UNIVERSITAS INDONESIA

RANCANG BANGUN SISTEM MOTION CAPTURE DAN DATABASE MOTION UNTUK ROBOT HUMANOID DENGAN PERANGKAT MICROSOFT KINECT BERBASIS ROS (*ROBOT OPERATING SYSTEM*)

SKRIPSI

WISNU INDRAJIT

0806315944

FAKULTAS TEKNIK

PROGRAM STUDI TEKNIK ELEKTRO

DEPOK

JUNI 2012

UNIVERSITAS INDONESIA

RANCANG BANGUN SISTEM MOTION CAPTURE DAN DATABASE MOTION UNTUK ROBOT HUMANOID DENGAN PERANGKAT MICROSOFT KINECT BERBASIS ROS (*ROBOT OPERATING SYSTEM*)

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar Sarjana Teknik

WISNU INDRAJIT

0806315944

FAKULTAS TEKNIK

PROGRAM STUDI TEKNIK ELEKTRO

DEPOK

JUNI 2012

HALAMAN PERNYATAAN ORISINALITAS

Skripsi ini adalah hasil karya saya sendiri,
dan semua sumber baik yang dikutip maupun dirujuk
telah saya nyatakan dengan benar.

Nama	**: Wisnu Indrajit**
NPM	**: 0806 315944**
Tanda Tangan	**:**
Tanggal	**: 25 Juni 2012**

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh :

Nama : Wisnu Indrajit

NPM : 0806315944

Program Studi : Teknik Elektro

Judul Skripsi : Rancang Bangun Sistem Motion Capture dan Database Motion untuk Robot Humanoid dengan Perangkat Microsoft Kinect Berbasis ROS (*Robot Operating System*)

Telah berhasil dipertahankan dihadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Teknik pada Program Studi Teknik Elektro, Fakultas Teknik, Universitas Indonesia.

DEWAN PENGUJI

Pembimbing :

Dr. Abdul Muis S.T., M.Eng ( )

Penguji 1 :

Ir. Wahidin Wahab M.Sc., Ph.D. ( )

Penguji 2 :

Prof. Dr.Eng. Drs. Benyamin Kusumoputro M.Eng. ( )

Ditetapkan di : Depok

Tanggal : 25 Juni 2012

KATA PENGANTAR

Alhamdulillah. Puji Syukur penulis sampaikan kepada Allah SWT atas limpahan karunia sehingga penyusunan skripsi ini dapat diselesaikan dengan baik. Skripsi ini diajukan untuk memenuhi salah satu persyaratan memperoleh gelar sarjana teknik dan bagian dari usaha penulis untuk dapat berkontribusi dalam bidang robotika.

Penulis menyadari bahwa tanpa bantuan dan bimbingan dari berbagai pihak, akan sangat sulit bagi penulis untuk menyelesaikan skripsi ini Oleh karena itu, penulis mengucapkan terima kasih kepada:

1. Dr. Abdul Muis, S.T., M.Eng selaku dosen pembimbing yang telah memberikan bimbingan, arahan, fasilitas, dann kepercayaan dalam melakukan penelitian;
2. Orang tua dan seluruh anggota keluarga penulis yang telah memberikan dukungan material dan moral, serta semangat untuk belajar setiap saat;
3. Vektor Dewanto, yang memperkenalkan saya pada Microsoft Kinect dan ROS.
4. Teman-teman bimbingan, Keluarga Besar TRUI, dan Keluarga Besar Elektro Komputer 2008, atas masukkan teknis, akomodasi peralatan, semangat kebersamaan, dan sebagai tempat bernaung penulis dalam mencari ilmu serta berbagi canda-tawa yang meredakan kepenatan;

Akhir kata penulis berharap dapat membalas kebaikan segala pihak yang telah membantu dan berdoa kepada Allah SWT agar memberi pahala yang pantas. Semoga laporan ini dapat bermanfaat bagi pengembangan ilmu pengetahuan dan teknologi.

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI SKRIPSI UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Universitas Indonesia, saya yang bertanda tangan di bawah ini:

Nama : Wisnu Indrajit

NPM : 0806315944

Program Studi : Teknik Elektro

Departemen : Teknik Elektro

Fakultas : Teknik

Jenis karya : Skripsi

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Noneksklusif (*Non-exclusive Royalty-Free Right*)** atas karya ilmiah saya yang berjudul:

"RANCANG BANGUN SISTEM MOTION CAPTURE DAN DATABASE MOTION UNTUK ROBOT HUMANOID DENGAN PERANGKAT MICROSOFT KINECT BERBASIS ROS (*ROBOT OPERATING SYSTEM*)"

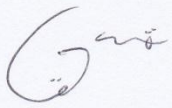
beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/formatkan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan mempublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis / pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di: Depok

Pada tanggal : 25 Juni 2012

Yang menyatakan,



(Wisnu Indrajit)

ABSTRAK

Nama : Wisnu Indrajit

Program Studi : Teknik Elektro

Judul : Rancang Bangun Sistem Motion Capture dan Database Motion untuk Robot Humanoid dengan Perangkat Microsoft Kinect berbasis ROS (*Robot Operating Sistem*)

Para *roboticist* selalu berusaha agar robot dapat menghampiri bentuk dan kemampuan manusia sehingga robot dapat berinteraksi bersama manusia dengan baik. Untuk dapat berinteraksi dengan manusia, robot sebisa mungkin dibuat agar memiliki gerakan yang mirip dengan manusia. *Imitation Learning* atau sering disebut dengan *Motion Capture*, adalah salah satu teknik pengendalian robot humanoid dengan manusia sebagai aktor dan robot sebagai agen yang akan mengimitasi gerakan aktor. Metode ini menawarkan kefleksibelan dan kemudahan dalam mememodifikasi sistem robot. Pada penelitian ini telah dikembangkan sebuah sistem motion capture untuk mentransformasikan gerakan lengan manusia ke lengan robot humanoid secara real time, dengan setiap lengan terdiri dari 3 DOF serta dilakukan perancangan database motion agar robot dapat melakukan gerakan yang telah dilakukannya. Proses *tracking* dengan Microsoft Kinect dilakukan pada rate frekuensi 20 Hz dengan dengan satu loop proses komputasi *mapping* membutuhkan waktu rata-rata 340 us. Rata-rata error pendeteksian vektor skeleton yang dideteksi adalah 1.74 cm.

Kata Kunci: *Kinect, Motion Capture, ROS, Dynamixel*

ABSTRACT

Name : Wisnu Indrajit
Study Program : Electrical Engineering
Title : Motion Capture System and Database Motion Development for Humanoid Robot using Microsoft Kinect Based on ROS (Robot Operating System)

The roboticist always trying to get the robot to approach the form and abilities so that the robot can interact with humans as well. To be able to interact with humans, robot made as much as possible in order to have similar movement to human. Imitation Learning or often called Motion Capture, is one of the humanoid robot control techniques with human as an actor and the robot as an agent who will imitate the movement of the actor. This method offers flexibility and ease to modify robot system. In this research, we have developed a motion capture system to transform human arm movement to humanoid robot in real time, with each arm consisting of 3 DOF and we have designed database motion so that robot can redo the movement which it can do previously. Tracking process with Microsoft Kinect performed at frequency of 20 Hz with a single loop computation mapping process takes an average of 340 us. The average error detection of skeleton vector is 1.74 cm.

Keyword: *Kinect, Motion Capture, ROS, Dynamixel*

DAFTAR ISI

DAFTAR GAMBAR

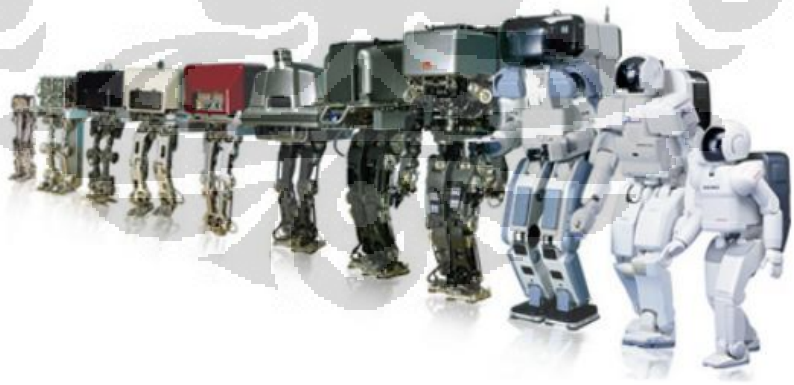
DAFTAR TABEL

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Penulis berpendapat bahwa para *roboticist*, insinyur robot, akan merasa lega saat robot sudah menjadi bagian dari kehidupan manusia. Gambaran terbaik tentang itu tersaji dalam film *I,Robot*. Beberapa cuplikan darinya adalah sebuah robot lari dengan cekatan mengambilkan alat bantu pernapasan tuannya yang tertinggal di rumah, sebuah robot sedang membantu ibu memotong kentang dengan cepat di dapur, dan robot-robot yang membersihkan sampah di jalan dengan tekun. Robot-robot tersebut membuat hidup manusia lebih sejahtera. Mereka biasa disebut sebagai *service robot.*

Service robot memiliki tingkat kompleksitas yang tinggi. Para insinyurnya selalu berusaha agar bisa menghampiri bentuk dan kemampuan manusia; ciptaan Tuhan yang paling sempurna. Kompleksitas tersebut dapat diurai menjadi berbagai macam bidang, misal: tentang struktur dan mekanisme sistem mekanik, kehandalan dan kapabilitas pengendali, sensor, dan aktuator, kecerdasan buatan, teknologi mobilitas, teknologi komunikasi dan kerjasama, teknik pentautan sub-sistem, dan sebagainya.



Gambar 1.1 Perkembagan Service Robot Produksi Honda

Salah satu bentuk *service robot* yang merupakan sebuah topik panas saat ini adalah humanoid robot. Belakangan ini, kemajuan teknologi disekitar telah memungkinkan robot untuk menyerupai bentuk dan tingkah laku manusia. Film-

film, buku fiksi, dan televisi telah menjanjikan kita bahwa humanoid robot akan memasak untuk kita, membersihkan rumah, menjadi sahabat, bahkan jatuh cinta dengan manusia. Robot humanoid adalah sebuah bentuk desain robot ideal untuk berinteraksi dengan manusia [12].

Tantangan dari humanoid robot adalah perancangan dan pengendalian *trajectory* dari sendi-sendi robot. Pemrograman secara manual atau dengan bahasa pemrograman biasa dilakukan untuk menggerakkan joint robot dari satu posisi ke posisi lain kemudian merekam state posisi joint robot. Namun, pemrograman seperti ini tidak fleksibel dan sulit untuk dilakukan. Sehingga muncullah teknik-teknik pengendalian joint robot humanoid, salah satunya adalah *imitation learning* atau sering disebut *motion capture*.

Pemrograman dengan demonstrasi, atau juga dikenal dengan *imitation learning* atau *motion capture* ini merupakan sebuah jawaban dari masalah pengendalian sendi-sendi robot humanoid. Metode ini menawarkan kefleksibelan serta modifikasi yang mudah dalam pengendalian robot [10]. Manusia sebagai aktor akan melakukan gerakan, kemudian robot sebagai agen akan melakukan gerakan-gerakan yang dilakukan oleh manusia. Sistem ini sangat berguna untuk melatih robot untuk melakukan gerakan-gerakan dan kedepannya dapat digunakan untuk membuat gerakan robot humanoid untuk mengerjakan pekerjaan rumah, pekerjaan berbahaya, maupun untuk hiburan yang mana kebanyakan orang membuat gerakan robot secara manual dengan tangan atau dengan latihan intensif yang mengandalkan pengalaman.

Dalam melakukan proses *imitation learning* pada robot, terdapat beberapa tahapan. Dimulai dari proses *tracking* manusia, *mapping* gerakan manusia ke robot, pengendalian posisi dan kecepatan robot, penyimpanan gerakan, serta proses mengulang gerakan-gerakan yang telah dilakukan. Pada penelitian ini akan dilakukan proses motion capture dari manusia terhadap model humanoid robot dengan menggunakan perangkat capture Microsoft Kinect dengan bantuan *Robot Operating Sistem* (ROS).

1.2 Tujuan Penelitian

Tujuan dan kontribusi dari penelitian ini adalah untuk membuat sistem *motion capture* dan perancangan *database motion* sebagai solusi dari perancangan dan pengendalian trajectory humanoid robot yang rumit. Sistem yang dirancang ini mampu membuat robot untuk melakukan gerakan yang telah diajarkan manusia kepada robot.

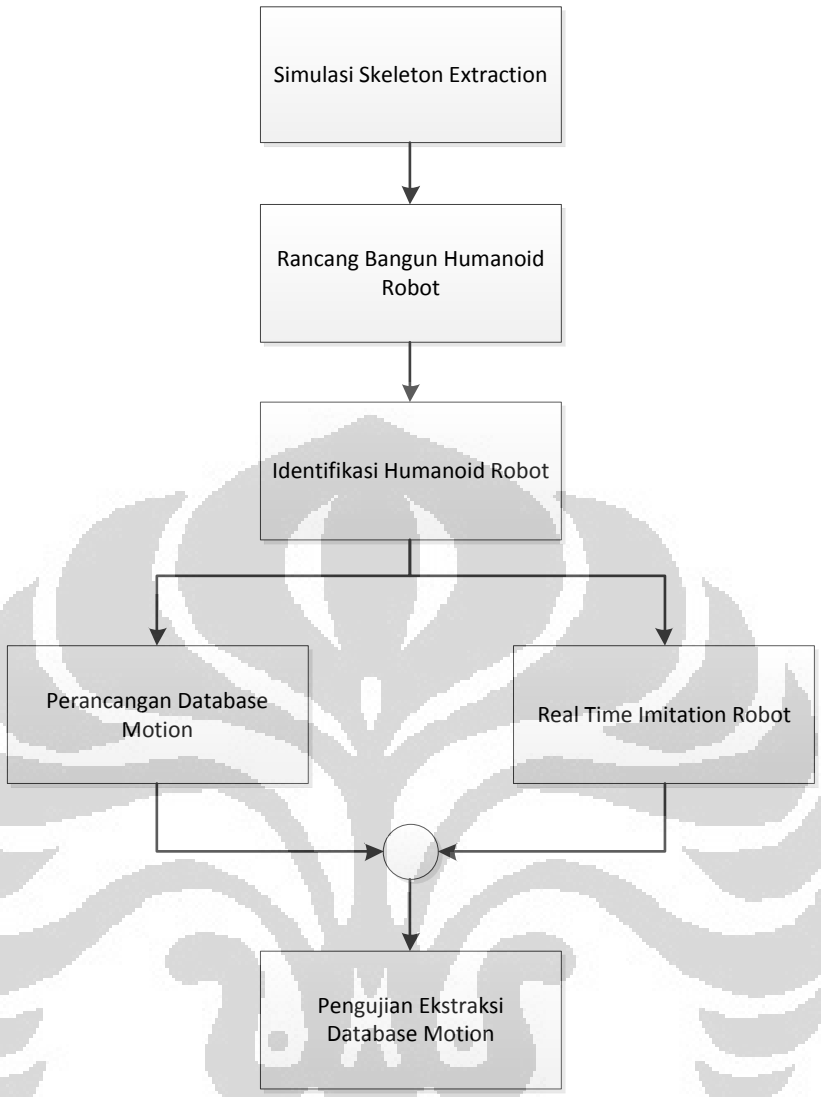
1.3 Pembatasan Masalah

Masalah pokok dari penelitian ini adalah perancangan algoritma motion capture dan perancangan database motion untuk membuat *intelligent* humanoid robot yang dapat melakukan gerakan yang sama dengan manusia. Dalam penelitian ini, objek riset yang digunakan adalah model robot humanoid dua lengan serta Microsoft Kinect sebagai perangkat *capture*. Proses motion capture yang dilakukan hanya ditransformasikan pada kedua lengan robot dengan setiap lengan terdiri dari tiga sendi berupa motor servo Dynamixel AX-12.

Microsoft Kinect akan melakukan capture pada lima belas titik pada tubuh manusia yang menjadi aktor. Dari proses *capture* akan dilakukan proses *mapping* posisi dan kecepatan sudut gerakan manusia ke robot. Gerakan robot yang telah dilakukan dapat disimpan dalam sebuah database motion dan diekstraksi kembali untuk dilakukan oleh robot.

1.4 Metodologi Penelitian

Langkah-langkah dalam melakukan penelitian ini digambarkan dalam diagram alir berikut.



Gambar 1.2 Metodologi Penelitan

Lebih jelasnya, metodologi penelitian yang dilakukan pada skripsi ini adalah:

a. Perancangan sistem humanoid robot
b. Proses *tracking* manusia menggunakan Microsoft Kinect
c. Perumusan proses *mapping* dari manusia ke robot yang digunakan
d. Desain database motion dan proses ekstraksi database motion untuk melakukan gerakan yang telah dilakukan sebelumnya
e. Pengujian, pengambilan data, dan analisis dari sisem yang telah dirancang.

Dalam menjalankan metodologi tersebut, penulis melakukan pendekatan tinjauan pustaka, yaitu dengan melakukan studi literature dari buku-buku, sumber dari internet dan beberapa jurnal serta melakukan pendekatan diskusi dengan dosen pembimbing skripsi yang berkaitan dengan topik bahasan skripsi.

1.5 Sistematika Penulisan

Laporan ini akan dibagi menjadi enam bab utama dengan pembagian den penjelasan ringkas sebagai berikut:

1. Pendahuluan

   Pada bab ini akan dijelaskan tentang latar belakang permasalahan yang muncul dan alasan pentingnya penyelesaian permasalahan tersebut, tujuan penelitian, pembatasan permasalahan yang dibahas dalam laporan ini, metode penelitian, serta susunan penulisan skripsi.

2. Teori Pendukung

   Pada bagian ini akan dipaparkan aspek umum dari teori-teori yang digunakan dalam penelitian ini. Bab ini akan menjelaskan teori mengenai proses *tracking* pada motion capture, proses pemetaan, dan contoh-contoh robot yang telah dapat melakukan proses imitasi.

3. Sistem Pendukung Penelitian

   Pada bab ketiga ini, akan dipaparkan perangkat keras dan perangkat lunak yang digunakan dalam melakukan penelitian.

4. Proses Tracking dan Mapping

   Bagian ini menjelaskan bagaimana proses *tracking* dan transformasi gerakan manusia ke robot dengan perangkat lunak yang telah dirancang. Bagian ini diawali dengan penjelasan proses skeleton tracking, identifikasi model robot, dan pengujian proses imitasi

5. Perancangan Database Motion

   Bab kelima ini menjelaskan perancangan database motion dan bagaimana proses ekstraksi database tersebut sehingga model robot mampu melakukan gerakan yang telah dipelajarinya.

6. Pengujian Sistem Motion Capture

   Pada bab ini akan dijelaskan beberapa pengujian terhadap sistem motion capture yang telah dirancang dan juga berisi analisis terhadap hasil penelitian.

7. Penutup

   Bab terakhir ini berisi perihal-perihal yang diperoleh selama penelitian, terutama yang berhubungan dengan proses penyelesaian masalah yang ada dan saran yang dapat dilakukan kedepannya.

BAB 2
TEORI PENDUKUNG

Pada bagian ini akan dipaparkan aspek umum dari imitasi dan bagaimana imitasi sangat berpengaruh pada banyak bidang sebagai bagian dalam proses pembelajaran. Faktanya, imitasi adalah sebuah hal penting yang membuat manusia dan juga binatang memiliki kapasitas untuk belajar, dan bidang robotika mencoba untuk menerapkan imitasi dengan membuat robot yang dapat belajar melalui proses imitasi. Bab ini menjelaskan proses imitasi, teknik-teknik yang digunakan untuk mengekstrasi informasi posisi dari manusia, dan seputar robot humanoid yang telah dapat melakukan proses imitasi.

2.1 Imitation Learning

Banyak peneliti berpendapat bahwa terdapat relasi penting antara ilmu biologi dan sistem imitasi pada robot. Pada kenyataannya, telah banyak adaptasi dari ilmu biologi yang telah diterapkan pada teknologi robotika. Salah satu bidang biologi yang merupakan sumber inspirasi untuk pengembangan robot adalah pembelajaran imitasi. Seperti yang dilakukan binatang yang baru lahir, yaitu berusaha untuk mengikuti perilaku induknya. Membuat gerakan robot, terutama robot humanoid dengan bahasa pemrograman untuk bergerak seperti gerakan manusa sangatlah sulit. Gerakan manusia merupakan gerakan yang sangat kompleks. Cara termudah untuk membuat robot dapat berperilaku mendekati manusia adalah dengan memberikan robot untuk mengimitasi perilaku manusia. Imitasi pada robot merupakan kemampuan robot untuk meng-*copy* gerakan yang dilakukan manusia.

Seperti yang telah dipaparkan pada [17], terdapat dua masalah dasar yang muncul pada proses imitasi. Yang pertama adalah bagaimana robot mengetahui aksi manusia yang dapat ditiru? Dan yang kedua adalah bagaimana melakukan pemetaan pose manusia ke dalam robot? Jawaban dari pertanyaan pertama akan dijelaskan pada bagian 2.3 dan pertanyaan kedua akan dipaparkan pada bagian 2.4.

2.2 Motion Capture

Motion capture, *motion tracking*, atau *mocap* adalah terminologi yang digunakan untuk mendeskripsikan proses dari perekaman gerakan dan pengartian gerakan tersebut menjadi model digital. *Motion capture* telah digunakan dalam dunia hiburan, olahraga, aplikasi medis, dan robotika. Pada dasarnya, motion capture berarti merekam aksi dari aktor manusia dan menggunakan informasi tersebut untuk menganimasi karakter digital ke dalam model animasi computer dua dimensi atau tiga dimensi bahkan dapat langsung ke sebuah object.

Motion capture dapat dilakukan dengan beberapa metode. Salah satu metode yang telah dilakukan adalah dengan menggunakan Metode *Marked-Human,* yaitu dengan memasang perangkat capture ke tubuh manusia. Gambaran umum dari perangat proses ini adalah seperangkat alat yang dlekatkan pada tubuh sang aktor (manusia) dimana didalam perangkat tersebut telah diberikan semacam alat pemancar gelombang ataupun *magnetic markers*. Ketika sang aktor bergerak alat tersebut memancarkan sinyal yang kemudian ditangkap oleh kamera yang mampu mendeteksi sinyal yang dikeluarkan oleh marker tersebut dan diproses didalam komputer guna menghasilkan data-data gerakan tersebut [7].

