Segmentasi Citra pada Robot Sepak Bola Beroda Menggunakan *Multilayer Neural Network* dan Fitur Warna HSV

Alam Ar Raad Stone, Nanik Suciati, dan Dini Adni Navastara
Departemen Informatika, Fakultas Teknologi Informasi dan Komunikasi, Institut Teknologi Sepuluh Nopember (ITS)
e-mail: dini_navastara@if.its.ac.id

***Abstrak*—Robot sepak bola beroda merupakan robot beroda yang dikembangkan untuk bermain sepak bola secara *full autonomous*. Robot bertanding secara tim melawan tim lain pada lapangan *indoor* yang telah disesuaikan ukurannya. Selama pertandingan, tidak diperbolehkan adanya campur tangan manusia. Pada robot, telah di pasang sebuah kamera dengan tujuan untuk menemukan di mana objek penting berada. Salah satu tahapan sebelum mendeteksi objek adalah segmentasi. Segmentasi citra bertujuan untuk memisahkan objek dengan latar belakang atau membagi citra ke dalam beberapa daerah dengan setiap daerah memiliki kemiripan atribut. Salah satu cara untuk melakukan segmentasi citra adalah dengan mengklasifikasikan tiap piksel pada citra sebagai objek tertentu maupun latar belakang. Pada penelitian ini, dilakukan klasifikasi tiap piksel pada ruang warna HSV menjadi 6 kelas. Yaitu: kawan (cyan), lawan (magenta), lapangan (hijau), garis lapangan (putih), bola (orange), dan objek lain (hitam). Proses klasifikasi dilakukan dengan menerapkan model *Multilayer Neural Network*. Kemudian hasil klasifikasi tersebut digunakan untuk membangun *lookup table* yang akan digunakan untuk klasifikasi tiap piksel warna secara cepat pada komputer robot. Dari hasil uji coba dan *fine tuning* terhadap *hyperparameter* dan arsitektur pada *multilayer neural network*, didapatkan nilai error rata-rata terkecil yaitu 0.16%. Kemudian dari evaluasi hasil segmentasi, diperoleh error rata-rata sebesar 19.37%.**

***Kata Kunci*—*Lookup Table*, *Neural Network*, Segmentasi Citra.**

I. PENDAHULUAN

ROBOT merupakan alat yang dapat membantu manusia baik dengan kontrol oleh manusia maupun secara otomatis dengan program yang telah ditanamkan dahulu pada robot. Untuk memacu perkembangan keilmuan di bidang robotika, maka banyak sekali kontes yang diadakan di seluruh dunia setiap tahunnya dengan tema yang beraneka ragam. Salah satunya adalah Kontes Robot Sepak Bola Indonesia Beroda (KRSBI beroda). Kontes Robot Sepak Bola Indonesia Beroda ini merupakan salah satu kegiatan yang merupakan bagian dari Kontes Robot Indonesia (KRI) sebagai ajang kompetisi rancang bangun dan rekayasa dalam bidang robotika [1]. Kontes Robot Sepakbola Indonesia Beroda diselenggarakan berdasarkan aturan pada lomba internasional *RoboCup Middle Size League* (MSL).

Di antara banyak bidang riset pada Robot Sepak Bola Beroda, visi komputer merupakan salah satu yang paling menantang. Hal ini dikarenakan robot menangkap citra yang memiliki karakteristik perpindahan lingkungan yang cepat. Pemain kawan, pemain lawan, dan bola berpindah dengan cepat, dan sering kali tidak dapat diprediksi. Robot diharuskan menangkap citra lingkungan tersebut menggunakan kamera, dan menemukan di mana objek penting berada. Tidak ada waktu untuk menjalankan algoritma yang kompleks. Semuanya harus diperhitungkan dalam waktu singkat, atau hanya akan menjadi sia-sia [2].

