_roll": angles_R["ShoulderRoll"],
    "l_sho_roll": angles_L["ShoulderRoll"],
    "r_el_pitch": angles_R["ElbowTwist"],
    "l_el_pitch": angles_L["ElbowTwist"],
    "r_el_roll": angles_R["ElbowFlexion"],
    "l_el_roll": angles_L["ElbowFlexion"],
    "head_pan": angles_head["HeadPan"],
    "head_pitch": angles_head["HeadTilt"],
    "head_roll": angles_head["HeadRoll"],
}

results[str(frame_idx)] = frame_result
```

Setelah seluruh frame diproses, data hasil konversi disimpan dalam berkas JSON menggunakan fungsi json.dump(). Berikut contoh kode penyimpanan:

Potongan Kode 5.12 Penyimpanan hasil gerakan dalam file JSON

```
output_file = "/home/luthfai/Devel/skripsi/pose_angles2.json"
with open(output_file, "w") as f:
    json.dump(results, f, indent=2)
```

Format file JSON ini berisi daftar sudut sendi per frame dalam satuan derajat, yang kemudian dapat digunakan sebagai masukan untuk perintah gerakan pada robot humanoid ROBOTIS-OP3. Contoh isi file JSON hasil penyimpanan dapat dilihat pada Potongan Data berikut:

Potongan Kode 5.13 Contoh data sudut sendi dalam format JSON

```
{
    "0": {
        "r_sho_pitch": -7.799538951688305,
        "l_sho_pitch": -10.71019173677621,
        "r_sho_roll": -46.40881222111872,
        "l_sho_roll": 48.35393974799908,
        "r_el_pitch": 121.56508280420996,
        "l_el_pitch": -103.9285816547141,
        "r_el_roll": -32.71889721964388,
        "l_el_roll": -17.327496671902992,
        "head_pan": -15.129533767700195,
        "head_pitch": -2.2988436222076416,
        "head_roll": 1.3416086435317993
    },
}
```

Data ini menunjukkan sudut sendi dalam satuan derajat untuk frame ke-0 pose 3D, yang kemudian dapat diproses sebagai perintah gerakan robot.

5.1.9 Eksekusi Gerakan pada Robot

Tahap akhir dalam sistem ini adalah menjalankan pergerakan lengan robot berdasarkan data sudut sendi yang telah dikonversi dan disimpan dalam berkas JSON. Proses ini dilakukan dengan memuat data sudut untuk setiap frame, mengonversi

sudut derajat menjadi nilai posisi servo Dynamixel, kemudian mengirimkan perintah posisi ke semua aktuator secara sinkron.

File JSON yang berisi urutan sudut sendi dibaca dan diurutkan berdasarkan nomor frame. Data sudut kemudian dikonversi menggunakan fungsi *deg_to_dxl* untuk menghasilkan nilai posisi servo yang sesuai dengan protokol pengendalian Dynamixel. Hasil konversi disimpan dalam variabel *joint_trajectories* yang memuat data untuk seluruh frame.

Pada tahap playback, program melakukan iterasi melalui setiap frame. Nilai posisi servo dikirimkan ke aktuator menggunakan metode *write4ByteTxRx()* atau melalui mekanisme sinkronisasi paket (*sync write*), sehingga semua sendi bergerak serempak sesuai dengan urutan gerakan yang diinginkan. Kecepatan eksekusi dapat diatur menggunakan fungsi *sleep()* untuk mengatur interval antar frame.

Contoh kode Python yang digunakan dalam proses eksekusi gerakan robot adalah sebagai berikut:

Potongan Kode 5.14 Kode eksekusi playback gerakan pada robot humanoid

```
for frame_idx in range(len(angle_data)):
    for joint_name, joint_id in JOINTS.items():
        pos = joint_trajectories[joint_name][frame_idx]

        # Konversi posisi menjadi 4 byte
        param_goal_position = [
            DXL_LOBYTE(DXL_LOWORD(pos)),
            DXL_HIBYTE(DXL_LOWORD(pos)),
            DXL_LOBYTE(DXL_HIWORD(pos)),
            DXL_HIBYTE(DXL_HIWORD(pos))
        ]
        groupSyncWrite.addParam(joint_id,
                               bytarray(param_goal_position)
                               )
    groupSyncWrite.txPacket()
    groupSyncWrite.clearParam()

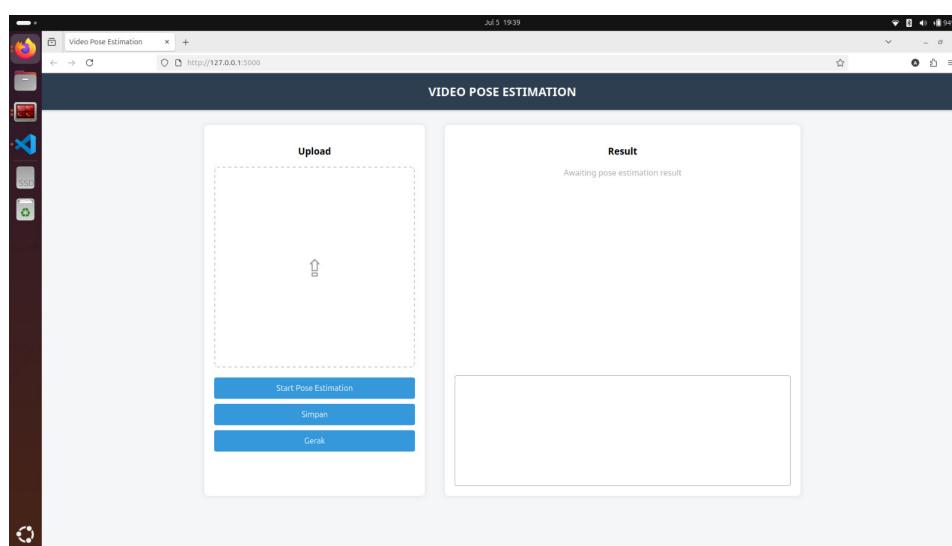
    time.sleep(1.0 / 30.0)
```

Proses ini memastikan robot dapat mereplikasi urutan gerakan hasil estimasi pose secara konsisten dengan kecepatan yang telah ditentukan.

5.2 Implementasi Website

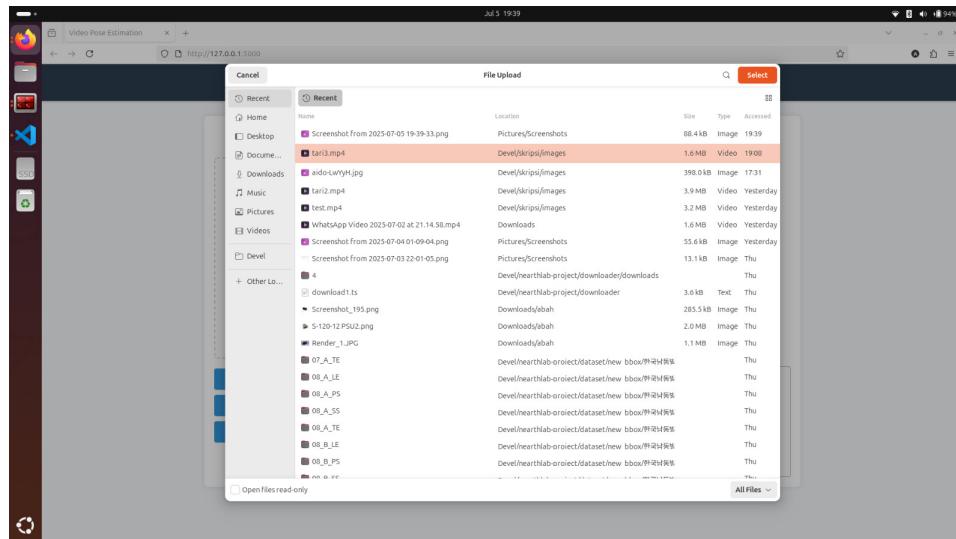
Website yang dikembangkan pada penelitian ini berfungsi sebagai antarmuka utama untuk memproses video gerakan tari, melakukan estimasi pose 3D, menyimpan hasil konversi sudut, serta mengirim perintah gerakan ke robot humanoid. Sistem website dikembangkan menggunakan framework Flask, sedangkan proses inferensi model pose estimation dijalankan secara terpisah menggunakan Uvicorn dan FastAPI. Pemisahan ini dilakukan karena lingkungan Python untuk model estimasi membutuhkan pustaka tambahan seperti TensorFlow dan Torch yang berbeda versi dengan dependensi Flask.