Gambar 2.1Titik-titik ekstraksi pada Motion Capture

Data gerakan tersebut kemudian dipetakan (*mapping*) kedalam model yang dibuat didalam aplikasi 2D/3D atau ke objek nyata lainnya, sehingga akhirnya objek tersebut akan bergerak layaknya sang aktor. Titik yang diambil merupakan titik pertemuan tulang atau engsel yang ada pada tubuh sang aktor yang akan dijadikan *keymotion* terhadap pegerakan itu sendiri. Titik-titik yang diambil ditunjukkan pada Gambar 2.1 diatas.

Namun, metode *marked-human based* ini masih memiliki beberapa kelemahan, antara lain:

1. Memerlukan hardware dan software yang spesifik, harga aplikasi dan perangkat yang dibutuhkan akan menjadi kendala terutama bagi studio animasi kelas kecil dan menengah
2. Pada beberapa teknologi *capture* yang ada membutuhkan tempat khusus yang dirancang untuk melakukan proses tersebut.
3. Secara teknis, akan menjadi masalah ketika objek yang dibuat memiliki bentuk karakter yang tidak proporsional dengan objek yang diambil, sehinga perlu dilakukan justifikasi manual dari sistem maupun hasil yang didapat.

Dari kelemahan-kelemahan diatas, pada penelitian ini dirancang sebuah sistem motion capture menggunakan metode *markless detection* menggunakan Microsoft Kinect sebagai perangkat capture. Microsoft Kinect dipilih sebagai perangkat capture karena harganya yang ekonomis, tidak membutuhkan tempat khusus, serta *open source software*.

2.3 Proses *Mapping*

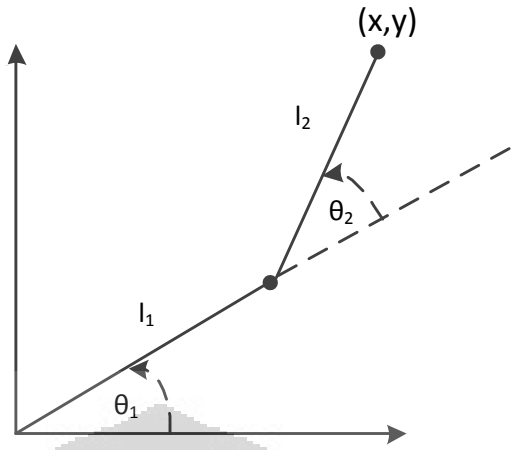
Untuk melakukan proses pemetaan antara gerakan robot dan gerakan manusia secara garis besar terdapat dua cara yang berbeda untuk melakukannya. Cara pertama adalah robot melakukan *motion primitives* yang telah dibuat sebelumnya saat software imitasi mendeteksi gerakan manusia yang mengintepretasikan gerakan yang berkorespondensi dengan *motion primitives* yang ada. *Motion primitives* ini terlebih dulu harus diprogramkan ke robot menggunakan metode *inverse kinematik* atau perangkat lunak lain yang dapat membuat program. Setelah menyimpan gerakan-gerakan yang diinginkan,

perangkat lunak penangkap citra diprogramkan sedemikian rupa untuk menentukan motion primiteves mana yang akan dilakukan robot dari citra yang ditangkap oleh perangkat *capture*. Masalah dari teknik ini adalah gerakan yang dilakukan robot dapat sangat berbeda dengan gerakan aktor pendemonstrasi. Agar dapat melakukan imitasi yang baik harus dibuat database yang besar dari *motion primitives*, dan perancangan perangkat lunak yang baik, yang dapat membedakan gerakan-gerakan yang dilakukan oleh aktor. Masalah lain pada teknik ini sulitnya mendapatkan transisi antar gerakan yang halus.

Teknik lain yang digunakan untuk melakukan pemetaan adalah dengan menggunakan metode inverse kinematik. Kinematika adalah bagian dari ilmu fisika yang membahas pergerakan robot dengan mengabaikan gaya pada objek tersebut. Dengan kinematika, telah dikembangkan pengetahuan lebih lanjut tentang posisi, kecepatan, percepatan, dan derivative dari posisi dengan orde yang lebih tinggi. Umumnya, robot terdiri dari link-link yang dihubungkan dengan joint (sendi) yang memungkinkan pergerakan terhadap link yang saling bertetanggan.

Dalam kinematika dikenal istilah forward kinematik dan inverse kinematik [3]. Masalah yang terdapat pada forward kinematik adalah, "Dimana posisi end effector apabila diberikan semua sudut pada setiap joint robot ($x=f(\theta)$)?" Sedangkan inverse kinematik mencoba menjawab pertanyaan "Berapa nilai sudut pada setiap joint robot apabila diberikan nilai poisi yang diinginkan ($\theta = f(x)$)?" Ketika manusia menggerakan tangan menuju suatu posisi tertentu, secara tidak langsung manusia melakukan penyelesaian masalah inverse kinematik. Metode ini memetakan titik-titik yang didapat dari objek tracking menjadi sudut pada manipulator. Metode ini cukup sulit dilakukan apabila jumlah joint robot kurang dari jumlah joint pada aktor.

Contoh permasalahan forward kinematik dan inverse kinematik adalah pada model robot serial link planar dua sendi seperti pada berikut.



Gambar 2.2 Serial Link Planar Dua Sendi

Persamaan forward kinematik pada kasus diatas adalah:

$$x = l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2)$$
$$y = l_1 \sin(\theta_1) + l_2 \sin(\theta_1 + \theta_2)$$

Sedangkan nilai θ_1 dan θ_2 dari serial link diatas dapat memiliki solusi inverse kinematik sebagi berikut:

$$\theta_2 = \arccos\left(\frac{x^2 + y^2 - l_1^2 - l_2^2}{2 l_1 l_2}\right)$$

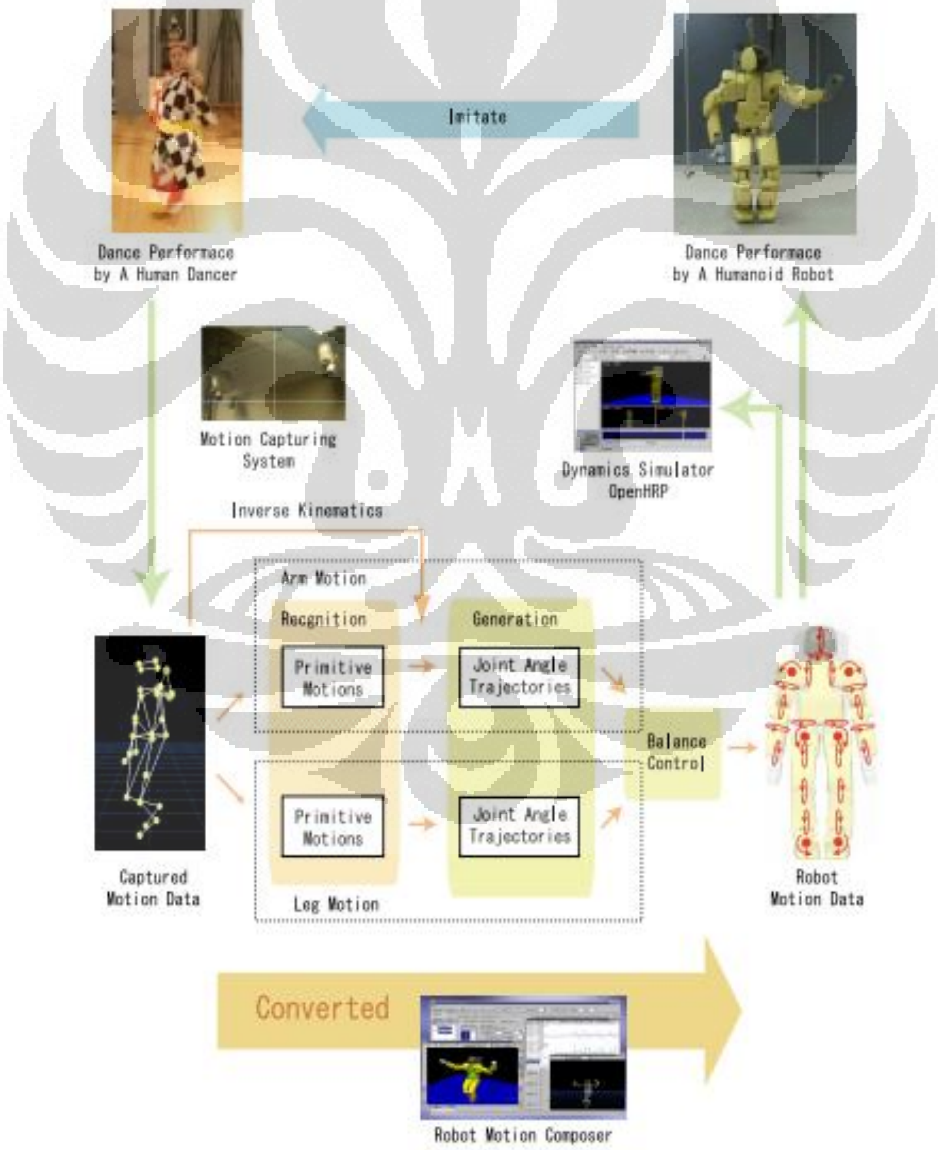
$$\theta_1 = \arctan\left(\frac{y(l_1 + l_2\cos(\theta) - x.l_2\sin(\theta_2)}{x(l_1 + l_2\cos(\theta) + y.l_2\sin(\theta_2)}\right)$$

Untuk melakukan transformasi manusia ke robot, pada penelitian ini digunakan metode inverse kinematik. Metode ini lebih mudah digunakan dibandingkan dengan metode ekstraksi dari *motion primitive*.

2.4 Robot Pengimitasi dan Aplikasinya

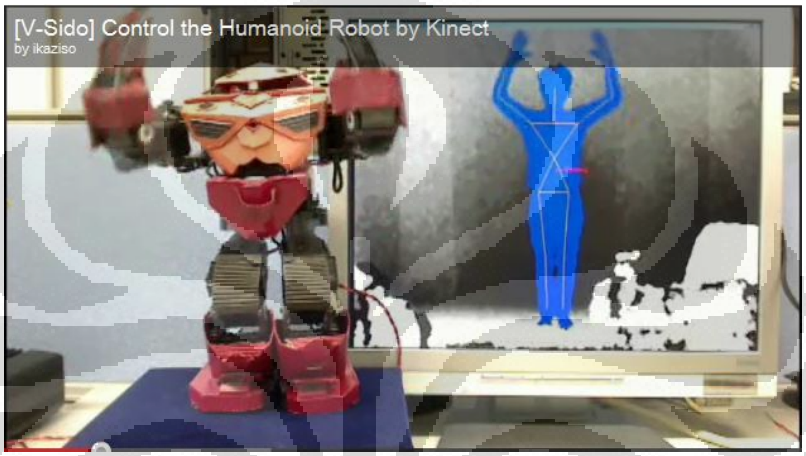
Beberapa sistem telah dikembangkan sebelumnya untuk mengeksplorasi imitasi pada robot. Beberapa diantaranya pada torso bagian atas, robot secara utuh, riset yang lain melakukan imitasi pada ekspresi wajah seperti pada robot DARWIN-OP. Salah satu yang paling popoler dieksplorasi adalah lengan robot sebagai bahan pembelajaran dalam melakukan pekerjaan fisik melalui demonstrasi.

Nakaoka dan grup risetnya telah melakukan pengendalian pose humanoid dengan imitasi untuk melindungi tarian tradisional Jepang *"Jongara-Bushi"* [11]. Aset budaya ini direkam melalui 8 buah kamera sebagai perangkat capture untuk mendapatkan informasi 3 dimensi dari 30 titik yang kedapannya dibutuhkan untuk melakukan proses imitasi. Objek dari riset ini adalah sebuah robot humanoid HRP 1-S. Untuk pengendalian lengan dilakukan proses inverse kinematik dari informasi 3 dimensi yang didapat, sedangkan untuk pengendalian kaki masih digunakan data hasil rekaman untuk menjaga keseimbangan robot. Gambaran riset ini dapat dilihat pada gambar dibawah ini.



Gambar 2.3 Motion Capture pada Robot HRP untuk melakukan tarian "Jongara-Bushi"

Riset lain dalam pengendalian robot humanoid berbasis motion capture dengan menggunakan kinect adalah proyek Vsido. Vsido adalah *software* robot yang dapat memindahkan pose orang secara langsung ke dalam robot seperti pada Gambar 2.4 . *Software* ini tidak hanya mampu memetakan lengan, tetapi juga dapat untuk memetakan pergerakan kaki dengan tetap menjaga keseimbangan robot.



Gambar 2.4 Proyek Motion Capture Vsido

BAB 3
SISTEM PENDUKUNG PENELITIAN

Bagian ini memaparkan sistem perangkat keras dan perangkat lunak pendukung yang digunakan dalam penelitian pembuatan sistem motion capture ini. Perangakat keras yang digunakan antara lain adalah model robot humanoid, USB to Dynamixel, Power Supply, dan Microsoft Kinect sedangkan perangkat lunak pendukung penelitian ini adalah ROS (*Robot Operating System*).

3.1 Perangkat Keras Pendukung

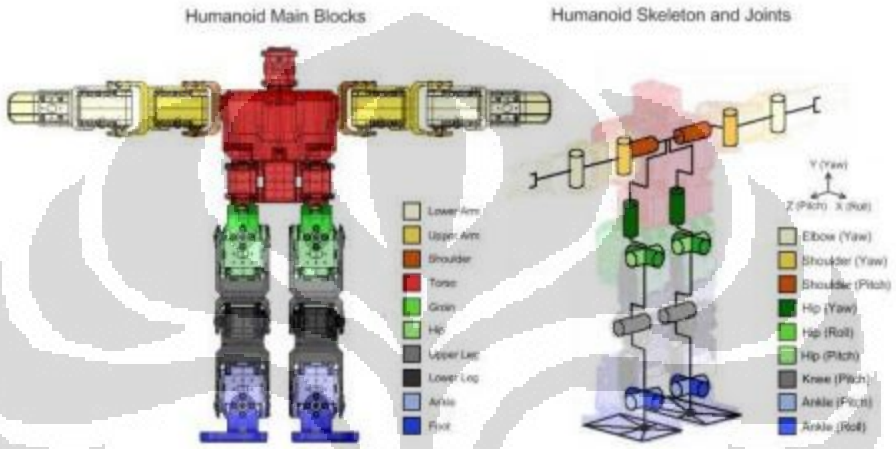
Untuk membuat sebuah robot humanoid yang mampu mengimitasi gerakan manusia perlu dirancang sebuah model humanoid robot yang memiliki konfigurasi yang mirip dengan manusia. Sistem perangkat keras pada penelitian ini mencakup humanoid robot, USB to Dynamixel, dan Microsoft Kinect.

3.1.1 Humanoid Robot

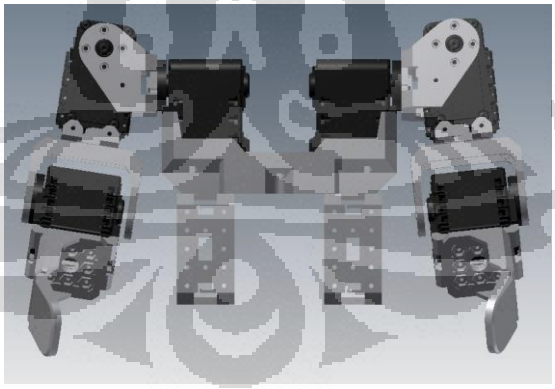
Robot yang digunakan pada penelitian ini adalah humanoid robot yang terdiri dari dua lengan dengan setiap lengan terdiri dari tiga dynamixel produksi ROBOTIS. Humanoid robot ini memiliki actuator motor servo yang memiliki feedback posisi, kecepatan, tegangan, temperature, dan beban. Beberapa platform robot humanoid seperti robot Kondo KHR-1HV / KHR – 2HV / Manoi / Robonova juga memiliki fitur yang sama. Namun, dokumentasi actuator dari robot-robot tersebut tidak cukup baik seperti yang dimiliki oleh ROBOTIS.

Gambar 3.1 Jenis Humanoid Robot di Pasaran

Sebenarnya, desain dari robot ini hampir menyerupai Bioloid Comprehensive kit.. Model bioloid comprehensive yang original memiliki konfigurasi seperti ditunjukkan pada Gambar 3.2. Pada riset ini tidak semua bagian dari robot akan diimitasi. Bagian robot yang akan diimitasi hanyalah kedua lengan. Oleh karena itu, konfigurasi robot yang digunakan dalam penelitian ini ditunjukkan pada Gambar 3.3. Kelebihan konfigurasi ini adalah memiliki jangkauan gerak yang cukup luas dan lebih mirip dengan manusia.



Gambar 3.2 Konfigurasi Original Bioloid Comprehensive



Gambar 3.3 Konfigurasi Model Robot yang digunakan

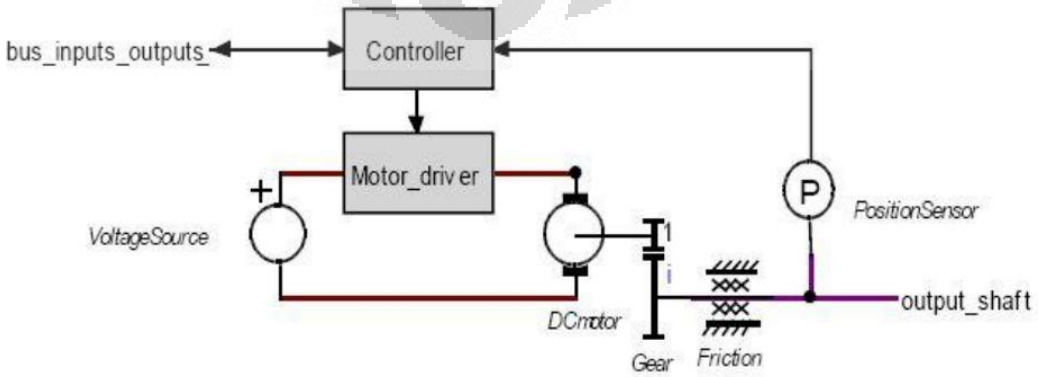
Konfigurasi ini tersusun dari dua lengan dengan masing-masing lengan memiliki tiga derajat kebebasan. Pergerakan robot ini diatur secara langsung oleh PC melalui USB2 Dynamixel. Identifikasi konfigurasi dari model robot ini akan dijelaskan pada bagian selanjutnya. Sebagai sumber pencatu daya pada robot ini,

digunakan *power supply* komputer. Berikut akan dijelaskan beberapa komponen pada robot ini.

3.1.1.1 Servo Dynamixel AX-12

Gambar 3.4 Dynamixel AX-12

Servo Dynamixel AX-12 merupakan 'otot' pada model robot yang digunakan. Dynamixel AX-12 adalah salah satu jenis dari banyak tipe dynamixel serial servo. Servo serial ini memiliki gear yang terbuat dari plastik yang kuat. Didalamnya terdapat pengendali ATmega8 sehingga dengan bantuan fitur-fitur pada ATMEGA8, dynamixel mampu memberikan *feedback* posisi, kecepatan, temperatur, dan tegangan. Dynamixel mendapat setpoint melalui komunikasi serial dan diproses untuk menggerakkan motor DC pada dynamixel tersebut. Output yang dihasilkan berupa posisi yang dibaca dengan sensor posisi untuk mengukur posisi saat ini sebagai umpan balik pengendali.



Gambar 3.5 Internal Blok Diagram pada Dynamixel AX-12

Selain itu dynamixel AX-12 ini memiliki *precision control* yang memiliki resolusi 1024 step serta *compliance driving* sehingga sudut dari penyesuaian torsi dapat diatur. Dynamixel juga memiliki sistem alarm yang dapat aktif dari parameter yang diingikan pengguna seperti *overheat, over voltage*, dan *overload*. Dynamixel AX-12 menggunakan distributed control yang berarti posisi, kecepatan, dan torsi dapat diatur dengan sebuah paket data. Berikut adalah spesifikasi utama dari Dynamixel AX-12:

Tabel 3.1 Spesifikasi Dynamixel AX-12

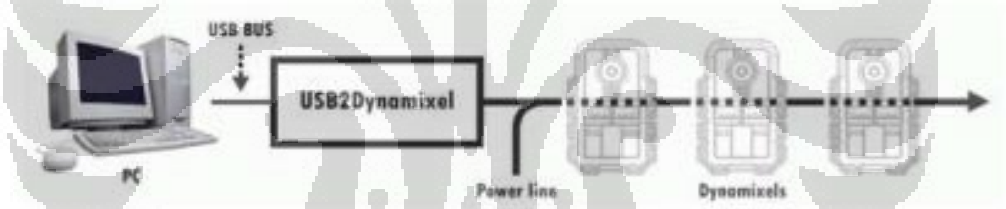
Parameter	Nilai
Limit Tegangan	7-14 Volt
Arus Maksimum	900mA
Torsi	12-16.5 kgf.cm
Kecepatan	0.269 sec/60° ~ 0.196 sec/60°
Resolusi	0.29°/step (1024 step)
Sudut Operasi	300°, *Endless Turn*
Temperatur Operasi	-5° C ~ +85° C
Sinyal Pengendali	*Digital Packet*
Tipe Protokol	*Half Duplex Asynchronous Serial Communication (8 bit, 1 stop, No Parity)*
BaudRate	7343bps – 1Mbps
Level Sinyal	TTL
ID	254 ID

3.1.2 USB to Dynamixel

USB to Dynamixel buatan ROBOTIS ini berfungsi untuk menghubungkan PC dengan Dynamixel tanpa memerlukan mikrokontroler dalam proses komunikasinya. Device ini sangat ideal digunakan dalam penelitian ini karena tujuan penelitian ini bukan untuk membuat algoritma untuk meningkatkan autonomi dari robot, melainkan ingin membuat robot memiliki kemampuan untuk mengimitasi gerakan manusia dengan pengolahan data dari komputer.

Gambar 3.6 USB to Dynamixel

PC memiliki kemampuan untuk melakukan komputasi yang lebih tinggi dan hal tersebut merupakan hal yang penting untuk mengimplementasikan Real Time Imitation. Pada gambar 3.7 diperlihatkan bagaimana cara menghubungkan Dynamixel AX-12 dengan PC melalui Port USB. Untuk melakukan hal tersebut, setiap AX-12 dihubungkan secara seri dengan yang lain dan salah satu dari Dynamixel AX-12 tersebut dihubungkan dengan USB to Dynamixel.



