SinarFe7
Seminar Nasional Fortei Regional 7

ISSN (Print) : 2621-3540
ISSN (Online) : 2621-5551

GESTUR TANGAN UNTUK KENDALI *TURTLEBOT* DAN *OPENMANIPULATOR*

Alexander Edward, [2] Petrus Santoso
Program Studi Teknik Elektro Universitas Kristen Petra Surabaya, Indonesia
[1]alexander.hutomo@yahoo.com, [2] petrus@petra.ac.id

Abstrak – NUI atau *Natural User Interface* adalah sebuah bentuk teknologi yang sedang berkembang saat ini. Salah satu bentuknya adalah *gesture control* terhadap perangkat atau mesin yang digunakan manusia. Proyek ini membuat sebuah bentuk NUI yang digunakan untuk mengontrol gerakan robot manipulator (*Turtlebot* dan OpenManipulator) menggunakan gerakan tangan. Program ini dibuat pada *platform* ROS (*Robot Operating System*) dan terdiri dari 3 *node*. *Node* pendeteksi tangan memanfaatkan *library leap motion* untuk mendapatkan data dari tangan yang terdeteksi oleh kamera *leap*, memproses dan mengirimkan data tersebut. *Node* kontrol *turtlebot* merupakan *node* yang dibuat khusus untuk mengontrol gerakan *turtlebot*. *Node* kontrol *OpenManipulator* merupakan program yang dibuat untuk menghitung serta mengirimkan perintah gerak robot. Program yang dibuat dapat mendeteksi bentuk tangan dengan cukup baik dimana bentuk tangan terbuka dan mengepal dapat 100% terdeteksi, namun bentuk tangan lain seperti menjepit kurang dapat dikenali. Waktu proses yang dibutuhkan program untuk menghitung dan mengirimkan perintah ke robot juga dapat dianggap cukup cepat dimana waktu proses untuk *node turtlebot* adalah 0.1 detik dan untuk robot lengan adalah 0.6 detik. Selain itu, didapati *node* kontrol *OpenManipulator* tidak dapat berjalan dengan tingkat kesuksesan yang sempurna, yaitu 70%.

Kata kunci: *leap-motion*, *turtlebot, openmanipulator,* kontrol, tangan

Abstract – NUI or Natural User Interface is a form of technology that is currently being developed. A form of NUI is to use gesture control to interact with machines. This project's goal is to make a form of NUI that is used to control a manipulator robot using hands' movements. This program was mad using ROS (Robot Operating System) as the platform and the program consists of 3 nodes. Hand detection node makes use of leap motion's library to get detected hands' data, process and send that data. Turtlebot control node is a node that was made to specifically control turtlebot. OpenManipulator control node was made to calculate and send commands to the manipulator robot. The program can detect hands' shapes quite well, which are 100% success rate for closed and open hand shapes, but was unable to reliably detect pinch and other hands' shapes. The time needed to process and send the data are fairly quick as well which are 0.1 second for turtlebot node and 0.6 second for OpenManipulator node. Furthermore, the success rate for OpenManipulator node in controlling the manipulator is not perfect, which is 70%.

Keyword: leap-motion, turtlebot, openmanipulator, control, hands

I. PENDAHULUAN

Dalam beberapa tahun terakhir, teknologi interaksi antara manusia dan mesin, yang disebut juga dengan *user interface* adalah salah satu teknologi yang paling berkembang. Teknologi *user interface* terus berkembang sampai akhirnya muncul istilah *Natural User Interface* (NUI). *Natural User Interface* atau NUI adalah sebuah sistem interaksi antara manusia dan komputer dimana pengguna melakukan sesuatu melalui aksi yang intuitif berhubungan dengan tindakan manusia yang natural dan dilakukan sehari hari [1]. NUI memanfaatkan beberapa kemampuan alami manusia dan membangunnya untuk mengintegrasikan teknologi buatan ke dalam interaksi manusia yang telah dipelajari. Kata *natural* disini digunakan karena kebanyakan hubungan antarmuka komputer dengan manusia menggunakan alat kontrol buatan yang perlu dipelajari penggunaannya. NUI bergantung pada pengguna yang dapat melakukan gerakan relatif alami yang dengan cepat ditemukan untuk mengontrol aplikasi komputer atau memanipulasi konten pada layar [2].