Saat aplikasi pertama kali dijalankan, pengguna akan diarahkan pada halaman utama yang menampilkan judul Video Pose Estimation di bagian atas. Di sisi kiri halaman terdapat area unggah video, dengan ikon upload berukuran besar sebagai pusat interaksi. Bagian ini memungkinkan pengguna memilih video gerakan tari tradisional yang akan diproses. Setelah file dipilih, nama video akan muncul pada area log di bagian kanan bawah halaman, sehingga pengguna dapat memastikan bahwa file berhasil dimuat. Tampilan awal antarmuka website dapat dilihat pada Gambar 5.3.



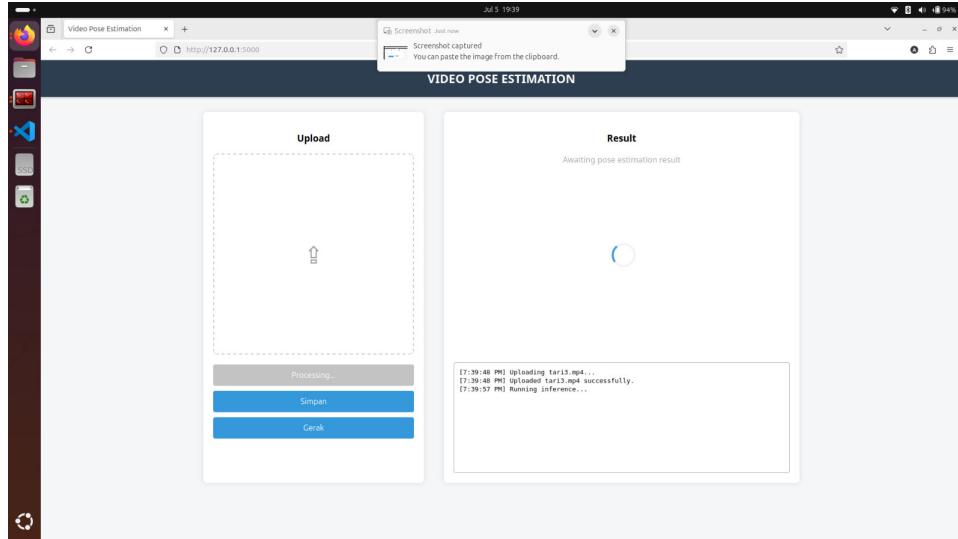
Gambar 5.3 Tampilan awal website sebelum interaksi pengguna

Pengguna dapat mengunggah video gerakan tari tradisional melalui tombol unggah. Setelah video dipilih, sistem akan menampilkan nama file pada area log. Contoh tampilan proses pemilihan video ditunjukkan pada Gambar 5.4.



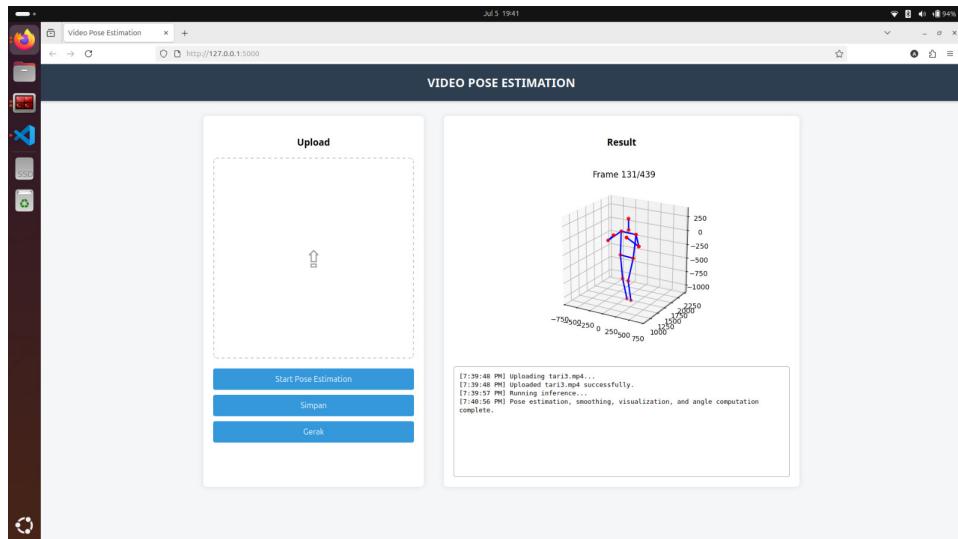
Gambar 5.4 Proses upload dan pemilihan video

Setelah video berhasil diunggah, pengguna dapat menekan tombol Start Pose Estimation yang terletak di bawah area unggah video. Ketika tombol ini ditekan, sistem akan melakukan proses estimasi pose 3D pada video secara bertahap. Proses ini dijalankan oleh server Uvicorn yang secara independen menangani komputasi model TensorFlow. Selama proses inferensi berlangsung, sistem menampilkan progress log dan visualisasi kerangka pose yang diperbarui secara real-time. Gambar 5.5 menunjukkan contoh tampilan proses estimasi pose.



Gambar 5.5 Tampilan proses estimasi pose 3D sedang berjalan

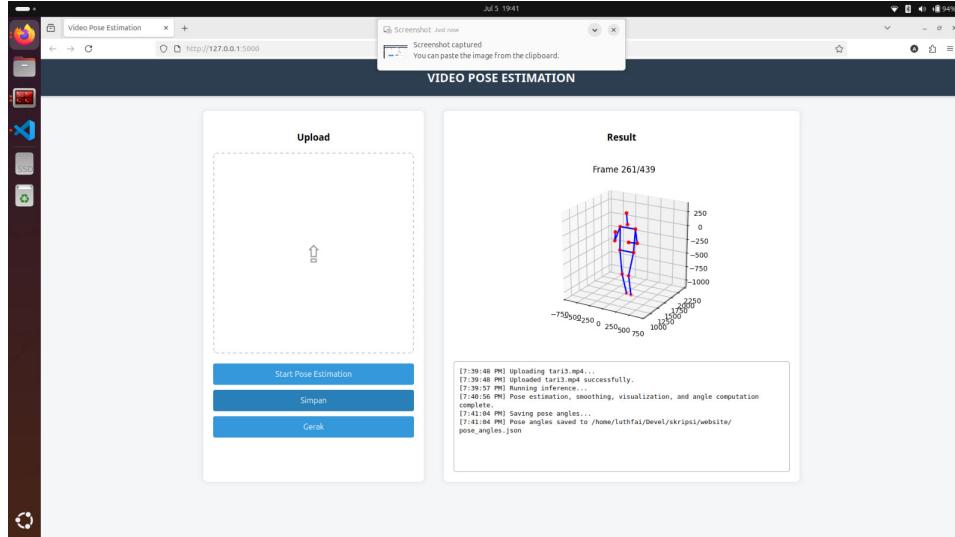
Setelah estimasi selesai, hasil prediksi pose akan divisualisasikan dalam bentuk skeleton 3D pada sisi kanan halaman. Visualisasi ini dilengkapi keterangan frame saat ini dan total jumlah frame, sehingga pengguna dapat memantau hasil pose estimation secara lebih detail. Contoh hasil visualisasi ditunjukkan pada Gambar 5.6.



