CESS (Journal of Computer Engineering System and Science) **p-ISSN :2502-7131**
Vol. 5 No. 2 Juli 2020 **e-ISSN :2502-714x**

PENGENALAN BENTUK TANGAN SECARA *REAL TIME* MENGGUNAKAN *LEAP MOTION* DAN *K-NEAREST NEIGHBORS* SEBAGAI SISTEM KENDALI ROBOT BERODA

Supria[1], Depandi Enda[2], Muhamad Nasir[3]

[1,2] *Politeknik Negeri Bengkalis*
Jl. Bathin alam, Sei. Alam, Bengkalis - RIAU
[1]Phiya@polbeng.ac.id, [2]Depandienda@polbeng.ac.id, [3]Nasir@polbeng.ac.id

Abstrak— **Sistem kendali robot saat ini telah banyak dibuat dengan menggunakan berbagai metode seperti sensor accelerometer, sensor suara, leap motion. Pada penelitian ini diusulkan pengenalan bentuk tangan secara real time menggunakan leap motion dan K-Nearest Neighbors (KNN) sebagai sistem kendali robot beroda. Leap motion digunakan untuk mendeteksi titik koordinat posisi tangan pada pandangan leap motion. pembentukan fitur dilakukan dengan mengukur jarak Euclidean distance antara palm position dengan finger tip. KNN digunakan untuk menentukan kelas pada data testing. Uji coba dilakukan menggunakan tangan penulis dengan 5 jenis instruksi yaitu maju, mundur, stop, belok kanan, belok kiri dan setiap instruksi di ujicoba sebanyak 10 kali percobaan. Dari hasil ujicoba yang dilakukan menunjukkan bahwa sistem yang diusulkan memiliki tingkat akurasi pengenalan 94%.**

Kata Kunci— **Sistem kendali, robot beroda, leap motion, pembentukan fitur, k-nearest neighbors.**

Abstract— **Today's robot control systems have been made using various methods such as accelerometer sensors, sound sensors, and motion leap. This research proposes the hand shapes recognition in real time using leap motion and K-Nearest Neighbors (KNN) as wheeled robot control systems. Leap motion is used to detect the coordinates of the hand position in the leap motion view. Extraction feature is done by measuring the distance between the palm position and the finger tip using the euclidean distance. KNN is used to determine the class in the testing data. The trial was conducted using the author's hand with 5 types of instructions, namely forward, backward, stop, turn right, turn left and each instruction was tested 20 times. From the results of the tests conducted showed that the proposed system has a 94% recognition accuracy rate.**

Keywords— **Remote control, wheeled robot, leap motion, feature exraction, k-nearest neighbors.**

I. PENDAHULUAN

Sebuah robot adalah kombinasi antara perangkat keras dan perangkat lunak. Robot tidak akan beraksi jika tidak memiliki instruksi-instruksi. Instruksi untuk menggerakkan robot dapat secara otomatis ataupun manual. Instruksi otomatis berarti robot akan bergerak secara otomatis sesuai dengan instruksi-instruksi berupa bahasa pemrograman tertentu yang ditanamkan pada kontroller atau mikrokontroller di robot itu sendiri. Sedangkan instruksi manual berarti robot akan bergerak jika menerima instruksi dari pengendali robot seperti remote kontrol.

Remote kontrol robot beroda telah banyak dibuat dengan berbagai metode. Ada remote kontrol menggunakan *leap motion* untuk menangkap gerakan tangan [1], sistem kendali menggunakan *smartphone android* dengan memanfaatkan interface, sensor suara, maupun sensor *accelerometer* [2] [3] [4] [5] [6].

Leap motion adalah sensor yang berfungsi untuk mendeteksi titik-titik koordinat tangan mulai dari siku sampai dengan ujung jari [7]. Perangkat ini dapat beroperasi pada jarak dekat dan memiliki presisi yang tinggi. Sensor pada leap motion mengarah pada sumbu *y* memiliki bidang pandang 150 derajat. Jarak pandang yang efektif adalah 25 – 600 mm. Leap motion banyak diterapkan pada aplikasi [8] [9] [10] [11].