NUI memiliki kemampuan untuk mengurangi jarak antara mesin dan manusia sekaligus meningkatkan kekuatan manusia terhadap mesin tersebut yang membuat pengembangan komputer dapat berkembang lebih jauh. Salah satu pengaplikasian NUI yang sedang berkembang sekarang adalah penggunaan gerakan tubuh pengguna untuk mengoperasikan teknologi. Telah banyak institusi yang mengembangkan teknologi yang memanfaatkan gerakan tubuh seperti *Microsoft* dengan sensor *Kinect*, Kamera *RGBD (Red, Green, Blue, Depth), Real Sense Software Development Kit 2* oleh Intel, serta kamera pendeteksi gestur tangan oleh *Leap Motion*. Salah satu penelitian yang menyerupai proyek ini adalah publikasi oleh Edwin C. Montiel-Vazquez dan timnya dengan judul MOBMA [3]. Publikasi ini juga membuat sebuah sistem kontrol menggunakan gestur untuk robot *mobile manipulator*. Proyek ini memiliki ide yang sama, namun menggunakan metode dan platfor yang berbeda, dimana MOBMA memanfaatkan LabVIEW sebagai platform dan java sebagai bahasa pemorgramannya sedangkan proyek yang dibuat pada tulisan ini memanfaatkan platform ROS (*Robot Operating System*) dan bahasa pemrograman Python. Selain itu, program ini memanfaatkan *library* dan program dari *turtlebot* dan *openmanipulator* sendiri untuk melakukan perhitungan dan kontrol sedangkan MOBMA membuat algoritma kontrol sendiri yang membuat waktu proses kontrol lebih tidak

ISSN (Print) : 2621-3540
ISSN (Online) : 2621-5551

responsif dikarenakan banyaknya proses dan perhitungan yang perlu dilakukan.

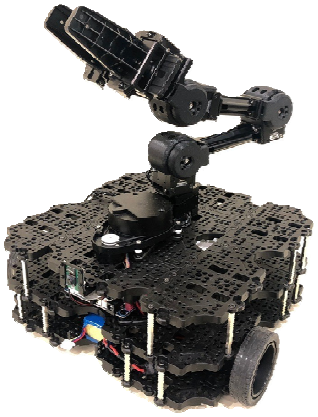
Dengan semakin berkembangnya teknologi ini, kita dapat memanfaatkan gerakan tangan kita untuk mengendalikan konten pada layar komputer atau mengendalikan robot sesuai keinginan kita. Salah satu riset yang mengembangkan kontrol robot lengan menggunakan *Leap Motion* dilakukan oleh Bassily dan timnya [4]. Berdasarkan hal tersebut, penulis ingin membuat suatu aplikasi untuk mengontrol robot, dimana robot yang digunakan adalah *turtlebot* dan O*penManipulator,* menggunakan *Leap Motion*. Aplikasi ini akan memproses gestur tangan yang dideteksi oleh kamera *leap motion* dan melakukan kontrol gerakan dari robot *mobile* dan *manipulator*.

II. PERENCANAAN & IMPLEMENTASI SISTEM

Proyek ini berfokus pada pembuatan program kontrol robot *mobile manipulator* menggunakan gestur tangan dan memanfaatkan kamera *leap motion* sebagai pendeteksi gerakan tangan untuk mengontrol gerakan robot. Program kontrol robot dibuat menggunakan ROS (*Robot Operating System*) dan terdiri dari 3 *node* utama, *node* pendeteksi tangan, *node* kontrol *turtlebot*, dan *node* kontrol *OpenManipulator.*

A. Hardware Robot, Turtlebot with OpenManipulator

Program kontrol ini memanfaatkan sebuah modul robot yang dikembangkan oleh *ROBOTIS* bernama *turtlebot with OpenManipulator* [5]. *Turtlebot* merupakan *mobile robot* beroda 2 yang pada robot ini, menjadi platform bergerak bagi modul ini. *OpenManipulator* merupakan robot lengan dan merupakan bagian *manipulator* dari robot *mobile manipulator* ini.



Gambar 1. *Turtlebot with OpenManipulator*
Sumber: Turtlebot3: Robotis E-manual, n.d.