Gambar 5.6 Hasil visualisasi pose 3D setelah proses estimasi selesai

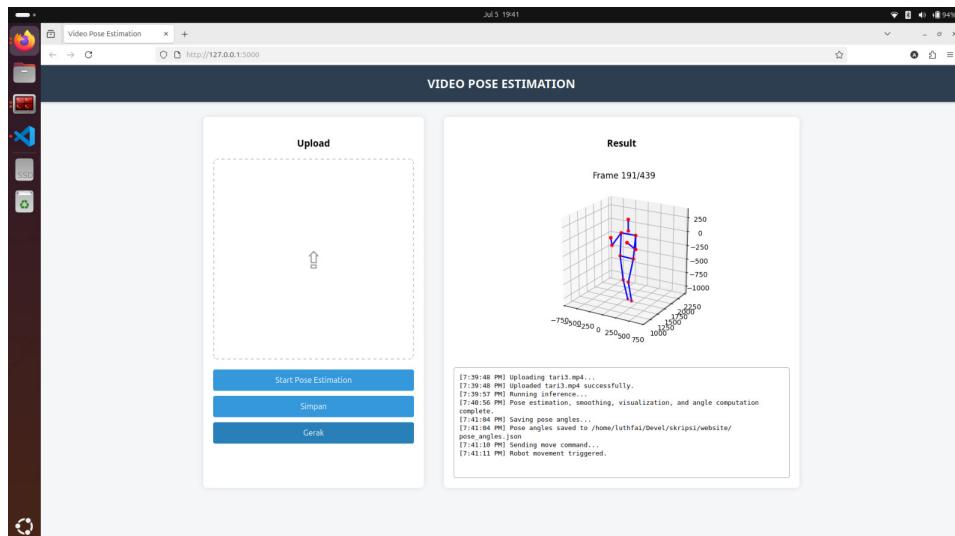
Di bagian bawah area unggah video, terdapat tombol Simpan dan Gerak. Tombol Simpan digunakan untuk menyimpan hasil konversi sudut sendi dalam format JSON. File JSON yang dihasilkan berisi nilai sudut sendi untuk setiap frame, yang kemudian dapat digunakan sebagai input gerakan robot. Setelah penyimpanan

selesai, sistem akan menampilkan notifikasi log yang menginformasikan direktori penyimpanan file. Contoh tampilan proses penyimpanan ditunjukkan pada Gambar 5.7.



Gambar 5.7 Tampilan saat proses penyimpanan data sudut sendi

Tombol Gerak digunakan untuk memulai eksekusi playback gerakan pada robot humanoid ROBOTIS-OP3. Saat tombol ini ditekan, sistem akan membaca file JSON yang telah disimpan sebelumnya dan mengirimkan perintah posisi servo secara sinkron melalui protokol Dynamixel. Gambar 5.8 menunjukkan tampilan antarmuka ketika proses gerakan robot dijalankan.



Gambar 5.8 Tampilan saat proses playback gerakan robot berjalan

Dengan desain ini, seluruh proses mulai dari unggah video, estimasi pose, penyimpanan sudut sendi, hingga playback gerakan dapat dilakukan secara terpadu melalui satu antarmuka web yang sederhana namun lengkap. Pemisahan server Flask dan Uvicorn memungkinkan sistem berjalan lebih stabil karena proses komputasi model TensorFlow tidak memengaruhi kinerja web server utama.

5.3 Pengujian Fungsional Sistem

Pengujian fungsional dirancang untuk memastikan seluruh fitur utama pada sistem dapat berjalan sesuai dengan spesifikasi yang telah ditentukan. Metode pengujian menggunakan pendekatan *black-box testing*, dengan skenario uji yang disusun berdasarkan masing-masing fitur yang terdapat pada sistem.

Pada tahap ini, pengujian fungsional aplikasi tidak dilakukan secara langsung oleh penulis, melainkan direncanakan untuk dieksekusi oleh anggota tim robotik yang bertugas mengelola implementasi pada perangkat robot. Pengujian ini mencakup alur penggunaan aplikasi, mulai dari input video hingga eksekusi gerakan pada robot ROBOTIS-OP3.

Tabel 5.2 berikut menyajikan rencana skenario pengujian fungsional yang telah disusun.

Tabel 5.2 Rencana Pengujian Fungsional Sistem

No	Fitur	Skenario Pengujian	Hasil yang Diharapkan
1	Upload Video	Pengguna memilih dan mengunggah file video gerakan tari tradisional.	File berhasil diunggah dan nama file muncul di area log.
2	Estimasi Pose dan Filtering	Pengguna menekan tombol <i>Start Pose Estimation</i> .	Proses estimasi dan filtering berjalan hingga selesai tanpa error.
3	Tampilan Estimasi Pose	Sistem menampilkan skeleton 3D hasil estimasi setiap frame.	Visualisasi skeleton muncul sesuai urutan frame video.

No	Fitur	Skenario Pengujian	Hasil yang Diharapkan
4	Status Log	Sistem mencatat status proses pada area log.	Log status muncul pada setiap tahap proses.
5	Penyimpanan Data	Pengguna menekan tombol <i>Simpan</i> setelah estimasi pose selesai.	File JSON disimpan dalam direktori output.
6	Playback Robot	Pengguna menekan tombol <i>Gerak</i> untuk menjalankan robot.	Robot bergerak sesuai data sudut hasil estimasi pose.

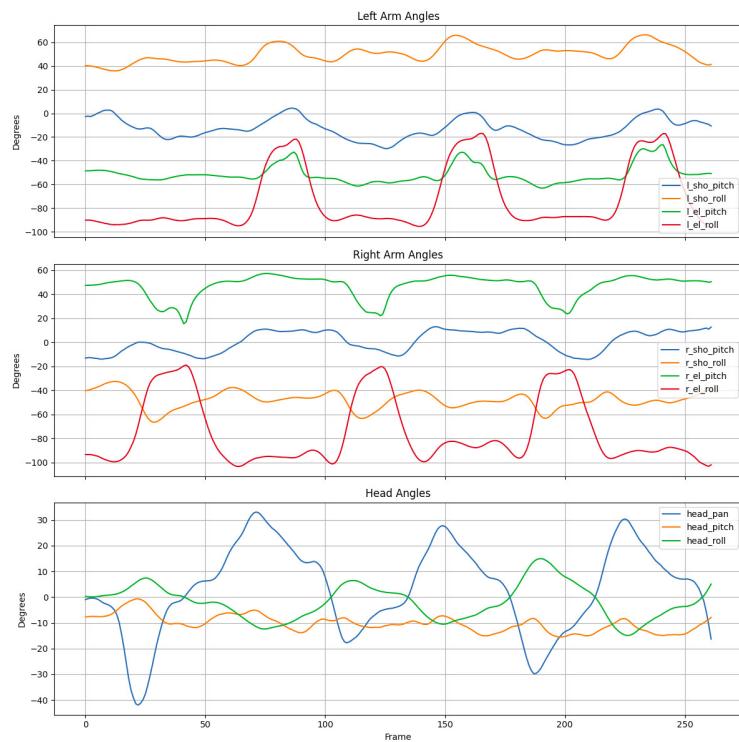
BAB VI. HASIL DAN PEMBAHASAN

6.1 Hasil dan Pembahasan Sistem

6.1.1 Hasil dan Pembahasan Implementasi pada Robot

Pada tahap ini dilakukan implementasi sistem ke dalam robot humanoid ROBOTIS-OP3 untuk mereplikasi gerakan manusia berdasarkan hasil estimasi pose yang telah diolah sebelumnya. Proses ini melibatkan beberapa tahapan penting, mulai dari konversi data pose 3D menjadi sudut servo, validasi area pergerakan robot, hingga eksekusi gerakan secara fisik. Berikut adalah pembahasan hasil implementasi pada masing-masing tahap.