Pada penelitian dengan judul Sistem Remote Control Robot Beroda Menggunakan Teknologi Leap Motion memiliki tingkat akurasi 90% [1]. Sistem ini cukup baik dalam mengenal bentuk tangan, namun pengenalannya menggunakan pengenalan secara manual yaitu dengan menentukan range dari rata-rata nilai fitur. Data fitur yang digunakan memiliki range [1 - 2000] atau tidak di lakukan normalisasi, sehingga membutuhkan resource yang cukup besar sehingga komputasinya berat.

Pada penelitian ini diusulkan pengenalan bentuk tangan menggunakan leap motion dan KNN sebagai sistem kendali robot beroda. Leap motion digunakan untuk menangkap titik koordinat tangan. Titik koordinat yang digunakan adalah titik tengah tangan (*palm* position) dan titik koordinat ujung jari (*finger tip*) pada setiap jari yaitu jari jempol (*thumb*), jari

telunjuk (*index*), jari tengah (*middle*), jari manis (*ring*), dan jari kelingking (*pinky*). Pembentukan fitur dilakukan dengan mengukur jarak antara *palm* dengan *tip* pada setiap jari. Jadi ada 5 jarak yang akan dijadikan fitur (5 fitur). Data jarak atau fitur akan direkam sebanyak 10 data pada setiap bentuk dan ada 5 bentuk tangan yang akan digunakan sebagai kelas, sehingga total ada 50 data (10 x 5). Data dinormalisasi pada range [0-1] agar mengurangi penggunaan resource pada proses komputasi [12]. Setiap data bentuk tangan akan diberi kelas secara manual kemudian disimpan dalam file *.csv yang akan dijadikan sebagai data sampel. KNN digunakan untuk proses pengenalan data testing (data baru) atau menentukan kelas pada data testing berdasarkan pada data sampel [13].

II. METODE PENELITIAN

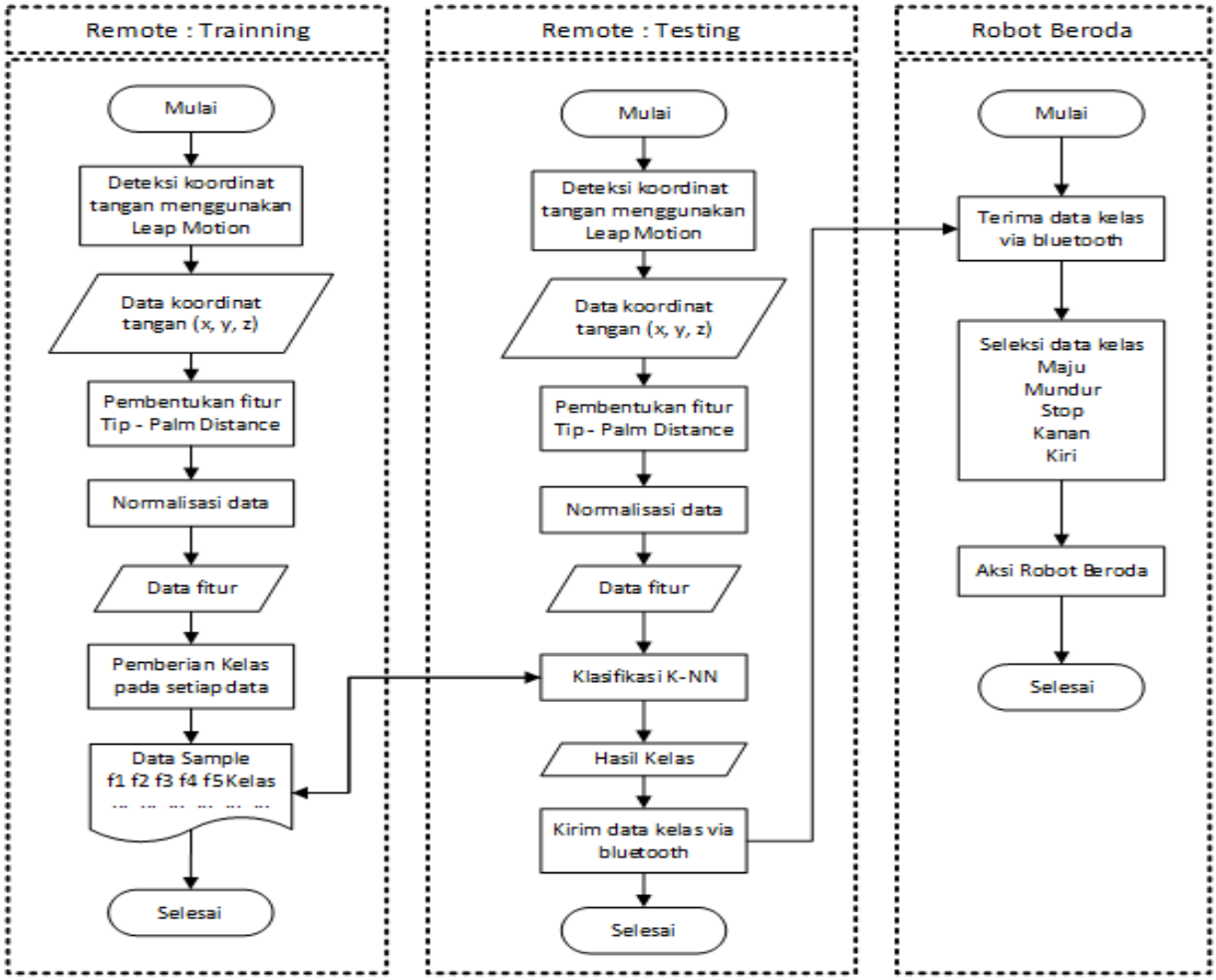
Pada sesi metode penelitian akan dibahas tentang metode dari sistem yang diusulkan. Ada tiga tahapan pada sistem yang diusulkan yaitu training, testing dan sistem di robot beroda. Adapun gambaran sistem secara umum dapat ditunjukkan pada Gambar 1.