Salah satu tahapan sebelum mendeteksi objek adalah segmentasi. Segmentasi citra bertujuan untuk memisahkan objek dengan latar belakang atau membagi citra ke dalam beberapa daerah dengan setiap daerah memiliki kemiripan atribut. Salah satu metode segmentasi citra yang sederhana dan sering digunakan adalah *thresholding*. Metode *thresholding* dilakukan dengan cara mengganti nilai piksel dengan nilai baru apabila nilai piksel lama berada di antara nilai konstan tertentu. Metode ini berlaku pada citra dengan skala keabuan [3]. Pada citra berwarna, segmentasi dilakukan dengan cara memisahkan tiap komponen warna pada citra, kemudian dilakukan *thresholding* secara terpisah, lalu digabungkan kembali menggunakan operator *and*. Saat ini, ada beberapa representasi warna dalam pemrosesan gambar. RGB, CMYK, dan YIQ merupakan representasi warna *hardware-oriented*. Sementara itu, HSV merupakan representasi warna *user-oriented* [4]. Model ruang warna yang sering digunakan di dalam proses *thresholding* pada citra berwarna adalah ruang warna HSV. Bagaimanapun, terkadang sulit untuk menentukan nilai konstan yang akan digunakan pada proses *thresholding*.

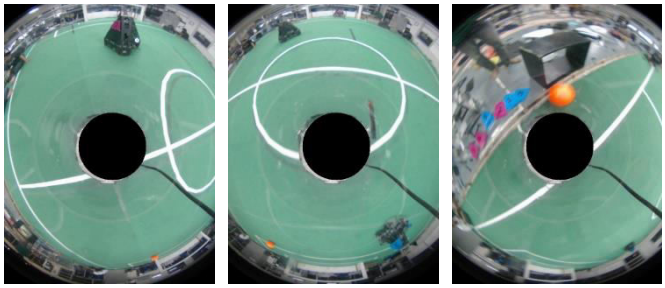
Cara lain untuk melakukan segmentasi citra adalah dengan mengklasifikasikan tiap piksel pada citra sebagai objek tertentu maupun latar belakang. Dalam penelitian ini, akan dilakukan klasifikasi tiap piksel pada ruang warna HSV menjadi 6 kelas. Yaitu: kawan, lawan, lapangan, garis lapangan, bola, dan objek lain. Proses klasifikasi dilakukan dengan menerapkan model *multilayer neural network* yang dilatih pada komputer lain menggunakan data yang diambil dari robot sepak bola beroda pada waktu yang beragam. Kemudian hasil klasifikasi tersebut digunakan untuk membangun *lookup table* yang akan digunakan untuk klasifikasi tiap piksel warna secara cepat pada komputer robot.

II. METODOLOGI

A. *Pembentukan Model Multilayer Neural Network*

Sistem dimulai dengan menerima citra RGB berukuran 480x640 dari kamera Logitech C922. Pada bagian depan

kamera ini, dipasang lensa *fisheye* 235°. Karakter lensa *fisheye* yang cembung menghasilkan citra yang terdistorsi melengkung. Lensa *fisheye* digunakan untuk meningkatkan *angles of view* [5]. Kamera dipasang pada bagian atas robot dengan arah menghadap ke bawah. Contoh citra RGB yang didapatkan dapat dilihat pada Gambar 1.



Gambar 1. Contoh citra RGB yang didapatkan dari kamera pada robot.

Kemudian, citra dikonversi ke ruang warna HSV. Persamaan 1-7 dari [6] digunakan untuk konversi citra dari ruang warna RGB ke HSV. Citra yang telah dikonversi, piksel-pikselnya diklasifikasi secara manual ke dalam 6 kelas yang telah ditentukan, yaitu: kawan (cyan), lawan (magenta), lapangan (hijau), garis lapangan (putih), bola (orange), dan objek lain (hitam).

$$V \leftarrow max(R,G,B) \tag{1}$$

$$S \leftarrow \begin{cases} \dfrac{V - min(R,G,B)}{v} & if\ V \neq 0 \\ 0 & otherwise \end{cases} \tag{2}$$

$$H \leftarrow \begin{cases} 60(G-B)/(V-min(R,G,B)) & if\ V = R \\ 120 + 60(B-R)/(V-min(R,G,B)) & if\ V = G \\ 240 + 60(R-G)/(V-min(R,G,B)) & if\ V = B \end{cases} \tag{3}$$