6.1.1.1 Konversi Koordinat ke Sudut Servo



Gambar 6.1 Grafik perubahan sudut sendi robot per frame pada lengan kiri, lengan kanan, dan kepala

Gambar 6.1 menunjukkan hasil visualisasi perubahan sudut sendi (servo) pada robot humanoid untuk mengontrol pergerakan lengan kiri, lengan kanan, dan kepala dalam satu sekuens gerakan tari tradisional. Tari tersebut melakukan gerakan

menekuk lengan secara bergantian dengan kepala menengok ke kanan dan ke kiri secara bergantian.

Grafik bagian atas memperlihatkan *Left Arm Angles*, yaitu perubahan sudut pada *Shoulder Pitch*, *Shoulder Roll*, *Elbow Pitch*, dan *Elbow Roll* di lengan kiri. Kurva menunjukkan variasi sudut yang mencerminkan pergerakan lengan naik, turun, dan rotasi sepanjang gerakan.

Grafik bagian tengah menampilkan *Right Arm Angles*, dengan pola perubahan sudut serupa pada lengan kanan. Terlihat perubahan signifikan pada sudut *Elbow Pitch* dan *Elbow Roll*, terutama pada rentang frame 50 hingga 150 yang menandai fase gerakan tangan intens.

Grafik bagian bawah memperlihatkan *Head Angles*, yaitu perubahan sudut kepala pada tiga sumbu: *Head Pan* (rotasi mendatar), *Head Pitch* (mendongak atau menunduk), dan *Head Roll* (kemiringan samping). Meskipun variasinya lebih kecil dibandingkan pergerakan lengan, grafik ini tetap menunjukkan dinamika gerakan kepala sesuai dengan input pose.

Secara keseluruhan, visualisasi ini menunjukkan bahwa sistem berhasil mengonversi data pose 3D menjadi perintah sudut servo yang halus, kontinu, dan sesuai dengan variasi gerakan tari. Pola sudut yang stabil dan tidak fluktuatif secara ekstrem mendukung proses eksekusi gerakan robot secara natural dan aman.

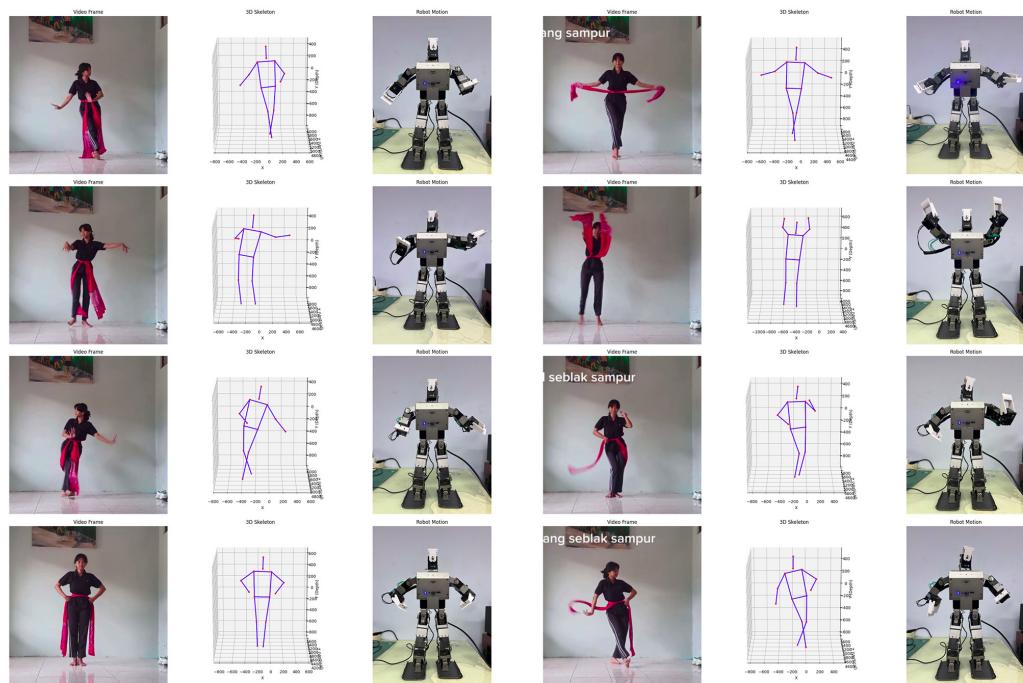
6.1.2 Motion Imitation Pada Robot

Pada hasil *motion imitation*, dilakukan dua jenis percobaan untuk menguji sistem. Percobaan pertama dilakukan dengan input gambar berpose khusus untuk menguji cakupan pergerakan robot. Input ini meliputi berbagai variasi gerakan seperti tangan merentang ke samping (*T-pose*), tangan ke depan, tangan ke atas, dan kombinasi lainnya yang mencakup area maksimal pergerakan lengan dan kepala robot. Percobaan ini bertujuan untuk mengetahui seberapa baik sistem dapat memetakan berbagai pose manusia ke dalam ruang gerak (range of motion) robot ROBOTIS-OP3.

Percobaan kedua dilakukan pada gerakan tari tradisional, dengan rangkaian gerakan yang lebih dinamis dan variatif. Hasil percobaan ditunjukkan dengan

menampilkan tiga komponen utama secara berdampingan dalam satu gambar, yaitu frame asli, hasil estimasi rangka 3D, dan gerakan robot yang menirukan pose. Gambar berikut memperlihatkan contoh rangkaian gerakan tari tradisional yang diproses oleh sistem.

Pada gambar bagian kiri ditampilkan frame video input yang menjadi dasar proses estimasi pose. Bagian tengah menunjukkan hasil prediksi koordinat 3D yang divisualisasikan dalam bentuk rangka, sedangkan bagian kanan memperlihatkan robot humanoid ROBOTIS-OP3 yang menjalankan pose hasil konversi sudut sendi.



Gambar 6.2 Visualisasi motion imitation pada gerakan tari tradisional: frame video asli, rangka hasil estimasi 3D, dan gerakan robot

Pada Gambar 6.2, ditampilkan hasil percobaan *motion imitation* pada gerakan tari tradisional, yaitu tari Denok Semarangan. Robot berhasil menirukan gerakan tari dengan cukup baik, terutama pada pergerakan tangan dan kepala yang menjadi fokus utama dalam penelitian ini. Posisi lengan dan kepala robot mengikuti pola pose dari data rangka secara real-time, dengan pergerakan yang halus dan stabil antar frame.