B. Leap Motion

Program ini membutuhkan media yang dapat mendeteksi gerakan ataupun bentuk tangan. Salah satu alat yang dapat dipakai adalah kamera *leap motion*. Kamera *Leap Motion* didesain secara khusus untuk mendeteksi kondisi tangan yang terdeteksi [6]. Dengan menggunakan *library* dari API *Leap Motion*, program dapat langsung mengambil data-data yang diperlukan dan menyerahkan proses *image processing* dan pengelompokan data kepada *service* dari *driver leap motion.*

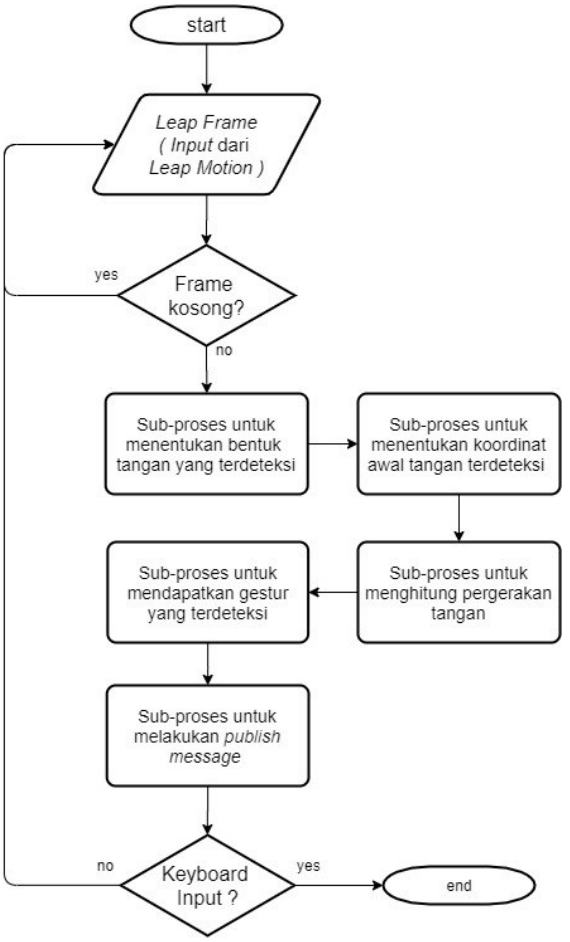
Gambar 2 Kamera *Leap Motion*
Sumber: stampa3dusd.it, n.d

C. Node Pendeteksi Tangan

Tahap pertama dalam proses dari program ini adalah mendeteksi tangan dari pengguna. Menggunakan *sample* dari API *Leap Motion* sebagai referensi dan *library* dari *leap motion*, *node* ini dapat mendeteksi berbagai properti dari tangan yang dideteksi kamera seperti jumlah tangan, posisi dan arah tangan, kiri atau kanan, kondisi jari, dan lain-lain. Melalui data tersebut dapat juga dicari data lain yang tidak dapat langsung dicari menggunakan *library* dari *leap motion* seperti bentuk tangan, perubahan koordinat posisi tangan, dan lain-lain.

Node ini memproses data *frame* yang didapat menggunakan *library leap motion* dan di dalam *frame* tersebut terdapat nilai – nilai sehubungan dengan tangan yang terdeteksi di dalam *frame* tersebut. Setiap *frame* yang diterima program akan menjalankan fungsi *callback* yang didalamnya terdapat beberapa subproses untuk menghitung nilai yang diperlukan dan mengirimkannya ke *node* kontrol. Gambar 3 menunjukkan *flowchart* dari proses di dalam *node* pendeteksi tangan.

ISSN (Print) : 2621-3540
ISSN (Online) : 2621-5551



Gambar 3 *Flowchart* Program Pendeteksi Tangan

Sub-proses pertama yang terjadi saat memasuki fungsi *callback* adalah penentuan bentuk tangan dari seluruh tangan yang terdeteksi. Bentuk yang dapat dideteksi dari algoritma ini adalah bentuk tangan terbuka, mengepal, menjepit, dan tidak diketahui. Penentuan bentuk tangan didasarkan 3 variabel data dari *frame leap motion*, yaitu jumlah jari yang terbuka, kekuatan genggaman, dan kekuatan jepitan jari.

Tabel 1 Nilai Dominan Saat Pendeteksian Bentuk Tangan Tertentu

Hand Shape	Grab	Pinch	Fingers
Open	0	0	5
Close	1	1	0
Pinch Fist	0	1	2
Pinch Open	0	1	3

Data pada *Tabel 1* merupakan data nilai dominan yang didapat dari 200 data *frame* saat bentuk tangan tertentu dibuat di depan kamera *leap moton*. Nilai tersebut dijadikan penentu bentuk tangan dari algoritma pendeteksi bentuk tangan.