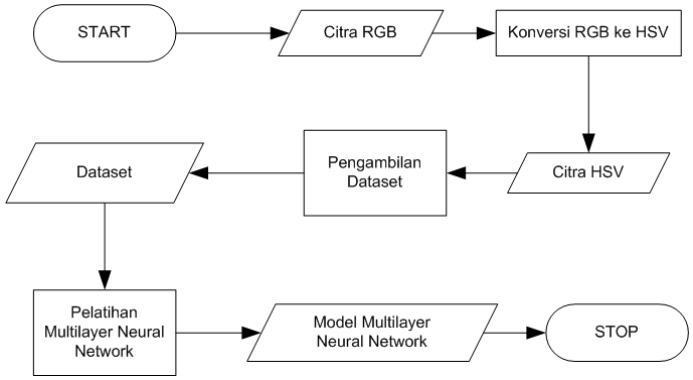
$$if\ H < 0,\ then\ H \leftarrow H + 360 \tag{4}$$

$$V \leftarrow 255V \tag{5}$$

$$S \leftarrow 255S \tag{6}$$

$$H \leftarrow H/2 \tag{7}$$

Karena keterbatasan perangkat keras, tidak dimungkinkan menggunakan seluruh data untuk melatih model *multilayer neural network*. Sehingga, dari 8.234.496 sampel piksel yang telah terklasifikasi secara manual, dipilih 500.000 data secara random. Dari data terpilih, 80 persennya digunakan sebagai data training, kemudian sisanya sebagai data testing.

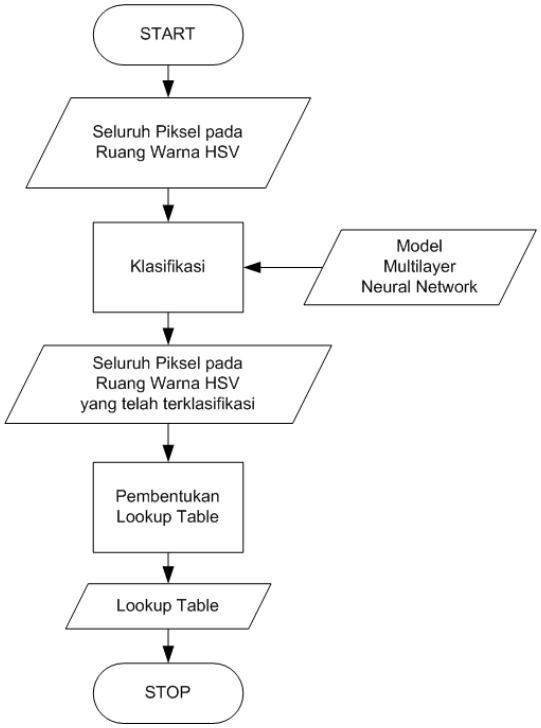


Gambar 2. *Flowchart* program pembentukan model *multilayer neural network*.

Dataset yang dihasilkan dari tahap pengambilan dataset ialah piksel-piksel HSV yang telah terklasifikasi secara manual kedalam 6 kelas. Dataset kemudian digunakan pada tahap pelatihan *multilayer neural network*. *Multilayer neural network* yang akan dilatih terdiri dari *input layer* dengan 3 *neuron* (nilai *hue*, nilai *saturation*, dan nilai *value*), 1 *hidden layer* dengan jumlah neuron yang akan dilakukan uji coba, dan *output layer* dengan 6 neuron, serta menggunakan fungsi aktivasi *sigmoid*. Secara umum, *flowchart* program pembentukan model *multilayer neural network* dapat dilihat pada Gambar 2.

B. Pembentukan Lookup Table

Implementasi sebenarnya pada robot, *lookup table* digunakan untuk menggantikan operasi perhitungan *multilayer neural network* dengan operasi indeks. Dengan menggunakan *lookup table* waktu untuk komputasi secara signifikan dapat lebih cepat karena mendapatkan data dari memori memakan waktu yang relatif singkat dibandingkan dengan melakukan operasi perhitungan.