Dari Gambar 1 menunjukkan bahwa ada 3 bagian utama pada sistem yang diusulkan. Training digunakan untuk melakukan training dengan tujuan untuk membentuk data sampel yang akan digunakan untuk data pada saat klasifikasi. Testing dilakukan untuk mendapatkan kelas dari data yang di inputkan berdasarkan kelas pada data sampel. Sistem robot beroda adalah sistem yang ada pada robot beroda dengan menggunakan mikrokontroler.

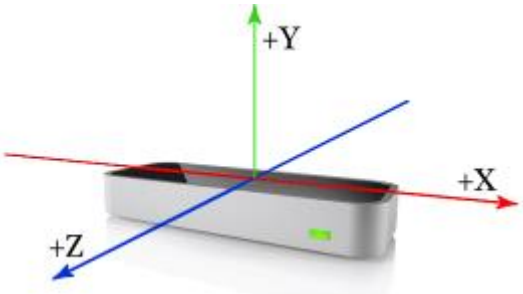
A. *Deteksi koordinat tangan*

Leap motion adalah sebuah sensor yang berfungsi untuk mendeteksi titik-titik koordinat setiap bagian tulang pada tangan dari siku sampai dengan ujung jari. Pada sistem ini, deteksi koordinat tangan dilakukan dengan menggunakan *Leap motion*. Titik koordinat yang digunakan adalah titik koordinat tengah tangan (*palm position*) dan titik koordinat ujung setiap jari (*finger tip*).

Leap motion memiliki nilai pada ruang 3-dimensi yaitu X, Y dan Z. Titik 0 terletak pada permukaan leap motion. adapun sistem koordinat leap motion dapat ditunjukkan pada Gambar 2.

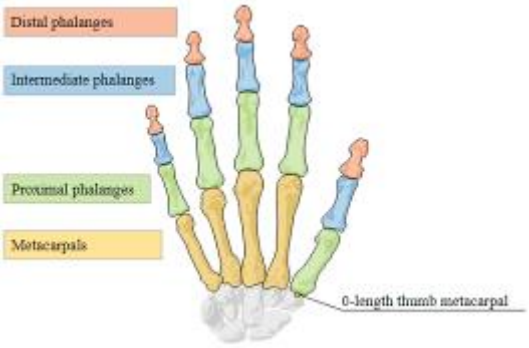


Gbr 1. Diagram sistem yang diusulkan



Gbr 2. Sistem koordinat leap motion.