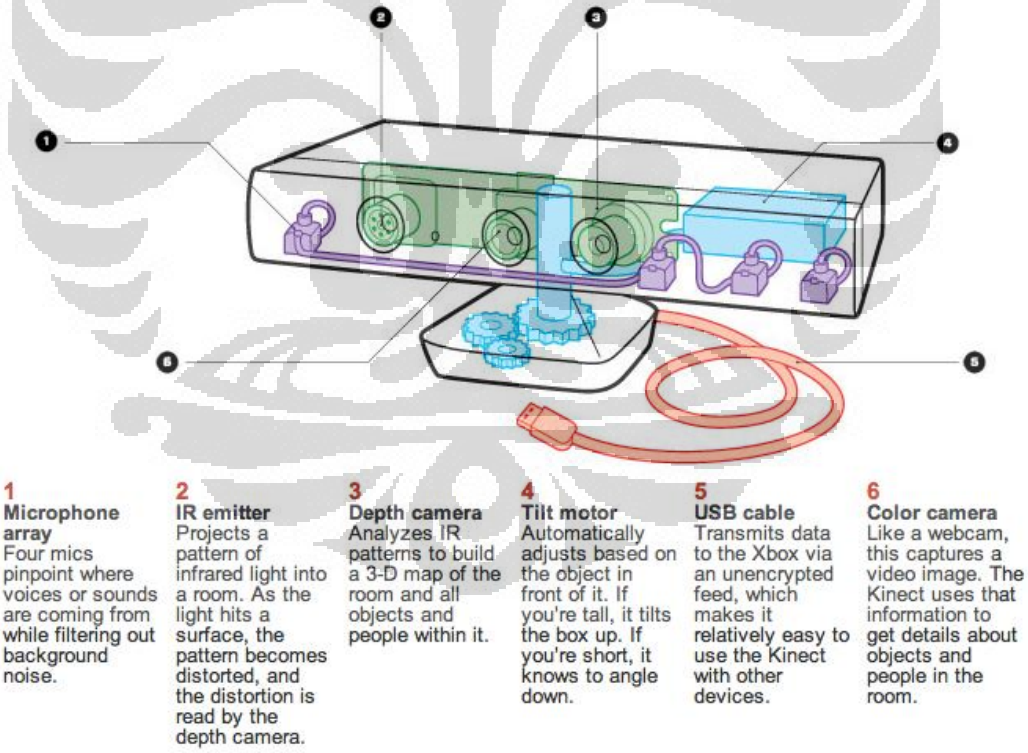
Gambar 3.7 Konfigurasi Pemasangan USB to Dynamixel

3.1.3 Microsoft Kinect

Microsoft Kinect adalah sebuah aksesori untuk platform Microsoft Xbox 360. Kinect merupakan sebuah *motion sensing unit device* yang dikembangkan untuk menginterpretasikan gerak tubuh manusia. Kinect memiliki sensor-sensor yang terhubung dengan *motorized pivot based*. Device ini memiliki Kamera RGB, *depth sensor*, dan *multi-array microphone*. Dengan sensor-sensor tersebut Kinect dapat menyajikan kemampuan melakukan *full-body 3D motion capture, facial recognition,* dan *voice recognition*. Kinect memiliki performansi yang baik, dimensi yang kecil, *open source libraries* yang mampu membuat Kinect digunakan pada beberapa platform OS (Windows, Linux, dan Mac), serta memiliki harga yang cukup murah.

Gambar 3.8 Microsoft Kinect XBOX 360

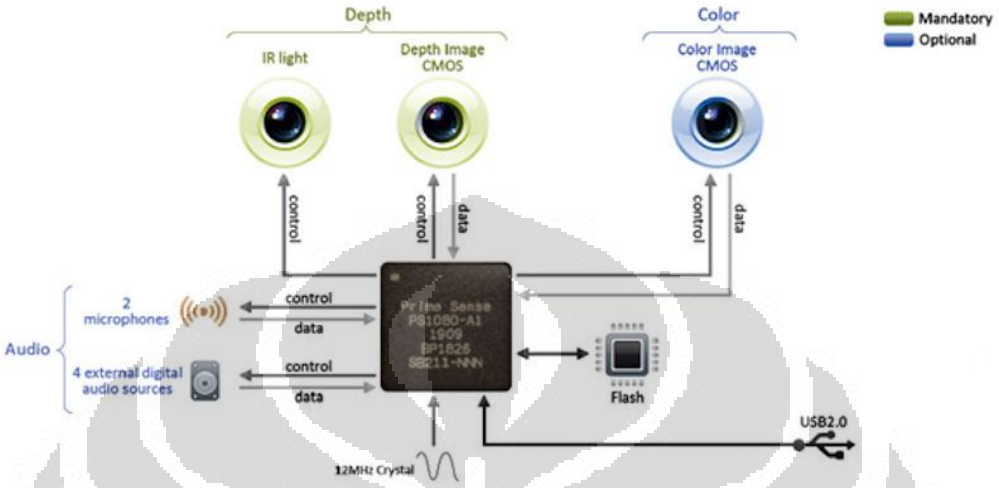
Sensor kinect mampu mengeluarkan output video dengan frame rate 30 Hz sebagai kombinasi dari dua sensor yang berbeda. Yang pertama dihasilkan oleh kamera RGB dengan resolusi 8 bit VGA (640x480 pixel) dengan Bayer Color Filter, dan yang kedua dihasilkan oleh *monochrome depth sensor* dengan resolusi yang sama dan 2048 tingkat sensitivitas (11 bit). Ilustrasi dari sensor-sensor kinect ditunjukkan pada gambar berikut.



Gambar 3.9 Ilustrasi Bagian-Bagian dari Microsoft Kinect

Bagimana sensor-sensor pada Kinect terhubung ditunjukkan pada gambar 3.10. Sistem tersebut terdiri dari *Depth Camera* dan *Color Camera* yang

menggunakan tekologi CMOS serta Microphone sebagai sensor Audio. Seluruh sensor tersebut dikontrol dan diolah dalam sebuah middleware yang dikembangkan oleh PrimeSense.



Gambar 3.10 Internal Blok Diagram dari Microst Kinect

3.2 Perangkat Lunak Pendukung

Perangkat lunak merupakan suatu kebutuhan yang harus dipenuhi suatu sistem. Pada bagian ini akan dijelaskan mengenai perangkat lunak yang digunakan dalam penelitian. Perangkat lunak pendukung pada penelitian ini berfungsi sebagai driver dan abstraksi hardware Microsoft Kinect dan Dynamixel pada robot yang digunakan.

3.2.1 ROS (Robot Operating Sistem)

Gambar 3.11 Logo ROS (Robot Operating Sistem)

ROS (Robot Operating Sistem) adalah sebuah kerangka kerja untuk pengembangan perangkat lunak robot. Pada awalnya ROS dikembangkan oleh

Stanndford Artificial Intelligence Laboratory, tetapi pada tahun 2008 pengembangan lebih lanjut dilakukan oleh Willow Garage dengan lebih dari 20 institusi berkolaborasi dalam pengembangan ROS. Saat ini ROS telah mendukung pengembangan berbagai robot sepeti ang ditunjukkan pada Gambar 3.12 [1].

ROS menyediakan layanan standar sebuah OS (Operating Sistem) seperti abstraksi perangkat keras, low level device control, implementasi dari fungsi umum sebuah OS, message-passing antar proses dan package management. Hal ini didasarkan pada arsitektur grafik dimana suatu proses terletak pada sebuah node yang dapat mengirim dan menerima multipleks sensor, mengontrol, mengetahui state, aktuator, dan pesan lainnya. Library ROS berbasis sistem UNIX (diutamakan mendukung Ubuntu Linux, sedangkan variasi lain seperti Fedora dan Mac OS X masih berstatus "eksperimental").

Gambar 3.12 Robot-robot yang telah dikembangkan dengan ROS

ROS memiliki dua dasar sistem, yaitu meta operating sistem bernama ROS seperti yang dijelaskan sebelumnya, dan ros-pkg, sebuah suite dari pengguna berupa paket yang dapat melakukan fungsi secara simultan lokalisasi dan pemetaan, perencanaan, persepsi, simulasi, dan lain-lain. ROS dirilis dibawah lisensi BSD dan merupakan perangkat lunak open source.

Pengembangan ROS juga didukung oleh universitas-universitas seperti Brown University, Stanford University, Tokyo University, dan lain-lain. Universitas-universitas ini ikut mengembangkan ROS dengan menyumbang

repository package pada ROS. Beberapa Universitas juga telah menjadikan ROS bagian dalam pelajaran mata kuliah robotika.

3.2.1.1 Arsitektur Grafis Komputasi ROS

Seperti yang telah dijelaskan sebelumnya, ROS dibangun berbasis arsitektur grafik. Dari sudut pandang secara grafis, ROS digambarkan jaringan peer to peer dari proses ROS yang berbagi data. Gambaran grafis tersebut dapat dikelompokkan ke dalam bagian berikut:

1. *Nodes*

   Sebuah node secara substansi adalah sebuah proses yang melakukan komputasi. Dapat dikatakan node adalah tempat dimana semua operasi utama dilakukan, sebuah entitas utama dalam grafik. Node saling berkomunikasi satu sama lain dengan menggunakan streaming topics, services, atau Parameter Server
2. *Messages*

   Dalam ROS, *Messages* adalah sebuah struktur data sederhana yang didefinisikan oleh pembuat packages. Fitur *Messages* mendukung tipe primitif standar sebagaimana array dari tipe primitive atau sebelumnya didefinisikan dalam pesan.
3. *Topics*

   Topics adalah sebuah nama bus tempat dimana node saling bertukar pesan.
4. *Services*

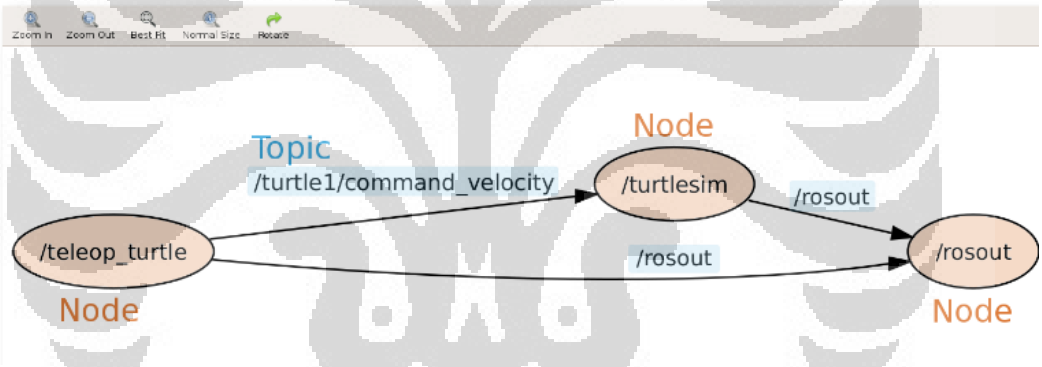
   Model *publish/subscribe* adalah sebuah paradigma komunikasi yang sangat fleksibel. Permintaan atau balasan dilakukan melalui layanan yang didefinisikan oleh sepasang pesan, satu untuk permintaan dan satu untuk balasan.
5. *Parameter Server*

   Parameter Server adalah sebuah "kamus" bersama yang dapat diakses melalui jaringan API. Node dapat menggunakan server ini untuk menyimpan dan mengambil parameter pada saat runtime.

6. *Bags*

   Merupakan sebuah format untuk menyimpan dan memainkan kemali data pesan pada ROS. Bags adalah mekanisme penting untuk menyimpan data, seperti data sensor yang mungkin sulit dikumpulkan tetapi diperlukan untuk mengembangkan dan menguji algoritma.

Sebuah contoh dari grafik yang terbentuk dalam suatu sesi ROS dapat dilihat pada Gambar 3.13. Setiap node dapat terhubung dengan sebuah atau beberapa topik. Mekanisme pertukaran pesan ini dikenal dengan sebutan *publish* dan *subscribe*. Pada gambar 3.13 dpat dilihat bahwa node /teleop_turtle mempublish topik /turtle1/command_velocity yang disubscribe oleh node /turtlesim.



Gambar 3.13 Grafik Representatif pada sebuah sesi aktif dalam ROS (Node-elips, topic-rectangle)

3.2.1.2 Important Library

Untuk menunjang keberlangsungan eksperimen ini, dibutuhkan beberapa library penting dari ROS. Paket-paket ini dapat diunduh langsung dari www.ros.org. Berikut adalah beberapa library penting yang digunakan dalam eksperimen ini:

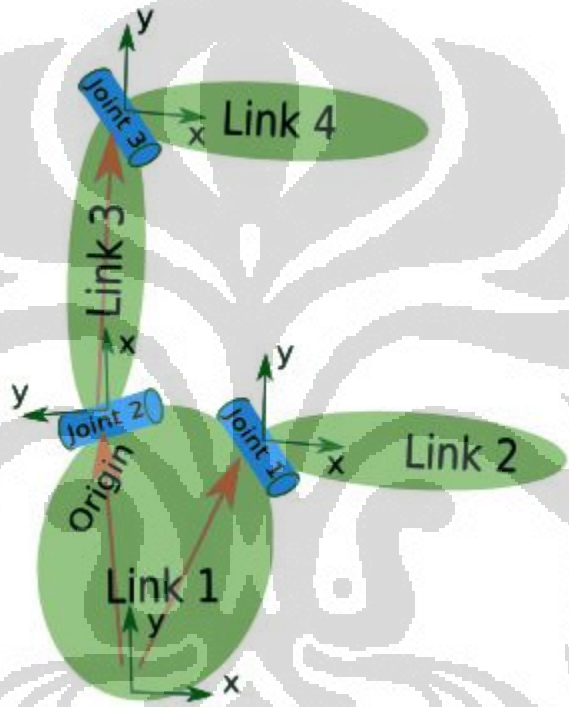
1. Openni_kinect

   Merupakan Library yang menyediakan driver kinect dengan computer dan beberapa package pendukung seperti openni_camera, openni_tracker,

depth_image_proc, dan lain-lain. Library ini juga memungkinkan untuk memunculkan *point cloud data* dari *depth sensor* pada kinect.

2. URDF (*Unified Robot Description Format*)/ XACRO

URDF adalah sebuah spesifikasi XML untuk menggambarkan bentuk fisik dari robot. Pada format URDF robot dianggap sebuah struktur pohon dari sebuah rigid links yang dihubungkan dengan joint. Sebuah contoh struktur sederhana dari sebuah model URDF digambarkan dalam Gambar 3.14**.**



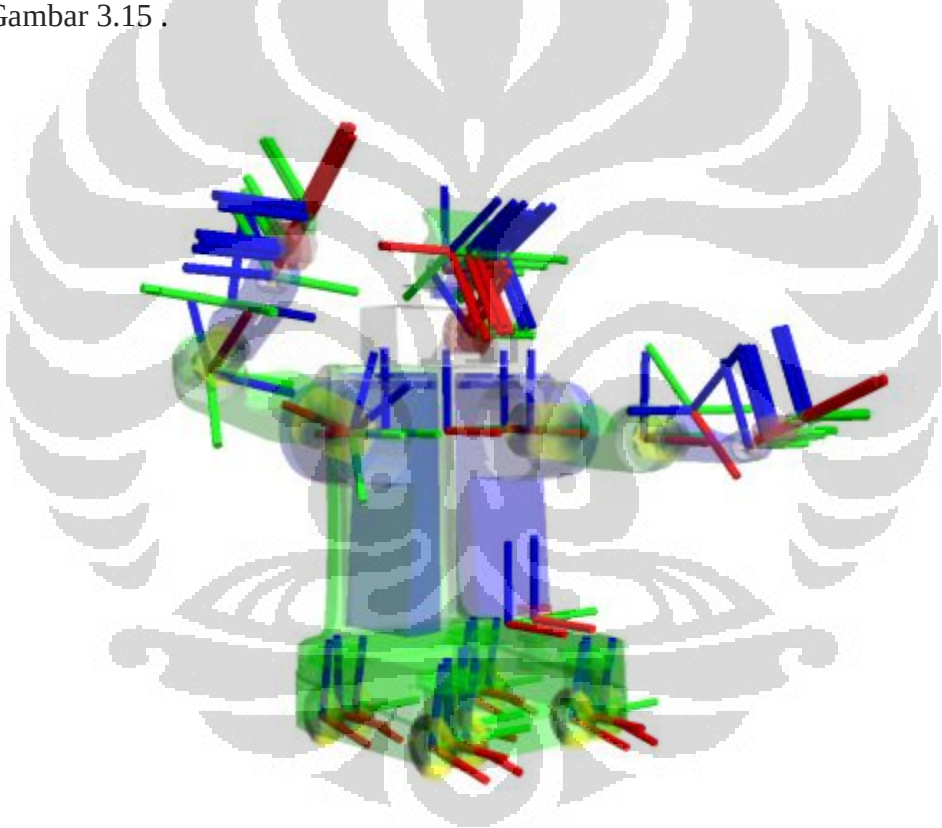
Gambar 3.14 Model URDF sederhana

Seperti yang telah dijelaskan, URDF memiliki dua buah tipe elemen, yaitu link dan joint (sendi). Link digunakan untuk medeskripsikan atau menggambarkan visual objek beserta property dari benda tersebut, misal inertia dan *collision* (tabrakan). Sedangkan joint menggambarkan kinematika dan dinamika dari sebuah sendi serta batas keamanannya. ROS juga menyediakan Xacro, sebuah makro XML yang sangat berguna saat bekerja dengan dokumen XML besar seperti deskripsi robot. Deskripsi lebih dalam tentang penulisan URDF dapat dilhat pada http://www.ros.org/wiki/urdf/XML/Link dan http://www.ros.org/wiki/urdf/XML/Joints.

3. TF

Sebuah sistem robot biasanya memiliki banyak frame koordinat 3D yang berubah dari waktu ke waktu, sebagai conth world frame, base frame, gripper frame, dan lain-lain. Library tf digunakan untuk melacak koordinat dan orientasi semua frame dari waktu ke waktu.

Library tf dapat digunakan dengan spesifikasi URDF untuk beroperasi dengan robot dalam koordinat ruang. Library ini dapat digunakan untuk mendapatkan lokasi frame tertentu (link) dari robot yang berhubungan dengan "dunia". Gambaran tentang penggunaan bersama antara URDF dan tf terlihat pada Gambar 3.15 .



Gambar 3.15 Model Kompleks Sebuah Robot yang dibaca dari URDF Description.

BAB 4
PROSES TRACKING DAN MAPPING

Pada bagian ini akan dijelaskan metode *tracking* dan *mapping* dari skeleton yang dideteksi kinect untuk ditransformasikan ke model robot. Bagian ini diawali dengan proses tracking dari kinect menggunakan ROS, dan proses mapping yang berisi tentang penjelasan struktur konfigurasi model robot humanoid yang digunakan, penurunan model kinematik dari robot, serta proses pemetaan posisi dan kecepatan skeleton ke robot.

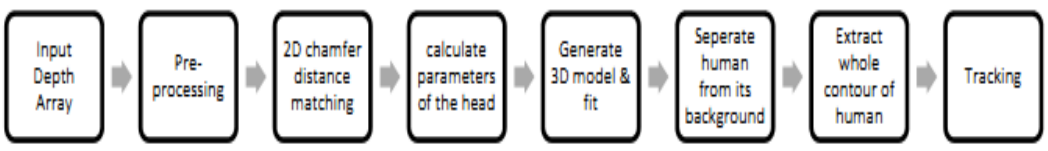
4.1 Proses Tracking

Tracking skeleton manusia merupakan bagian penting dalam sistem motion capture. Titik-titik pertemuan tulang atau engsel yang ada pada tubuh manusia akan diambil dan direpresentasikan dalam sebuah vektor tiga dimensi. Pada bagian ini akan dijelaskan tahapan proses tracking pada kinect dan bagaimana data-data skeleton diekstrasi.

4.1.1 Skeleton Tracking dengan Microsoft Kinect

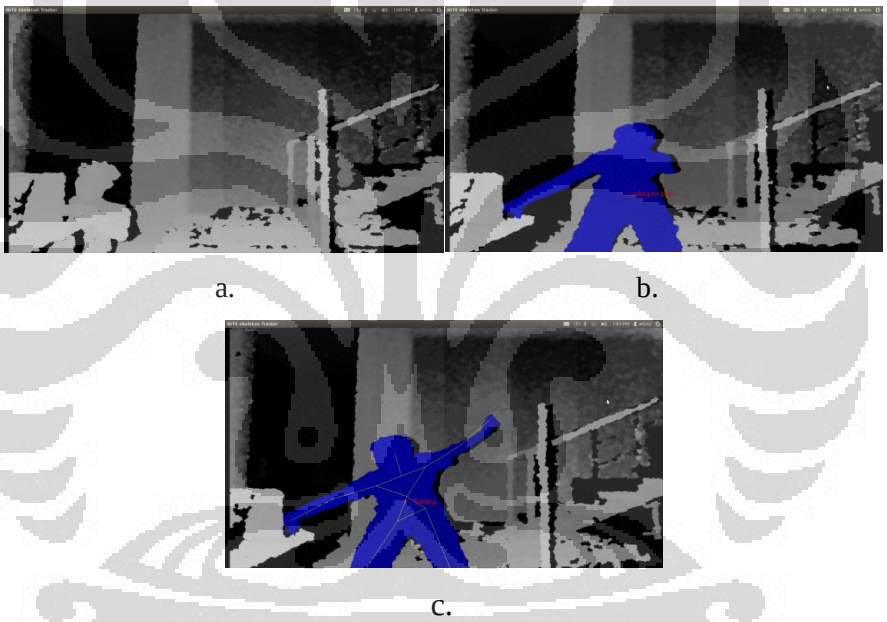
Microsoft Kinect memiliki beberapa driver yang dapat digunakan untuk memanfaatkan sensor-sensornya. Beberapa driver yang ada antara lain adalah openni, libfreenect, library dari Microsoft, dan open_kinect. Driver-driver ini memiliki kemampuan untuk mengolah data dari sensor kinect menjadi data-data yang dapat dimanfaatkan oleh pengguna. Pada penelitian ini, depth sensor kinect akan mengambil data-data beberapa vektor-vektor penting pada tubuh manusia yang akan dimanfaatkan dalam sistem motion capture ini.