Gambar 3. Flowchart program pembentukan *lookup table*

Pada tahap ini, model *multilayer neural network* digunakan untuk mengklasifikasikan seluruh piksel pada ruang warna HSV. Kemudian hasil klasifikasi disimpan pada *lookup table*. Indeks *lookup table* merupakan hasil operasi *or* antara nilai *hue* (8 bit) yang telah dilakukan *shift left* sebanyak 16, nilai *saturation* (8 bit) yang telah dilakukan *shift left* sebanyak 8, dan nilai *value* (8 bit).

Lookup table disimpan pada file untuk digunakan pada komputer robot. File yang dihasilkan berupa *binary file* berukuran sekitar 16 megabyte. *Flowchart* program pembentukan *lookup table* dapat dilihat pada Gambar 3. Ilustrasi *lookup table* dapat dilihat pada Tabel 1.

Tabel 1.
Ilustrasi *lookup table.*

H	S	V	Indeks (bin)	Indeks (int)	Kelas
0	0	0	000000000000 000000000000	0	3
0	0	1	000000000000 000000000001	1	3
. . .	. . .	. . .	. . .	. . .	. . .
0	1	0	000000000000 000100000000	256	3
0	1	1	000000000000 000100000001	257	3
. . .	. . .	. . .	. . .	. . .	. . .
255	255	255	111111111111 111111111111	16777215	5

III. UJI COBA DAN EVALUASI

A. *Lingkungan Uji Coba*

Lingkungan implementasi pada penelitian ini adalah sebuah *personal computer* (PC). PC yang digunakan pada penelitian ini memiliki prosesor Intel Core i5-7200U dengan kecepatan 2,6 GHz, dan *Random Access Memory* (RAM) sebesar 4 GB. Program dibangun menggunakan Python 3.6 pada sistem operasi Windows dengan menggunakan IDE PyCharm, serta C++ menggunakan IDE Visual Studio 2015 dengan pustaka pendukung OpenCV.

Spesifikasi tersebut cukup memadai untuk melatih model *multilayer neural network* dengan kecepatan komputasi yang baik, namun RAM masih cenderung kurang, sehingga tidak semua *dataset* dapat digunakan, melainkan hanya sebagian kecil saja.

B. *Fine Tuning Hyperparameter*

Fine tuning hyperparameter bertujuan untuk mencari konfigurasi *hyperparameter* yang paling baik pada *multilayer neural network* pada dataset yang telah diperoleh. Pada penelitian ini akan dilakukan *fine tuning* pada *learning rate* dan *lambda*. Untuk mengurangi *fluktuasi* pada kualitas model, akan diambil nilai rata-rata dari 2 pengujian pada masing-masing konfigurasi.

Learning Rate

Hyperparameter yang dilakukan *fine tuning* pertama kali ialah *learning rate*. Pada proses *fine tuning learning rate*, jumlah *epoch* yang digunakan berjumlah 50, dengan *batch size* sebanyak 10. *Fine tuning* pada proses ini tidak menggunakan L2 *regularization*, sehingga *lambda* diset dengan nilai 0,0. Konfigurasi yang digunakan dapat dilihat pada Tabel 2.

Dari hasil pengujian menggunakan berbagai nilai *learning rate* yang berbeda, didapatkan rata-rata *error* terkecil pada nilai *learning rate* 0,3. Sehingga, selanjutnya proses *fine tuning* akan dilakukan dengan menggunakan nilai *learning rate* 0,3. Hasil dari pengujian *learning rate* dapat dilihat pada Tabel 3. Dari informasi pada Tabel 2, didapatkan rata-rata *error* terkecil pada nilai *learning rate* 0,3. Sehingga, selanjutnya pengujian akan dilakukan dengan menggunakan nilai *learning rate* 0.3.

Tabel 2.
Hyperparameter dan arsitektur tetap pengujian *learning rate*.

Nama	Nilai
Hyperparameter Jaringan	
Epoch	50
Batch Size	10
Lambda	0,0
Arsitektur Jaringan	
Neuron tiap *Layer*	3 – 6 – 6

Tabel 3.
Hasil pengujian *learning rate*.