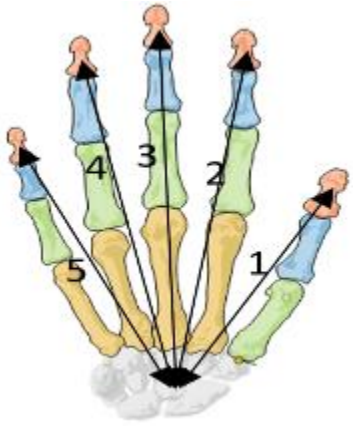
Pada setiap jari memiliki 4 titik koordinat yang dapat dideteksi *leap motion*. Adapun titik-titik koordinat pada setiap jari dapat ditunjukkan pada Gambar 3.



Gbr 3. Posisi koordinat tulang tangan.

B. Pembentukan fitur

Pembentukan fitur dilakukan untuk membedakan setiap bentuk tangan. Ada 5 fitur yang digunakan - pada sistem ini yaitu jarak antara palm dengan ujung setiap jari (*palm – tumb tip, palm - index tip, palm – middle tip, palm – ring tip, palm – pinky tip*). Adapun pengukuran jarak tersebut dapat ditunjukkan pada Gambar 4.



Gbr 4. Pembentukan fitur

Keterangan :

1. Jarak palm ke ujung jari jempol (palm – thumb tip).
2. Jarak palm ke ujung jari telunjuk (palm – index tip).
3. Jarak palm ke ujung jari tengah (palm – middle tip).
4. Jarak palm ke ujung jari manis (palm – ring tip).
5. Jarak palm ke ujung jari kelingking (palm – pinky tip).

Pengukuran jarak dilakukan dengan menggunakan *Euclidean Distance* yang dapat ditunjukkan pada Persamaan 1.

$$D_{palm-tip} = \sqrt{(x_{palm}-x_{tip})^2 + (y_{palm}-y_{tip})^2 + (z_{palm}-z_{tip})^2} \quad (1)$$

Dimana :

$D_{palm-tip}$: jarak antara titik koordinat palm dengan tip pada ruang 3 dimensi (xyz).
x_{palm} : nilai x pada titik koordinat palm.
x_{tip} : nilai x pada titik koordinat tip.
y_{palm} : nilai y pada titik koordinat palm.
y_{tip} : nilai y pada titik koordinat tip.
z_{palm} : nilai z pada titik koordinat palm.
z_{tip} : nilai z pada titik koordinat tip.

Terdapat 5 fitur yang digunakan dan 5 kelas atau instruksi. Adapun contoh data fitur dapat ditunjukkan pada Tabel 1.

TABEL I
CONTOH DATA TRAINNING SEBELUM DINORMALISASI

Fitur 1	Fitur 2	Fitur 3	Fitur 4	Fitur 5	Kelas
48.284	91.709	60.366	55.532	47.457	F
51.964	59.181	93.997	52.732	45.430	B
76.650	95.427	97.104	92.485	87.608	S
53.870	49.949	50.123	44.540	68.340	R
78.267	44.050	42.768	38.801	36.370	L

C. Normalisasi

Normalisasi data fitur (*feature scalling*) dilakukan untuk merubah data fitur ke dalam range tertentu [0 – 1] sehingga data fitur akan lebih proporsional [12] [7]. Normalisasi data dilakukan dengan tujuan untuk mengurangi penggunaan resource pada saat komputasi. Normalisasi data fitur dilakukan dengan menggunakan Persamaan 2.

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (2)$$

dimana :

X' : nilai data fitur setelah dinormalisasi.

X : nilai data fitur sebelum dinormalisasi.

X_{min} : nilai data fitur minimum atau nilai data fitur terkecil.

X_{max} : nilai data fitur maksimum atau nilai data fitur terbesar.

Hasil data fitur yang sudah dinormalisasi, nilainya akan berada pada range [0-1]. Adapun contoh data fitur yang sudah dinormalisasi dapat ditunjukkan pada Tabel 2.