Secara umum, gambaran umum proses *human detection* dengan kinect ditunjukkan pada gambar dibawah ini.



Gambar 4.1 Tahapan Proses Skeleton Tracking pada Kinect

Dari input *depth array* yang didapat, dilakukan proses filtering untuk mereduksi noise dan melakukan proses penghalusan (*smoothing*) dari array tersebut. Digunakan dua tahap proses pendeteksian untuk melokalisasi keberadaan manusia. Pertama dilakukan eksplorasi terhadap batas dari informasi *depth array* yang didapatkan untuk menentukan kandidat daerah yeng memiliki kemungkinan terdapat manusia. Proses ini melakukan segmentasi daerah berdasarkan kedalaman (depth). Setelah dilakukan proses segmentasi, dilakukan fitting model dari hasil proses segmentasi tersebut terhadap data model yang telah dimiliki library kinect. Kemudian apabila orang terdeteksi melakukan pose seperti pada gambar 4.4, kinect akan melakukan skeleton tracking.



a. b.

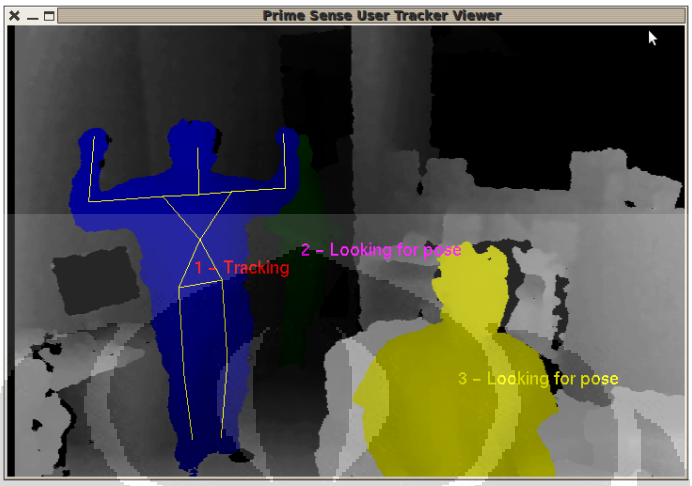
c.

Gambar 4.2 Proses Skeleton Tracking dengan Kinect, a. Segmentasi, b. Human Detection, c. Skeleton Tracking

ROS adalah sebuah sistem besar, dengan banyak kode *open source* yang telah ditulis oleh banyak *developer*. Telah banyak package-package pada ROS yang telah dikembangkan para developer sebagai abstraksi dan driver dari hardware-hardware yang secara umum digunakan dalam dunia robotika.

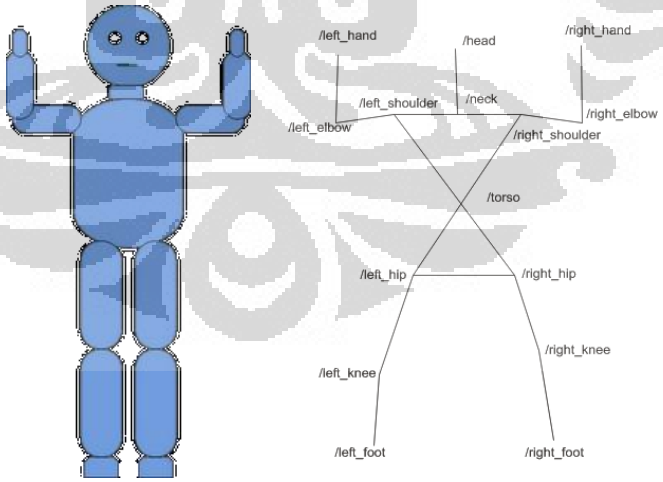
Stack openni_kinect merupakan driver yang telah dikembangkan developer ROS sebagai abstraksi hardware kinect dengan komputer. Salah satu

package yang terdapat pada openni_kinect adalah openni_tracker yang berfungsi untuk mendapatkan posisi tiga dimensi dari joint-joint pada tubuh manusia.



Gambar 4.3 Skeleton Tracking pada package openni_tracker

Untuk melakukan proses skeleton tracking, sesorang perlu berdiri didepan kinect dan melakukan pose "menyerah" (mengangkat kedua tangan). Saat melakukan pose ini, kinect akan melakukan kalibrasi skeleton manusia. Pose ini digambarkan pada Gambar 4.4.



Gambar 4.4 Pose Kalibrasi untuk melakukan Skelton Tracking

Node openni_tracker ini akan mempublish suatu set transformasi dengan nama frame sebagai berikut:

- /head
- /neck
- /torso
- /left_shoulder
- /left_elbow
- /left_hand
- /right_shoulder
- /right_elbow
- /right_hand
- /left_hip
- /left_knee
- /left_foot
- /right_hip
- /right_knee
- /right_foot

4.1.2 Perancangan Software Tracking

Dalam penelitian ini, untuk melakukan proses *tracking* dan *mapping* secara real time, pengolahan data motion, proses penyimpanan dan pengulangan perlu dibuat sebuah ROS package. Pada penelitian ini dibuat sebuah package dengan nama skripsi_mocap yang memiliki *dependencies* dengan beberapa package, antara lain roscpp, rospy, std_msgs, geometry_msgs, nav_msgs, tf, urdf, rviz, kdl, nite, dynamixel_driver, dynamixel_controllers, dan dynamixel_msgs. Untuk membuat package digunakan perintah roscreate-pkg

```
$ roscreate-pkg [package_name] [depend1] [depend2] [depend3]
```

Setelah membuat ROS package, dibuat source code yang akan di-*build* beserta file-file lainnya, seperti message, service, definisi parameter dan robot model dalam format urdf. Proses Building Packages dilakukan dengan menggunakan perintah rosmake. Pada penelitian ini beberapa library didapat dari package pi_tracker yang sebelumnya telah dikembangkan oleh Patrick Goebel untuk membuat pi_robot.

```
$ rosmake skripsi_mocap
```

Dalam package skripsi_mocap ini dibuat beberapa launch file untuk memanggil node-node ROS yang telah dibuat. Berikut adalah deskripsi dari setiap launch file.

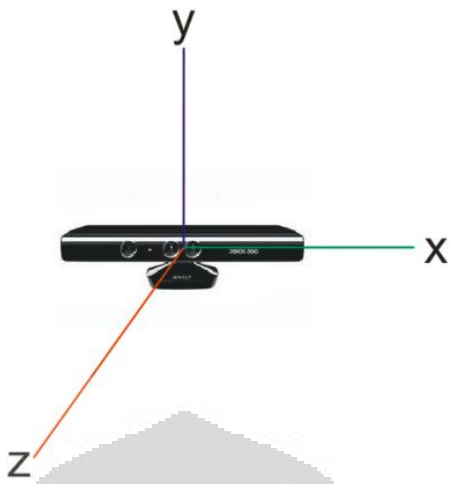
Tabel 4.1 Deskripsi Launch File pada Package skripsi_mocap

Launch File	Node	Subscribed Topic	Published Topic
my_skeleton	skeleton_tracker	-	skeleton
my_tracker	tracker_command	skeleton	-
	tracker_joint_controller	skeleton	-
		<joint_controller_name>/state	-
my_dynamixel	dynamixel_manager	-	<joint_controller_name>/state
		-	<joint_controller_name>/command
	dynamixel_controller_spawner_ax12	-	-
record	record_motion	<joint_controller_name>/state	-
play_recorded_motion	play_recorded_motion	<joint_controller_name>/state	-

Untuk melakukan proses imitasi secara real time, launch file yang perlu diaktifkan adalah my_skeleton.launch, my_dynamixel.launch, dan my_tracker.launch. File my_skeleton.launch berfungsi untuk mengaktifkan proses pembacaan citra tiga dimensi dari kinect. Dari file my_skeleton.launch tersebut akan muncul node skeleton_tracker yang mem-publish topic skeleton. Topik skeleton merupakan suatu paket data yang berisi data-data sebagai berikut:

```
Header header
int32 user_id
string[] name
float32[] confidence
geometry_msgs/Vector3[] position
geometry_msgs/Quaternion[] orientation
```

Dari subtopic position yang berupa vektor dimensi pada topik skeleton akan dicari solusi invers kinematik setiap joint pada robot. Representasi nilai x, y, dan z pada topic skeleton ditunjukkan dari frame kinect ditunjukkan pada gambar berikut dengan satuan meter.

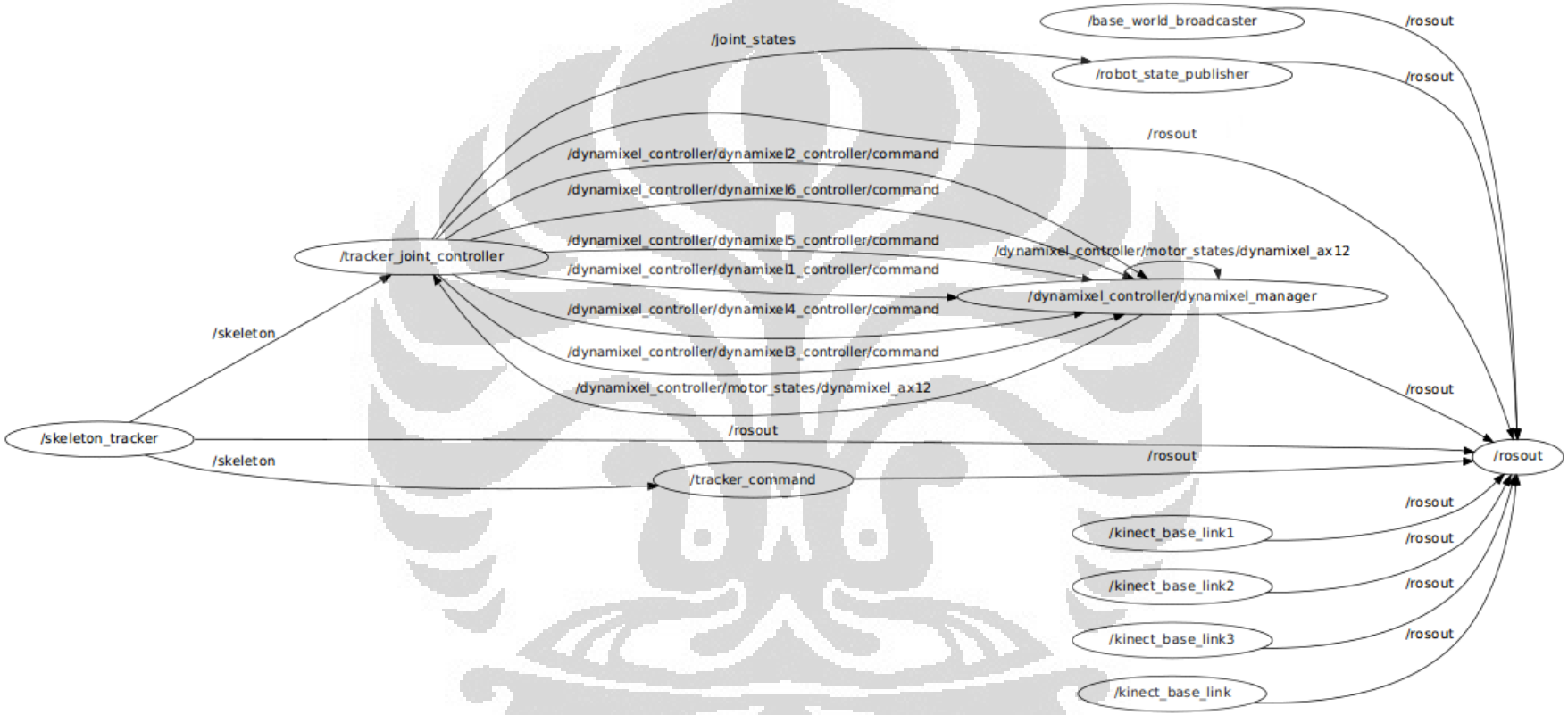


Gambar 4.5 Frame Microsoft Kinect

File kedua yang perlu diaktifkan adalah my_dynamixel.launch. Launch file ini akan mengaktifkan node dynamixel_manager dan dynamixel_spawner. Secara umum kedua node tersebut adalah node yang menerima dan mengirimkan data dari dan ke Dynamixel AX-12 melalui USB to Dynamixel. Node ini akan mempublish topic <joint_controller_name>/state dan <joint_controller_name>/command.

Launch file utama yang berperan penting adalah tracker2.launch. Launch file ini memanggil dua buah node, yaitu tracker_joint_controller dan tracker_command. Node tracker command sebagai untuk mengirimkan perintah mulai untuk melakukan tracking apabila kalibrasi tracking manusia dari node skeleton tracker telah berhasil dilakukan. Sedangkan node tracker_joint_controller berfungsi untuk melakukan mapping yang lebih jelasnya akan dijelaskan pada bagian setelah ini.

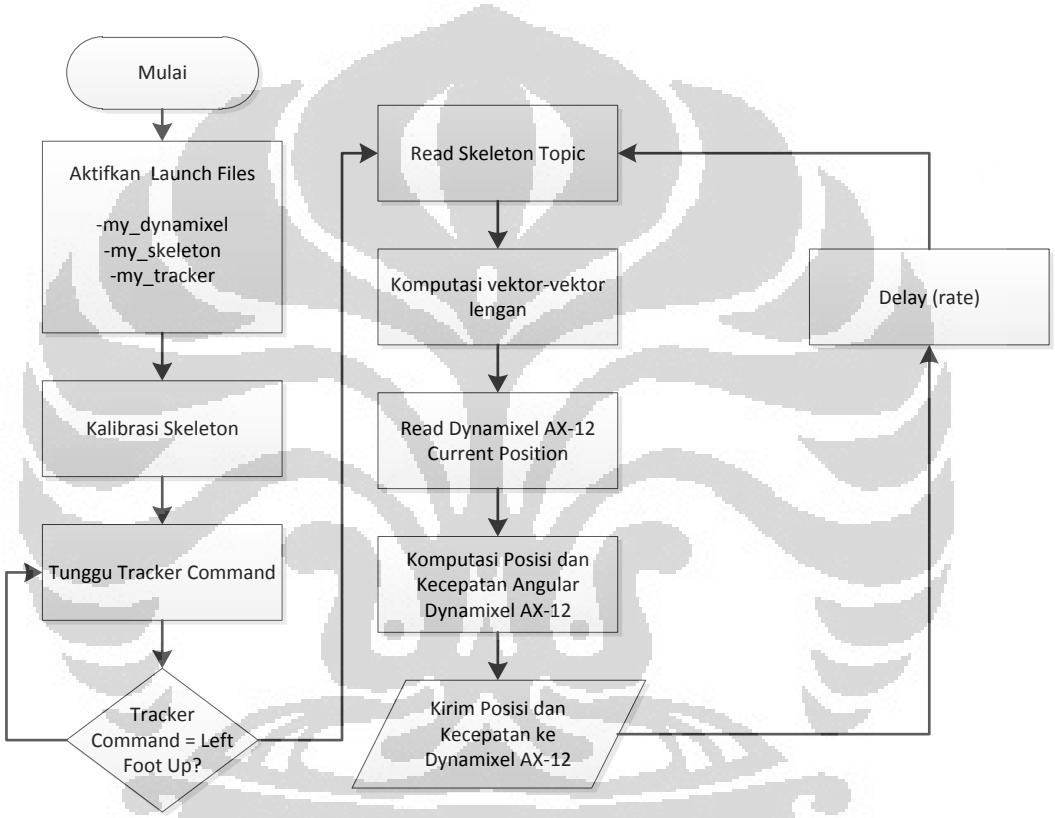
Contoh gambaran proses dari setiap node dalam melakukan proses imitasi secara real time ditunjukkan pada gambar berikut.



Gambar 4.6 Gambaran Node pada Proses Imitasi Secara Real Time

Dengan men-subscribe topic skeleton, node tracker_joint_controller berfungsi untuk mengolah data posisi dan orientasi dari topik skeleton menjadi data posisi angular sendi dan kecepatan angular pada robot. Kemudian data ini akan di-*publish* ke node dynamixel_manager untuk diberikan ke setiap dynamixel pada robot.

Diagram alir dari proses imitasi secara real time dijelaskan sebagai berikut.

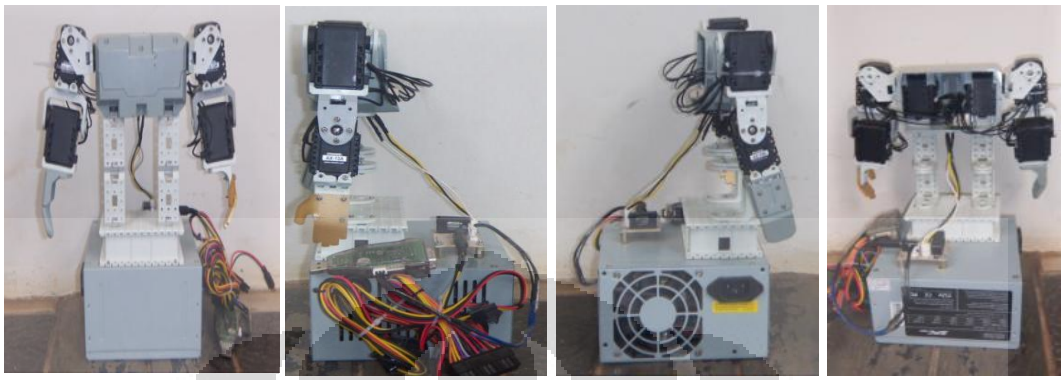


Gambar 4.7 Diagram Alir Proses Real Time Imitation

Salah satu keunggulan dari ROS adalah *topic* yang di-*publish* atau d-*isubscribe* oleh setiap node dapat diatur frekuensinya. Dalam penelitian ini, untuk melakukan tracking dan update posisi dynamixel digunakan frekuensi 20Hz. Sebenarnya, Kinect memiliki kemampuan untuk mengambil citra 3 dimensi dengan kecepatan kurang lebih 30 fps, tetapi untuk menjaga proses pengambilan citra agar masih dapat dilakukan, digunakan frekuensi kurang dari 30 Hz, yaitu 20 Hz.

4.2 Identifikasi Robot dan Proses Mapping

Berikut adalah penampakan model robot yang digunakan pada penelitian ini.



Gambar 4.8 Model Robot (tampak depan, tampak kiri, tampak kanan, tampak belakang)

Robot ini terdiri dari dua lengan dengan setiap lengan terdiri dari tiga derajat kebebasan. Pada bagian identifikasi model robot ini dijelaskan konfigurasi model robot, perbandingan konfigurasi lengan manusia dan model robot, persamaan model kinematik, serta solusi inverse kinematik model robot.

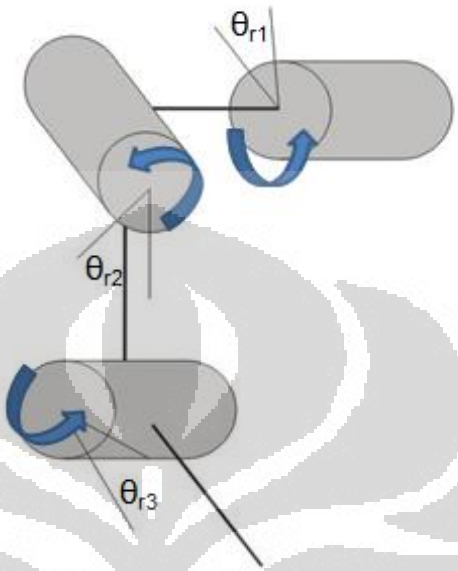
4.2.1 Konfigurasi Model Robot

Seperti yang telah ditunjukkan pada gambar 4.8 diatas. Konfigurasi robot yang digunakan memiliki dua lengan dengan setiap lengan terdiri dari dua derajat kebebasan. Posisi dari setiap dynamixel dan ID dari dynamixel pada setiap lengan ditunjukkan pada tabel berikut.

Tabel 4.2 ID dan Posisi Dynamixel Pada Model Robot

AX-12 ID	AX-12 Position	Sudut yang di bentuk
1	AX-12 Pertama pada bahu kanan	θ_{r1}
3	AX-12 Kedua pada bahu kanan	θ_{r2}
5	AX-12 pada siku kanan	θ_{r3}
2	AX-12 Pertama pada bahu kiri	θ_{l1}
4	AX-12 Kedua pada bahu kiri	θ_{l2}
6	AX-12 pada siku kiri	θ_{l3}

Gambar berikut menunjukkan tiga derajat kebebasan dari konfigurasi lengan kanan.



Gambar 4.9 Konfigurasi Lengan Kanan Model Robot

Secara lengkap, posisi robot dengan setiap sendi diberikan posisi normal (nilai digital 512 atau sudut 150^{o}) ditunjukkan pada gambar berikut.

Gambar 4.10 Konfigurasi Robot Pada Posisi Normal (150^o atau 512 dalam encoder unit)

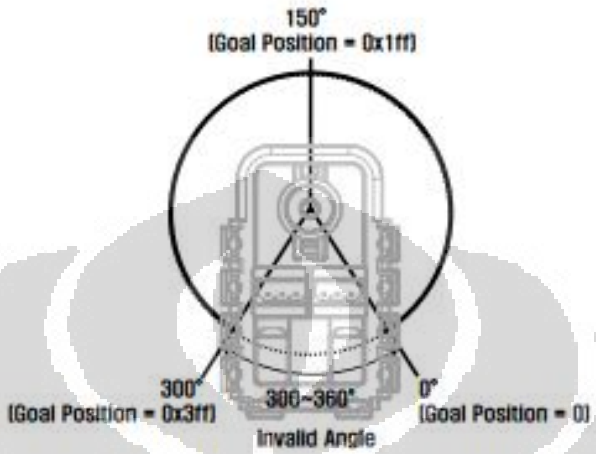
Struktur tangan ini cukup fleksibel dan secara umum dapat mencapai berbagai posisi dalam area kerjanya. Untuk mencegah gerakan diluar gerakan yang tidak dapat dicapai manusia, setiap joint pada lengan diberikan batasan-batasan posisi yang ditunjukkan pada gambar berikut.

Secara keseluruhan, batasan dari gerakan lengan tersebut dapat dilihat pada tabel berikut.