Learning Rate	Rata-rata *Error Training*	Rata-rata *Error Testing*
0,01	0,79%	0,77%
0,1	0,32%	0,33%
0,2	0,40%	0,42%
0,3	0,26%	0,25%
0,4	0,29%	0,28%

Lambda

Setelah dilakukan proses *fine tuning learning rate*, proses selanjutnya yaitu *fine tuning lambda*. *Lambda* merupakan *hyperparameter* pada L2 *regularization*. Pada proses *fine tuning lambda*, jumlah *epoch* yang digunakan berjumlah 50, dengan *batch size* sebanyak 10, serta *learning rate* 0,3 yang merupakan nilai terbaik yang didapatkan dari proses *fine tuning learning rate*. Konfigurasi yang digunakan dapat dilihat pada Tabel 4.

Tabel 4.
Hyperparameter dan arsitektur tetap pengujian *lambda*

Nama	Nilai
Hyperparameter Jaringan	
Epoch	50
Batch Size	10
Learning Rate	0,3
Arsitektur Jaringan	
Neuron tiap *Layer*	3 – 6 – 6

Dari hasil pengujian, didapatkan rata-rata *error* terkecil pada nilai *lambda* 0,1. Sehingga, selanjutnya pengujian akan dilakukan dengan menggunakan nilai *lambda* 0,1. Hasil dari pengujian *lambda* dapat dilihat pada Tabel 5.

Tabel 5.
Hasil pengujian *lambda*.

Lambda	Rata-rata *Error Training*	Rata-rata *Error Testing*
0	0,26%	0,25%
0,1	0,21%	0,21%
0,01	0,27%	0,26%
0,001	0,43%	0,43%

C. Perbandingan Arsitektur

Secara umum, *multilayer neural network* dapat memiliki *hidden layer* dan *neuron* pada tiap *layer* dengan jumlah berapapun. Namun, pada penelitian ini, arsitektur jaringan yang digunakan dibatasi dengan menggunakan 1 *hidden layer* saja, dengan jumlah *neuron* pada *input layer* berjumlah 3 (nilai *hue*, nilai *saturation*, dan nilai *value*), dan jumlah *neuron* pada *output layer* berjumlah 6 (kawan, lawan, lapangan, garis lapangan, bola, dan objek lain), serta menggunakan fungsi aktivasi *sigmoid*. Sehingga perbandingan arsitektur dilakukan dengan membandingkan jumlah *neuron* pada *hidden layer*. Konfigurasi yang digunakan pada pengujian arsitektur merupakan hasil *fine tuning* yang telah dilakukan sebelumnya, yaitu *learning rate* dengan nilai 0,3 dan *lambda* dengan nilai 0,1. Konfigurasi yang digunakan dapat dilihat pada Tabel 6.

Tabel 6.
Hyperparameter pengujian arsitektur jaringan.

Nama	Nilai
Hyperparameter Jaringan	
Epoch	50
Batch Size	10
Learning Rate	0,3
Lambda	0,1
Arsitektur Jaringan	
Hidden Layer	1

Dari hasil pengujian, didapatkan rata-rata *error* terkecil sebesar 0,16% pada arsitektur jaringan dengan 15 *neuron* pada *hidden layer*. Berdasarkan hasil pengujian, terlihat bahwa jumlah *neuron* pada *hidden layer* berbanding lurus dengan *run time*, yang berarti bahwa proses komputasi akan semakin lama apabila jumlah *neuron* semakin banyak. Namun demikian, semakin banyak *neuron* tidak selalu menghasilkan *error* yang lebih kecil. Hasil dari pengujian arsitektur *multilayer neural network* dapat dilihat pada Tabel 7. Dari hasil perbandingan arsitektur ini, arsitektur jaringan *multilayer neural network* dengan rata-rata error terkecil akan digunakan untuk klasifikasi tiap piksel pada ruang warna HSV yang akan digunakan pada proses selanjutnya.

Tabel 7.
Hasil pengujian arsitektur jaringan.