Gambar 6.3 Visualisasi motion imitation untuk cakupan gerakan: frame video asli, rangka hasil estimasi 3D, dan gerakan robot

Berdasarkan Gambar 6.3, dapat dilihat bahwa robot mampu mengikuti

berbagai variasi gerakan dengan baik. Pada pengujian cakupan gerakan, sistem berhasil memetakan gerakan lengan seperti merentang ke samping, mengangkat ke atas, dan mengarah ke depan tanpa melebihi batas pergerakan mekanis robot. Gerakan yang dihasilkan tetap stabil dan tidak terdapat pergerakan yang melebihi limit servo, sehingga sistem berhasil menjaga keamanan aktuator. Hal ini menunjukkan bahwa proses *motion retargeting* telah berhasil menyesuaikan koordinat tubuh manusia dengan batas kemampuan robot ROBOTIS-OP3 secara efektif.

Walaupun terdapat perbedaan proporsi anatomi antara manusia dan robot, sistem tetap mampu menghasilkan gerakan yang menyerupai tari tradisional secara visual. Hasil ini menunjukkan bahwa pipeline mulai dari input video, estimasi rangka 3D, konversi sudut sendi, hingga eksekusi gerakan robot berjalan dengan baik secara *end-to-end*.

6.1.3 Hasil dan Pembahasan Fungsional Sistem

Pengujian fungsional sistem dilakukan untuk memastikan bahwa seluruh fitur utama berjalan sesuai spesifikasi yang telah dirancang. Pengujian telah dilakukan oleh anggota tim robotik terhadap aplikasi yang telah dikembangkan. Terdapat 6 fitur utama dengan total 6 skenario pengujian, dan seluruh skenario berhasil dijalankan tanpa kendala. Persentase keberhasilan pengujian fungsional sistem adalah $\frac{6}{6} \times 100\% = 100\%$. Dengan demikian, seluruh fitur pada sistem dinyatakan telah memenuhi kebutuhan fungsional sesuai perancangan.

Tabel 6.1 berikut menyajikan hasil lengkap pengujian fungsional sistem.

Tabel 6.1 Hasil Pengujian Fungsional Sistem

No	Fitur	Skenario Pengujian	Hasil Pengujian	Kesimpulan
1	Upload Video	Pengguna mengunggah file video gerakan tari tradisional	File video berhasil diunggah	Berhasil

No	Fitur	Skenario Pengujian	Hasil Pengujian	Kesimpulan
2	Estimasi Pose dan Filtering	Pengguna menekan tombol <i>Start Pose Estimation</i>	<i>Pose estimation</i> berhasil dijalankan	Berhasil
3	Tampilan Estimasi Pose	Sistem menampilkan skeleton 3D pada setiap frame video	Sistem menampilkan skeleton 3D dari video (frame awal hingga frame akhir)	Berhasil
4	Status Log	Sistem mencatat proses pada area log setiap tahapannya	Sistem menampilkan setiap proses yang berlangsung	Berhasil
5	Penyimpanan Data	Pengguna menyimpan hasil estimasi ke dalam file JSON	File estimasi dari setiap frame didapatkan dalam bentuk JSON	Berhasil
6	Playback Robot	Pengguna menekan tombol <i>Gerak</i> untuk menjalankan robot	Robot bergerak sesuai hasil estimasi proses sebelumnya	Berhasil

Dari hasil pengujian fungsional, dapat disimpulkan bahwa sistem telah memenuhi seluruh kebutuhan fungsi utama yang dirancang. Proses mulai dari input video, estimasi pose, penyimpanan data, hingga eksekusi gerakan pada robot ROBOTIS-OP3 berhasil dijalankan dengan baik tanpa kendala teknis.

Keberhasilan ini menunjukkan bahwa sistem dapat digunakan sebagai media

representasi gerakan tari tradisional secara otomatis melalui robot humanoid. Implementasi *black-box testing* berhasil memvalidasi bahwa alur aplikasi sesuai dengan rancangan antarmuka dan kebutuhan operasional.

6.1.4 Hasil dan Pembahasan Performa Model

Bagian ini menjelaskan hasil pengujian performa model DeciWatch, mencakup proses training, perbandingan performa antar model, serta visualisasi hasil estimasi pose dan pergerakan sudut sendi.

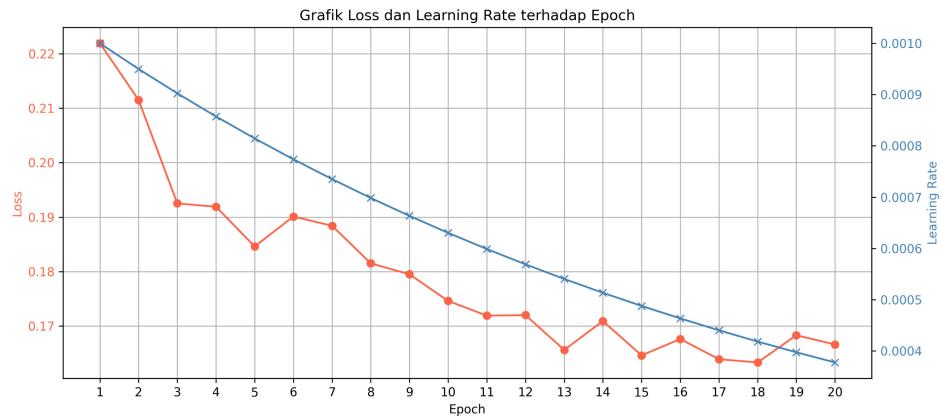
6.1.4.1 Proses Pelatihan dan Evaluasi Model

Model DeciWatch dilatih menggunakan dataset AIST++ dalam dua sesi training dengan total 45 epoch. Tujuan pelatihan adalah untuk meningkatkan akurasi estimasi pose 3D secara temporal dengan mengoreksi prediksi awal dari model baseline. Pelatihan dilakukan dengan GPU NVIDIA RTX 3060 Mobile. Sesi pertama berlangsung selama 20 epoch, sedangkan sesi kedua dilakukan dari epoch ke-21 hingga ke-45 dengan memuat parameter dari sesi sebelumnya.

Selama proses training, dicatat nilai loss dan Mean Per Joint Position Error (MPJPE). MPJPE input berasal dari prediksi awal model SPIN (yang tetap konstan karena tidak dilatih ulang), sedangkan MPJPE output dihasilkan dari model DeciWatch setelah proses filtering temporal.