TABEL 2
CONTOH DATA TRAINNING SETELAH DINORMALISASI

Fitur 1	Fitur 2	Fitur 3	Fitur 4	Fitur 5	Kelas
0,0615	0,9201	0,3175	0,3116	0,2163	F
0,1767	0,2921	0,9236	0,2595	0,1768	B
0,9493	0,9918	0,9796	0,9998	0,9999	S
0,2363	0,1139	0,1329	0,1069	0,6239	R
1,0000	0,0001	0,0004	0,0001	0,0000	L

D. Klasifikasi

Proses kalsifikasi dilakukan untuk proses pengenalan atau penentuan kelas terhadap data testing. Proses klasifikasi dilakukan dengan menggunakan metode KNN.

E. Koneksi data

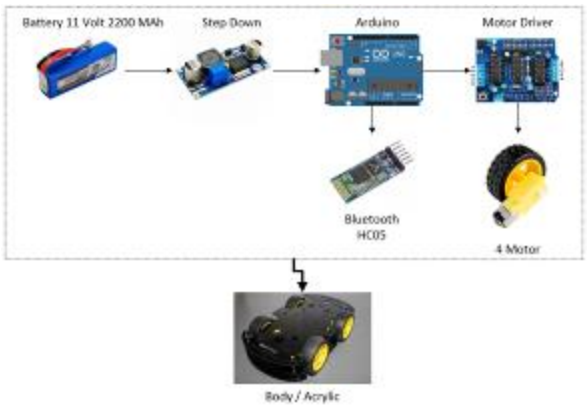
Data kelas yang dihasilkan dari proses klasifikasi akan digunakan sebagai kata kunci perintah untuk menggerakkan robot beroda. Untuk mengirim data kelas dari PC ke robot beroda dilakukan dengan menggunakan komunikasi via bluetooth. Ada bluetooth pengirim (*transfer*) pada PC dan bluetooth penerima (*receiver*) pada robot beroda.

F. Perancangan robot beroda

Terdapat 2 tahapan dalam perancangan robot beroda yaitu perancangan hardware dan perancangan software. Perancangan hardware dilakukan dengan menghubungkan komponen-komponen robot beroda dan mengatur daya yang masuk ke arduino menggunakan step down. Sedangkan perancangan software dilakukan dengan merancang algoritma dan pemrograman yang akan ditanamkan ke arduino.

G. Perancangan hardware

Ada beberapa komponen yang digunakan untuk perancangan robot beroda. Adapun komponen yang dibutuhkan adalah battery Turnigy dengan kapasitas 11 Volt dan 2200Mah, step down 3-35 Volt, Arduino Uno, Bluetooth HC-05, Motor driver, motor roda 4 buah, dan *Acrylic*. *Battery* digunakan sebagai *supply* daya utama yang akan dihubungkan ke *stepdown* yang digunakan untuk setting tegangan yang dibutuhkan oleh Arduino dan komponen lain. Arduino berfungsi sebagai *controller* yang akan ditanamkan instruksi-instruksi dalam bentuk bahasa pemrograman Arduino/c. Bluetooth digunakan sebagai *Bluetooth* penerima atau *receiver* dari *Bluetooth transfer* yang ada di perangkat sistem kendali atau laptop. Sedangkan motor driver digunakan untuk mengatur motor roda sebanyak 4 buah. Adapun perancangan robot beroda dapat ditunjukkan pada Gambar 3.