Neuron tiap *Layer*	Rata-rata *Run Time*	Rata-rata *Error Training*	Rata-rata *Error Testing*
3 – 6 – 6	51,72 menit	0,21%	0,21%
3 – 10 – 6	56,12 menit	0,19%	0,20%
3 – 13 – 6	55,87 menit	0,17%	0,17%
3 – 15 – 6	57,52 menit	0,16%	0,16%
3 – 17 – 6	61,41 menit	0,18%	0,19%
3 – 20 – 6	55,36 menit	0,22%	0,20%
3 – 50 – 6	53,25 menit	0,23%	0,22%
3 – 100 – 6	65,08 menit	0,21%	0,21%
3 – 200 – 6	68,00 menit	0,19%	0,18%
3 – 300 – 6	73,53 menit	0,22%	0,23%
3 – 400 – 6	77,73 menit	0,18%	0,17%
3 – 500 – 6	90,02 menit	0,27%	0,25%

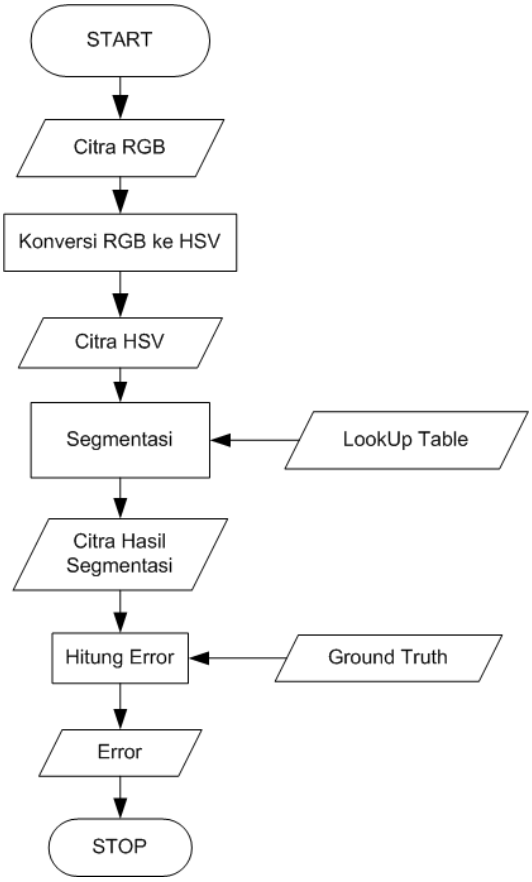
D. Uji Coba Segmentasi

Uji coba segmentasi dilakukan untuk memastikan apakah *lookup table* yang telah dibuat berdasarkan hasil klasifikasi menggunakan *multilayer neural network* telah layak diimplementasikan pada robot. Uji coba dilakukan dengan membandingkan tingkat *error* segmentasi dan waktu pemrosesan antara sistem lama yang telah diimplementasikan pada robot sebelumnya (menggunakan metode *thresholding*) dan sistem yang digunakan pada penelitian ini.

Tiap piksel citra hasil segmentasi dari kedua sistem akan dibandingkan dengan *ground truth*. *Error* dihitung dengan menggunakan Persamaan 8. dengan total data merupakan panjang citra dikalikan dengan lebar citra, dan prediksi salah merupakan jumlah piksel yang tidak sesuai dengan *ground truth*. Jumlah citra yang digunakan untuk uji coba adalah 4 citra, dengan *ground truth* merupakan citra uji coba yang telah dilakukan segmentasi secara manual oleh manusia.

$$error = \frac{prediksi\ salah}{total\ data} \times 100\% \quad (8)$$

Metode segmentasi yang digunakan pada penelitian ini ialah dengan melakukan klasifikasi tiap piksel pada citra yang telah dikonversi ke HSV berdasarkan *lookup table* yang dibangun dari hasil klasifikasi tiap piksel pada ruang warna HSV menggunakan *multilayer neural network*. *Flowchart* pengujian hasil segmentasi menggunakan metode ini dapat dilihat pada Gambar 4.