Tabel 4.3 Tabel Batasan Gerakan Dynamixel pada Model Robot

	ID	CW Angle Limit		CCW Angle Limit	
		Digital Command	Degree	Digital Command	Degree
Tangan kanan	1	0	0^o	1023	300^o
	3	205	60^o	819	240^o
	5	205	60^o	511	60^o
Tangan kiri	2	0	0^o	1023	300^o
	4	205	60^o	819	240^o
	6	511	150^o	819	240^o

Area kerja dari Dynamixel AX-12 sebelum dilakukan pembatasan posisi adalah 300^o, dengan resolusi perintah digital 10 bit. Memberikan perintah 511 akan membuat dynamixel bergerak diposisi tengah (150^o). Area kerjaDynamixel AX-12 ditunjukkan pada gambar berikut.



Gambar 4.11 Range Kerja Dynamixel AX-12

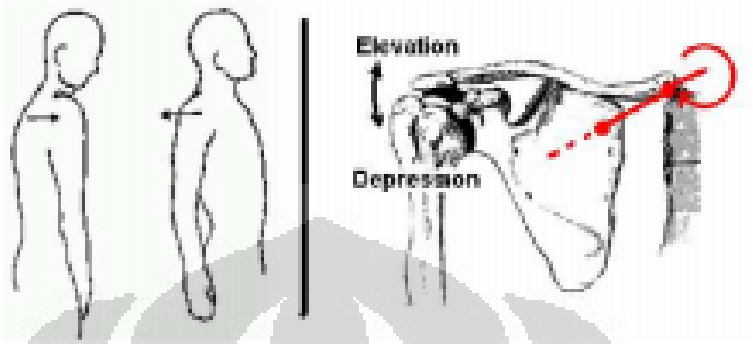
4.2.2 Perbandingan Konfigurasi Lengan Manusia dan Model Robot

Lengan manusia memiliki mekanisme yang terdiri dari sendi peluru dan sendi engsel dengan sumbu-sumbu rotasi untuk melakkan gerakan abduksi-adduksi, fleksi-ekstensi, dan rotasi internal pada lengan atas. Pada bagian ini akan dijelaskan bagaimana algoritma mapping ini dikembangkan, yaitu dengan menggunakan informasi dari $vektor_{shoulder\_wrist}$ yang didapat dari proses *tracking*. Pada setiap loop, informasi yang didapat dari vektor ini akan diproses untuk mendapatkan sudur dan kecepatan angular dari setiap servo Dynamixel pada lengan.

Hal pertama yang perlu diperhatikan adalah prosedur untuk mengadaptasi gerakan dari manusia ke robot. Pada penelitian ini, digunakan tiga buah servo untuk setiap lengan robot agar dapat menyerupai posisi pergerakan lengan pada manusia. Pada bagian ini akan disajikan gerakan manusia yang terdapat pada sendi bahu dan sendi lengan. Bahu manusia terdiri dari tiga tulang, yaitu clavicle, scapula, dan humerus. Artikulasi dari tiga tulang tersebut membentuk sendi-sendi pada bahu. Pergerakan utama pada bahu adalah sebagai berikut:

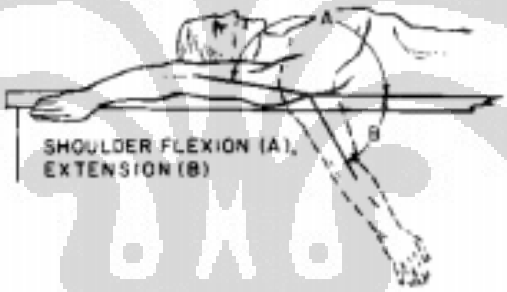
Scapular retraksi dan protaksi: Pergerakan ini mengizinkan bahu untuk bergerak secara posterior dan anterior

Scapular elevasi dan depresi: scapula naik atau turun bahu



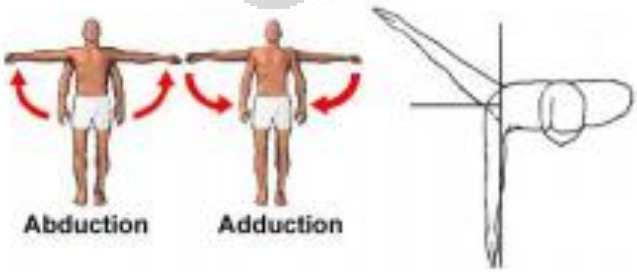
Gambar 4.12 Gerakan Scapular Elevasi dan Depresi

Fleksi dan Ekstensi: lengan berputar mengelilingi sumbu parallel menuju klavikula melalui kedua bahu,



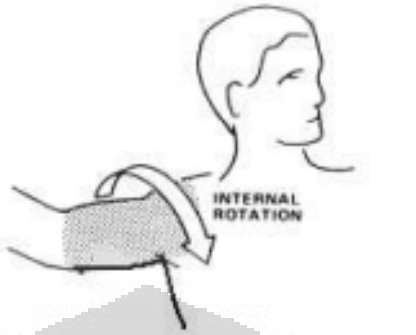
Gambar 4.13 Gerakan Fleksi dan Ekstensi pada Bahu

Abduksi dan Adduksi Bahu - Abduksi adalah gerakan lengan menjauhi tubuh. Sedangkan adduksi adalah gerakan sebalikknya ketika lengan mendekati tubuh.



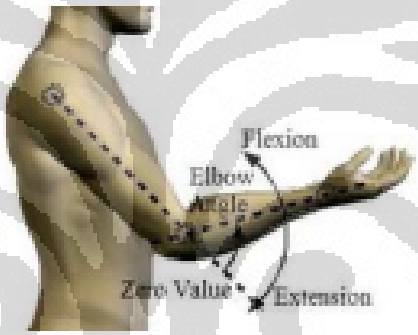
Gambar 4.14 Gerakan Abduksi dan Adduksi pada Bahu

Rotasi internal dari lengan sepanjang sumbu longitudinal.



Gambar 4.15 Rotasi Internal Pada Lengan

Sedangkan pergerakan utama dari siku adalah Flexi dan extensi antara humerus dan ulna seperti pada gambar berikut.

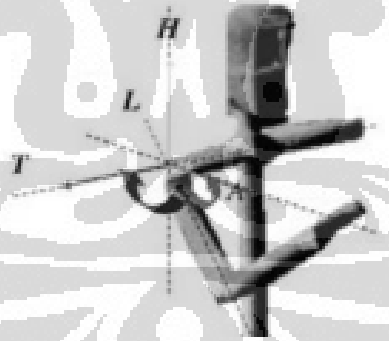


Gambar 4.16 Gerakan Fleksi dan Ekstensi pada Manusia

Pronasi dan Supinasi: pergerakan memutar dari lengan bawah mengelilingi longitudinal axis yang membuat telapak tangan menghadap atas atau bawah. Artikulasi antara siku dan pergelangan tangan berperan dalam gerakan ini.

Gambar 4.17 Gerakan Supinasi dan Pronasi

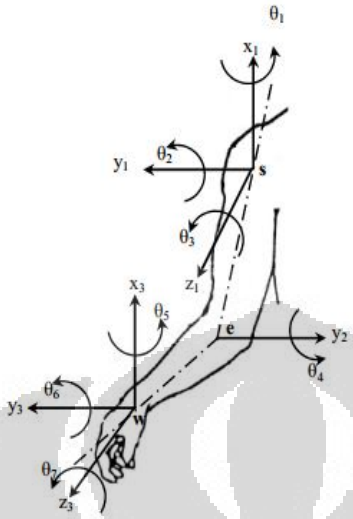
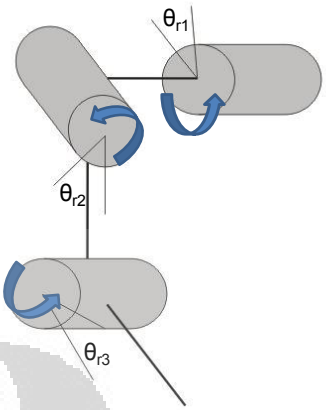
Seperti yang telah dilihat, pergerakan manusia memiliki kompleksitas yang tinggi untuk memuat kemungkinan-kemungkinan gerakan yang ada. Namun, robot yang dibuat pada penelitian ini memiliki struktur lengan yang simpel dengan derajat kebebasan yang lebih sedikit dibandingkan manusia. Untuk memetakan posisi manusia menjadi posisi robot perlu dilakukan pengelompokan gerakan yang sama yang mungkin dilakukan oleh robot dan manusia. Karena model robot hanya memiliki tiga sendi di setiap lengan, setiap lengan robot dirancang hanya untuk melakukan gerakan flexi dan extensi pada bahu, abduksi dan adduksi pada bahu, serta fleksi dan extensi pada siku. Gerakan tiga dimensi dari bahu dapat dilihat pada gambar dibawah ini dimana servo pertama (Dynamixel 1) membuat robot melakukan gerakan fleksi dan extensi yang direpresentasikan oleh sudut antara sumbu L dan sumbu A. Servo kedua (Dynamixel 3) membuat robot dapat melakukan gerakan abduksi dan adduksi yang direpresentasikan oleh sudut yang dibentuk antara sumbu L dan T, bidang yang dibentuk tergantung oleh sudut fleksi/ekstensi, karena servo ini merupakan sambungan dari servo pertama.



Gambar 4.18 Visualisasi Gerakan Dynamixel pada Lengan

Perbandingan antara gerakan yang dapat dilakukan oleh manusia dan model robot yang digunakan dalam penelitian ini disajikan dalam tabel berikut:

Tabel 4.4 Perbandingan Lengan Manusia dan Model Robot

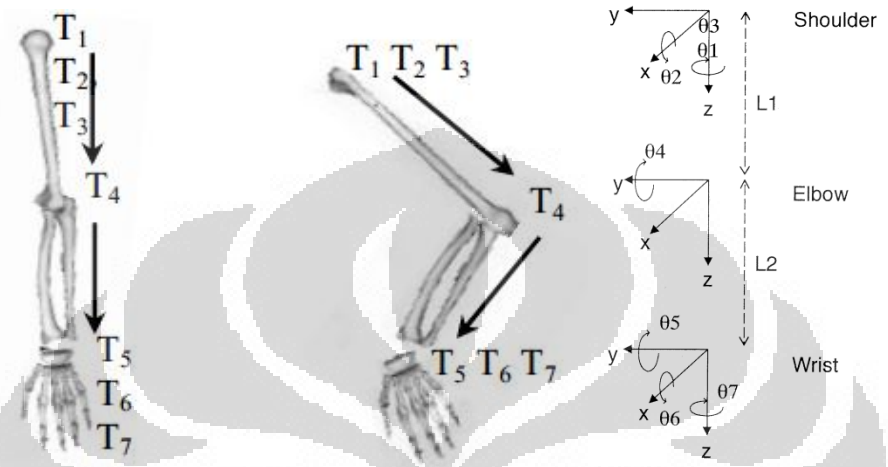
	Manusia	**Model Robot**
Gambaran Fisik		
Jumlah DOF	7 DOF	3 DOF
Cakupan gerakan	▪ Fleksi – Ekstension Bahu ▪ Abduksi – Adduksi Bahu ▪ Rotasi Internal Bahu ▪ Fleksi – Ekstensi Siku ▪ Pronasi dan Supinasi	▪ Fleksi – Ekstension Bahu ▪ Abduksi – Adduksi Bahu ▪ Fleksi – Ekstensi Siku

Dari perbandingan pada table diatas, sudut-sudut yang ada pada model robot, yaitu θ_{r1}, θ_{r2}, dan θ_{r3} merepresentasikan sudut θ_2, θ_3, dan θ_4 pada manusia. Setelah mendapatkan representasi sudut dari manusia dapat dicari persamaan forward kinematik dari bahu hingga ujung tangan pada model robot. Dari persamaan forward kinematik ini akan diturunkan persamaan inverse kinematik yang akan digunakan untuk memetakan koordinat yang didapat dari perangkat capture Microsoft Kinect menjadi sudut-sudut pada model robot.

4.2.3 Persamaan Model Kinematik Lengan Robot

Pada gambar konfigurasi lengan manusia yang terdapat pada table 4.4 dapat dilihat pada lengan manusia memiliki 7 DOF yang memungkinkan lengan manusia untuk melakukan gerakan-gerakan. Dari 7 DOF tersebut dapat dibuat Matrix Forward Kinematik yang memetakan posisi telapak tangan dari bahu (bahu

sebagai world frame). Matrix Forward Kinematik ini dapat direpresentasikan sebagai 4x4 matrix homogenous yang memperlihatkan rotasi dan translasi dari setiap frame pada robot. Untuk mendapatkan matrix ini digunakan parameter Denavit Hartenberg pada setiap frame.



Gambar 4.19 Transformasi Kinematik Pada Lengan Manusia

Gambar diatas menunjukkan rangkaian kinematik dilengkapi dengan transisi setiap frame. Matrix Forward Kinematik dari bahu hingga telapak tangan didapatkan dengan mengalikan seluruh matrix transformasi dari T_1 sampai dengan T_7.

$$G_{human} = T_1\ T_2\ T_3\ T_4\ T_5\ T_6\ T_7$$

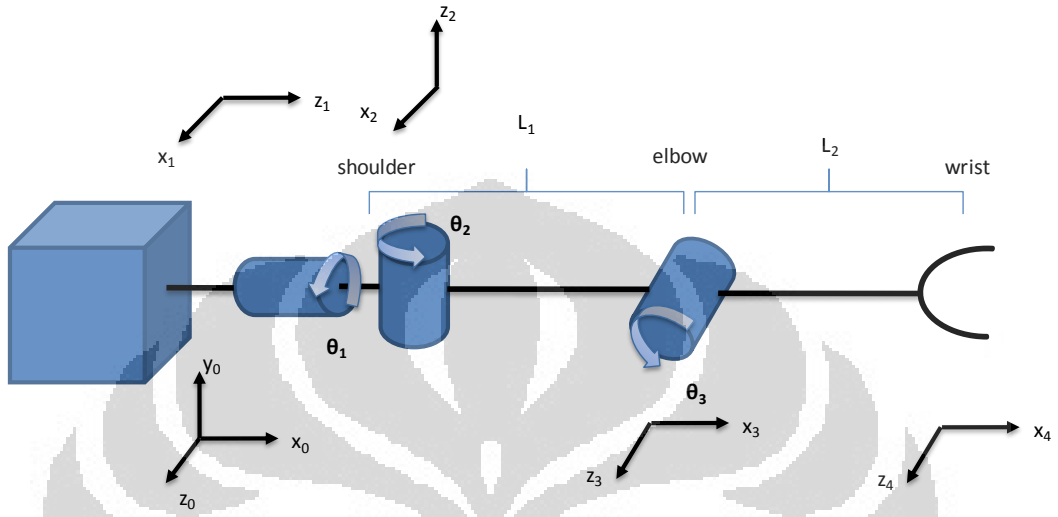
Matrix G merepresentasikan orientasi dan posisi dari end effector (telapak tangan) pada world frame dengan bagian rotasi direpresentasikan pada matrix R_g berdimensi 3x3 dan bagian translasi direpresentasikan dalam bagian t_g 3x1.

$$G_{4x4} = \begin{bmatrix} & R_g & & t_g \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Namun, karena pada model robot hanya memiliki tiga buah joint, maka matrix homogeneus transform G_{model_robot} adalah:

$$G_{model\_robot} = T_2\ T_3\ T_4$$

Untuk mencari model kinematik lengan model robot, dapat digunakan konfigurasi lengan seperti yang ditunjukkan pada gambar berikut.



Gambar 4.20 Konfigurasi Lengan Kanan Robot dan Transformasinya

Pergerakan rotasi sumbu-sumbu setiap joint dalam gambar diatas adalah sama dengan model robot. Berikut matrix transformasi dari konfigurasi lengan kanan pada gambar. Transformasi pada matrix-matrix berikut mengacu pada koordinat frame 0. Arah-arah sumbu koordinat kartesian pada frame 0 mengacu kepada frame yang dibaca oleh kinect.

$$T_1 = R_x(\theta_1) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta_1) & -\sin(\theta_1) & 0 \\ 0 & \sin(\theta_1) & \cos(\theta_1) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_2 = R_y(-\theta_2)T(L1,0,0) = \begin{bmatrix} \cos(-\theta_2) & 0 & \sin(-\theta_2) & L1 \\ 0 & 1 & 0 & 0 \\ -\sin(-\theta_2) & 0 & \cos(-\theta_2) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_3 = R_z(\theta_3)T(L2,0,0) = \begin{bmatrix} \cos(\theta_3) & -\sin(\theta_3) & 0 & L2 \\ \sin(\theta_3) & \cos(\theta_3) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Maka matrix transformasi model robot G adalah:

$$G = T_1 T_2 T_3$$

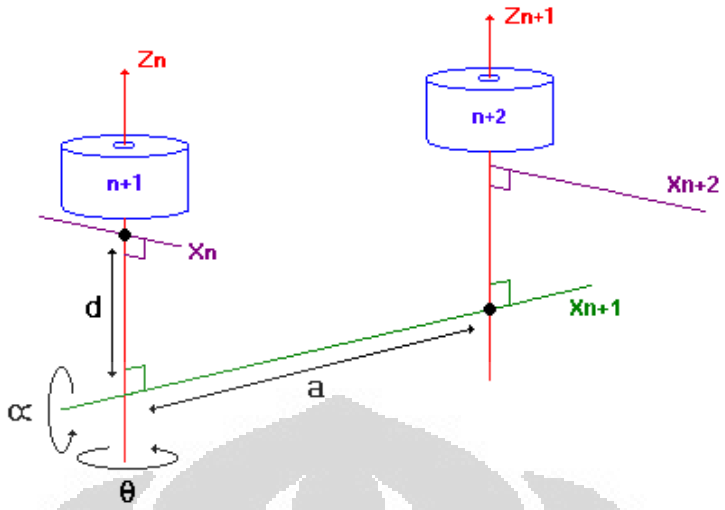
$$G = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta_1) & -\sin(\theta_1) & 0 \\ 0 & \sin(\theta_1) & \cos(\theta_1) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(-\theta_2) & 0 & \sin(-\theta_2) & L1 \\ 0 & 1 & 0 & 0 \\ -\sin(-\theta_2) & 0 & \cos(-\theta_2) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta_3) & -\sin(\theta_3) & 0 & L2 \\ \sin(\theta_3) & \cos(\theta_3) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$G = \begin{bmatrix} g11 & g12 & g13 & g14 \\ g21 & g22 & g23 & g24 \\ g31 & g32 & g33 & g34 \\ g41 & g42 & g43 & g44 \end{bmatrix}$$

Nilai koordinat x, y, dan z dari end effector terhadap frame 1 ditunjukkan pada nilai elemen matrix g14, g24, dan g34. Nilai elemen matrix g14, g24, dan g34 adalah fungsi dari L1, L2, θ_1, θ_2, dan θ_3. Berikut adalah persamaan x, y, dan z. Persamaan ini merupakan persamaan forward kinematik dari model robot.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} g31 \\ g32 \\ g33 \end{bmatrix} = \begin{bmatrix} L_1 \cos(-\theta_2) + L_2 \cos(-\theta_2)\cos(-\theta_3) \\ L_1 \sin(\theta_1)\sin(-\theta_2) + L_2 \cos(\theta_1)\sin(-\theta_3) + L_2 \sin(\theta_1)\sin(-\theta_2)\cos(-\theta_3) \\ -L_1 \cos(\theta_1)\sin(-\theta_2) + L_2 \sin(\theta_1)\sin(-\theta_3) - L_2 \cos(\theta_1)\sin(-\theta_2)\cos(-\theta_3) \end{bmatrix}$$

Untuk memvalidasi nilai matrix G dapat digunakan parameter Denavit-Hartenberg. Metode Denavit Hartenberg diusulkan oleh R.S. Denavit dan J. Hartenberg pada tahun 1947 digunakan untuk menyederhanakan dan standarisasi pemodelan robot manipulator. Metode ini menggunakan empat parameter utama dengan tiga parameter tetap dan satu variable parameter. Keempat parameter ini adalah α_i, a_i, d_i, dan θ_i.



Gambar 4.21 Penentuan Parameter DH pada Link Robot

Keempat parameter DH, yaitu α_i, a_i, d_i, dan θ_i dapat dijelaskan sebagai berikut.

a_i : jarak antara z_i menuju z_{i+1} sepanjang sumbu x_i

α_i : sudut antara z_i menuju z_{i+1} terhadap sumbu x_i

d_i : jarak antara x_{i-1} menuju x_i sepanjang sumbu z_i

θ_i : sudut antara x_{i-1} menuju x_i terhadap sumbu z_i

Keempat parameter dari setiap joint pada robot akan direpresentasikan dalam bentuk matrix Denavit Hartenberg sebagai berikut.

$$
\begin{aligned}
{}^{i-1}_{i}T_{(\alpha_{i-1},a_{i-1},\theta_i,d_i)} &= R_x(\alpha_{i-1})D_x(a_{i-1})R_z(\theta_i)D_z(d_i) \\
&= \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & a_{i-1} \\ \sin\theta_i & \cos\theta_i\cos\alpha_{i-1} & -\sin\alpha_{i-1} & -\sin\alpha_{i-1}d_i \\ \sin\theta_i & \cos\theta_i\sin\alpha_{i-1} & \cos\alpha_{i-1} & \cos\alpha_{i-1}d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned}
$$

Dalam mencari nilai parameter tersebut terdapat aturan dasar penggunaan sumbu dalam metode Denavit Hartenberg. Berikut adalah aturan sumbu yang digunakan.

Sumbu x :

- Menuju ke arah lengan
- Menuju ke panjang lengan

Sumbu y :

- Tidak digunakan
- Arah disesuaikan dengan sumbu x dan z

Sumbu z :

- Menuju ke tinggi lengan
- Sebagai sumbu perputaran poros (engsel)

Nilai parameter Denavit-Hartenberg pada model konfigurasi lengan robot yang ditunjukkan pada gambar 4.13 adalah sebagai berikut:

Tabel 4.5 Parameter Denavit Hartenberg Konfigurasi Model Robot

I	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	θ_1^o
2	-90	0	0	$-\theta_2$
3	90	L1	0	$-\theta_3$
4	0	L2	0	0

Dari hasil perhitungan D-H yang telah dilakukan, nilai koordinat x, y, dan z yang didapat dari metode Denavit-Hartenberg telah sama dengan menggunakan penghitungan matrix transformasi model robot.