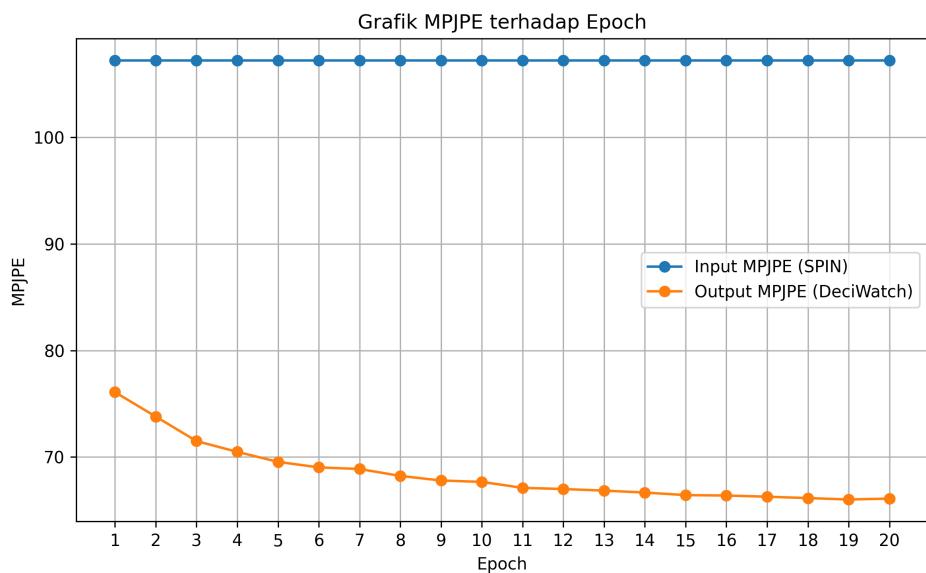
1. Sesi Pertama (Epoch 1–20)

Pada sesi pertama, model dilatih selama 20 epoch. Gambar 6.4 menunjukkan penurunan loss secara konsisten seiring dengan penurunan *learning rate* menggunakan skema eksponensial decay sebesar 0,95 per epoch.



Gambar 6.4 Perbandingan nilai loss dan learning rate terhadap epoch (1–20)

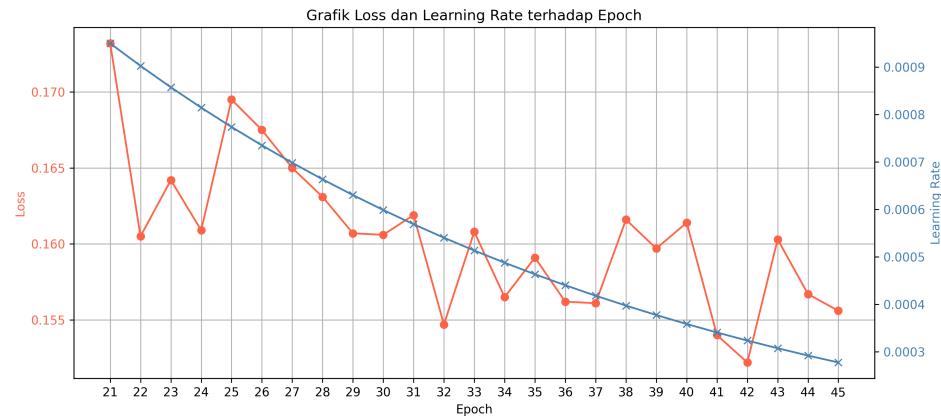
Gambar 6.5 memperlihatkan tren penurunan MPJPE output secara bertahap, sementara MPJPE input tetap konstan. Hal ini menunjukkan bahwa DeciWatch mulai berhasil mengurangi error prediksi pada sesi pertama.



Gambar 6.5 Perbandingan MPJPE antara input dan output pada sesi pelatihan pertama

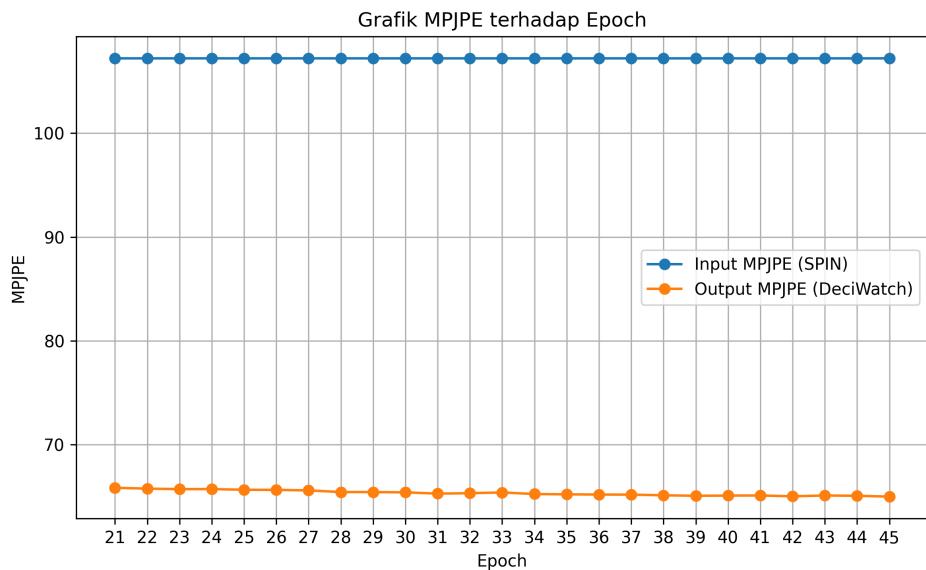
2. Sesi Kedua (Epoch 21–45)

Pada sesi kedua, proses pelatihan dilanjutkan dengan memuat parameter dari sesi pertama. Gambar 6.6 menunjukkan bahwa penurunan loss tetap berlanjut, meskipun laju penurunannya lebih kecil karena *learning rate* terus berkurang.



Gambar 6.6 Perbandingan nilai loss dan learning rate terhadap epoch (21–45)

Gambar 6.7 memperlihatkan penurunan MPJPE output yang konsisten hingga epoch ke-45. Ini menunjukkan bahwa model semakin mampu mengoreksi kesalahan prediksi secara temporal.



Gambar 6.7 Perbandingan MPJPE antara input dan output pada sesi pelatihan kedua

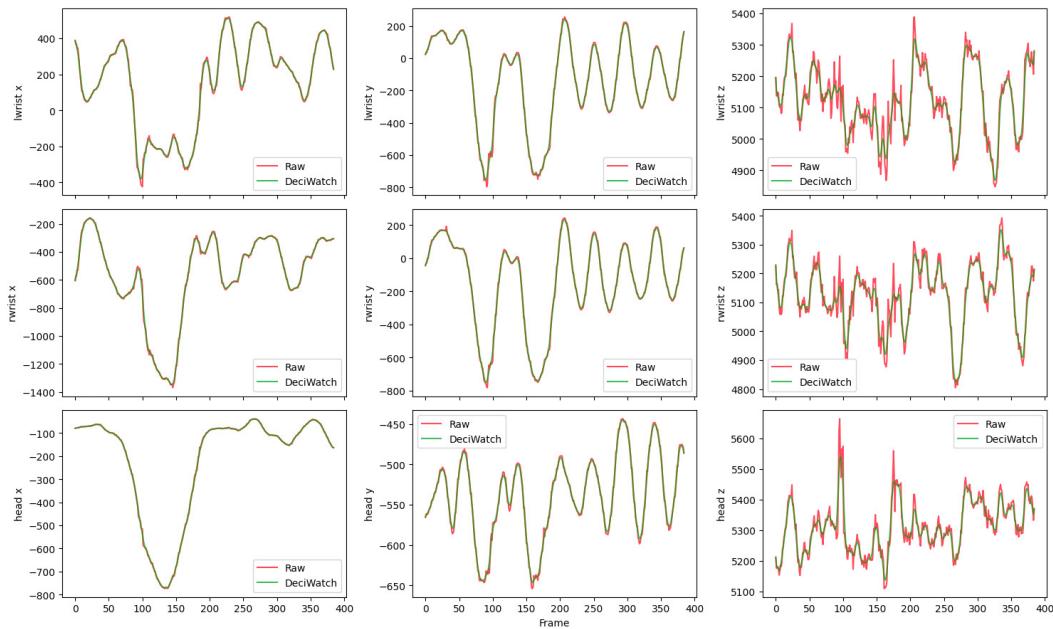
Tabel 6.2 merangkum hasil akhir dari dua sesi training, termasuk MPJPE output, loss akhir, dan durasi pelatihan. Berdasarkan tabel tersebut, dapat dilihat bahwa model berhasil menurunkan MPJPE output dari 66,10 mm menjadi 64,98 mm dengan penurunan loss dari 0,1666 ke 0,1522 selama dua sesi pelatihan.