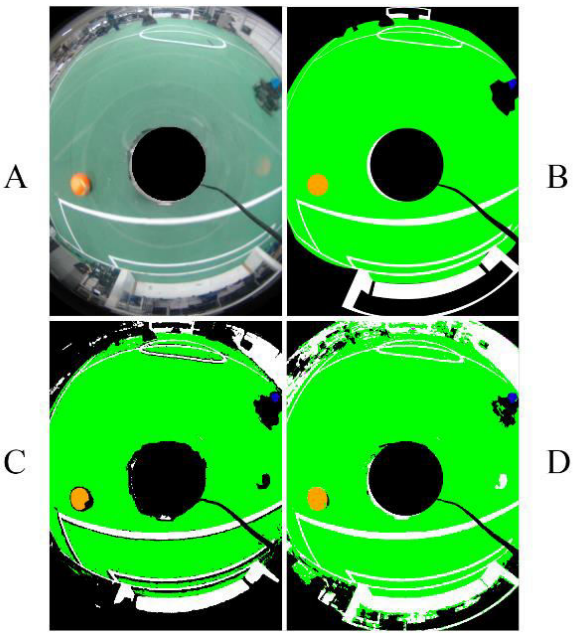


Gambar 4. *Flowchart* pengujian hasil segmentasi.

Sedangkan, metode *thresholding* dilakukan dengan cara mengganti nilai piksel dengan nilai baru sesuai dengan kelasnya, apabila nilai piksel lama berada di antara nilai konstan tertentu. Parameter tetap yang digunakan pada sistem lama dengan metode *thresholding* dapat dilihat pada Tabel 7.

Tabel 8.
Parameter tetap metode *thresholding*.

Kelas	*Hue*		*Saturation*		*Value*	
	min	max	min	max	min	max
Hijau	37	64	37	121	125	255
Putih	0	255	0	50	177	255
Orange	60	125	52	255	202	255
Cyan	16	64	196	255	144	255
Magenta	127	148	66	215	121	240
Hitam	0	255	0	255	0	83



Gambar 5. (A) Citra asli, (B) *ground truth*, (C) hasil segmentasi menggunakan sistem lama, dan (D) hasil segmentasi menggunakan sistem baru.

Berdasarkan hasil uji coba segmentasi, *error* rata-rata yang didapatkan pada sistem baru lebih besar daripada sistem lama. Hal ini dikarenakan banyaknya *noise* yang terdapat pada citra, seperti warna tembok yang mirip dengan garis lapangan. Namun, secara visual, terlihat bahwa hasil segmentasi menggunakan sistem baru memberikan hasil yang lebih baik di dalam lapangan dibandingkan dengan sistem lama. Sedangkan kecepatan segmentasi menggunakan sistem baru lebih cepat daripada sistem lama. Hal ini dikarenakan operasi baca memori memakan waktu yang relatif singkat dibandingkan dengan melakukan operasi perhitungan dan operasi logika. Hasil dari uji coba segmentasi dapat dilihat pada Tabel 8. Sedangkan contoh citra hasil segmentasi dapat dilihat pada Gambar 5.

Tabel 9.
Hasil uji coba segmentasi.

Keterangan	Citra ke-	*Run Time*	*Error*
Sistem Lama	1	0.1700s	9.55%
	2	0.1660s	12.75%
	3	0.1770s	8.87%
	4	0.1630s	13.18%
	Rata-rata	0.1690s	11.09%
Sistem Baru	1	0.0620s	14.92%
	2	0.0710s	16.05%
	3	0.0560s	12.96%
	4	0.0550s	33.57%
	Rata-rata	0.0610s	19.38%

Sistem lama yang dimaksud pada Tabel 8 merupakan metode segmentasi yang telah diimplementasikan pada robot sebelumnya, yaitu menggunakan *thresholding*. Sedangkan sistem baru yang dimaksud pada tugas akhir ini ialah metode segmentasi yang akan diimplementasikan pada robot, yaitu menggunakan *lookup table* yang dibangun dari hasil klasifikasi tiap piksel pada ruang warna HSV menggunakan *multilayer neural network*.