4.2.4 Solusi Inverse Kinematik Model Robot

Inverse kinematik merupakan sebuah metode yang menjawab berapakah sudut yang harus diberikan jika end effector harus berada di posisi tertentu. Sudut hasil dari solusi inverse kinematik ini nantinya akan dikirimkan kepada model robot. Untuk menghitung solusi inverse kinematik perlu diketahui vektor posisi dari posisi bahu (*shoulder*), posisi siku (elbow), dan posisi telapak tangan (*wrist*).

Dari koordinat vektor tiga dimensi titik-titik pada skeleton yang didapatkan dari proses tracking akan dicari vektor-vektor yang membantu dalam mencari persamaan solusi inverse kinematik. Vektor-vektor tersebut adalah:

$$v_{shoulder\_elbow} = v_{shoulder} - v_{elbow}$$

$$v_{shoulder\_wrist} = v_{shoulder} - v_{wrist}$$

$$v_{elbow\_wrist} = v_{elbow} - v_{wrist}$$

Mengacu pada gambar 4.13, matrix transformasi forward kinematik dari bahu ke siku dengan acuan frame 0 dapat dinyatakan dengan:

$$G_{shoulder\_elbow} = T_1 T_2 = [Rot_x(\theta_1)][Rot_y(-\theta_2)transl(L_1,0,0)]$$

$$G_{shoulder\_elbow} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta_1) & -\sin(\theta_1) & 0 \\ 0 & \sin(\theta_1) & \cos(\theta_1) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(-\theta_2) & 0 & \sin(-\theta_2) & L_1 \\ 0 & 1 & 0 & 0 \\ -\sin(-\theta_2) & 0 & \cos(-\theta_2) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}_{shoulder\_elbow} = \begin{bmatrix} L_1 \cos(-\theta_2) \\ L_1 \sin(\theta_1) \sin(-\theta_2) \\ -L_1 \cos(\theta_1) \sin(-\theta_2) \end{bmatrix}$$

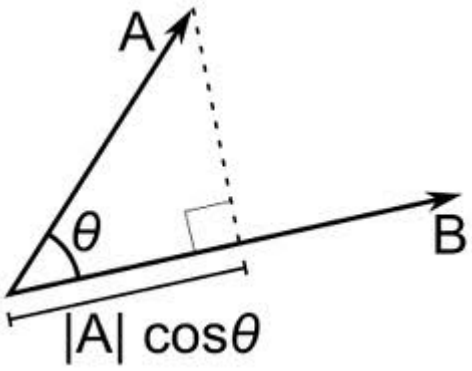
Dari persamaan diatas dapat dicari solusi inverse kinematik dari θ_1 dan θ_2. Yaitu:

$$\theta_2 = -\arccos\left(\frac{x_{shoulder\_elbow}}{L1}\right)$$

$$\theta_1 = \arcsin\left(\frac{y_{shoulder\_elbow}}{\sin(\theta_2)}\right)$$

dengan L_1 merupakan panjang dari $v_{shoulder\_elbow}$. Sudut θ_1 merupakan sudut yang membuat lengan mampu melakukan gerakan fleksi dan ekstensi, sedangkan sudut θ_2 adalah sudut yang membuat lengan dapat melakukan gerakan abduksi dan adduksi.

Sudut θ_3 merupakan sudut internal siku, yaitu sudut yang dibentuk antara $v_{shoulder\_elbow}$ dan v_{elbow_wrist}. Untuk mencari θ_3 dilakukan dengan memanfaatkan persamaan dot product vektor. Dalam geometri Euclidean, sudut yang dibentuk antara dua vektor adalah perkalian dot dari vektor yang dinyatakan dalam sebuah basis ortonormal berhubungan dengan panjang dan sudut yang dibentuk antara vektor tersebut. Hubungan tersebut direpresntasikan pada gambar berikut.



Gambar 4.22 Representasi Dot Vektor dalam Geometri Euclidean

Jika A dan B adalah sebuah vektor dan θ adalah sudut yang dibentuk kedua vector tersebut, maka dot product dari vector A dan B dinyatakan sebagai:

$$A \bullet B = \|A\|\|B\| \cos\theta$$

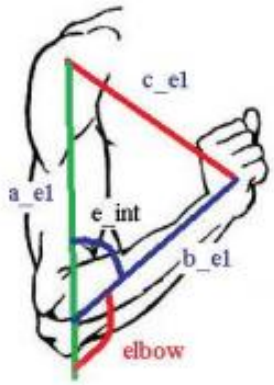
Sehingga nilai sudut θ dapat dicari dengan persamaan berikut.

$$\theta = \arccos\left(\frac{A \bullet B}{\|A\|\|B\|}\right)$$

Bentuk $\frac{A}{\|A\|}$ merupakan bentuk vektor satuan $\hat{A}$, sehingga sudut θ dapat dinyatakan sebagai:

$$\theta = \arccos\left(\hat{A} \bullet \hat{B}\right)$$

Untuk mencari nilai θ_3 dapat dilihat pada gambar 4.16 berikut. Panjang a_el, b_el, dan _c_el merupakan panjang dari vektor $v_{shoulder\_elbow}$, v_{elbow_wrist}, dan $v_{shoulder\_wrist}$. Nilai θ_3 merupakan sudut e_int pada gambar 4.16. Sudut θ_3 merupakan arcus cosines dari perkalian dot product $v_{shoulder\_elbow}$ dan v_{elbow_wrist}.



Gambar 4.23 Representasi Vektor pada Lengan Manusia

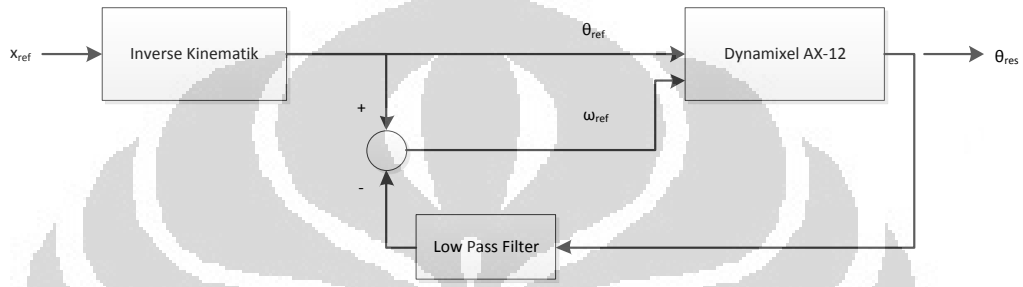
Solusi inverse kinematik θ_1, θ_2, dan θ_3 ini dapat berlaku pada lengan bagian kiri maupun lengan bagian kanan dapat digunakan cara yang sama, nilai setiap sudut yang diberikan pada setiap ID dynamixel AX-12 dapat dilihat pada tabel berikut.

Tabel 4.6 Solusi Inverse Kinematik Setiap Joint

	Sudut Joint Dynamixel	Solusi Invers Kinematik
Lengan Kanan	θ_{r1}	$\theta_{r1} = \arcsin\left(\frac{y_{right\_shoulder\_elbow}}{\sin(\theta_2)}\right)$
	θ_{r2}	$\theta_{r2} = -\arccos\left(\frac{x_{right\_shoulder\_elbow}}{L1}\right)$
	θ_{r3}	$\theta_{r3} = \arccos\left(\hat{v}_{right\_shoulder\_elbow} \cdot \hat{v}_{right\_elbow\_wrist}\right)$
Lengan Kiri	θ_{l1}	$\theta_{l1} = \arcsin\left(\frac{y_{left\_shoulder\_elbow}}{\sin(-\theta_{l2})}\right)$
	θ_{l2}	$\theta_{l2} = \arccos\left(\frac{-x_{left\_shoulder\_elbow}}{L1}\right)$
	θ_{l3}	$\theta_{l3} = \arccos\left(\hat{v}_{left\_shoulder\_elbow} \cdot \hat{v}_{left\_elbow\_wrist}\right)$

4.2.5 Proses Pemetaan Kecepatan Angular

Hasil solusi inverse kinematik dari proses sebelumnya yang ditunjukkan pada tabel 4.6 akan dikirimkan kepada setiap dynamixel pada robot. Namun, untuk membuat robot mampu melakukan gerakan yang lebih mirip dengan manusia perlu ditambahkan dimensi kecepatan angular. Blok diagram proses pengiriman posisi angular dan kecepatan angular ditunjukkan pada gambar berikut.



Gambar 4.24 Blok Diagram Proses Pengendalian Posisi dan Kecepatan Dynamixel

Nilai vektor x_{ref} merupakan nilai vektor-vektor yang didapat dari proses skeleton tracking yang telah dijelaskan pada bagian sebelumnya. Posisi angular yang akan dikirimkan ke dynamixel sebagai θ_{ref} didapatkan dari blok solusi invers kinematik. Kecepatan angular yang akan dikirimkan ke robot dirumuskan sebagai berikut:

$$\omega_{ref} = \frac{\theta_{ref} - \theta_{old}}{rate}$$

Dengan *rate* merupakan waktu sampling yang digunakan yaitu 0.05 detik pada penelitian ini.

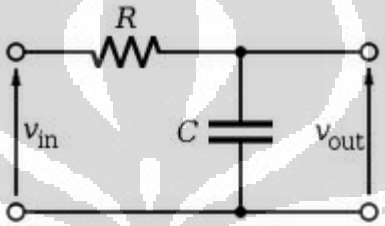
Pada blok diagram diatas, pembacaan feedback θ_{res} oleh sensor posisi dari dynamixel dapat dilakukan dengan mensubscribe topik MotorState. Sama seperti topic skeleton, topik MotorState merupakan paket data dengan format sebagai berikut.

```
Float64 timestamp     #motor state is at this time
int32 id              #motor id
int32 goal            #commanded position (in encoder units)
int32 position        #current position (in encoder units
```

```
int32 error           #difference between current and goal pos
int32 speed           #current speed (0.111 rpm per unit)
float64 load          #current load
float64 voltage       #current voltage
int32 temperature     #current temperature (degrees celcius)
bool moving           #whether the motor is currently in motion
```

Dalam melakukan pengambilan feedback dari θ_{res} diberikan sebuah low pass filter dari pembacaan subtopic position untuk meredam gangguan-gangguan pada pembacaan data yang didapat.

Realisasi filter ini dalam bentuk diskrit didasari dari rangkaian berikut.



Gambar 4.25 Rangkaian Low Pass Filter Sederhana

Dari rangkaian diatas dengan menggunakan hukum Kirchoff didapatkan persamaan berikut.

$$v_{in}(t) - v_{out}(t) = Ri(t)$$

Dengan *i(t)* adalah

$$i(t) = C\frac{dv_{out}}{dt}$$

Sehingga

$$v_{in}(t) - v_{out}(t) = RC\frac{dv_{out}}{dt}$$

Persamaan diatas dapat diubah dalam bentuk diskrit dengan mengambil sampel input dan output dalam suatu jeda waktu Δt. Misalkan v_{in} direpresentasikan dalam sebuah sekuens input (x_1, x_2, x_3, …, x_n) dan v_{out} direpresentasikan dalam (y_1, y_2, y_3, …, y_n). Maka dapat dibentuk sebuah persamaan diskrit sebagai berikut.

$$x_i - y_i = RC\frac{y_i - y_{i-1}}{\Delta t}$$

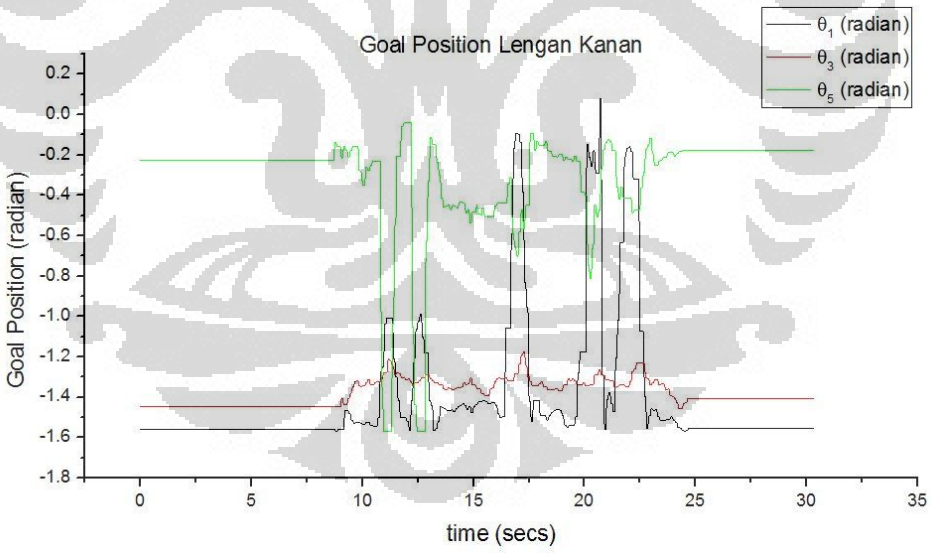
Dengan mengolah persamaan diatas, dapat dibentuk persamaan keluaran low pass filter sebagai berikut.

$$y_i = x_i \overbrace{\left(\frac{\Delta t}{RC + \Delta t}\right)}^{kontribusi\_input} + y_{i-1} \overbrace{\left(\frac{RC}{RC + \Delta t}\right)}^{inertia\_dari\_input\_sebelumnya}$$
$$= \alpha x_i + (1 - \alpha) y_{i-1}$$

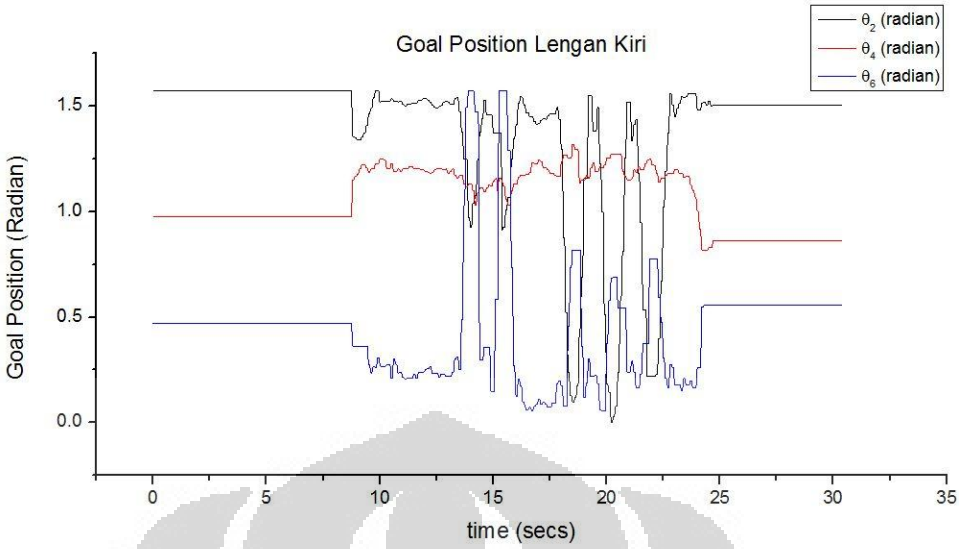
Dengan $\alpha = \frac{\Delta t}{RC + \Delta t}$ adalah *smoothing factor* yang memiliki nilai $0<\alpha<1$.

4.2.6 Penentuan Smoothing Factor (α) pada Low Pass Filter

Pada bagian ini akan dilakukan proses penentuan nilai dari *smoothing factor* ini. Nilai terbaik adalah nilai keluaran yang paling kecil terpengaruhi *noise*, atau nilai dengan error terkecil. Percobaan dilakukan dengan memberikan setpoint posisi pada setiap servo kemudian mencari error rata-rata dari setiap posisi sudut yang terbaca. Setpoint posisi yang diberikan adalah sebagai berikut.



Gambar 4.26 Goal Position Lengan Kanan



Gambar 4.27 Goal Position Lengan Kiri

Dari set point posisi diatas, dilakukan pembacaan hasil low pass filter dengan berbagai variasi α. Error rata-rata dari setiap sudut ditunjukkan pada table berikut.

Tabel 4.7 Pengujian Variasi Smoothing Factor

α (Smoothing Factor)	Error Rata-Rata Per Dynamixel (Radian)						Error Rata-Rata Keseluruhan
	θ1	θ2	θ3	θ4	θ5	θ6	
0.1	0.1998	-0.3442	0.4600	-0.3339	0.1828	-0.0429	0.0203
0.2	0.2717	-0.5311	0.6004	-0.4210	0.2240	-0.0158	0.0214
0.3	0.2715	-0.4054	0.5046	-0.3687	0.2383	-0.0274	0.0355
0.4	0.1958	-0.4027	0.5117	-0.3704	0.2471	-0.0511	0.0217
0.5	0.1443	-0.2975	0.4373	-0.3103	0.1621	-0.0085	0.0212
0.6	0.3657	-0.6419	0.7754	-0.5493	0.3090	0.0023	0.0435
0.7	0.1930	-0.3405	0.5019	-0.3482	0.2134	0.0009	0.0368
0.8	0.2274	-0.4194	0.4976	-0.3589	0.1949	-0.0133	0.0214
0.9	0.2365	-0.4179	0.4650	-0.3698	0.2100	-0.0065	0.0196
1	0.2589	-0.4170	0.4938	-0.3546	0.2305	0.0084	0.0367

Dari table tersebut perubahan variasi alpha tidak menunjukkan sebuah pola tertentu. Error terkecil terjadi pada nilai alpha sebesar 0.9. Pada percobaan ini digunakan frekuensi sampling sebesar 20 Hz atau periode sebesar 50 ms. Semakin besar nilai alpha semakin kecil nilai inertia dari keluaran sebelumnya. Konstanta waktu (RC) pada nilai alpha 0.9 dengan periode sampling sebesar 50 ms adalah 0.00556. Nilai alpha 0.9 ini akan digunakan sebagai nilai tetap dalam sistem motion capture.

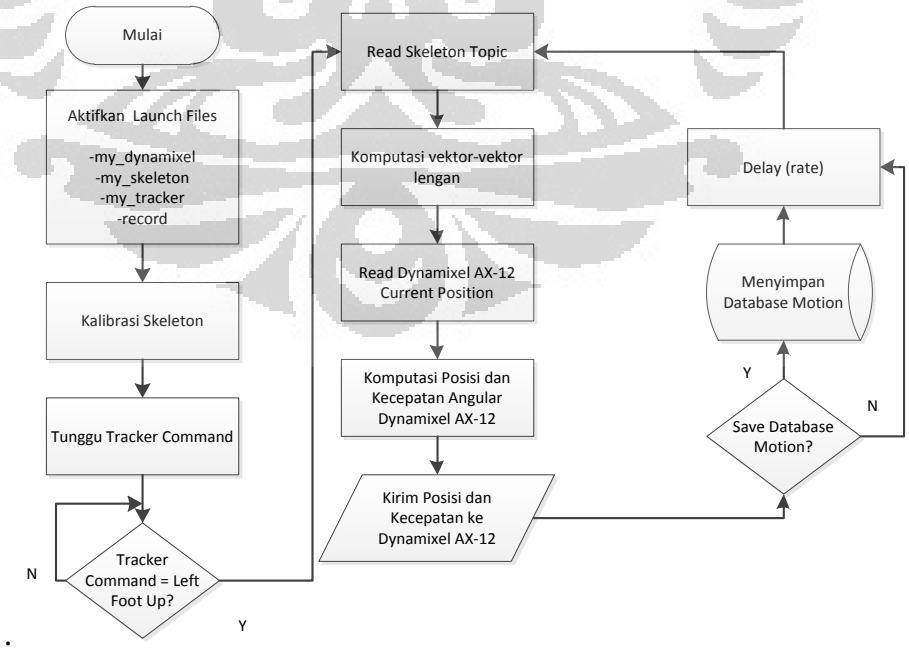
BAB 5
PERANCANGAN DATABASE MOTION

Dalam proses pembelajaran dengan metode imitasi, robot harus diberikan kemampuan untuk mampu melakukan pengulangan gerakan yang diimitasinya. Untuk itu perlu dibuat sebuah modul untuk merekam gerakan manusia. Database hasil rekaman ini mampu untuk dimainkan kembali oleh robot. Bagian ini menjelaskan proses perancangan dan proses ekstraksi motion yang telah disimpan ke robot.

5.1 Model Perancangan Database Motion

Hal terpenting dari proses imitasi adalah agen dapat meniru perilaku aktor berulang-ulang pada saat kapan pun. Agar robot memiliki kecerdasan untuk mampu melakukan gerakan yang telah diajarkan kepadanya, perlu dirancang sebuah database motion yang merekam gerakan manusia yang akan diimitasi oleh robot.

Untuk merekam database dirancang sebuah launch file record.launch yang didalamnya terdapat node record.py. Berikut diagram alir untuk melakukan proses perekaman database motion



Gambar 5.1 Diagram Alir Proses Perekaman Database Motion

Data yang direkam pada database ini adalah data sudut referensi (goal_position) yang akan dikir imkan ke dynamixel. Database disimpan dalam file berekstensi .csv dengan format penyimpanan per baris yang ditunjukkan pada gambar 5.2.

Time	Goal_Pos[0]	Goal_Pos[1]	Goal_Pos[2]	Goal_Pos[3]	Goal_Pos[4]	Goal_Pos[5]

Gambar 5.2 Format Penyimpanan database Motion

CSV adalah salah satu bentuk file text yang digunakan untuk menyimpan data dalam struktur table. Data yang tersimpan dalam CSV disimpan dalam tiap baris dengan setiap kolom dipisahkan dengan karakter pemisah (*delimiter*). Karakter pemisah bisa dalam betuk koma (,), titik koma (;), atau karakter tabulasi.

Nilai Time´didapatkan dari fungsi `rospy.get_rostime()` dengan return varaiabel berupa time.secs dan time.nsecs. Sedangkan untuk Goal_Pos didapat dengan men-subscribe topik MotorState dari setiap dynamixel.