Dari bentuk tangan yang terdeteksi, akan ditentukan kapan memulai perhitungan perubahan posisi tangan. Perhitungan perubahan posisi tangan dilakukan saat *node* mendeteksi bentuk tangan mengepal dan dimulai dengan menentukan posisi awal untuk perhitungan ini. Oleh karena itu, posisi tangan pada *frame* pertama yang mendeteksi bentuk tangan mengepal akan dijadikan titik 0 / referensi dari perhitungan perubahan posisi. *Frame* berikutnya, apabila tangan terus mengepal, nilai koordinatnya akan dikurangi dengan nilai koordinat referensi menjadi nilai perubahan posisi yang dicari. Apabila terdeteksi bentuk tangan lain selain mengepal, posisi awal tangan akan di-*reset*. Selain itu, *library leap motion* juga memiliki kemampuan untuk mendeteksi gerakan tertentu seperti *swipe, circle*, dan lain-lain dan kemampuan ini juga dimanfaatkan untuk mendeteksi gestur khusus yang dapat digunakan untuk perintah tertentu.

Setelah semua data selesai diproses, data tersebut dimasukkan ke dalam *message* yang kemudian akan di-*publish* ke *node* kontrol melalui *topic "/hands"* dengan frekuensi 2 *Hz*. Selain mengirimkan data tersebut, *node* juga akan mem-*publish* sinyal *shutdown* apabila *node* ini dimatikan. Cara mematikan *node* ini adalah dengan memberikan *keyboard input "Enter"*. *Message* sinyal *shutdown* ini berfungsi untuk mematikan *node* lain bersamaan dengan *node* ini dimatikan.

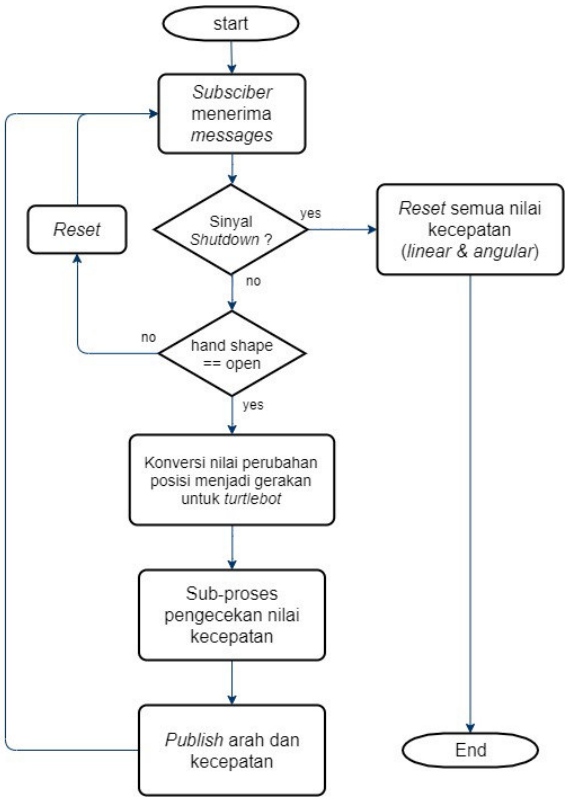
D. Node Kontrol Turtlebot

Node ini merupakan program untuk melakukan kontrol terhadap bagian platform dari robot, yaitu *turtlebot*. Program ini memanfaatkan program original dari *package turtlebot* yaitu *turtlebot_teleop_key*. Program referensi yang digunakan merupakan *node* kontrol *turtlebot* yang menggunakan *input keyboard* untuk menentukan arah dan kecepatan gerak robot. Modifikasi yang dilakukan pada program ini terdapat pada metode menggerakkan robot yaitu dari *input keyboard* menjadi *message* dari *node* pendeteksi tangan. Karena itu, seluruh proses untuk menggerakkan *turtlebot* diletakkan pada fungsi *callback* dari ROS *subscriber*.

Fungsi *main* dari *node* berisikan inisialisasi *node* , inisialisasi *publisher* dan *subscriber,* inisialisasi *leap motion* dan *loop rospy.sleep* untuk mengatur frekuensi *subscribe*. *Node* ini menerima pesan melalui *topic "/hands"* dimana terdapat *message* dari *node* pendeteksi tangan. Dari data yang diterima, tidak semua nilai digunakan oleh *node* ini. Data yang digunakan dari *message* tersebut hanya bentuk tangan yang terdeteksi, sinyal *shutdown* dan nilai perubahan posisi tangan. Setiap kali *node* menerima *message*, fungsi *callback* akan dijalankan dimana di dalam fungsi ini terdapat beberapa sub-proses yang fungsinya menentukan gerak untuk *turtlebot* berdasarkan nilai yang diterima lewat *message*. Gambar 4

ISSN (Print) : 2621-3540
ISSN (Online) : 2621-5551

menunjukkan *flowchart* proses kerja dari program dalam melakukan kontrol terhadap *turtlebot*.