IV. KESIMPULAN/RINGKASAN

Berdasarkan hasil *fine tuning hyperparameter*, didapatkan *error* pengujian terkecil pada nilai *learning rate* sebesar 0,3, dan nilai *lambda* sebesar 0,1. Kemudian, dari serangkaian hasil uji coba, didapatkan model akhir dengan nilai *error* pada data testing sebesar 0,16%, atau mendapat nilai akurasi sebesar 99,84%. Pelatihan dari model tersebut berjalan selama 57,52 menit. Berdasarkan hasil uji coba, selisih nilai *error training* dan *error testing* bernilai sangat kecil. Sehingga dapat disimpulkan bahwa tidak terjadi *overfitting* pada model yang telah dibangun. Berdasarkan uji coba segmentasi, didapatkan error rata-rata sebesar 19,38% pada sistem baru, lebih besar daripada sistem lama. Hal ini dikarenakan banyaknya noise yang terdapat pada citra, seperti warna tembok yang mirip dengan garis lapangan. Kecepatan segmentasi menggunakan *lookup table* sedikit lebih cepat daripada *thresholding*. Hal ini dikarenakan operasi baca memori memakan waktu yang relatif singkat dibandingkan dengan melakukan operasi perhitungan dan operasi logika. Sehingga, segmentasi menggunakan *lookup table* layak diimplementasikan pada robot yang memiliki visi dengan karakteristik perpindahan lingkungan yang cepat seperti robot sepak bola beroda.

Pembuatan program pelatihan model *multilayer neural network* pada penelitian ini dilakukan tanpa bantuan pustaka pihak ketiga, sehingga komputasi dilakukan belum menggunakan GPU. Pada penelitian selanjutnya, disarankan untuk menggunakan pustaka pihak ketiga yang mendukung komputasi menggunakan GPU pada tahap pelatihan *multilayer neural network*, sehingga waktu pelatihan dapat dilakukan dengan lebih cepat. Kemudian, perlu adanya proses dan pengembangan lebih lanjut sehingga hasil segmentasi dapat diolah menjadi data yang berguna bagi robot.

DAFTAR PUSTAKA

[1] D. J. P. d. K. Direktorat Kemahasiswaan, "Buku Panduan Kontes Robot Sepakbola Indonesia Divisi Beroda," Kementerian Riset, Teknologi dan Pendidikan Tinggi Republik Indonesia, 2018.

[2] A. J. R. Neves, A. J. Pinho, D. A. Martins and B. Cunha, "An Efficient Omnidirectional Vision System For Soccer Robots: From Calibration To Object Detection," *Mechatronics,* vol. 21, pp. 399-410, 2011.

[3] A. Rachmawan, "Penentuan Posisi Robot Sepak Bola Beroda Menggunakan Rotary Encoder dan Kamera," Fakultas Teknologi Elektro, Institut Teknologi Sepuluh Nopember, Surabaya, 2017.

[4] N. Suciati, D. Herumurti and A. Y. Wijaya, "Fractal-based Texture and HSV Color Features for Fabric Image Retrieval," *International Conference on Control System, Computing and Engineering,* pp. 178-182, 2015.

[5] Y. Prakoso, "Desain Dan Implementasi Pengukuran Posisi Bola Menggunakan Kamera 360 Derajat Pada Robot Sepak Bola," Fakultas Teknologi Elektro, Institut Teknologi Sepuluh Nopember, Surabaya, 2017.

[6] "OpenCV: Color conversions," [Online]. Available: https://docs.opencv.org/3.4.1/de/d25/imgproc_color_conversions.html#color_convert_rgb_hsv. [Accessed 27 Juni 2018].

[7] D. Svozil, V. Kvasnicka and J. Pospichal, "Introduction to Multi-layer Feed-forward Neural Networks," *Chemometrics and Intelligent Laboratory Systems,* no. 39, pp. 43-62, 1997.

[8] J. Bruce, T. Balch and M. Veloso, "Fast and Inexpensive Color Image Segmentation for Interactive Robots," in *Intelligent Robots and Systems, 2000. (IROS 2000).*, Takamatsu, Japan, 2000.

[9] M. A. Morshidi, M. H. Marhaban and A. Jantan, "Color segmentation using multi layer neural network and the HSV color space," in *International Conference on Computer and Communication Engineering, 2008. ICCCE 2008*, Kuala Lumpur, Malaysia, 2008.

[10] M. A. Nielsen, "Neural Networks and Deep Learning," Determination Press, 2015.