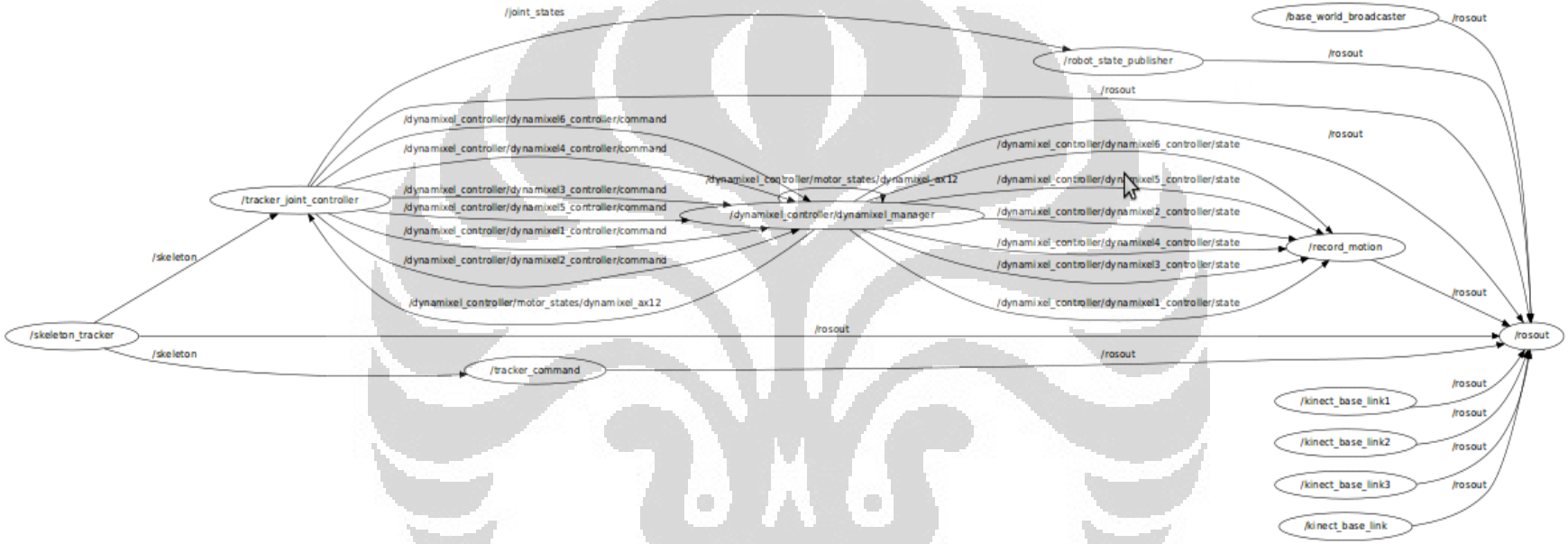
Contoh hasil database yang telah disimpan, ditunjukkan pada gambar berikut.

goal_pos6.11.2.csv | goal_pos6.11.3.csv

```
153  1339562698.937511920,-1.570,1.570,-1.570,1.570,0.000,0.000
154  1339562698.987543106,-1.570,1.570,-1.570,1.570,0.000,0.000
155  1339562699.037475109,-1.570,1.570,-1.570,1.570,0.000,0.000
156  1339562699.087476968,-1.570,1.570,-1.570,1.570,0.000,0.000
157  1339562699.137531995,-1.570,1.570,-1.570,1.570,0.000,0.000
158  1339562699.187649011,-1.570,1.570,-1.570,1.570,0.000,0.000
159  1339562699.237519979,-1.570,1.570,-1.570,1.570,0.000,0.000
160  1339562699.287507057,-1.570,1.570,-1.570,1.570,0.000,0.000
161  1339562699.337511062,-1.570,1.570,-1.570,1.570,0.000,0.000
162  1339562699.387468099,-1.570,1.570,-1.570,1.570,0.000,0.000
163  1339562699.437495946,-1.570,1.570,-1.570,1.570,0.000,0.000
164  1339562699.487494945,-1.570,1.570,-1.370,1.570,0.000,0.828
165  1339562699.537476062,-1.570,1.570,-1.370,1.570,0.000,0.828
166  1339562699.587482929,-1.570,1.012,-1.329,1.508,0.000,0.828
167  1339562699.637548923,-1.570,1.012,-1.329,1.508,0.000,0.828
168  1339562699.687515020,-1.570,0.987,-1.324,1.508,-0.276,0.828
169  1339562699.737611055,-1.570,0.987,-1.324,1.508,-0.276,0.828
170  1339562699.787514925,-1.468,0.972,-1.309,1.508,-0.245,0.828
171  1339562699.837558031,-1.468,0.972,-1.309,1.508,-0.245,0.828
```

Gambar 5.3 Database Motion dalam Format CSV

Visualisasi node-node dan topic saat melakukan proses perekaman ini ditunjukkan pada gambar rxgraph berikut.



Gambar 5.4 Gambaran Node dan Topic dalam Proses *Record Motion*

5.2 Proses Ekstrasksi Database Motion

Dari file database yang telah disimpan, dirancang sebuah node play_recorded_motion.py untuk mengekstraksi database yang telah disimpan. Proses ekstraksi robot dijelaskan dalam pseudocode berikut.

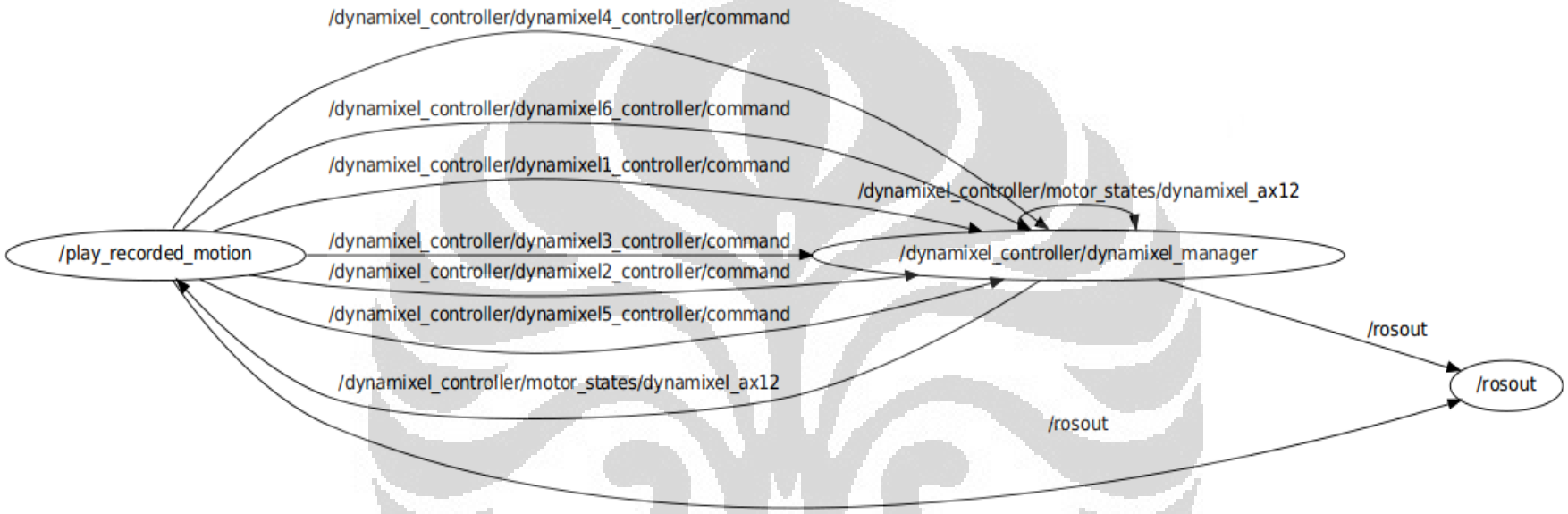
```
while not rospy.is_shutdown():
# Execute the behavior appropriate for the current command.
  rawdataread = datalog.readline()
  rawdataread2 = rawdataread.split()
  for idx,val in enumerate(rawdataread2):
       datareadnow[idx] = eval(rawdataread2[idx])
  self.dt = datareadnow[0] - datareadold[0]  #calculating dt
  if self.dt == 0.0: exit()
  self.goal_pos[0:6] = datareadnow[1:7]
  if firststart == 1:
       firststart = 0
       for name in sorted(dynamixels):
        try:
          self.servo_speed(self.default_joint_speed)
       else:
          self.servo_speed(abs((self.goal_pos-self.current_pos)/
self.dt))

  #send position topics
  self.dynamixel_pub.publish(self.goal_pos)
```

Gambar 5.5 Pseudocode Ekstraksi Database Motion

Node ini membaca setiap baris pada database yang telah dibuat (datalog). Baris yang dibaca merupakan sebuah string variable, sehingga untuk membaca setiap data pada baris perlu dilakukan pemisahan dengan fungsi split(). Hasil pemisahan ini akan menghasilkan 7 data dengan format seperti yang ditunjukkan pada Gambar 5.2. Data pertama merupakan data ros waktu, sedangkan enam data berikutnya merupakan data goal position setiap dynamixel. Sebelum mengirimkan posisi, dilakukan komputasi untuk menghitung kecepatan angular yang akan diberikan kepada dynamixel.

Visualisasi node-node dan topic saat melakukan proses perekaman ini ditunjukkan pada gambar rxgraph berikut.



Gambar 5.6 Visualisasi Node dan Topic pada Proses Playing Record Motion

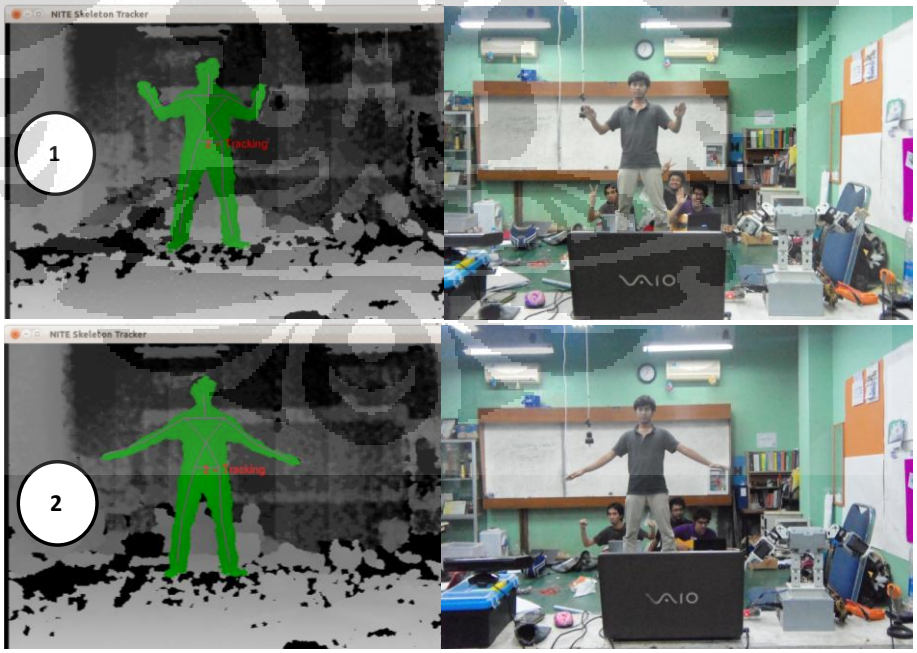
BAB 6
PENGUJIAN SISTEM MOTION CAPTURE

Untuk mengetahui kehandalan dari sistem yang telah dibuat perlu dilakukan pengujian pada sistem tersebut. Pengujian yang dilakukan pada sistem ini antara lain adalah uji coba real time imitation, perekaman dan ekstraksi database motion, serta uji batasan tracking.

6.1 Pengujian Validasi Solusi Inverse Kinematic

Pada bab 4.2 telah diturunkan solusi inverse kinematik dari model robot. Solusi inverse kinematik yang telah didapatkan dapat dilihat pada tabel 4.6.

Pada bagian ini akan dilakukan validasi solusi inverse kinematik yang didapatkan. Untuk menguji nilai posisi sudut yang akan diberikan, dilakukan perbandingan hasil forward kinematik dari sudut yang akan diberikan dengan nilai vektor reference yang didapatkan dari pengolahan topic skeleton. Pengujian dilakukan dengan melakukan gerakan-gerakan berikut.



Gambar 6.1 Pose Uji Validasi Solusi Inverse Kinematik

Dari gerakan-gerakan tersebut didapatkan vektor posisi dari titik left_shoulder, left_elbow, left_wrist, right_shoulder, right_elbow, dan right_wrist. Kemudian data-data tersebut diolah untuk mendapatkan vektor-vektor yang telah disebutkan pada bagian 4.2.4. Berikut adalah pengolahan data vektor untuk lengan kiri dan kanan dari keempat posisi tersebut.

Tabel 6.1 Tabel Pengolahan Validasi Solusi Inverse Kinematik Lengan Kiri

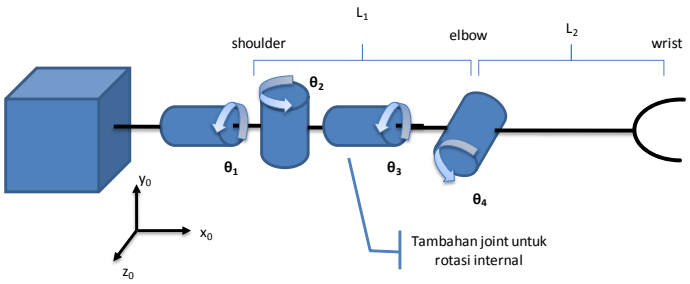
	POSISI					
	1			2		
	x (m)	y (m)	z (m)	x (m)	y (m)	z (m)
Left Shoulder	-0,294409	0,695051	3,257723	-0,305092	0,694969	3,24055
Left Elbow	-0,536357	0,571074	3,097391	-0,620074	0,478969	3,155274
Left Wrist	-0,565449	0,687998	2,807476	-0,910897	0,341379	3,055016
Vlse	-0,241948	-0,123977	-0,160332	-0,314982	-0,216	-0,085276
Vlew	-0,029092	0,116924	-0,289915	-0,290823	-0,13759	-0,100258
Vlsw	-0,27104	-0,007053	-0,450247	-0,605805	-0,35359	-0,185534
Llse	0,3156			0,3913		
Llew	0,3140			0,3370		
Llsw	0,5256			0,7256		
q2 (degree)	-23,6008			12,7945		
q4 (degree)	-50,0733			-53,6273		
q6 (degree)	66,8404			9,9999		
Vlse (FK)	-0,241948	-0,123977	-0,160332	-0,314982	-0,216	-0,085276
Error Vlse	0	0	0	0	0	0
Vlsw (FK)	-0,3367	-0,1529	-0,3996	-0,5823	-0,0649	-0,4053
Error Vlsw	24,225207	2067,8718	11,2487146	3,879882	81,6454085	118,450527

Tabel 6.2 Tabel Pengolahan Validasi Solusi Inverse Kinematik Lengan Kanan

	POSISI					
	1			2		
	x (m)	y (m)	z (m)	x (m)	y (m)	z (m)
Right Shoulder	0,0285	0,6977	2,8075	0,0187	0,6995	3,2847
Right Elbow	0,2088	0,5426	3,1497	0,2848	0,4293	3,1727
Right Wrist	0,2534	0,6550	2,8475	0,5718	0,2988	3,0518
Vrse	0,1804	-0,1551	0,3422	0,2660	-0,2702	-0,1120
Vrew	0,0445	0,1124	-0,3022	0,2870	-0,1305	-0,1209
Vrsw	0,2249	-0,0427	0,0400	0,5531	-0,4007	-0,2329
Lrse	0,4167			0,3954		
Lrew	0,3255			0,3377		
Lrsw	0,2324			0,7216		
q1 (degree)	-10,3233			-21,4327		
q3 (degree)	25,6568			42,3094		
q5 (degree)	146,3467			20,3791		
Vrse (FK)	0,1804	-0,1551	0,3422	0,2660	-0,2702	-0,1120
%Error Vrse	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000
Vrsw (FK)	0,0631	-0,0664	-0,1616	0,4792	0,0882	-0,5331
%Error Vrsw	71,9397	55,5764	503,6569	13,3556	122,0099	128,9005

Pada kedua tabel tersebut terlihat bahwa error Vlse dan Vrse sudah 0, artinya nilai solusi inverse kinematik dari θ_1 dan θ_2 atau dapat dikatakan bahwa solusi inverse kinematik dari bahu ke lengan sudah valid. Namun, pada error Vrsw terdapat kesalahan yang cukup besar. Kesalahan yang terjadi besar karena pada manusia, dari tangan hingga siku sebenarnya terdapat empat buah sendi. Sedangkan pada model robot yang digunakan dan dalam melakukan validasi ini digunakan model transformasi forward kinematik tiga sendi untuk setiap lengan.

Untuk membuktikan asumsi tersebut, penulis telah mencoba membuat persamaan forward kinematik dengan empat sendi, yaitu dengan menambahkan sebuah model joint untuk melakukan pergerakan rotasi internal dari bahu ke siku sehingga model lengan menjadi seperti pada gambar berikut.

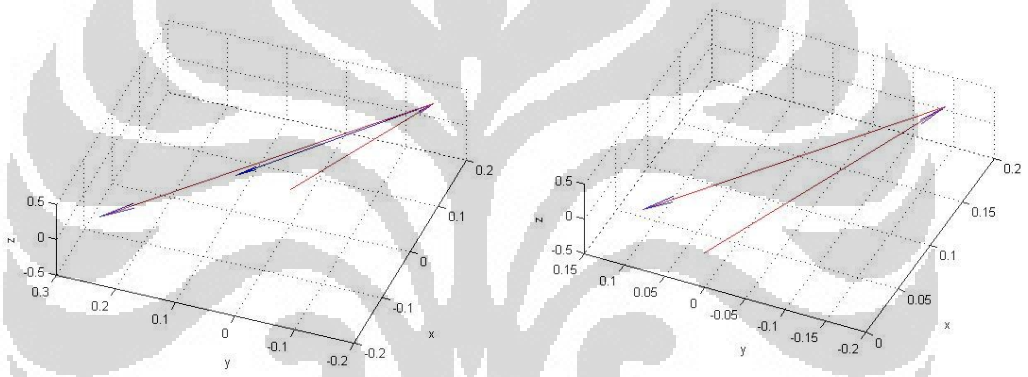


Gambar 6.2 Konfigurasi Lengan 4 DOF

Persamaan kinematik dari bahu ke pergelangan tangan menggunakan empat sendi dapat dilihat pada persamaan berikut.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}_{rsw4DOF} = \begin{bmatrix} L_1\cos(-\theta_2) + L_2\cos(-\theta_2)\cos(-\theta_4) + L_2\sin(-\theta_2)\sin(\theta_3)\sin(-\theta_4) \\ L_2\sin(-\theta_4)(\cos(\theta_1)\cos(\theta_3) - \sin(\theta_1)\cos(-\theta_2)\sin(-\theta_3)) + \\ L_1\sin(\theta_1)\sin(-\theta_2) + L_2\sin(\theta_1)\sin(-\theta_2)\cos(-\theta_4) \\ L_2\sin(-\theta_4)(\sin(\theta_1)\cos(\theta_3) + \cos(\theta_1)\cos(-\theta_2)\sin(-\theta_3)) - \\ L_1\cos(\theta_1)\sin(-\theta_2) - L_2\cos(\theta_1)\sin(-\theta_2)\cos(-\theta_4) \end{bmatrix}$$

Untuk memvalidasi persamaan tersebut, dilakukan ujicoba validasi persamaan tersebut pada hasil vektor yang didapat pada posisi satu. Dan didapatkan hasil validasi sebagai berikut.



a. Validasi Dengan Persamaan 3 DOF b. Validasi Dengan Persamaan 4 DOF

Gambar 6.3 Perbandingan Validasi Solusi Invers Kinematik menggunakan persamaan 3 DOF dan 4 DOF

Pada gambar 6.3 tersebut, vektor biru merupakan vektor acuan yang didapatkan dari topic skeleton, sedangkan vektor merah merupakan validasi forward kinematik dari solusi invers kinematik yang didapatkan. Dapat dilihat bahwa pada validasi menggunakan persamaan forward kinematik 4 DOF, hasil validasi dibandingkan vektor acuan telah berimpit sedangkan pada validasi dengan persamaan forwad kinematik 3 DOF tidak berimpit. Hal ini menunjukkan bahwa hasil pembacaan vektor skeleton manusia yang didapatkan kinect merupakan suatu set transformasi untuk solusi joint sebanyak 4 DOF, dikarenakan lengan manusia memiliki kemampuan untuk melakukan rotasi internal seperti gerakan pada sudut θ_3 pada gambar 6.2.

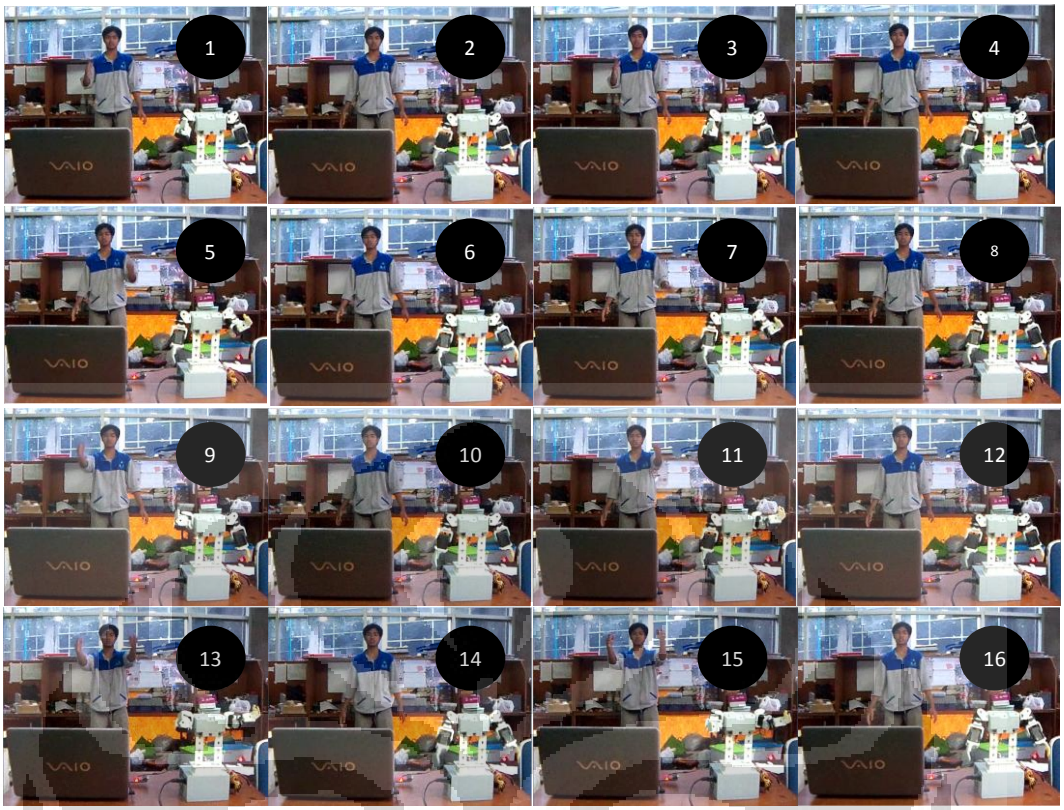
Namun, dikarenakan persamaan solusi inverse kinematik sudut siku dengan model 3 DOF ataupun 4 DOF adalah sama, maka error validasi yang besar ini dapat diabaikan. Untuk kedepannya, dalam melakukan *mapping* lengan manusia ke robot sebaiknya dilakukan dengan jumlah minimal 4 buah joint pada robot.