Gambar 4 *Flowchart* Program Kontrol *Turtlebot*

Di awal fungsi *callback,* program akan membaca sinyal *shutdown* dan akan mematikan *node* apabila sinyal *shutdown* yang diterima memiliki nilai *true*. Sebelum melakukan *shutdown*, *node* akan memberikan perintah *reset* dan menghentikan seluruh gerakan robot agar robot tidak terus bergerak meskipun program kontrol mati. Apabila *sinyal shutdown* bernilai *false*, program akan melanjutkan ke proses berikutnya.

Sub-proses berikutnya mendeteksi bentuk tangan yang sedang terdeteksi dan karena *node* ini memanfaatkan nilai perubahan posisi yang hanya terdeteksi saat tangan mengepal, proses akan memberikan perintah *reset* dan berhenti apabila terdeteksi bentuk tangan membuka. Apabila bentuk tangan yang terdeteksi adalah mengepal, program akan berlanjut menuju proses konversi nilai perubahan posisi tangan menjadi gerakan untuk *turtlebot*. Proses konversi ini terdiri dari konversi satuan dan *scaling*. Konversi satuan dilakukan karena perbedaan satuan yang digunakan oleh masing-masing kontrol. *Scaling* dilakukan untuk menyesuaikan sensitifitas perubahan nilai terhadap gerakan robot sehingga dapat membuat robot lebih responsive terhadap gerakan tangan atau sebaliknya. Setelah itu, dilakukan validasi nilai yang didapat apakah sesuai dengan batas spesifikasi robot, seperti batas kecepatan linear dan angular, dan lain-lain.

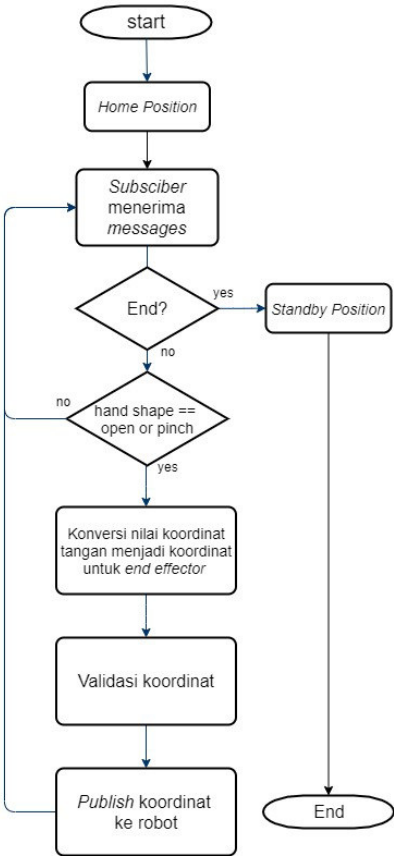
DI akhir proses-proses tersebut, nilai akan di *publish* menuju robot melalui *topic "/cmd_vel"*. Namun sebelum di-*publish,* nilai tersebut akan diubah agar pernambahan nilai kecepatan robot tidak terlalu ekstrim dan robot dapat dipercepat dan diperlambat secara perlahan.

E. Node Kontrol OpenManipulator

Node ini merupakan program untuk melakukan kontrol terhadap bagian manipulator dari robot, yaitu *OpenManipulator*. Program ini memanfaatkan program original dari *package OpenManipulator* dan *package Turtlebot_manipulation* sebagai referensi. *Node* ini memiliki struktur program yang sama dengan *node* kontrol *turtlebot* dimana fungsi *main* dari program ini terdiri dari inisialisasi *node*, inisialisasi *Subscriber* dan *loop rospy.sleep* yang berfungsi untuk mengatur frekuensi *Subscriber* dari *node* ini.

Meski serupa, *node* ini memiliki beberapa perbedaan yaitu *node* ini tidak melakukan *publish* dan frekuensi dari *rospy.sleep* yang digunakan pada *node* ini adalah 1 *Hz.* Alasan dari tidak adanya *publish* pada *node* ini karena pengiriman pesan ini dikerjakan oleh *library* yang digunakan untuk mengontrol robot lengan. Alasan dari frekuensi *subscribe* antara *node* ini dengan *node* sebelumnya adalah waktu proses yang dibutuhkan oleh program ini. *Node* ini juga menerima pesan melalui topik *"/hands"*, namun nilai yang digunakan oleh *node* ini sedikit berbeda. Pada awal *node* menyala, *node* akan memberikan perintah kepada robot untuk melakukan gerakan menuju posisi tertentu yang disebut dengan *home position.* Dari *message* yang diterima, *node* akan membaca sinyal *shutdown*, bentuk tangan yang terdeteksi, serta koordinat posisi tangan pada saat ini. Gambar 5 menunjukkan *flowchart* proses kerja dari program kontrol *OpenManipulator.*