Gbr 5. Perancangan *hardware*

H. Perancangan software

Algoritma pada robot beroda adalah sebagai berikut:

```
#include <AFMotor.h>

char val;
AF_DCMotor motor1(1, MOTOR12_64KHZ);
AF_DCMotor motor2(2, MOTOR12_64KHZ);
AF_DCMotor motor3(3, MOTOR12_64KHZ);
AF_DCMotor motor4(4, MOTOR12_64KHZ);

void setup() {
        Serial.begin(9600);
}

void loop() {
        if ( Serial.available() > 0 ) {
                val = Serial.read();
                Serial.println(val);
        }
        if ( val == '1' ) {
                maju();
        }
        else if ( val == '2' ) {
                mundur();
        }
        else if ( val == '3' ) {
                berhenti();
        }
        else if ( val == '4' ) {
                kanan();
        }
        else if ( val == '5' ) {
                kiri();
        }
}
```

```
void setKecepatan(int SA, int SB, int SC, int SD)
{
        motor1.setSpeed(SA);
        motor2.setSpeed(SB);
        motor3.setSpeed(SC);
        motor4.setSpeed(SD);
}

void maju() {
        setKecepatan(100, 100, 100, 100);
        motor1.run(FORWARD);
        motor2.run(FORWARD);
        motor3.run(FORWARD);
        motor4.run(FORWARD);
}

void mundur() {
        setKecepatan(100, 100, 100, 100);
        motor1.run(BACKWARD);
        motor2.run(BACKWARD);
        motor3.run(BACKWARD);
        motor4.run(BACKWARD);
}

void kanan() {
        setKecepatan(150, 150, 150, 150);
        motor1.run(BACKWARD);
        motor2.run(FORWARD);
        motor3.run(FORWARD);
        motor4.run(BACKWARD);
}

void kiri() {
        setKecepatan(150, 150, 150, 150);
        motor1.run(FORWARD);
        motor2.run(BACKWARD);
        motor3.run(BACKWARD);
        motor4.run(FORWARD);
}

void berhenti() {
        motor1.run(RELEASE);
        motor2.run(RELEASE);
        motor3.run(RELEASE);
        motor4.run(RELEASE);
}
```

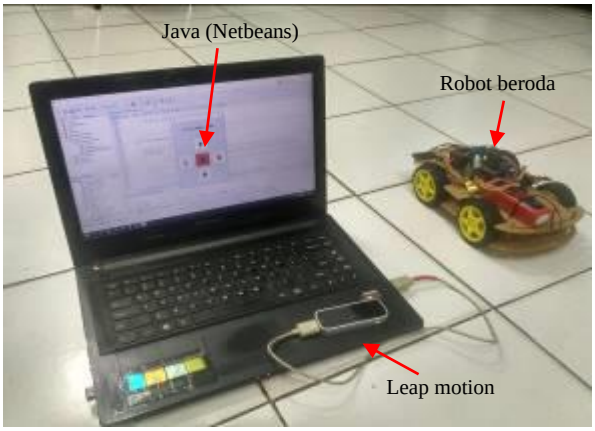
III. HASIL UJI COBA DAN PEMBAHASAN

A. *Data*

Hardware yang digunakan adalah laptop Lenovo AMD A8, 1 unit leap motion, dan 1 set robot beroda dengan menggunakan *microcontroller* Arduino serta menggunakan Bluetooth HC-05. Software yang digunakan adalah NetBeans 8.2 untuk tools developer bahasa pemrograman java, driver leap motion dan IDE Arduino. Data sampel yang digunakan adalah data hasil rekaman dengan menggunakan tangan penulis sendiri. Jumlah kelas yang digunakan ada 5 kelas, dan setiap kelas menggunakan 20 data sampel, sehingga total jumlah data sampel yang digunakan adalah 100 data sampel. Data sampel disimpan ke dalam 1 file dengan format file ***.csv**. Sedangkan untuk data uji coba yang digunakan adalah hasil rekaman langsung dengan menggunakan tangan penulis sendiri secara real time.

B. *Implementasi*

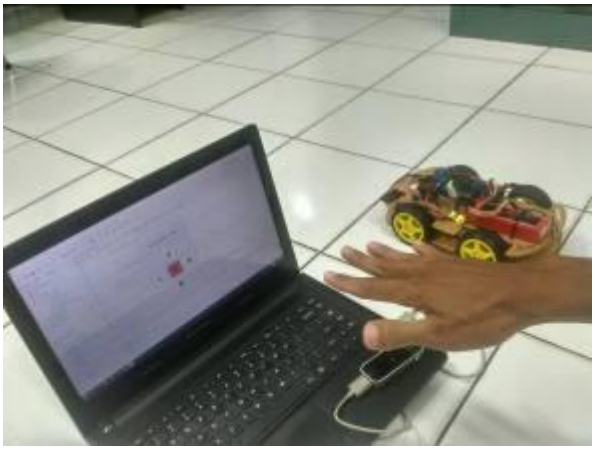
Implementasi dilakukan dengan menguji langsung sistem yang dibuat. Ada 5 instruksi yang uji yaitu maju, mundur, berhenti, belok kanan, dan belok kiri. Hasil sistem yang dibuat dapat ditunjukkan pada Gambar 6. Hasil ujicoba untuk instruksi maju dapat ditunjukkan pada Gambar 7. Hasil uji coba dengan instruksi mundur dapat ditunjukkan pada Gambar 8. Hasil uji coba dengan instruksi berhenti dapat ditunjukkan pada Gambar 9. Hasil uji coba dengan instruksi belo kanan dapat ditunjukkan pada Gambar 10. Hasil uji coba dengan instruksi belok kiri dapat ditunjukkan pada Gambar 11.