Tabel 6.2 Ringkasan Hasil Pengujian Training Model DeciWatch

Sesi	Epoch	MPJPE Output (mm)	Loss Akhir	Durasi Training
1	1–20	66.10	0.1666	24 jam
2	21–45	64.98	0.1522	26 jam

6.1.4.2 Visualisasi Perbandingan Koordinat 3D

Visualisasi hasil prediksi dilakukan untuk membandingkan koordinat 3D antara model baseline (Metrabs) dan DeciWatch. Grafik berikut menunjukkan perbandingan koordinat X, Y, dan Z untuk beberapa joint seperti pergelangan tangan kanan (*rwrists*), pergelangan tangan kiri (*lwrists*), dan kepala (*head*).



Gambar 6.8 Perbandingan koordinat X, Y, dan Z hasil estimasi pose 3D antara Metrabs (*Raw*) dan DeciWatch (*Filtered*)

Kurva berwarna merah menunjukkan data mentah dari Metrabs, sedangkan kurva hijau menunjukkan hasil DeciWatch. Berdasarkan visualisasi tersebut, dapat dilihat bahwa pada data mentah (*Raw*), khususnya pada sumbu Z, terdapat fluktuasi yang lebih signifikan dibandingkan sumbu X dan Y. Hal ini disebabkan karena sumbu Z merepresentasikan kedalaman (*depth*), yang merupakan komponen paling sulit diprediksi dari input 2D. Model estimasi pose seperti Metrabs harus mengonversi

informasi gambar dua dimensi menjadi prediksi posisi tiga dimensi, sehingga prediksi pada sumbu Z cenderung lebih rentan terhadap noise.

Dengan penerapan DeciWatch sebagai model *temporal filter*, noise pada sumbu Z berhasil direduksi secara signifikan. DeciWatch memanfaatkan konteks waktu untuk melakukan *smoothing* pada urutan pose, sehingga menghasilkan data gerakan yang lebih halus dan stabil. Hal ini penting agar proses konversi ke sudut sendi (melalui *inverse kinematics*) menjadi lebih akurat dan menghindari gerakan robot yang tidak realistik akibat data yang terlalu berfluktuasi.

6.1.4.3 Perbandingan Model Custom dan Pre-trained

Pengujian dilakukan untuk membandingkan performa model DeciWatch hasil training custom dengan model pre-trained bawaan DeciWatch. Model custom menggunakan *skip frame* 5, sedangkan model pre-trained menggunakan nilai default 10.

Hasil perbandingan menunjukkan bahwa model custom memiliki MPJPE output lebih rendah, yaitu 64,98 mm dibandingkan pre-trained sebesar 71,27 mm. Hal ini disebabkan oleh resolusi temporal yang lebih rapat pada model custom, sehingga mampu mempelajari variasi gerakan secara lebih detail.

Tabel 6.3 menunjukkan perbandingan performa kedua model.

Tabel 6.3 Perbandingan Performa Model Custom dan Model Pre-trained

Model	MPJPE Output (mm)	Loss Akhir	Durasi Training
Custom Training	64.98	0.1522	50 jam
Pre-trained	71.27	-	-

Hasil ini menunjukkan bahwa proses training yang disesuaikan dengan konfigurasi dataset yang berupa data dance, memiliki gerakan yang cepat dan lambat dan parameter temporal dapat meningkatkan akurasi estimasi pose 3D dibandingkan dengan model pre-trained standar.

BAB VII. KESIMPULAN DAN SARAN

7.1 Kesimpulan

Berdasarkan hasil penelitian yang berjudul “*Implementasi Pose Estimation untuk Pemodelan Gerak Tari Tradisional Indonesia pada Robot Humanoid ROBOTIS-OP3*”, dapat diambil beberapa kesimpulan sebagai berikut:

1. Penelitian ini berhasil mengembangkan sistem estimasi pose 3D berbasis deep learning untuk menganalisis gerakan manusia. Model Metrabs digunakan sebagai baseline untuk deteksi awal pose, sedangkan DeciWatch digunakan untuk menyempurnakan prediksi secara temporal. Hasil pengujian menunjukkan bahwa MPJPE output berhasil diturunkan dari 107 mm (baseline) menjadi 64,98 mm setelah proses filtering temporal oleh DeciWatch.
2. Data pose 3D yang dihasilkan berhasil diaplikasikan untuk menggerakkan robot humanoid ROBOTIS-OP3. Proses konversi dari koordinat 3D ke sudut servo dilakukan menggunakan metode *Inverse Kinematics Transform*, sehingga robot mampu menirukan gerakan tari tradisional Indonesia. Pergerakan sudut sendi pada robot menunjukkan perubahan yang kontinu dan relatif halus pada setiap frame, dengan nilai sudut servo yang stabil sesuai data per frame.
3. Aplikasi berbasis website berhasil dirancang dan dibangun untuk mengintegrasikan proses input video, estimasi pose 3D, konversi ke sudut servo, hingga eksekusi gerakan pada robot. Sistem ini memudahkan pengguna dalam mengoperasikan *pipeline* secara terpadu dari antarmuka yang telah disediakan.
4. Model DeciWatch hasil custom training dilatih selama dua sesi dengan total 45 *epoch*, menghasilkan penurunan *loss* dari 0,1666 pada sesi pertama menjadi 0,1522 pada sesi kedua. MPJPE output pada sesi pertama adalah 66,10 mm, sedangkan pada sesi kedua berhasil turun menjadi 64,98 mm dengan durasi total pelatihan 50 jam (24 jam pada sesi pertama dan 26 jam pada sesi kedua). Model custom ini memiliki performa lebih baik dibandingkan model

pre-trained DeciWatch, yang memiliki MPJPE output sebesar 71,27 mm.

7.2 Saran

Berdasarkan hasil penelitian yang telah dilakukan, masih terdapat beberapa aspek yang dapat ditingkatkan dalam pengembangan sistem estimasi pose dan motion imitation pada penelitian berikutnya. Beberapa saran yang dapat diberikan antara lain:

1. Menambah variasi dataset yang digunakan, termasuk data gerakan tari tradisional Indonesia dengan kondisi pencahayaan dan sudut pengambilan gambar yang lebih beragam, agar model lebih robust terhadap variasi lingkungan.
2. Meningkatkan kompleksitas model estimasi pose dengan mengeksplorasi arsitektur lain yang memiliki resolusi temporal lebih tinggi, atau mengkombinasikan beberapa model dalam pendekatan ensemble learning.
3. Mengembangkan metode inverse kinematics yang lebih adaptif terhadap keterbatasan fisik robot, sehingga perbedaan proporsi anatomi antara manusia dan robot dapat diminimalkan, dan hasil gerakan menjadi lebih natural.
4. Melakukan pengujian lebih lanjut pada robot secara langsung dalam durasi lebih panjang untuk mengevaluasi ketahanan sistem kontrol gerakan dan kestabilan servo saat mereplikasi gerakan tari yang dinamis.