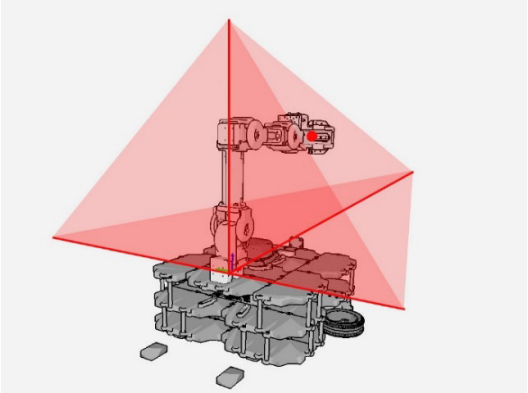
ISSN (Print) : 2621-3540
ISSN (Online) : 2621-5551



Gambar 5 *Flowchart* Program Kontrol *OpenManipulator*

Sama dengan *node* sebelumnya, proses utama dari program ini juga terletak pada fungsi *callback* dan terdiri dari beberapa sub-proses antara lain konversi nilai posisi tangan menjadi koordinat tujuan *end-effector*, validasi koordinat tujuan, kemudian *publish* nilai tujuan menggunakan *library moveit_commander*. Di awal fungsi *callback*, program akan membaca sinyal *shutdown* di dalam *message* dan akan melakukan *shutdown* apabila nilai yang diterima adalah *true*. Apabila program memutuskan untuk melakukan *shutdown*, program akan memerintahkan robot untuk menuju posisi yang disebut posisi *standby* sebelum *node* benar-benar mati. Apabila nilai yang diterima adalah *false*, node akan melanjutkan prosesnya dan akan membaca bentuk tangan yang terdeteksi di saat itu. Pada *node* ini, karena robot memiliki *end effector* berupa penjepit yang dapat juga dikontrol, *node* ini selain mendeteksi bentuk tangan mengepal, juga akan mendeteksi bentuk tangan menjepit. Tujuan membaca bentuk menjepit adalah untuk mendapatkan nilai kekuatan jepitan yang bervariasi sehingga dapat digunakan untuk mengontrol *end effector*.

Apabila bentuk tangan yang terdeteksi sesuai, maka program akan memasuki proses berikutnya yaitu konversi nilai koordinat posisi tangan menjadi koordinat tujuan *end effector*. Proses konversi nilai ini hanya terdiri dari konversi satuan sebelum kemudian nilai tersebut akan dicek apakah terdapat di dalam *workspace* dari *OpenManipulator* atau tidak. Metode yang digunakan adalah dengan melakukan cek apakah koordinat tujuan terdapat di dalam volume area yang mewakili *workspace Openmanipulator*. Tujuan dari validasi ini adalah untuk mengurangi kemungkinan *error* yang dapat terjadi akibat koordinat tujuan di luar kemampuan robot lengan. Gambar 6 menunjukkan visualisasi area kerja algoritma validasi yang digunakan untuk mewakili *workspace* sebenarnya dari robot lengan.



Gambar 6 Visualisasi *Workspace* dari *OpenManipulator*

Perhitungan dilakukan dengan memanfaatkan rumus trigonometri dan rumus luas segitiga dengan Langkah sebagai berikut.

1. Mencari titik-titik ujung segitiga pada ketinggian tertentu sesuai dengan nilai z dari koordinat tangan menggunakan rumus (1), (2), dan (3).

$$\frac{X\,max}{Z\,max} \times Z\ input = Point\ A\ dari\ \Delta \quad (1)$$

$$\frac{Y\,max}{Z\,max} \times Z\ input = Point\ B\ dari\ \Delta \quad (2)$$

$$-\ PointB = Point\ C\ dari\ \Delta \quad (3)$$

2. Mencari Luasan segitiga tersebut menggunakan rumus (4).

$$Luas = \left| \frac{XA(YB-YC) + XB(YC-YA) + XC(YA-YB)}{2} \right| \quad (4)$$

3. Mengganti Salah satu titik dari rumus tersebut dengan titik yang ingin dicari

ISSN (Print) : 2621-3540
ISSN (Online) : 2621-5551

4. Melakukan Langkah 3 sebanyak 3 kali dengan komposisi yang berbeda-beda
5. Menjumlah 3 Area yang ditemukan di Langkah 4
6. Apabila Area hasil penjumlahan sama dengan luas area segitiga dari Langkah 2, maka titik koordinat (x,y) tersebut berada di dalam segitiga.