Gbr 6. Posisi awal


Gbr 7. Posisi maju


Gbr 8. Posisi mundur



Gbr 9. Posisi berhenti

Gbr 10. Posisi belok kanan

Gbr 11. Posisi belok kiri

C. Hasil uji coba

Untuk mengukur keberhasilan sistem yang diusulkan maka dilakukan uji coba terhadap sistem. Uji coba dilakukan sebanyak 20 kali setiap kelas atau instruksi. Untuk mengukur tingkat akurasi dari sistem yang diusulkan maka diukur dengan menggunakan persamaan 3.

$$Akurasi = \frac{PB}{TP} \; x \; 100\% \qquad (3)$$

dimana :

$Akurasi$ *:* Tingkat akurasi pengenalan bentuk tangan.
PB : Jumlah percobaan benar
TP : Total jumlah percobaan yang dilakukan

Hasil uji coba diimplementasikan dalam bentuk *confussion matrik* untuk mengetahui hasil tinggkat akurasi proses klasifikasi. Adapun hasil *confusion matrik* dapat ditunjukkan pada Gambar 12.

	F	B	S	R	L
F	20	0	0	0	0
B	1	19	0	0	0
S	0	0	20	0	0
R	0	0	0	17	3
L	0	0	0	2	18

Gbr 12. Confussion matrix hasil uji coba

Hasil akurasi pengenalan bentuk tangan pada setiap instruksi memiliki tinggkat akurasi yang baik dengan tingkat rata-rata akurasi 94%. Sedangkan tingkat rata-rata error adalah 6%. Adapun hasil akurasi setiap instruksi dan rata-rata nya dapat ditujukkan pada Tabel 3.

TABEL 3
HASIL AKURASI UJI COBA

Instruksi	Total ujicoba	Ujicoba benar	Ujicoba salah	Akurasi (%)	Error rate (%)
Maju	20	20	0	100	0
Mundur	20	19	1	95	5
Stop	20	20	0	100	0
Kanan	20	17	3	85	15
Kiri	20	18	2	90	10
Rata-rata				**94**	**4**

Uji coba waktu komputasi dilakukan untuk membandingkan waktu yang dibutuhkan untuk proses komputasi pada data sebelum dilakukan normalisasi dengan data sesudah dilakukan normalisasi. Adapun hasil waktu komputasi dapat ditunjukkan pada Tabel 3.

TABEL 3
HASIL WAKTU PROSES

Instruksi	Waktu komputasi (ms)	
	Sebelum Normalisasi	Setelah Normalisasi
Maju	1500	1340
Mundur	1520	1430
Stop	1510	1390
Kanan	1560	1380
Kiri	1600	1400

D. Pembahasan

Deteksi koordinat tangan dengan menggunakan *leap motion* dapat dilakukan dengan baik ketika dalam kondisi pencahayaan yang stabil atau jelas. Karena leap motion merupakan sensor kamera infrared yang membutuhkan pencahayaan yang cukup untuk mendeteksi objek.

Proses pengenalan bentuk tangan dengan menggunakan KNN memiliki akurasi yang cukup baik dibandingkan dengan tidak menggunakan metode klasifikasi.

Normalisasi data dilakukan dengan merubah data ke range tertentu [0-1]. Sehingga dengan normalisasi dapat mengurangi waktu yang dibutuhkan saat komputasi.