DAFTAR PUSTAKA

- Ahmad, A. (2017). Mengenal artificial intelligence, machine learning, neural network, dan deep learning. *J. Teknol. Indones.*, no. October, 3.
- Cholissodin, I., Sutrisno, S., Soebroto, A., Hasanah, U., & Febiola, Y. (2020). Ai, machine learning and deep learning. *Fakultas Ilmu Komputer, Universitas Brawijaya, Malang*.
- Damar, S. I., Firdaus, M. R., Hamdi, N., Firmansyah, R. C., Sendari, S., Mahandi, Y. D., & Zaeni, I. A. E. (2023). Football humanoid goalkeeper robot movement system using optimization of head movements. In *2023 8th international conference on electrical, electronics and information engineering (iceeie)* (pp. 1–6).
- Dhimas, W. A., Santoso, K. S., & Hartayu, R. (2024). Pengembangan kontrol robot mobil dengan kerangka kerja robot operating system. *Elsains: Jurnal Elektro*.
- Imania, A., Subur, J., Taufiqurrohman, M., & Thaha, S. (2023). Identifikasi frekuensi musik pada robot humanoid seni tari untuk penyelaras gerak tarian menggunakan metode k-nearest neighbor. In *Seminar nasional teknik elektro*.
- Jalil, A. (2016). Rancang bangun robot humanoid. *Proceeding, Konferensi Nasional Ilmu Komputer (KONIK APTIKOM)*, ISSN, 2338–2899.
- Kim, J.-W., Choi, J.-Y., Ha, E.-J., & Choi, J.-H. (2023). Human pose estimation using mediapipe pose and optimization method based on a humanoid model. *Applied sciences*, 13(4), 2700.
- Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In F. Pereira, C. Burges, L. Bottou, & K. Weinberger (Eds.), *Advances in neural information processing systems* (Vol. 25). Curran Associates, Inc. Retrieved from https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf
- Li, R., Yang, S., Ross, D. A., & Kanazawa, A. (2021). Ai choreographer: Music

- conditioned 3d dance generation with aist++. In *Proceedings of the ieee/cvpr international conference on computer vision* (pp. 13401–13412).
- Listiani, W., & Rohaeni, A. J. (2024). Pemanfaatan estimasi pose gerak pada penari trance dalam ritual adat ngalaksa. *Jurnal Budaya Nusantara*, 6(3), 365–371.
- Maharaj, A. (2008, 08). Some insights from research literature for teaching and learning mathematics. *South African Journal of Education*, 28, 401-414. doi: 10.15700/saje.v28n3a182
- Marpaung, F., Aulia, F., Nabila, R. C., et al. (2022). *Computer vision dan pengolahan citra digital*. PUSTAKA AKSARA.
- Mehta, D., Sridhar, S., Sotnychenko, O., Rhodin, H., Shafiei, M., Seidel, H.-P., ... Theobalt, C. (2017). Vnect: Real-time 3d human pose estimation with a single rgb camera. *Acm transactions on graphics (tog)*, 36(4), 1–14.
- Meng, S., Qiu, S., Liang, T., & Ren, Q. (2023). Motion imitation of a humanoid robot via pose estimation. In *2023 35th chinese control and decision conference (ccdc)* (p. 1526-1532). doi: 10.1109/CCDC58219.2023.10327198
- Palazzolo, A. B. (1976). Formalism for the rotation matrix of rotations about an arbitrary axis. *Classical Mechanics*. (Derives 3×3 axis-angle rotation matrix)
- Robotis. (2023a). Dynamixel overview [Computer software manual]. Retrieved from <https://emanual.robotis.com/docs/en/dxl/> (Diakses pada Juli 2025)
- Robotis. (2023b). Dynamixel protocol 2.0 [Computer software manual]. Retrieved from <https://emanual.robotis.com/docs/en/dxl/protocol2/> (Diakses pada Juli 2025)
- Robotis. (2023c). Dynamixel sdk [Computer software manual]. Retrieved from https://emanual.robotis.com/docs/en/software/dynamixel/dynamixel_sdk/overview/ (Diakses pada Juli 2025)
- Robotis. (2023d). Robotis-op3 e-manual [Computer software manual]. Retrieved from <https://emanual.robotis.com/docs/en/platform/op3/specifications/> (Diakses pada Juli 2025)
- Sárándi, I., Linder, T., Arras, K. O., & Leibe, B. (2021). MeTRAbs: metric-scale truncation-robust heatmaps for absolute 3D human pose estimation. *IEEE*

- Transactions on Biometrics, Behavior, and Identity Science (T-BIOM)*, 3(1), 16–30.
- Song, H., Bronfman, G., Zhang, Y., Sun, Q., & Kim, J. H. (2024). Mixed reality interface for whole-body balancing and manipulation of humanoid robot. In *2024 21st international conference on ubiquitous robots (ur)* (pp. 642–647).
- Sun, K., Xiao, B., Liu, D., & Wang, J. (2019). Deep high-resolution representation learning for human pose estimation. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 5693–5703).
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need.(nips), 2017. *arXiv preprint arXiv:1706.03762*, 10, S0140525X16001837.
- Wang, J., Tan, S., Zhen, X., Xu, S., Zheng, F., He, Z., & Shao, L. (2021). Deep 3d human pose estimation: A review. *Computer Vision and Image Understanding*, 210, 103225. Retrieved from <https://www.sciencedirect.com/science/article/pii/S1077314221000692> doi: <https://doi.org/10.1016/j.cviu.2021.103225>
- Xu, Y., Zhang, J., Zhang, Q., & Tao, D. (2022). Vitpose: Simple vision transformer baselines for human pose estimation. *Advances in Neural Information Processing Systems*, 35, 38571–38584.
- Zeng, A., Ju, X., Yang, L., Gao, R., Zhu, X., Dai, B., & Xu, Q. (2022). Deciwatch: A simple baseline for 10x efficient 2d and 3d pose estimation. In *European conference on computer vision* (pp. 607–624).
- Zhu, T., Qunfei, Z., Wan, W., & Xia, Z. (2017, 11). Robust regression-based motion perception for online imitation on humanoid robot. *International Journal of Social Robotics*, 9, 1-21. doi: 10.1007/s12369-017-0416-9

LAMPIRAN

Lampiran 1 Kuesioner *black-box testing*

No	Fitur	Skenario Pengujian	Hasil Pengujian	Kesimpulan
1	Upload Video	Pengguna mengunggah file video gerakan tari tradisional	File video berhasil diunggah	Berhasil
2	Estimasi Pose dan Filtering	Pengguna menekan tombol <i>Start Pose Estimation</i>	Pose estimation dapat dijalankan.	Berhasil
3	Tampilan Estimasi Pose	Sistem menampilkan skeleton 3D pada setiap frame video	Sistem memampulkan skeleton 3D dan video (frame awal → frame akhir)	Berhasil
4	Status Log	Sistem mencatat proses pada area log setiap tahapannya	Sistem memampulkan informasi setiap proses yang berlangsung	Berhasil
5	Penyimpanan Data	Pengguna menyimpan hasil estimasi ke dalam file JSON	File estimasi dari setiap frame didapatkan dengan format json	Berhasil

6	Playback Robot	Pengguna menekan tombol <i>Gerak</i> untuk menjalankan robot	Robot bergerak sesuai hasil estimasi proses sebelumnya.	Berhasil
Malang, 15 Juli 2025 Anggota Tim Robotik,  Marsanda Izzah Nabilah				

Lampiran 2 Dokumentasi saat pengambilan data oleh anggota tim robotik