Apabila koordinat tersebut berada di dalam segitiga, maka dilakukan kalkulasi *inverse kinematic* oleh *library ikpy* untuk mendapatkan *joint – joint* tujuan robot lengan. *Joint – joint* ini kemudian di *publish* ke robot lengan menggunakan perintah dari *library moveit_commander.*

III. PENGUJIAN & HASIL

A. Pengujian Algortima Pendeteksian Bentuk Tangan

Pengujian ini dilakukan dengan tujuan untuk mengetahui persentase keberhasilan algoritma pada program pendeteksi tangan dalam menentukan bentuk tangan yang terdeteksi. Diambil 200 data untuk setiap pengujian bentuk tangan dan akan dihitung persentase algoritma memberikan hasil yang benar.

Tabel 2 Hasil Pengujain Algoritma Pendeteksi Bentuk Tangan

Bentuk Tangan	Persentase Keberhasilan
Terbuka	100%
Mengepal	100%
Menjepit Terbuka	29%
Menjepit Tertutup	52%
Tidak Dikenal “1 jari”	100%
Tidak Dikenal “2 jari”	1.5%
Tidak Dikenal “3 jari”	100%
Tidak Dikenal “4 jari”	0%

Dari data *Tabel 2* dapat disimpulkan bahwa algortima dapat dengan mudah mendeteksi bentuk tangan terbuka dan mengepal, ditunjukkan dengan persentase keberhasilan 100%. Namun untuk bentuk tangan menjepit dan tidak dikenal, algoritma kesulitan menentukan bentuknya dengan benar dan sering salah menentukan bentuk tersebut.

B. Pengujian Waktu Proses Kontrol Turtlebot

Pengujian ini dilakukan untuk mengetahui waktu yang dibutuhkan proses *callback* untuk menyelesaikan fungsinya dan apakah frekuensi *subscribe* dari *node* kontrol *turtlebot* sesuai dengan kebutuhannya. Pengujian dilakukan dengan melakukan 20 kali pengiriman *message* kepada *node turtlebot* dan menghitung selisih waktu pada awal dan akhir fungsi *callback.*

Dari data pengujian ditemukan bahwa rata-rata waktu yang diperlukan fungsi *callback* adalah 0.1035 detik dan waktu maksimal yang dibutuhkan adalah 0.3471 detik. Nilai ini menyebabkan diberikannya frekuensi 2 *Hz* untuk proses *subscribe* pada *node*.

C. Pengujian Algoritma Validasi Koordinat Tujuan End Effector

Pengujian ini bertujuan untuk mengetahui apakah metode validasi ini dapat mendekati area kerja sebenarnya dari robot lengan yang digunakan. Pengujian dilakukan dengan melakukan 10 kali pengambilan data untuk kondisi benar dan 10 kali untuk kondisi salah. Kondisi benar adalah saat koordinat tujuan berada di dalam area kerja algoritma validasi. Hasil ini kemudian akan dibandingkan dengan hasil dari program original dari *package turtlebot_manipulation*. Koordinat tersebut dimasukkan ke dalam program original ini dan dijalankan. Algoritma dianggap berhasil apabila program ini dapat bergerak menuju koordinat yang dianggap benar oleh algoritma dan mengeluarkan *error* apabila diberi input koordinat yang dianggap salah oleh algoritma ini.

Tabel 3 Hasil Perbandingan Data Kondisi Benar

Di Dalam	Success	Fail
Algoritma	10	0
Program	10	0

Tabel 4 Hasil Perbandingan Data Kondisi Salah

Di Dalam	Success	Fail
Algoritma	0	10
Program	10	0

Dari data pengujian pada *Tabel 3* dan *Tabel 4* dapat disimpulkan bahwa area kerja dari algoritma sebenarnya lebih kecil daripada *workspace* robot lengan yang sebenarnya. Hal tersebut terlihat dari data pada Tabel 4 dimana meskipun algoritma melihat bahwa koordinat tersebut tidak valid, namun program masih dapat mengontrol robot menuju koordinat tersebut. Namun melihat tujuan dari algoritma ini adalah untuk menghilangkan kemungkinan tujuan yang tidak valid, algoritma ini dapat memenuhi fungsinya dengan cukup baik.