Pengenalan bentuk tangan dengan menggunakan tangan orang lain yang berbeda ukuran akan mengurangi tingkat akurasi.

Komunikasi untuk pengiriman data kelas dengan menggunakan bluetooth hanya dapat dilakukan dengan jarak sampai dengan 10 meter. Lebih dari 10 meter maka responsif bluetoorh tidak begitu baik.

IV. KESIMPULAN DAN SARAN

Sistem yang diusulkan adalah pengenalan bentuk tangan dengan menggunakan leap motion dan KNN. Berdasarkan hasil ujicoba yang telah dilakukan menunjukkan bahwa sistem yan diusulkan memiliki akurasi rata-rata 96%. Sedangkan proses normalisasi data dapat mengurangi waktu komputasi hingga 200ms.

Adapun saran untuk pengembangan penelitian ini adalah dengan melakukan kalibrasi agar dapat digunakan pada tangan yang berbeda-beda ukuran.

REFERENSI

[1] S. Supria and N. Fahmi, “Sistem Remote Control Robot Beroda Menggunakan Teknologi Leap Motion,” *Digit. Zo. J. Teknol. Inf. dan Komun.*, vol. 9, no. 1, pp. 1–11, 2018.

[2] A. N. Arvindan and D. Keerthika, “Experimental investigation of remote control via Android smart phone of arduino-based automated irrigation system using moisture sensor,” in *2016 3rd International Conference on Electrical Energy Systems, ICEES 2016*, 2016, pp. 168–175.

[3] Y. Hendriana and R. Hardi, “Remote control system as serial communications mobile using a microcontroller,” in *2016 International Conference on Information Technology Systems and Innovation, ICITSI 2016 - Proceedings*, 2017.

[4] A. Widiyanto and N. Nuryanto, “Rancang Bangun Mobil Remote Control Android dengan Arduino,” *Creat. Inf. Technol. J.*, vol. 3, no. 1, p. 50, 2016.

[5] J. P. Ventura, N. A. Cruz, and F. P. Lima, “A remote monitoring and control system for ecosystem replication experiments,” in *OCEANS 2016 MTS/IEEE Monterey, OCE 2016*, 2016.

[6] E. D. Tica, L. A. Perișoara, and A. Vasile, “An arduino platform for remote control and bus testing of programmable instruments,” 2017, pp. 308–311.

[7] S. Supria, D. Herumurti, and W. N. Khotimah, “Pengenalan Sistem Isyarat Bahasa Indonesia Menggunakan Kombinasi Fitur Statis Dan Fitur Dinamis LMC Berbasis L-GCNN,” *JUTI J. Ilm. Teknol. Inf.*, vol. 14, no. 2, p. 217, 2016.

[8] R. Fernando, S. Supria, and M. Nasir, “AplikasiMarawis Digital Menggunakan Sensor Leap Motion,” vol. 2, no. 2, 2017.

[9] K. Y. Fok, N. Ganganath, C. T. Cheng, and C. K. Tse, “A Real-Time ASL Recognition System Using Leap Motion Sensors,” in *Proceedings - 2015 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, CyberC 2015*, 2015, pp. 411–414.

[10] M. F. Humaira, S. Supria, D. Herumurti, and K. Widarsono, “Real Time SIBI Sign Language Recognition Based on K-Nearest Neighbor,” in *Proceeding of the Electrical Engineering Computer Science and Informatics*, 2018, vol. 5, no. 5, pp. 669–673.

[11] M. Mohandes, S. Aliyu, and M. Deriche, “Arabic sign language recognition using the leap motion controller,” in *IEEE International Symposium on Industrial Electronics*, 2014, pp. 960–965.

[12] W. N. Khotimah, R. A. Saputra, N. Suciati, and R. R. Hariadi, “Comparison between Back Propagation Neural Network and Genetic Algorithm Back Propagation Neural Network for Sign Language Recognition,” vol. 0, pp. 0–5, 2015.

[13] S. Tan, “Neighbor-weighted K-nearest neighbor for unbalanced text corpus,” *Expert Syst. Appl.*, vol. 28, no. 4, pp. 667–671, 2005.