6.2 Pengujian Real Time Imitation dan Record Motion

Seperti yang telah dijelaskan sebelumnya, proses imitasi real time terdiri dari dua tahapan, yaitu tracking dan mapping. Tahapan pertama untuk melakukan imitasi secara real time adalah dengan melakukan proses kalibrasi skeleton dengan melakukan pose yang telah dijelaskan pada Gambar 4.4. Setelah node skeleton terkalibrasi, untuk melakukan real time imitation diberikan perintah tracker command dengan mengangkat kaki kiri. Tujuan memberi perintah dengan gerakan adalah untuk memudahkan user memulai proses real time imitation. Proses kalibrasi dan pemberian tracker command pada pengujian ini ditunjukkan pada gambar berikut.

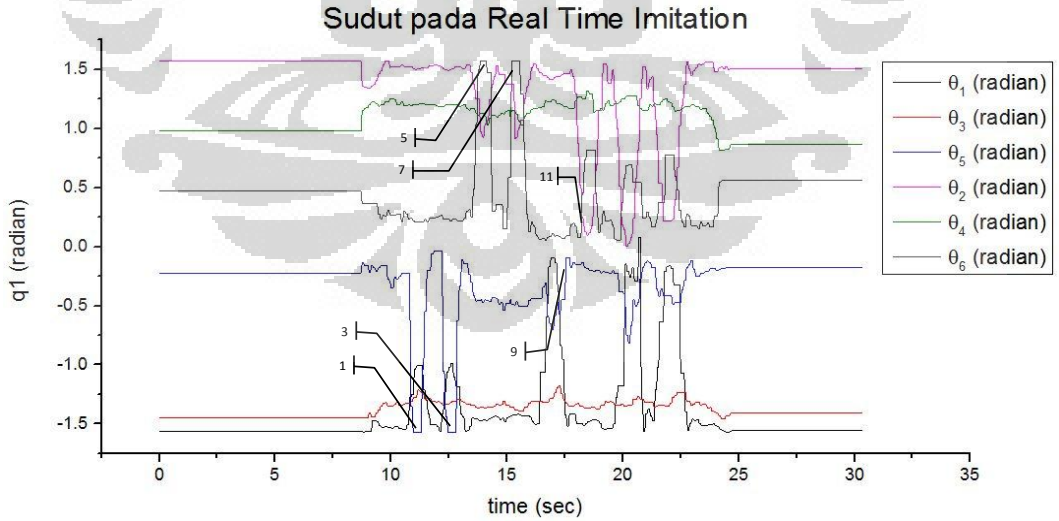
Gambar 6.4 Kalibrasi (Kiri) dan Pemberian Tracking Command (Kanan)

Berikut adalah tampilan proses *real time imitation* yang dilakukan dalam pengujian ini.



Gambar 6.5 Rangkaian Gerakan pada Pengujian Real Time Imitation

Posisi sudut yang dikirimkan ke robot ditunjukkan pada grafik dibawah ini.



Gambar 6.6 Sudut *Reference* pada setiap dynamixel pada Model Robot

Secara umum, proses real time imitation ini telah menunjukkan hasil yang bak seperti yang telah ditunjukkan pada rangkaian gambar 6.7 dan rekaman

video yang telah diunggah pada youtube yang menunjukkan bagaimana model robot ini mengimitasi gerakan dari demonstrator melalui sistem motion capture yang telah dirancang ini. Video tersebut dapat dilihat pada url http://www.youtube.com/watch?v=IZbPTS9VRpE&feature=my_liked_videos&list=LL7dqK1942CMWHXlsPJ8SLGw.

Hasil proses ekstraksi database juga menunjukkan hasil yang baik. Berikut adalah cuplikan dari proses ekstraksi database.

Gambar 6.7 Proses Ekstraksi Motion

6.3 Pengujian Batasan Sistem Motion Capture

Sebuah sistem yang baik, harus diuji kehandalannya dalam melakukan pekerjaan yang menjadi tugas sistem tersebut. Pengujian batasan sistem motion capture ini bertujuan untuk mengetahui sejauh mana ketahanan dan kehandalan sistem dalam melakukan proses motion capture serta untuk mengetahui kelemahan sistem agar kedepannya dapat dilakukan perbaikan. Pengujian batasan sistem yang dilakukan antara lain adalah variasi frekuensi tracking, pengujian jarak tracking, pengujian posisi yang tidak ter-tracking, probabilitas lost tracking, jumlah orang yang dapat di-tracking, dan orientasi tracking.

6.3.1 Variasi Frekuensi Tracking

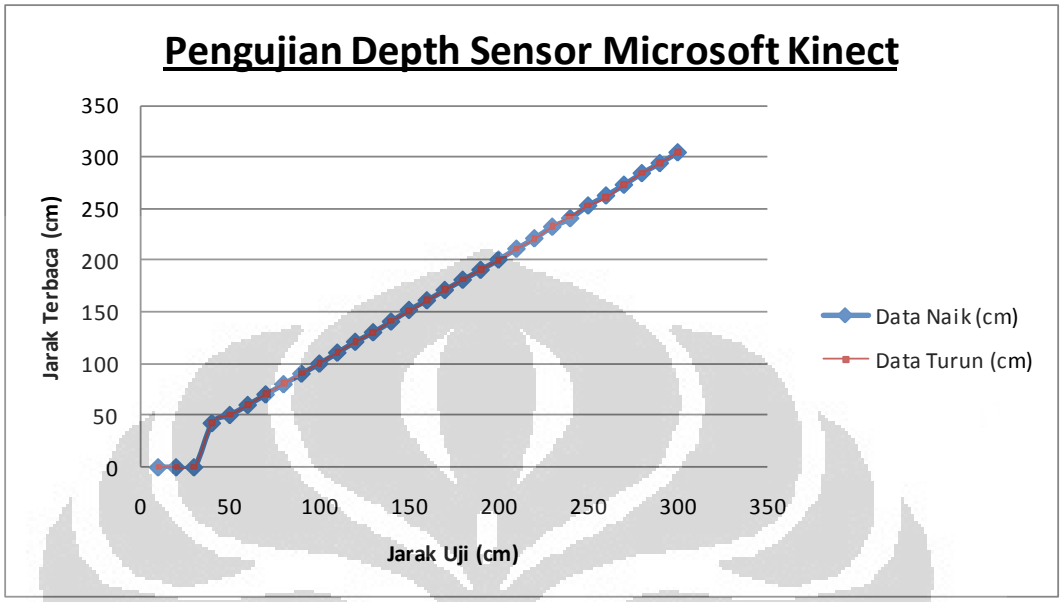
Dalam melakukan real time imitation, sangat penting untuk membuat periode sampling sekecil mungkin. Kinect merupakan stereo kamera yang memiliki kemampuan untuk mendapatkan citra 3 dimensi dengan kecepatan 30fps (frame per second). Pada pengujian ini akan diuji seberapa jauh frekuensi tracking dapat ditingkatkan.

Frekuensi yang telah diterapakan pada sistem ini telah divariasikan dari 2 Hz sampai dengan 20 Hz. Semakin tinggi frekuensi tracking, akan semakin halus gerakan-gerakan yang dipetakan dari manusia ke robot. Frekuensi tracking yang diprogramkan harus sedikit lebih lambat dibandingkan dengan waktu satu kali loop komputasit tracking – mapping. Komputasi yang dilakukan dalam satu loop tracking sampai mapping dalam melakukan proses real time imitation ini rata-rata adalah 340 us. Dari pengujian yang dilakukan didapatkan frekuensi tracking terbaik adalah 20 Hz. Pada frekuensi ini gerakan robot terlihat lebih stabil dan halus.

6.3.2 Pengujian Batasan Jarak Tracking

Kinect merupakan sebuah stereo camera, oleh karena itu sangat penting untuk mengetahui error dan batasan dead zone dari depth sensor. Eksperimen ini dilakukan untuk batasan jarak tracking sekaligus untuk mengetahui apakah nilai terukur pada *depth sensor* sesuai dengan nilai ukur asli real pada dunia nyata. Proses yang dilakukan untuk melakukan eksperimen ini adalah dengan

mengambil data depth sensor terhadap torso manusia dengan perubahan jarak setiap 10cm.



Gambar 6.8 Hasil Eksperimen Validasi Data Depth Sensor Kinect

Dari hasil yang didapatkan dapat disimpulkan bahwa depth sensor pada kinect adalah sensor yang presisi dan akurat. Kepresisian ditunjukkan pada grafik diatas yang mana pada setiap pengukuran nilai jarak terbaca pada kondisi naik (menjauhi) atau turun (mendekat) telihat cukup berimpit. Hysterisis pada pengukuran pun nyaris sama pada saat pengukuran mendekati dan menjauhi kinect. Error rata-rata pada pengukurun menjauhi didapat sebesar 1,74 cm dan pada saat pengukuran mendekati didapatkan sebesar 1,74 cm

Pada pengukuran yang dilakukan, secara umum dapat dilihat bahwa error yang terjadi semakin besar apabila jarak uji coba depth sensor semakin jauh. Pada percobaan ini juga didapat bahwa *deadzone* kinect adalah 40 cm. Pada deadzone, depth sensor tidak dapat membaca jarak (pembacaan bernilai 0 cm).

Apa bila pada deadzone dilakukan pengujian skeleton tracking, depth sensor tidak dapat mendeteksi skeleton manusia seperti pada gambar berikut.

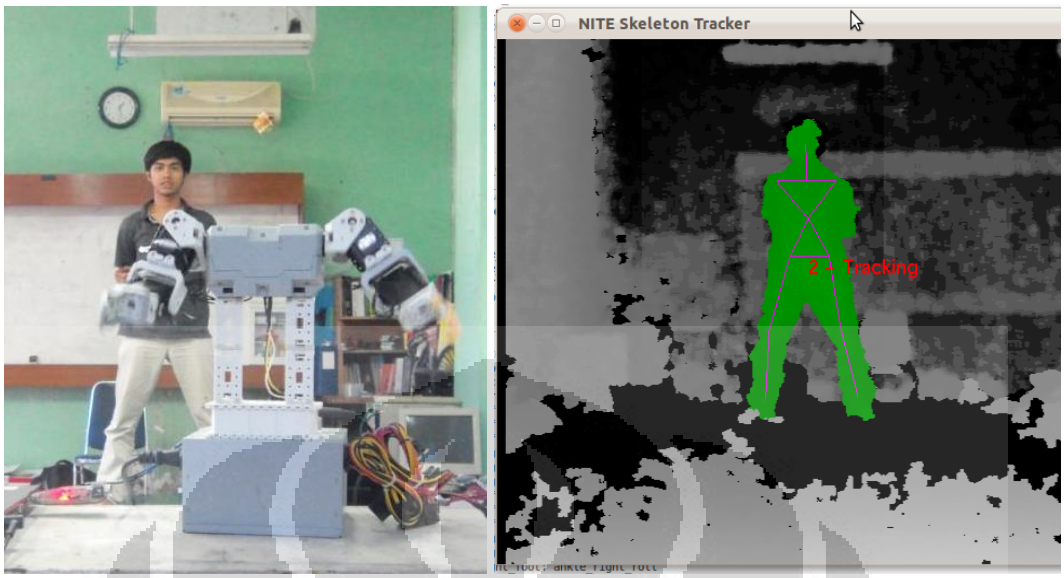
Gambar 6.9 Deadzone pada Kinect, terjadi pada jarak kurang dari 40 cm.

User juga tidak akan ter-tracking pada jarak lebih dari 4 meter.

Gambar 6.10 Posisi Tracking Lebih dari 4 meter

6.3.3 *Untracking Position*

Dalam melakukan skeleton tracking, terkadang kinect tidak selalu mendeteksi seluruh titik pada skeleton. Salah satu contoh kasus adalah sebagai berikut.



Gambar 6.11 Untracking Position

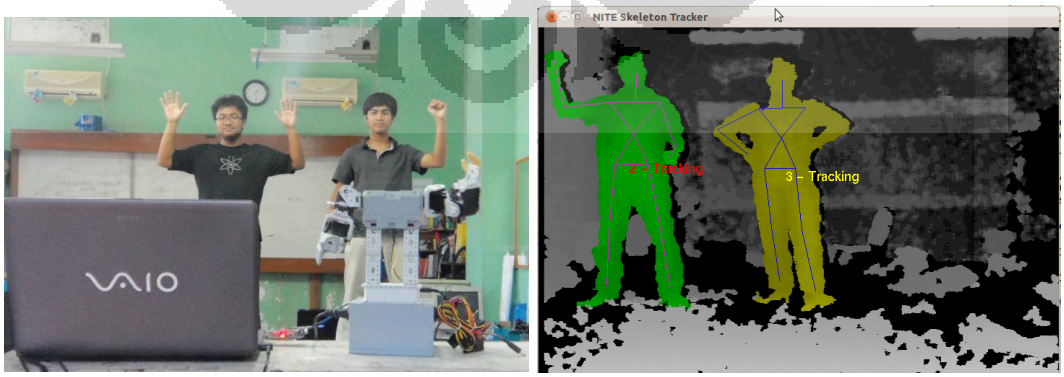
Pada kasus diatas, ketika user menyembunyikan kedua tangan dibalik punggung sehingga posisi vektor skeleton *left_elbow*, *left_hand*, *right_elbow*, dan *right_hand* tidak terdeteksi robot akan berada pada state mengikuti kedaan *tracking* sebelumnya. Hal yang sama akan terjadi jika terdapat salah satu vektor skeleton diluar jangkauan deteksi kinect.

Untuk kasus ketika user melakukan jangkauan gerakan yang tidak terjangkau oleh area kerja robot, komputasi tracking dan mapping inverse kinematik akan tetap dilakukan, tetapi apabila solusi inverse kinematik diluar jangkauan batasan dynamixel yang telah di set sebelumnya, dynamixel akan bergerak ke posisi batas.

Gambar 6.12 Posisi Robot ketika Gerakan User Diluar Area Kerja Robot

6.3.4 Jumlah User Ter-*tracking*

Salah satu fitur dari skeleton_tracker adalah dapat melakukan *tracking* untuk lebih dari satu orang. Namun, jika dua orang atau lebih telah melakukan tracking, manakah yang akan diimitasi robot. Berikut adalah gambaran ketika Kinect mentracking dua orang sekaligus.



Gambar 6.13 Tracking Multi User

Setiap user yang ter-tracking akan mendapatkan id dan warna tracking sebelum melakukan kalibrasi. Pada sistem yang dirancang ini, robot akan mengikuti user yang melakukan kalibarasi lebih dulu. Apabila user kedua juga melakukan pose kalibrasi, robot hanya mengikuti gerakan user pertama. Pada intinya, robot hanya akan melakukan imitasi pada satu user dan imitasi akan dilakukan terhadap user yang berhasil dikalibrasi dalam melakukan skeleton tracking terlebih dahulu.

BAB 7
PENUTUP

7.1 Kesimpulan

Dalam penelitian ini telah dikembangkan sebuah ros package yang disebut skripsi_mocap sebagai sebuah sistem motion capture untuk membuat sebuah model humanoid robot dapat melakukan imitasi terhadap gerakan manusia. Dari penelitian yang telah dilakukan dalam skripsi ini, dapat disimpulkan beberapa hal berikut.

1. ROS package skripsi_mocap yang telah dikembangkan berhasil menjawab tantangan proses imitasi dari humanoid robot serta mampu melakukan dan penyimpanan dan ekstraksi database motion yang telah dilakukan oleh manusia.
2. Ekstraksi solusi inverse kinematik, proses real time imitation, dan ekstraksi database pada robot telah menunjukkan hasil yang baik.
3. Pada Microsoft Kinect yang digunakan sebagai perangkat capture dalam penelitian ini digunakan frekuensi tracking sebesar 20Hz. Komputasi satu loop untuk melakukan proses mapping membutuhkan waktu rata-rata sebesar 340 us.
4. Batasan sistem motion capture yang telah dibuat dalam package skripsi_mocap ini antara lain adalah range kerja 0.4 – 4 meter dengan error rata-rata pembacaan setiap vektor skeleton sebesar 1.74cm.
5. Untuk mengatasi hasil tracking diluar jangkauan kerja robot, dilakukan pembatasan gerak pada Dynamixel AX-12.

7.2 Saran

Penelitian ini masih dapat dikembangkan dalam hal-hal berikut:

1. Proses imitasi seluruh bagian manusia ke robot
2. Pembuatan GUI untuk lebih memudahkan interaksi antara manusia dan perangkat yang telah dibuat.
3. Sistem motion capture yang telah dibuat dapat digunakan dalam kompetisi-kompetisi robotika.

7.3 Diskusi Aplikasi Motion Capture Pada KRSI dan KRCI RHSL

Penulis memulai karir dibidang robotika diawali dengan mengikuti Kontes Robot Nasional (KRI, KRI, dan KRCI) . Oleh sebab itu, salah satu motivasi untuk melakukan riset ini bermula dari keinginan penulis untuk memecahkan tantangan-tantangan dalam kompetisi tersebut khususnya KRSI (Kontes Robot Seni Indonesia) dan KRCI RSHL (Kontes Robot Cerdas Indonesia – Robot Humanoid Soccer League). Dalam kedua kontes tersebut digunakan robot humanoid untuk memecahkan tantangan yang diberikan.

Tantangan untuk membuat motion robot yang *robust* dengan menggunakan bahasa pemrograman atau trial error menggerakan joint robot kemudian merekamnya merupakan sesuatu yang sulit. Penelitian ini dapat diaplikasikan pada kedua kontes robot tersebut sebagai suatu teknik pemrograman humanoid robot melalui proses imitasi. Terlebih lagi apabila penelitian ini kedepannya dikembangkan suatu algoritma balancing motion dengan mempertimbangkan kinematika dan dinamika robot, akan tercipta sebuah sistem pemrograman robot humanoid yang mudah, fleksible, dan juga

Diakui dan dirasakan pula bahwa untuk mendapatkan predikat juara pada KRCI RSHL dan KRSI, dapat dilakukan dengan membuat sembarang motion (*trial and error*) tanpa perhitungan matematis kinematika dan dinamika. Namun, rasanya tidak elegan dan berkelas apabila menjuarai kontes robot dengan membuat sebuah robot humanoid tanpa "kontrol". Pemrograman dengan sistem motion capture ini merupakan solusi pemrograman robot humanoid yang lebih mudah dan lebih berunsur riset dibandingkan dengan melakukan pemrograman melalui bahasa pemrograman ataupun menggerakan joint-joint dengan tangan kemudia merekamnya.

DAFTAR REFERENSI

[1] http://www.ros.org/wiki (accessed Juni 10, 2012).

[2] Basso, Filippo. *RGB-D People Tracking by Detection for A Mobile Robot.* Bachelor Thesis, Padova: Universita degli Studi di Padova, 2011.

[3] Craig, John J. *Introduction to Robotics: Mechanics and Control.* USA: Addison Wesley Longman, 1989.

[4] Davies, Tracy. *Dawn of Kate's Gazebo Simulation.* Bachelor Thesis, California: California Polytechnic State University, San Luis Obispo, 2011.

[5] Erik B. Dam, Martin Koch, Martin Lillholm. *Quaternions, Interpolation, and Animation.* Technical Report, Copenhagen: University of Copenhagen, 1998.

[6] Goebel, Patrick. *ROS by Example: Head Tracking in 3D (Part 1).* http://www.pirobot.org/blog/0018/ (accessed Maret 20, 2012).

[7] K Saenko, S. Karayev, Y. Jia. "Practical 3-D Object Detection using Category and Instance Level Appearance Models." *International Conference on Intelligent Robots and Systems.* 2011.

[8] *Low-pass filter*. May 6, 2012 . en.wikipedia.org/wiki/Low-pass_filter (accessed June 10, 2012).

[9] M.Z. Al-Faiz, MIEEE. "Analytical Solution for Anthropomorphic Limbs Model, (IK of Human Arm)." *2009 IEEE Symposium on Industrial Electronics and Applications (ISIEA 2009).* Kuala Lumpur, 2009. 684-689.

[10] Meredith, M.J. *Adapting and Reconfiguring Human Figure Motion Capture Data through the Application of Inverse Kinematiks and Biomechanics-based Optimisation.* Master Thesis, University of Sheffield, 2005.

[11] Nakaoka, Shinichiro. *Generating Whole Body Motions for a Biped Robd from Captured Human Dances.* Master Thesis, Tokyo: University of Tokyo, 2003.

[12] Parul Gupta, Vineet Tirth, R.K. Srivastava. "Futuristic Humanoid Robots: An Overview." *First International Conference on Industrial and Information Systems.* Sri Lanka: IEEE, 2006. 247-254.

[13] Pitowarno, Endra. *Robotika: Desain, Kontrol, dan Kecerdasan Buatan.* Yogyakarta: ANDI Yogyakarta, 2006.

[14] R, Zhang, and Wang Y. "Research and Implementation from Point Cloud to 3D Model." *Second International Conference on Computer Modeling and Simulation.* 2010.

[15] Rossum, Guido van. *Python Tutorial.* PythonLabs, 2003.

[16] Shon, A.P. *Robotic Imitation from Human Machine Capture using Gaussian Process.* University of Washington, 2010.

[17] Siscart, Marc Rosanes. *Algorithms and Graphic Interface Design to Control and Teach a Humanoid Robot Through Human Imitation.* Master Thesis, Catalunya: Universitat Politecnica de Catalunya, 2011.

[18] Teodoro, Pedro Daniel Dinis. *Humanoid Robot: Development of a simulation environment of an entertainment humanoid robot.* Dissertation, Lisbon: Instituto Superior Tecnico, 2007.

[19] Xia, Lu. *Human Detection Using Depth Information by Kinect.* Austin: University of Texas.