D. Pengujian Kesuksesan Inverse Kinematics Solver Dalam Menemukan Solusi Joint OpenManipulator

Pengujian ini bertujuan untuk mengetahui frekuensi keberhasilan *library ikpy* dalam menemukan solusi *inverse kinematics* dari koordinat tujuan yang diberikan. Alasan dari

ISSN (Print) : 2621-3540
ISSN (Online) : 2621-5551

pengujian ini karena terdapat kemungkinan *library* menghasilkan hasil yang tidak mampu dicapai oleh robot lengan, baik dikarenakan adanya tabrakan, nilai *joint* diluar kemampuan robot, dan lain-lain.

Pengambilan data dilakukan sebanyak 20 kali menggunakan program yang telah dibuat. Data pengujian menunjukkan adanya beberapa kali program gagal dalam melakukan kontrol terhadap robot lengan. Persentase keberhasilan dari program yang didapat dari pengujian ini adalah sebesar 70% dan dapat dianggap bekerja dengan cukup baik.

E. Pengujian Waktu Proses Kontrol OpenManipulator

Pengujian dilakukan untuk mengetahui waktu yang dibutuhkan program kontrol *OpenManipulator* sampai *node* mengirimkan perintah kepada robot manipulator dan apakah frekuensi yang digunakan pada program kontrol *OpenManipulator* sesuai dengan yang dibutuhkan. Pengujian dilakukan dengan melihat selisih waktu di awal dan di akhir fungsi *callback*. Karena tingkat kesuksesan program tidak mencapai 100%, maka terdapat 2 kemungkinan yang dapat terjadi saat menjalankan program sehingga perlu diketahui waktu proses dari kedua kondisi tersebut.

Dari proses pengambilan data, diambil 10 data pertama dari setiap kondisi. Rata – rata dari waktu proses saat berhasil adalah 0.6458 detik sedangkan rata-rata waktu saat program gagal adalah 0.63 detik. Dapat disimpulkan bahwa waktu proses pada kedua kondisi hampir sama dan frekuensi *subscribe* dari *node* kontrol *turtlebot* mencukupi waktu proses yang dibutuhkan, yaitu 1 *Hz*.

IV. KESIMPULAN

Dari proses perancangan, pembuatan dan pengujian program dapat ditarik beberapa kesimpulan sebagai berikut:

1. Program pendeteksi tangan dapat dengan mudah mengidentifikasi bentuk tangan terbuka dan mengepal, namun kesulitan dalam mengidentifikasi bentuk tangan menjepit dan tidak diketahui. Hal ini disebabkan kurangnya variabel penentu bentuk tangan.
2. Program kontrol *turtlebot* dapat dengan cukup cepat mengontrol robot. Waktu yang dibutuhkan program untuk memproses perintah dan melakukan kontrol adalah 0.1035 detik dengan waktu maksimal 0.3471 detik.
3. Algoritma validasi dapat digunakan untuk mengurangi kemampuan *error* yang diakibatkan koordinat yang tidak valid, namun mengorbankan sebagian kecil *workspace* sebenarnya dari *OpenManipulator.*
4. Program kontrol *OpenManipulator* dapat dianggap cukup berhasil dalam melakukan kontrol terhadap robot lengan dengan tingkat kesuksesan sebesar 70%.
5. Waktu kontrol *OpenManipulator*, meskipun tidak secepat *turtlebot*, memiliki waktu yang relatif cukup cepat dengan rata-rata waktu 0.6458 detik saat berhasil dan 0.63 detik saat gagal.

DAFTAR PUSTAKA

[1] Rouse, M. (April 2011). *WhatIs : Tech Target*. Retrieved November 13, 2019, from Tech Target: https://whatis.techtarget.com/definition/natural-user-interface-NUI

[2] Kramer, K.-L. (2012). *User Experience in the Age of Sustainability.* Morgan Kaufmann.

[3] Edwin C. Montiel Vázquez et al. (2019). *MOBMA: Gesture driven mobile manipulator.*

[4] Bassily, D., Guettler, J., Georgoulas, C., & Linner, T. (2014). *Intuitive and Adaptive Robotic Arm Manipulation using the Leap Motion Controller. ISR ROBOTIK 2014*, (pp. 78-84). Munich.

[5] *Manipulation: OpenManipulator*. (u.d.). Retrieved November 12, 2019, from E-manual: http://emanual.robotis.com/docs/en/platform/turtlebot3/manipulation/#manipulation

[6] *Developer : Leap Motion*. (n.d.). Retrieved November 8, 2019, from Leap Motion: https://developer.leapmotion.com/documentation/